Master's thesis

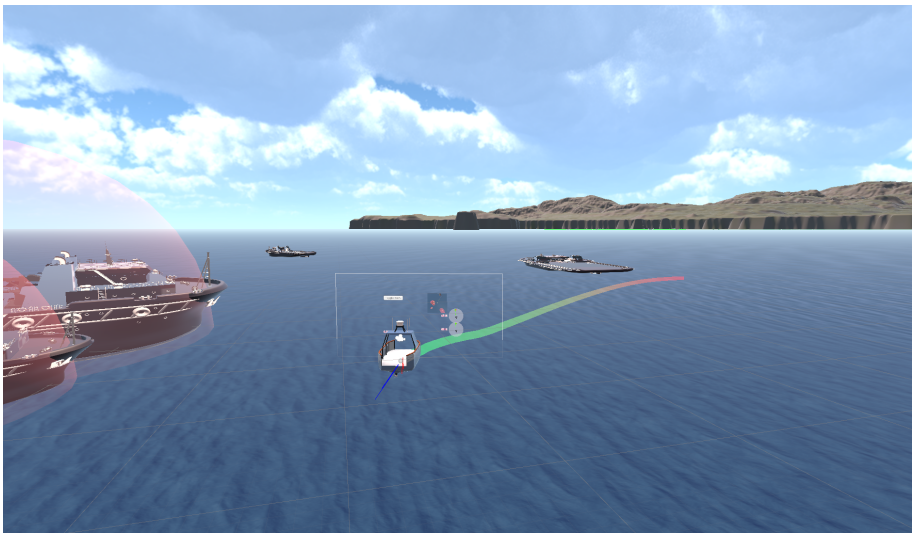
NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Svein Jostein Husa
Aksel Vaaler

Safe Reinforcement Learning in Marine Navigation and Control

Using a Predictive Safety Filter for Safety Verification on Autonomous Surface Vessels

Master's thesis in Cybernetics and Robotics
Supervisor: Adil Rasheed
June 2023



NTNU
Norwegian University of
Science and Technology

Svein Jostein Husa
Aksel Vaaler

Safe Reinforcement Learning in Marine Navigation and Control

Using a Predictive Safety Filter for Safety Verification on Autonomous Surface Vessels

Master's thesis in Cybernetics and Robotics
Supervisor: Adil Rasheed
June 2023

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

NTNU
Norwegian University of
Science and Technology

Abstract

The number of maritime systems being launched in the ocean is increasing every year, including the development of offshore wind farms, underwater robotics for ocean condition monitoring, and autonomous ship transport. Many of these activities are safety-critical, making it essential to have a closed-loop control system that satisfies constraints arising from underlying physical limitations and safety aspects in a robust manner. However, this is often challenging to achieve for real-world systems. For example, autonomous ships at sea have nonlinear and uncertain dynamics, and are subject to numerous time-varying environmental disturbances such as waves, currents, and wind. There is increasing interest in using machine learning-based approaches to adapt these systems to more complex scenarios, but there is currently no standard framework to guarantee the safety and stability of such systems.

Recently, predictive safety filters (PSF) have emerged as a promising method for ensuring constraint satisfaction in learning-based control, bypassing the need for explicit constraint handling in the learning algorithms themselves. The safety filter approach leads to a modular separation of the problem, allowing the usage of arbitrary control policies in a task-agnostic way. The filter takes in a potentially unsafe control action from the main controller and solves an optimization problem to compute a minimal perturbation of the proposed action, which adheres to both physical and safety-related constraints.

In this work, we combine reinforcement learning (RL) with predictive safety filtering in the context of marine navigation and control. The RL agent is trained on path following and safety adherence across a wide range of randomly generated environments, while the predictive safety filter continuously monitors the agents' proposed control actions and modifies them if necessary. The combined PSF/RL scheme is implemented on a simulated model of Cybership II, a miniature replica of a typical supply ship. Safety performance and learning rate are evaluated and compared with those of a standard, non-PSF, RL agent. It is demonstrated that the predictive safety filter is able to keep the vessel safe, while not prohibiting the learning rate and performance of the RL agent.

Sammendrag

Antall maritime stystemer som settes til havs øker for hvert år, inkludert utviklingen av havvindsparker, undervannsrobotikk, og autonom skipstransport. Mange av disse systemene er sikkerhetskritiske, noe som gjør det viktig å ha et lukket sløyfe kontrollsystem som tilfredsstiller underliggende fysiske begrensninger og sikkerhetsaspekter på en robust måte. Dette er imidlertid ofte utfordrende å oppnå for systemer i den virkelige verden. For eksempel har autonome skip ikke-lineær og usikker dynamikk, og er underlagt mange tidsvarierende miljøforstyrrelser som bølger, havstrømmer og vind. Det er økende interesse for å bruke tilnærminger basert på maskinlæring for å tilpasse disse systemene til mer komplekse scenarioer, men det finnes ingen standard rammeverk for å garantere sikkerheten og stabiliteten til slike systemer.

Nylig har prediktive sikkerhetsfiltre (PSF) dukket opp som en metode for å sikre oppfyllesle av begrensninger i læringbasert kontroll, og omgår behovet for eksplisitt håndtering av begrensninger i selve læringalgoritmene. Sikkerhetsfiltertilnærmingen fører til en modulær separasjon av problemet, og tillater bruk av vilkårlige kontrollpolitikker (engelsk: control policies) på en måte som er uavhengig av oppgaven. Filteret tar inn en potensielt usikker foreslått kontrollhandling fra hovedkontrolleren og løser et optimaliseringsproblem for å beregne en minimal endring av den foreslåtte handlingen, som overholder både fysiske og sikkerhetsrelaterte begrensninger.

I dette arbeidet kombinerer vi forsterkende læring (engelsk: reinforcement learning / RL) med prediktiv sikkerhetsfiltrering til marin navigasjon og kontroll. RL-agenten blir trent på sti-følging og sikkerhetsoverhodelse over et bredt spekter av tilfeldig genererte miljøer, mens det prediktive sikkerhetsfilteret kontinuerlig overvåker agentenes foreslåtte kontrollhandlinger og modifiserer dem om nødvendig. Den kombinerte PSF/RL-metoden er implementert på en simulert modell av Cybership II, en miniatyrreplika av et typisk forsyningsskip. Sikkerhetsytelse og læringsrate blir evaluert og sammenlignet med en standard RL-agent uten PSF. Det er demonstrert at det prediktive sikkerhetsfilteret er i stand til å holde fartøyet trygt, samtidig som det ikke forhindrer læringsraten eller ytelsen til RL-agenten.

Preface

This report is our submission to TTK4900 Engineering Cybernetics, Master's thesis, at the Norwegian University of Science and Technology (NTNU). The report was developed and written under the guidance of Prof. Adil Rasheed.

We would like thank Daniel Menges and Thomas Nakken Larsen for their support and insights, which greatly contributed to the quality of the thesis. Prof. Rasheed coordinated and supervised the work, and the results achieved would not have been possible without his guidance. Finally we would like to thank Haakon Robinson who first proposed the idea of applying predictive safety filters in marine control, and was instrumental in the development of the project report which served as the starting point for the thesis.

01.06.2023, Trondheim
Aksel Vaaler
Svein Jostein Husa

Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

Abbreviations

AI	Artificial Intelligence
RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
PSF	Predictive Safety Filter
MDP	Markov Decision Process
MPC	Model Predictive Control
COLAV	Collision Avoidance
CBF	Control Barrier Function
DOF	Degrees Of Freedom
NED	North-East-Down
ODE	Ordinary Differential Equation
CRI	Collision Risk Index
PPO	Proximal Policy Optimization
CPI	Conservative Policy Iteration
OCP	Optimal Control Problem
ASV	Autonomous Surface Vessel
LiDAR	Light Detection And Ranging
CNN	Convolutional Neural Network
SDP	Semi-Definite Programming
QP	Quadratic Programming
CTE	Cross-Track Error
AIS	Automatic Identification System

1 Introduction

It has become common practice to employ machine learning methods for autonomous control, given they can provide excellent performance for complex, non-linear control problems and also generalize previously learned rules to new scenarios [1]. One of the most direct approaches is reinforcement learning (RL), which without any prior information about the physical model or surrounding environment, iteratively can approximate optimal behavior. While RL methods are most known for succeeding in the domain of computer games and simple, deterministic simulated environments, more sophisticated algorithms and increased processing power has gradually shifted the focus of reinforcement learning toward real-world applications.

However, a major challenge in RL and learning-based control research in general is *safety*, or how to guarantee safe behavior when the main control algorithm is of a black-box nature [2]. This thesis explores a promising method, the *predictive safety filter*, for learning-based marine navigation and control. The filter ensures safe environment exploration while minimally interfering with the operation of the learning agent.

Autonomous surface vessels (ASVs) have the potential to improve many aspects of the maritime transportation sector [3]. From reducing shipping costs [4], to reducing the risk of lives lost at sea [5], ASVs have several potential benefits compared with traditional human-controlled ships. However, the successful implementation of autonomous surface vessels implicates many challenges ranging from handling continuously varying environmental forces impacting the vessel's dynamics to avoiding collisions with other objects in continuously changing environments. AI-driven methods, such as reinforcement learning, play a crucial role in the development of ASVs because of the capacity to learn from experience and generalize learned behaviors to different scenarios, properties which are essential to operate autonomously at sea. However, most RL-algorithms are inherently unpredictable and non-transparent, which limits their use in safety-critical applications. Classical control theory approaches, on the other hand, can handle safety constraints in a robust and predictable manner, but lack the ability to handle situations and dynamics which are not explicitly accounted for.

Hybrid solutions, such as hybrid analysis and modeling, reduce uncertainty and increase flexibility by combining physics-based and data-driven models, overcoming their individual weaknesses while maximizing their strengths [6]. Therefore, we propose a hybrid algorithm comprising a predictive safety filter and model-free

reinforcement learning for path following and collision avoidance to ensure both flexibility and safe operation of ASVs in complex sea environments.

1.1 Motivation

The European Maritime Safety Agency (EMSA) estimates that 59% of all marine accident events in EU member states from 2014 to 2021 was due to human error [7]. Another report based on data from the USA National Transportation Safety Board (NTSB) from 1975 to 2017 puts this figure at 75%, while claiming that around 46% of marine accidents are directly caused by the crew [8]. Hence, there is a potential to improve marine safety by employing autonomous solutions for better decision-making at sea. Furthermore, economic and environmental advantages are also possible with increased autonomy, resulting in greater energy efficiency and requiring fewer people for safe operation.

Because autonomous ships can be considered a safety-critical system, human supervision will likely be required in some way until a high level of autonomy is reached. One example is the *Bastø Fosen VI* ferry, carrying passengers from Horten to Moss. It utilizes automatic crossing and docking functionality, while still maintaining a full crew to monitor the vessel to ensure safety for the passengers [9]. Therefore, the deployment of an end-to-end controller could become more challenging if there is no feedback to verify the intended decisions of the autonomous system. RL agents make decisions based on a learned policy, which can be difficult for a human observer to interpret or use for understanding a long-term plan. In the proposed RL + PSF approach, the predictive safety filter automatically rejects unsafe actions suggested by the RL agent, reducing the need for human monitoring and addressing concerns about the interpretability of the RL controller. Additionally, the PSF by design computes the predicted system trajectory, which easily can aid in human supervision.

1.2 Background

1.2.1 Safe reinforcement learning

A human learning to control a system would in most cases intuitively have some notion of what is safe and not safe to do. For example, when an inexperienced driver tries to learn the skills needed to acquire a driver's license, they will naturally think about which traffic situations are safe for them to approach. In the beginning, driving back and forth on a parking lot could be challenging enough, then they would progress to some roads with low traffic, and finally begin training in an urban city area. If all goes well, and because safety is considered, the driver should get their license without any major accidents. In reinforcement learning (RL) one might think it would be desirable if similar behavior emerged naturally when an algorithm tries to learn how to control a physical system. However, since

RL is mainly done in simulations, where agents can explore different strategies without any consequences, safety is often less of a concern. Most algorithms instead focus on exploring different states as efficiently as possible, blind to the risk of the different actions [10].

Most of the success in RL research so far has come from applications in games such as Go [11] and Atari [12], where safety considerations are not necessary. In recent years, complex autonomous systems, such as self-driving vehicles, delivery drones or unmanned vessels, have been approaching integration into society. For deep RL to be deployed in these systems that operate in close proximity to humans, there is a need for safety precautions to avoid severe damage or, in worst case loss of lives. Consequently, people expect an increased focus on safety in RL as training agents in the real world is likely to become more common [13].

Defining safety. It is not straightforward to come up with a clear definition on the meaning of *safety* in an RL context. There are several different approaches in the literature trying to define what the requirements for safe RL should be [13, 14, 15]. One common approach is to require humans to label states in an environment as "safe" or "unsafe" [16]. Thus, an agent could be considered safe if it never enters unsafe states. Another approach is centered around the notion of ergodicity, which in essence means that an agent in any state can reach any other state by following a suitable policy [17]. In other words, if every mistake could be reversed, the system is safe. This requires finding the set of policies that preserves ergodicity, which can be shown to be NP-hard [17]. An approximation of the problem can be solved more easily, but this may lead to sub-optimal exploration. Additionally, it is impractical for many physical systems where the ergodicity assumption does not hold [17]. Finally, there are also approaches focusing on changing the optimization criteria by e.g. maximizing the worst-case return or minimizing the variance of the return [10]. In this work, we will refer to *safety* as the ability to adhere to constraints, as other approaches can be either impractical or insufficient [13].

Safe exploration and constrained RL. Alternatively, the question of safety in RL algorithms can be viewed as the problem of performing *safe exploration*. An RL agent learns by exploring the environment and often has no inherent way of knowing whether an unexplored state is safe or not. Therefore, a crucial step to ensuring safety in RL is forcing the agent to only explore states which are known to be safe with the available information. This relates to the exploration-exploitation dilemma, an important challenge in RL that involves deciding whether to exploit the currently available data, or to explore new data by taking possibly unsafe actions to gain new information.

Constrained RL solves the safe exploration problem by subjecting the agent to a set of constraints. It is one of the most used approaches in safe RL, and commonly relies on the framework of Constrained Markov Decision Processes (CMDPs) [18]. An introduction to Markov Decision Processes (MDP) can be found in section

2.2.1. In addition to a reward function, CMDPs are equipped with a set of cost functions that represent the constraints we want the agent to stay within. Just like standard RL, the goal is to maximize the reward, but there is also an additional requirement that the total cost has to be below a predetermined threshold.

This approach has been proven to be useful because constraints are a natural way of formulating safety requirements and also because it separates the problem of safety and maximizing the reward. In standard RL, it could be a difficult task to find the right trade-off parameter in the reward function between solving a task and avoiding dangerous situations [13]. If the penalty is too small, the agent will act unsafely, and if it is too large, the agent might not learn anything. Furthermore, it is not possible to guarantee safe behavior, since the penalty for unsafe behavior is always weighed against the reward of completing a task more efficiently. Constrained RL resolves these problems by formulating safety requirements as constraints instead [13]. In this work, we explore an approach related to constrained RL called the predictive safety filter.

1.2.2 Predictive safety filters

A promising method in the field of safe reinforcement learning is the use of the *predictive safety filter* (PSF). The idea was first developed in [19]. The PSF is based on the model predictive control (MPC) principle [20], and can be seen as a modular and minimally intrusive safety certification mechanism suitable for a wide range of control architectures. At every time-step the PSF module receives a proposed control input from the high-level control system. Using the proposed control input and the available domain knowledge, the PSF predicts the trajectory of the system over a finite horizon and calculates the minimal control input perturbation, which ensures that the system remains in a safe state. In practice, we can say that the PSF acts as a safety supervisor, which only intervenes when it is absolutely necessary to do so in order to avoid a hazardous situation. Otherwise, the nominal operation of the system is undisturbed. Figure 1.1 shows a diagram illustrating the concept.

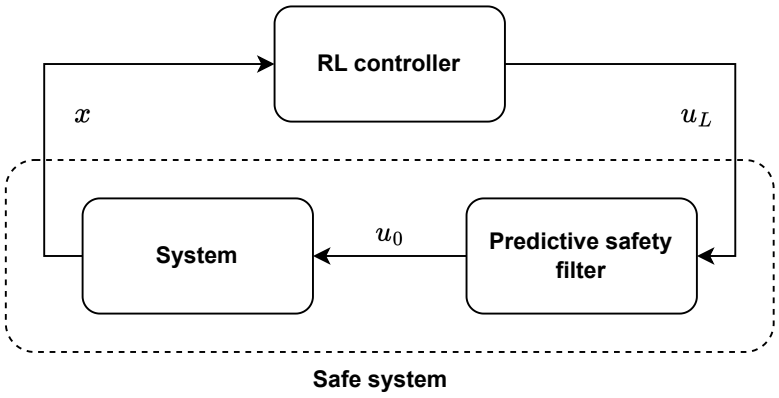


Figure 1.1: Predictive safety filter concept.

1.3 Contribution, Research Objectives and Research Questions

1.3.1 Contribution

The contribution of this thesis will be the design and verification of a predictive safety filter for marine collision avoidance and control, and analysis of the performance of the combined PSF/RL scheme with respect to safety adherence and quality of navigation, in sets of randomized simulated scenarios with varying difficulty levels. By evaluating the performance of an RL agent with and without a PSF, it is possible to highlight the advantages of utilizing a PSF, both in the training phase and for the final agent. The motivation is to make the use of RL in autonomous marine vessels more feasible to achieve in practice by providing an approach with a higher level of transparency and safety assurance than other state-of-the-art RL methods.

A thorough search of the relevant literature suggests that no previous studies have been conducted on the application of predictive safety filtering for learning-based marine craft navigation and control. However, other methods with similarities to the safety filter approach have been applied more generally to the field of marine vessel collision avoidance (COLAV). Thyri et al. [21] successfully developed a reactive control module based on the *control barrier function* (CBF) principle to act as a hazard avoidance mechanism for marine vessels. In Johansen et al. [22], a *scenario-based model predictive controller* (SBMPC), which uses a set of predefined course-altering strategies to find optimal collision-free paths, was shown to be an effective and viable method for collision avoidance in marine control. Furthermore, there are numerous attempts in applying RL for autonomous vessel navigation [23, 24, 25], which include the research that laid the foundation for this study. The main focus of these approaches has been to show the potential of RL as a feasible method for achieving autonomous navigation in marine vessels. This work goes further in focusing on the safety aspect of autonomous navigation, showing that the predictive safety filter can be a promising approach for guaranteeing safe navigation.

1.3.2 Objectives

Primary Objective: Examine whether the predictive safety filter is a viable method for safety certification in learning-based marine navigation and control. This encapsulates the design, implementation, and testing of the PSF over a wide range of simulated scenarios.

Secondary Objectives:

- Demonstrate real-time collision avoidance and safety verification using a predictive safety filter for marine control.

- Evaluate the impact of predictive safety filtering on the behavior and learning rate of reinforcement learning agents.
- Provide insights to how predictive safety filters can facilitate the use of reinforcement learning in complex real-world marine environments.

1.3.3 Research Questions

The following research questions were formulated to guide the study to achieve the presented research objectives:

- Can a PSF/RL scheme, capable of real-time collision avoidance and safety verification, be successfully realized within an ASV simulation environment?
- What effect does predictive safety filtering have on the performance and learning of reinforcement learning agents designed for autonomous surface vessels?
- Are predictive safety filters a feasible approach to incorporate reinforcement learning in real-world marine environments?

1.4 Structure of the Thesis

The thesis is structured as follows: Chapter 2 covers the theory underpinning the ship dynamics model, the predictive safety filter, and the reinforcement learning algorithm, as well as the steps necessary in order to design a predictive safety filter for the chosen dynamics model. In Chapter 3, the implementation of the PSF and the reinforcement learning algorithm is explained in detail, along with the structure and design of the various simulation experiments. In Chapter 4, simulation results are presented and analyzed, and the strengths and weaknesses of the thesis are discussed. Chapter 5 concludes the report, together with a brief discussion of the most interesting avenues for future research on the topic.

2 Theory

This chapter covers the details of the theory and equations used in our thesis. The chosen ship dynamics model is formulated, and the theory underpinning deep reinforcement learning and proximal policy optimization is presented. The nonlinear disturbance observer used to estimate the environmental disturbances affecting the system is outlined. Lastly, the predictive safety filter is explained, and the necessary equations for applying the predictive safety filter to the autonomous vessel are detailed.

2.1 Ship dynamics modelling

The ship used as basis for simulations and experiments is Cybership II, which is a 1:70 scale replica of a typical supply ship [26]. System identification parameters for the vessel were obtained from Skjetne (2004) [26]. To simulate the dynamics of the vessel, the rigid-body 3-DOF surge-sway-yaw model described in Fossen (2011) [27] was used.

2.1.1 Kinematics

We define the state of the system as

$$\begin{aligned} \boldsymbol{\eta} &= \begin{bmatrix} x & y & \psi \end{bmatrix}^T, \\ \boldsymbol{\nu} &= \begin{bmatrix} u & v & r \end{bmatrix}^T, \end{aligned} \tag{2.1}$$

where $\boldsymbol{\eta}$ denotes the coordinates and heading of the ship respectively, in the north, east, and down (NED) coordinate frame. $\boldsymbol{\nu}$ denotes the surge, sway, and yaw angular velocity of the vessel. The model kinematics are given by [27]

$$\dot{\boldsymbol{\eta}} = \boldsymbol{f}_{kinematic}(\psi, \boldsymbol{\nu}) = \boldsymbol{R}(\psi)\boldsymbol{\nu}, \tag{2.2}$$

where $\boldsymbol{R}(\psi)$ is the rotation matrix expressed by

$$\boldsymbol{R}(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.3}$$

2.1.2 Kinetics

The model kinetics are defined as [26]

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau} + \boldsymbol{\tau}_d, \tag{2.4}$$

where $\boldsymbol{M}$ denotes the mass matrix

$$\begin{aligned}
\boldsymbol{M} &= \begin{bmatrix} m_{11} & 0 & 0 \\ 0 & m_{22} & m_{23} \\ 0 & m_{32} & m_{33} \end{bmatrix} \\
m_{11} &= m - X_{\dot{u}} \\
m_{22} &= m - Y_{\dot{v}} \\
m_{23} &= m x_g - Y_{\dot{r}} \\
m_{32} &= m x_g - N_{\dot{v}} \\
m_{33} &= I_z - N_{\dot{r}}
\end{aligned} \tag{2.5}$$

$\boldsymbol{C}(\boldsymbol{\nu})$ is the Coriolis matrix

$$\begin{aligned}
\boldsymbol{C}(\boldsymbol{\nu}) &= \begin{bmatrix} 0 & 0 & c_{13} \\ 0 & 0 & c_{23} \\ -c_{13} & -c_{23} & 0 \end{bmatrix} \\
c_{13} &= -m_{11} v - m_{23} r \\
c_{23} &= m_{11} u
\end{aligned} \tag{2.6}$$

and $\boldsymbol{D}(\boldsymbol{\nu})$ characterizes the damping matrix

$$\begin{aligned}
\boldsymbol{D}(\boldsymbol{\nu}) &= \begin{bmatrix} d_{11} & 0 & 0 \\ 0 & d_{22} & d_{23} \\ 0 & d_{32} & d_{33} \end{bmatrix} \\
d_{11} &= -X_u - X_{|u|u}|u| - X_{uuu} u^2 \\
d_{22} &= -Y_v - Y_{|v|v}|v| - Y_{|r|v}|r| \\
d_{23} &= -Y_r - Y_{|v|r}|v| - Y_{|r|r}|r| \\
d_{32} &= -N_v - N_{|v|v}|v| - N_{|r|v}|r| \\
d_{33} &= -N_r - N_{|v|r}|v| - N_{|r|r}|r|
\end{aligned} \tag{2.7}$$

Furthermore, $\boldsymbol{\tau} = \begin{bmatrix} F_u, F_v, T_r \end{bmatrix}^T$ contains the surge force, sway force, and yaw moment applied to the ship. Since the mass matrix $\boldsymbol{M}$ is invertible, we can rewrite the kinetic equations as the explicit ODE

$$\dot{\boldsymbol{\nu}} = \boldsymbol{M}^{-1}(-\boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} - \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{\tau} + \boldsymbol{\tau}_d), \tag{2.8}$$

where $\boldsymbol{\tau}_d$ is a vector of generalized disturbance forces affecting the ship. In Skjetne (2004) [26] it is also shown that, given a suitable control allocation mechanism and sufficient vessel speed, the control force $\boldsymbol{\tau}$ can be reasonably approximated by

$$\boldsymbol{\tau} = \boldsymbol{B}\boldsymbol{u} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} F_u \\ T_r \end{bmatrix} \tag{2.9}$$

where F_u and T_r is the applied surge force and yaw moment, respectively. Thus, the resulting kinetics model becomes

$$\dot{\boldsymbol{\nu}} = \boldsymbol{f}_{kinetic}(\boldsymbol{\nu}, \boldsymbol{u}, \boldsymbol{\tau}_d) = \boldsymbol{M}^{-1}(-\boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} - \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{\tau}_d) \tag{2.10}$$

2.1.3 Collision risk

The collision risk index (CRI) is a metric used to quantify the risk of collision between two vessels. The approach used in the thesis is based on Heiberg et al. [28], which is designed to be compatible with the simulation environment. The CRI is calculated by the estimated time to the closest point of approach (TCPA) and the estimated distance to the closest point of approach (DCPA). Additionally, it considers the relative distance R, the relative speed V and the relative bearing θ_T. To do this, it uses fuzzy comprehensive evaluation methods, where the membership functions $u(\cdot) \in [0, 1]$ are used to determine the risk level associated with each risk factor.

$$\mathrm{CRI} = \alpha_{\mathrm{CPA}} \sqrt{u(\mathrm{DCPA}) \cdot u(\mathrm{TCPA})} + \alpha_{\theta_\mathrm{T}} u\left(\theta_\mathrm{T}\right) + \alpha_\mathrm{R} u(R) + \alpha_\mathrm{V} u(V), \tag{2.11}$$

where $\alpha_{\mathrm{DCPA}} + \alpha_{\mathrm{TCPA}} + \alpha_{\theta_\mathrm{T}} + \alpha_\mathrm{R} + \alpha_\mathrm{V} = 1$. Further details on the CRI calculation can be found in [28].

2.2 Deep Reinforcement Learning

In this section, we introduce the field of deep reinforcement learning to provide an understanding of the RL methods used in this work. Reinforcement learning is a sub-field of machine learning where a sequential decision-making agent learns a behavioral policy by iteratively acting in an environment and optimizing its parameters for the reward it receives [29]. The term "deep" refers to parameterizing the agent using deep neural networks, improving the RL methods' applicability and performance, particularly in complex environments with high-dimensional state spaces [11, 30]. Figure 2.1 shows a broad overview of the RL framework. It consists of two separate systems: an agent that makes decisions and an environment that is influenced by those decisions. Based on the feedback the agent obtains from the environment, it gradually learns more intelligent behaviour in order to maximize the rewards it receives. It does this by updating its *policy*, a mapping from states to action, which defines the agent's behaviour. While the RL framework could seem quite simplistic, it is inspired by how humans and animals learn and can lead to complex, goal-oriented behaviour [29]. By trial and error, adjusting behaviour based on feedback and the pursuit of maximizing rewards, intelligent behaviour can emerge through different types of RL algorithms.

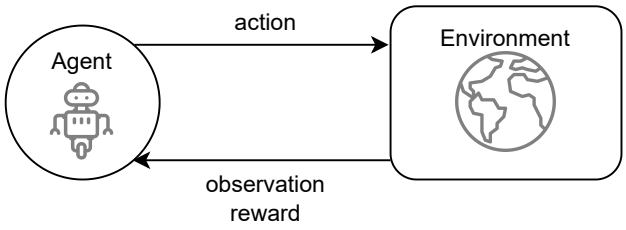


Figure 2.1: Simple overview of the RL framework. The agent performs an action which affects the state of the environment, and in return receives a reward and observation based on the new state.

2.2.1 RL preliminaries

One core assumption for RL is that the environment can be modeled as a Markov Decision Process (MDP), which is defined by the tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, p\left(s_0\right), \gamma\}$ where

- $\mathcal{S}$ is the state space,
- $\mathcal{A}$ is the action space,
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to p(\mathcal{S})$ is the transition function that represents the probability that an agent ends up in a new state $s' \in \mathcal{S}$ after taking a specific action $a \in \mathcal{A}$ from a state $s \in \mathcal{S}$,
- $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, which returns a scalar reward associated with the transition function,
- $p\left(s_0\right)$ is the initial state distribution, and
- $\gamma \in [0, 1]$ is the discount factor.

An agent acts in an environment according to a policy $\pi : \mathcal{S} \to p(\mathcal{A})$. Solving the MDP is equivalent to finding an optimal policy $\pi^\star$ that maximizes the reward. To do this, we need to introduce the action-value function $Q^\pi(s, a)$, defined as the expectation of the cumulative return for a given policy π.

$$Q^\pi(s, a) = \mathbb{E}_{\pi,\mathcal{T}}\left[\sum_{k=0}^{K} \gamma^k r_{t+k} \mid = s, a_t = a\right] \tag{2.12}$$

The optimal policy $\pi^\star$ can be defined as the policy that maximizes the expected return of $Q^\pi(s, a)$.

$$\pi^* = \arg\max_\pi Q^\pi(s, a) \tag{2.13}$$

$$= \arg\max_\pi \mathbb{E}_{\pi,\mathcal{T}}\left[\sum_{k=0}^{K} \gamma^k r_{t+k} \mid s_t = s, a_t = a\right] \tag{2.14}$$

2.2.2 Value-based and policy-based methods

There are two main approaches to training model-free RL agents; *value-based methods* learn an actor-value function $Q^{\pi}(s, a)$ and derive the optimal policy from it, while *policy-based methods* directly optimize a policy function $\pi_{\theta}(a \mid s)$ with parameters θ. A common way of doing this is by using policy gradients, which optimize the policy by gradient ascent on the expected reward.

While the value-based approach often is more sample efficient, they also tend to be less stable [31]. Examples of value-based methods algorithms are Deep-Q Networks (DQN) [12] and Hindsight Experience Replay (HER) [32]. Policy-based methods are more straightforward since they optimize the desired result, the policy. They are generally more reliable than value-based methods but can suffer from high sample complexity. Examples of policy-based methods are Trust Region Policy Optimization (TRPO) [33] and Proximal policy optimization (PPO) [34].

Lastly, *actor-critic methods* combine the two approaches by independently parameterizing a policy (the actor) and a value function (the critic) while learning both concurrently. With this, it overcomes many of the limitations of policy-based and value-based methods [35]. Examples of actor-critic algorithms are Soft Actor-Critic (SAC) [36] and Deep Deterministic Policy Gradient (DDPG) [37].

2.2.3 Proximal policy optimization

Proximal policy optimization (PPO) is a popular model-free algorithm in the family of policy gradient methods, originally developed by Shulman et al. [34]. By employing a trust region, it ensures that new policies do not deviate far from the old policy, avoiding large policy updates that could result in unstable learning. It does this by using a clipped surrogate objective function, which is a modified version of the standard policy gradient objective. The surrogate objective can be defined as follows:

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_{\theta} \left(a_t \mid s_t \right)}{\pi_{\theta_{\text{old}}} \left(a_t \mid s_t \right)} \hat{A}_t \right] = \hat{\mathbb{E}}_t \left[r_t(\theta) \hat{A}_t \right] \tag{2.15}$$

where $\hat{A}_t$ is the advantage estimate, which is a measure for how much better an action is compared to the average action in a given state. Moreover, the algorithm uses a modified version of this, known as the clipped surrogate objective function. This function ensures the probability ratio $r_t(\theta)$ is within the range $1-\epsilon$ and $1+\epsilon$.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta) \hat{A}_t, \operatorname{clip} \left(r_t(\theta), 1 - \epsilon, 1 + \epsilon \right) \hat{A}_t \right) \right] \tag{2.16}$$

Algorithm 1 shows pseudocode of the complete algorithm from the original paper [34]. The PPO algorithm is considered to be an efficient and robust algorithm, suited for various domains [34], and it has shown good results in previous work on the **gym-auv** simulation environment [25, 28]. Most notably, in Larsen et al. [38] the PPO algorithm was shown to outperform other state-of-the-art algorithms in a range of different scenarios in the **gym-auv** environment. Therefore, PPO was chosen as the RL method for this work.

Algorithm 1: Proximal Policy Optimization (PPO)

for iteration $k = 1, 2, \ldots$ **do**
  for actor $i = 1, 2, \ldots$ **do**
    Run policy $\pi_{\theta_{\text{old}}}$ in environment for T timesteps
    Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$
  end for
  Optimize surrogate L wrt θ, with K epochs and minibatch size $M \leq NT$
  $\theta_{\text{old}} \leftarrow \theta$
end for

2.3 Environmental disturbance observer

To estimate the environmental disturbances affecting the ship, we use the nonlinear disturbance observer algorithm developed in [39]. The observer system is defined by:

$$\begin{aligned} \hat{\boldsymbol{\tau}}_d &= \boldsymbol{\zeta} + \boldsymbol{\mu}(\boldsymbol{\nu}) \\ \dot{\boldsymbol{\zeta}} &= \boldsymbol{h}(\boldsymbol{\nu}, \hat{\boldsymbol{\tau}}_d) \end{aligned} \tag{2.17}$$

where $\hat{\boldsymbol{\tau}}_d$ is the estimate of the environmental disturbance forces affecting the ship, and $\boldsymbol{\zeta}$ is an observer variable. The estimated error and error dynamics are then described by:

$$\begin{aligned} \boldsymbol{z} &= \boldsymbol{\tau}_d - \boldsymbol{\zeta} - \boldsymbol{\mu}(\boldsymbol{\nu}) \\ \dot{\boldsymbol{z}} &= \dot{\boldsymbol{\tau}}_d - \dot{\boldsymbol{\zeta}} - \frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{\nu}} \dot{\boldsymbol{\nu}} \end{aligned} \tag{2.18}$$

Assuming that the true disturbance $\boldsymbol{\tau}_d$ is relatively slow-varying ($\dot{\boldsymbol{\tau}}_d \approx \mathbf{0}$) and inserting the ship model kinetics, the error dynamics become:

$$\dot{\boldsymbol{z}} = -\boldsymbol{h}(\boldsymbol{\nu}, \hat{\boldsymbol{\tau}}_d) - \frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{\nu}} \boldsymbol{M}^{-1}(\boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} - \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{\tau} + \boldsymbol{\tau}_d) \tag{2.19}$$

By defining $\boldsymbol{h}(\boldsymbol{\nu}, \hat{\boldsymbol{\tau}}_d)$ as:

$$\boldsymbol{h}(\boldsymbol{\nu}, \hat{\boldsymbol{\tau}}_d) = -\frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{\nu}} \boldsymbol{M}^{-1}(\boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} - \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{\tau} + \hat{\boldsymbol{\tau}}_d) \tag{2.20}$$

, [39] shows that the observer error converges by applying:

$$\frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{\nu}} = \boldsymbol{T} = \begin{bmatrix} \Gamma_1 \frac{1}{k_{11}} \sigma & 0 & 0 \\ 0 & \Gamma_2 \frac{1}{k_{22}} & -\Gamma_2 \frac{k_{23}}{k_{22} k_{33}} \\ 0 & -\Gamma_3 \frac{32}{k_{22} k_{33}} & \Gamma_3 \frac{1}{k_{33}} \end{bmatrix} \tag{2.21}$$

where $\sigma = 1 - \frac{k_{23} k_{32}}{k_{22} k_{33}}$ and k_{ij} are the elements of the mass matrix inverse $\boldsymbol{M}^{-1}$:

$$\boldsymbol{M}^{-1} = \begin{bmatrix} k_{11} & 0 & 0 \\ 0 & k_{22} & k_{23} \\ 0 & k_{32} & k_{33} \end{bmatrix} \tag{2.22}$$

$\Gamma_{1,2,3}$ are adaptive gains which will be chosen later. Using equations 2.20 and 2.21, the disturbance observer system (2.17) becomes:

$$\begin{aligned} \hat{\boldsymbol{\tau}}_d &= \boldsymbol{\zeta} + \boldsymbol{T}\boldsymbol{\nu} \\ \dot{\boldsymbol{\zeta}} &= -\boldsymbol{T}\boldsymbol{M}^{-1}(\boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} - \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{\tau} + \hat{\boldsymbol{\tau}}_d) \end{aligned} \tag{2.23}$$

2.4 Predictive Safety Filter

To ensure *safe exploration*, a *predictive safety filter* (PSF) is integrated into the control loop of the system. The main idea of the PSF, as first developed in [19], is to predict the future states of the system based on the current system state $\boldsymbol{x}(t)$ and the current proposed control action from the RL-agent $\boldsymbol{u}_L(t)$, and find the minimally perturbed control action $\boldsymbol{u}_0^*(\boldsymbol{x}(t), \boldsymbol{u}_L(t))$ which guarantees a safe state trajectory for all times $t' \in (t, \infty)$. At every time-step the PSF solves the finite-horizon optimal control problem (OCP):

$$\begin{aligned} &\min_{\boldsymbol{u}_{i|k}, \boldsymbol{x}_{i|k}} \; ||\boldsymbol{u}_{0|k} - \boldsymbol{u}_L(k)||_W^2 \\ &s.t. \quad (1) \quad \boldsymbol{x}_{0|k} = \boldsymbol{x}(k) \\ &\qquad (2) \quad \boldsymbol{x}_{i+1|k} = \boldsymbol{f}(\boldsymbol{x}_{i|k}, \boldsymbol{u}_{i|k}, \Delta T) \quad \forall i \in [0, N-1] \\ &\qquad (3) \quad \boldsymbol{x}_{i|k} \in \mathbb{X} \quad \forall i \in [0, N] \\ &\qquad (4) \quad \boldsymbol{u}_{i|k} \in \mathbb{U} \quad \forall i \in [0, N-1] \\ &\qquad (5) \quad \boldsymbol{x}_{N|k} \in \mathbb{X}_f \end{aligned} \tag{2.24}$$

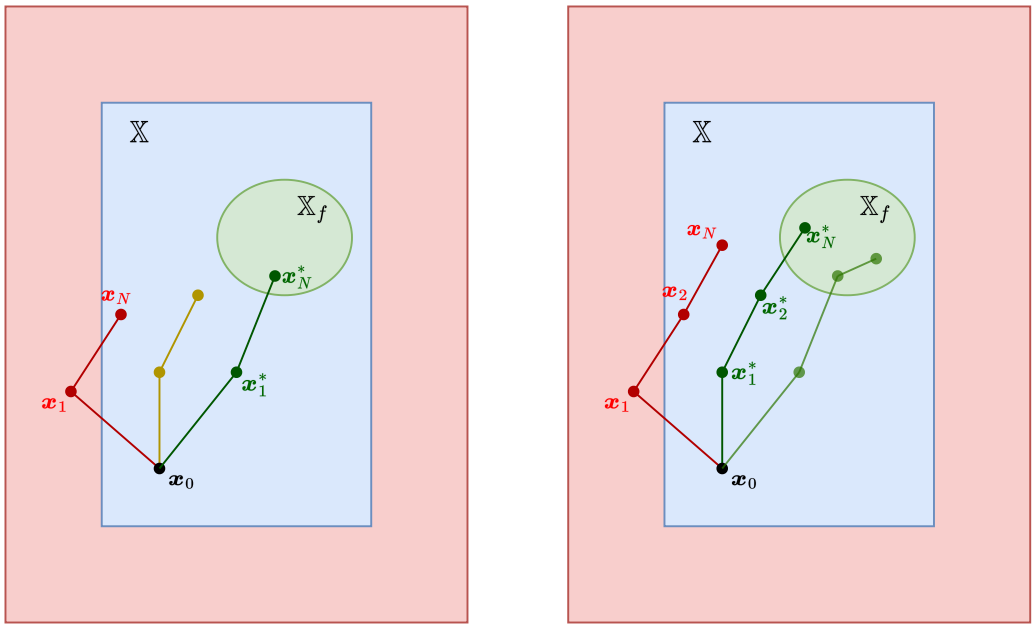
$\boldsymbol{x}(k)$ and $\boldsymbol{u}_L(k)$ are the system state and the proposed RL control action at the current time-step, respectively. W is the weighting matrix for the cost function. $\boldsymbol{f}(\cdot)$ are the discretized system model equations, where ΔT is the discretization step length. N is the number of states in the predicted trajectory (shooting nodes), which means that the prediction horizon is given by $T_f = N \cdot \Delta T$. $\mathbb{X} \subseteq \mathbb{R}^{n_x}$ denotes the set of feasible (safe) states, while $\mathbb{U} \subseteq \mathbb{R}^{n_u}$ is the set of feasible control inputs. The set $\mathbb{X}_f \subseteq \mathbb{X}$ is called the *terminal safe set* [19]. The terminal safe set is a *control invariant set*, which means that:

Definition 1. *(Control invariant set). The set $\mathbb{X}_f$ is a control invariant set if and only if, for all $\boldsymbol{x}|(\boldsymbol{x} \in \mathbb{X}_f)$, there exists a control law $\boldsymbol{u} = \boldsymbol{k}(\boldsymbol{x}) \subseteq \mathbb{U}$ such that $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{k}(\boldsymbol{x})) \in \mathbb{X}_f$.*

The terminal safety constraint $\boldsymbol{x}_{N|k} \in \mathbb{X}_f$ thus guarantees that the system will be able to stay within the terminal safe set $\mathbb{X}_f$ (and by extension the feasible set $\mathbb{X}$) for all time $t' \in (T_f, \infty)$. The size of the terminal set, as well as the length of the prediction horizon, are important factors in determining the conservatism of the predictive safety filter. With a large terminal set and a longer prediction horizon, the PSF has a higher degree of flexibility when computing trajectories, as the terminal set can more easily be reached within the horizon. Figure 2.2

visualizes how the PSF modifies unsafe trajectories, and highlights the impact of an increased prediction horizon on trajectory modification flexibility.

After solving the OCP, the control action that is applied to the system is chosen as $\boldsymbol{u}_0^*(\boldsymbol{x}(k), \boldsymbol{u}_L(k)) = \boldsymbol{u}_{0|k}^*$. The PSF algorithm is based on the classic model predictive control (MPC) algorithm [20]. The main difference is that while classical MPC minimizes the OCP with respect to a reference *trajectory*, the PSF only minimizes with respect to a reference *control input*.



(a) PSF trajectory modification with N = 2 (b) PSF trajectory modification with N = 3

Figure 2.2: Illustration of how the predictive safety filter modifies the nominal trajectory based on safe set $\mathbb{X}$, terminal set $\mathbb{X}_f$, and number of shooting nodes N. The nominal unsafe path in red has two possible modifications to become safe, where the dark green path is proposed by the PSF. In the left figure the yellow path is closer to the nominal path, but given the short prediction horizon, the PSF must take another path more directly towards the terminal set. In the right figure, with 1 additional shooting node, the PSF can compute a trajectory which lies closer to the nominal path, and still be able to reach the terminal set.

2.5 Formulation of predictive safety filter OCP for 3-DOF ASV model

Recall the definition of the PSF OCP:

$$\begin{aligned} \min_{\boldsymbol{u}_{i|k}, \boldsymbol{x}_{i|k}} \quad & ||\boldsymbol{u}_{0|k} - \boldsymbol{u}_L(k)||_W^2 \\ s.t. \quad (1) \quad & \boldsymbol{x}_{0|k} = \boldsymbol{x}(k) \\ (2) \quad & \boldsymbol{x}_{i+1|k} = \boldsymbol{f}(\boldsymbol{x}_{i|k}, \boldsymbol{u}_{i|k}, \Delta T) \quad \forall i \in [0, N-1] \\ (3) \quad & \boldsymbol{x}_{i|k} \in \mathbb{X} \quad \forall i \in [0, N] \\ (4) \quad & \boldsymbol{u}_{i|k} \in \mathbb{U} \quad \forall i \in [0, N-1] \\ (5) \quad & \boldsymbol{x}_{N|k} \in \mathbb{X}_f \end{aligned} \tag{2.24 revisited}$$

We define the discrete-time system equations as:

$$\begin{aligned} \boldsymbol{\eta}_{k+1} &= \hat{\boldsymbol{f}}_{kinematic}(\psi_{i|k}, \boldsymbol{\nu}_{i|k}, \Delta T) \\ \boldsymbol{\nu}_{k+1} &= \hat{\boldsymbol{f}}_{kinetic}(\boldsymbol{\nu}_{i|k}, \boldsymbol{u}_{i|k}, \hat{\boldsymbol{\tau}}_{d,k}, \Delta T) \end{aligned} \tag{2.25}$$

where the disturbance forces $\boldsymbol{\tau}_{d,k}$ have been replaced by the generalized environmental disturbance estimate $\hat{\boldsymbol{\tau}}_{d,k}$ obtained from the disturbance observer. Inserting $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{\eta} & \boldsymbol{\nu} \end{bmatrix}^T$ and using equation (2.25), we substitute constraints (1) and (2) with:

$$\begin{aligned} \boldsymbol{\eta}_{0|k} &= \boldsymbol{\eta}(k) \\ \boldsymbol{\nu}_{0|k} &= \boldsymbol{\nu}(k) \\ \boldsymbol{\eta}_{i+1|k} &= \hat{\boldsymbol{f}}_{kinematic}(\psi_{i|k}, \boldsymbol{\nu}_{i|k}, \Delta T) \quad \forall i \in [0, N-1] \\ \boldsymbol{\nu}_{i+1|k} &= \hat{\boldsymbol{f}}_{kinetic}(\boldsymbol{\nu}_{i|k}, \boldsymbol{u}_{i|k}, \hat{\boldsymbol{\tau}}_{d,k}, \Delta T) \quad \forall i \in [0, N-1] \end{aligned} \tag{2.26}$$

The state constraint $\boldsymbol{x}_{i|k} \in \mathbb{X}$ is deemed equivalent to the following:

1. All ship velocities are within specified upper and lower bounds: $\boldsymbol{\nu}_{lb} \leq \boldsymbol{\nu}_{i|k} \leq \boldsymbol{\nu}_{ub}$
2. The position of the vessel is a safe distance away from any observed obstacles: $d(\boldsymbol{p}_{i|k}, \mathbb{O}_{i|k}) \geq d_{safe}$

where $d(\cdot)$ is the Euclidean distance function, $\boldsymbol{p}_{i|k} = \begin{bmatrix} x & y \end{bmatrix}^T$ is the NED coordinate position of the vessel at prediction step i, time step k, $\mathbb{O} \subseteq \mathbb{R}^2$ is the union of all observed obstacles at prediction step i, time step k, and d_{safe} is the minimum safety distance. This can be stated formally as:

$$\begin{aligned} &\boldsymbol{x}_{i|k} \in \mathbb{X} \rightarrow \boldsymbol{\eta}_{i|k} \in \mathbb{X}_\eta \cap \boldsymbol{\nu}_{i|k} \in \mathbb{X}_\nu \\ &\boldsymbol{\eta}_{i|k} \in \mathbb{X}_\eta \leftrightarrow d(\boldsymbol{p}_{i|k}, \mathbb{O}_{i|k}) \geq d_{safe} \\ &\boldsymbol{\nu}_{i|k} \in \mathbb{X}_\nu \leftrightarrow \boldsymbol{\nu}_{lb} \leq \boldsymbol{\nu}_{i|k} \leq \boldsymbol{\nu}_{ub} \end{aligned} \tag{2.27}$$

The collision avoidance methods used in this work will be described in chapter 3. The set $\mathbb{U}$ is defined as:

$$\boldsymbol{u}_{i|k} \in \mathbb{U} \leftrightarrow \boldsymbol{u}_{lb} \leq \boldsymbol{u}_{i|k} \leq \boldsymbol{u}_{ub} \tag{2.28}$$

2.5.1 Control invariant terminal set formulation

The formulation of the terminal safe set $\mathbb{X}_f$ follows generally the procedure used in [40]. The main difference is that while [40] formulates the terminal safe set with respect to lateral track error and track relative heading error, we have chosen to formulate $\mathbb{X}_f$ only with respect to the velocities $\boldsymbol{\nu}$, which ensures asymptotic safety while retaining maximal flexibility with respect to the proposed actions of the RL-agent.

Assumption 1. *(Control invariant ellipsoidal set) Let the ellipsoidal set $\mathbb{X}_e \subseteq \mathbb{X}$ be defined by $\mathbb{X}_e := \{\boldsymbol{x} | \boldsymbol{x}^T \boldsymbol{P} \boldsymbol{x} \leq 1\}$ where $\boldsymbol{P}$ is a positive definite matrix. Assume that there exists a control law $\boldsymbol{k}(\boldsymbol{x}) \in \mathbb{U} | (\boldsymbol{x} \in \mathbb{X}_e)$ and $\boldsymbol{P}$ such that $\boldsymbol{x}_{i|k} \in \mathbb{X}_e \rightarrow \boldsymbol{f}(\boldsymbol{x}_{i|k}, \boldsymbol{k}(\boldsymbol{x}_{i|k})) \in \mathbb{X}_e$.*

Assumption 1 implies the existence of a control law $\boldsymbol{k}(\boldsymbol{x})$ and positive definite matrix $\boldsymbol{P}$ which guarantees that the state $\boldsymbol{x}$ will remain in $\mathbb{X}_e \subseteq \mathbb{X}$ for $t \in (t', \infty)$ as long as $\boldsymbol{x}(t') \in \mathbb{X}_e$. Let us now define the terminal state of the system as $\boldsymbol{x}_f = \begin{bmatrix} \boldsymbol{\eta}_f & \boldsymbol{\nu}_f \end{bmatrix}^T$. We now define the *terminal feasible set* as:

$$\mathbb{C}_f := \{\boldsymbol{x}_f | \quad ||\boldsymbol{p}_f - \boldsymbol{p}_{N|k}||_2 \leq d_f \ \cap \ \boldsymbol{\nu}_{lb} \leq \boldsymbol{\nu}_f \leq \boldsymbol{\nu}_{ub}\} \tag{2.29}$$

Assume now that the control law $\boldsymbol{k}_f(\boldsymbol{x}_f)$ and positive definite matrix $\boldsymbol{P}_f^{6\times 6}$ satisfy assumption 1 for the state $\boldsymbol{x}_f$ and the feasible set $\mathbb{C}_f$, which yields the terminal constraint:

$$\boldsymbol{x}_f^T \boldsymbol{P}_f \boldsymbol{x}_f \leq 1 \tag{2.30}$$

By adding the terminal collision avoidance constraint:

$$d(\boldsymbol{p}_{N|k}, \mathbb{O}_{N|k}) \geq d_{safe} + d_f \tag{2.31}$$

and inserting $\boldsymbol{x}_f = \boldsymbol{x}_{N|k}$, asymptotic safety with respect to obstacles is guaranteed by:

$$\begin{aligned} &\begin{bmatrix} -d(\boldsymbol{p}_{N|k}, \mathbb{O}_{N|k}) \\ \boldsymbol{x}_{N|k}^T \boldsymbol{P}_f \boldsymbol{x}_{N|k} \end{bmatrix} \leq \begin{bmatrix} -(d_{safe} + d_f) \\ 1 \end{bmatrix} \\ \rightarrow \quad &\begin{bmatrix} -d(\boldsymbol{p}_{N|k}, \mathbb{O}_{N|k}) \\ \begin{bmatrix} \boldsymbol{0}^{1\times 3} & \boldsymbol{\nu}_{N|k}^T \end{bmatrix} \boldsymbol{P}_f \begin{bmatrix} \boldsymbol{0}^{3\times 1} \\ \boldsymbol{\nu}_{N|k} \end{bmatrix} \end{bmatrix} \leq \begin{bmatrix} -(d_{safe} + d_f) \\ 1 \end{bmatrix} \\ \rightarrow \quad &\begin{bmatrix} -d(\boldsymbol{p}_{N|k}, \mathbb{O}_{N|k}) \\ \boldsymbol{\nu}_{N|k}^T \boldsymbol{P}_{f_\nu} \boldsymbol{\nu}_{N|k} \end{bmatrix} \leq \begin{bmatrix} -(d_{safe} + d_f) \\ 1 \end{bmatrix} \end{aligned} \tag{2.32}$$

where:

$$\boldsymbol{P}_{f_\nu} = \begin{bmatrix} p_{44} & p_{45} & p_{46} \\ p_{54} & p_{55} & p_{56} \\ p_{64} & p_{65} & p_{66} \end{bmatrix} \tag{2.33}$$

Equation (2.32) ensures, given the terminal velocities $\boldsymbol{\nu}_{N|k}$ and the control law $\boldsymbol{k}_f(\boldsymbol{x}_f)$, that the position of the ship beyond the prediction horizon ($\boldsymbol{p}_\infty$) will

deviate no more than a distance d_f from the terminal position $\boldsymbol{p}_{N|k}$. From this, we can conclude that:

$$\begin{aligned} ||\boldsymbol{p}_\infty - \boldsymbol{p}_{N|k}||_2 \leq d_f \quad &\cap \quad d(\boldsymbol{p}_{N|k}, \mathbb{O}_{N|k}) \geq d_{safe} + d_f \\ &\rightarrow \quad d(\boldsymbol{p}_\infty, \mathbb{O}_{N|k}) \geq d_{safe} \end{aligned} \tag{2.34}$$

The computation of the terminal control law $\boldsymbol{k}_f(\boldsymbol{x}_f)$ and matrix $\boldsymbol{P}_f$ will be detailed in chapter 3. Figure 2.3 illustrates a simple scenario in which the PSF modifies the ship trajectory because of the terminal safety constraint. The red arrows indicate the nominal path of the ship. While the nominal predicted position $\boldsymbol{p}_N$ satisfies the distance requirement d_{safe}, the point $\boldsymbol{p}_\infty(\boldsymbol{k}(\boldsymbol{x}))$ indicates that given the pose and velocity at $\boldsymbol{p}_N$, it is not possible to avoid a future safety violation given the terminal control law $\boldsymbol{k}(\boldsymbol{x})$, implying that terminal set constraint is violated. Therefore the PSF modifies the control input so that the optimal (predicted) safe path is taken instead, indicated by the green arrows.

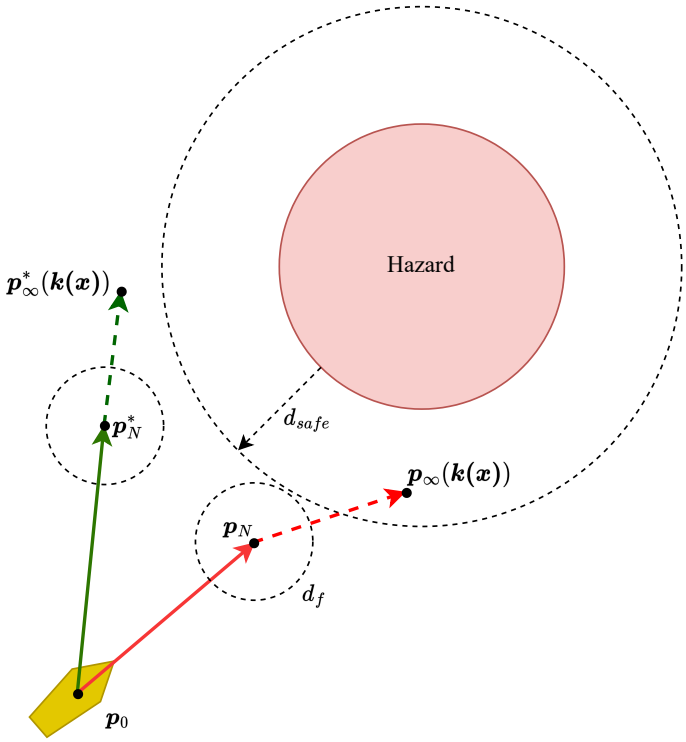


Figure 2.3: Visualization of ship trajectory modification caused by terminal safety constraint, with N = 1 for clarity. Red arrows indicate nominal (unsafe) trajectory, while green arrows indicate PSF modified trajectory.

The full PSF optimal control problem formulation for the 3-DOF ship model thus becomes:

$$
\begin{aligned}
&\min_{\boldsymbol{u}_{i|k},\boldsymbol{\eta}_{i|k},\boldsymbol{\nu}_{i|k}} ||\boldsymbol{u}_{0|k} - \boldsymbol{u}_L(k)||^2_W \\
&s.t. \quad (1) \quad \boldsymbol{\eta}_{0|k} = \boldsymbol{\eta}(k) \\
&\qquad (2) \quad \boldsymbol{\nu}_{0|k} = \boldsymbol{\nu}(k) \\
&\qquad (3) \quad \boldsymbol{\eta}_{i+1|k} = \hat{\boldsymbol{f}}_{kinematic}(\psi_{i|k}, \boldsymbol{\nu}_{i|k}, \Delta T) \quad \forall i \in [0, N-1] \\
&\qquad (4) \quad \boldsymbol{\nu}_{i+1|k} = \hat{\boldsymbol{f}}_{kinetic}(\boldsymbol{\nu}_{i|k}, \boldsymbol{u}_{i|k}, \hat{\boldsymbol{\tau}}_{d,k}, \Delta T) \quad \forall i \in [0, N-1] \\
&\qquad (5) \quad d(\boldsymbol{p}_{i|k}, \mathbb{O}_{i|k}) \geq d_{safe} \quad \forall i \in [0, N-1] \\
&\qquad (6) \quad d(\boldsymbol{p}_{N|k}, \mathbb{O}_{N|k}) \geq d_{safe} + d_f \\
&\qquad (7) \quad \boldsymbol{\nu}_{lb} \leq \boldsymbol{\nu}_{i|k} \leq \boldsymbol{\nu}_{ub} \quad \forall i \in [0, N] \\
&\qquad (8) \quad \boldsymbol{u}_{lb} \leq \boldsymbol{u}_{i|k} \leq \boldsymbol{u}_{ub} \quad \forall i \in [0, N-1] \\
&\qquad (9) \quad \boldsymbol{\nu}^T_{N|k} \boldsymbol{P}_{f_{\boldsymbol{\nu}}} \boldsymbol{\nu}_{N|k} \leq 1
\end{aligned}
\tag{2.35}
$$

3 Method

This chapter mainly covers the design and implementation of the reinforcement learning algorithm, predictive safety filter, nonlinear disturbance observer, and the test setup. In addition, the chosen simulation environment is presented, chiefly focusing on the most relevant features with regard to this work. The hardware and the various software components which we use are described, and hyperparameters are listed and explained. All code used in the simulation experiments can be found at: `https://github.com/sveinjhu/gym-auv-safety-filter`.

3.1 RL/PSF overview

An overview of the RL/PSF control architecture is shown in 3.1. Based on the current reward and observation, the RL agent proposes a control action. The proposed action u_L, along with the system state is passed to the predictive safety filter, which computes the minimally modified safe control action u_0. Then, u_0 is passed as the control input for the next iteration of the model simulation. The difference between the proposed action u_L and the modified action u_0, δ_u, is propagated to the reward function, along with the observation vector.

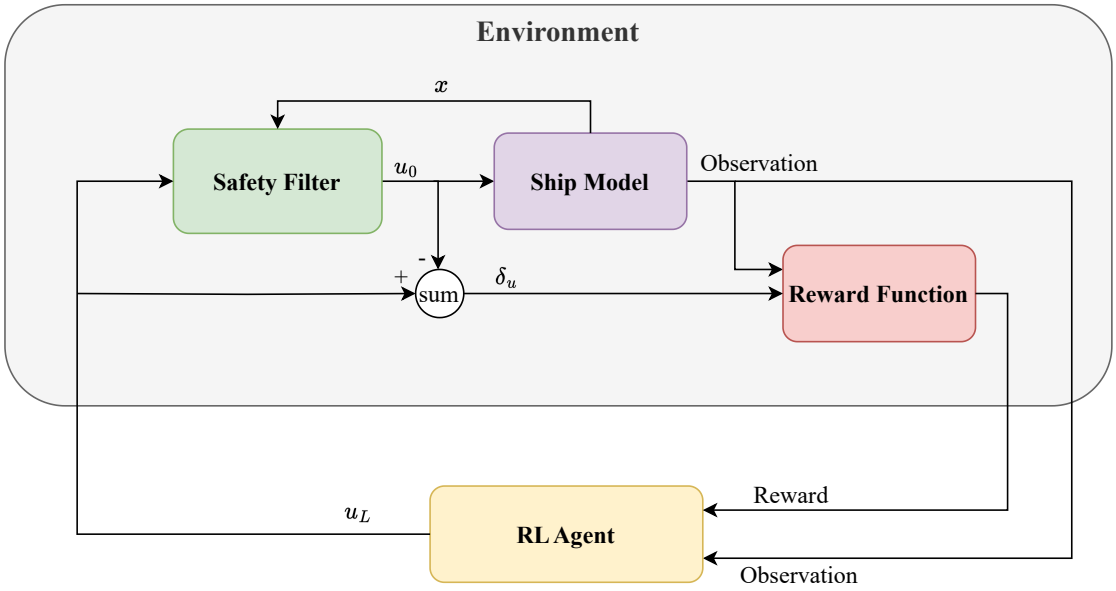


Figure 3.1: Illustration the RL + PSF control design. Note that the LiDAR perception features, environmental disturbances and obstacles have been omitted from this figure for clarity.

3.2 Training Environment

The **gym-auv** simulation framework by Meyer [25] was used for training and testing the RL agents. The framework is based on OpenAI Gym [41], a widely used toolkit for developing and comparing RL algorithms. The RL agent is trained by using Stable-Baselines3's PPO implementation [42], with hyperparameters identical to those applied in Larsen et al. [43].

3.2.1 Integrated LiDAR sensor suite

gym-auv features an integrated 2D LiDAR sensor suite which is used to detect potential hazards in the vicinity of the ownership. The simulated 2D LiDAR sensor consists of N_{ray} evenly spaced detection rays, each measuring the closest distance to an object along the direction of the ray, within the maximum detection distance R_{detect}. The rays are divided into N_{sector} non-overlapping sectors, each sector containing $\frac{N_{ray}}{N_{sector}}$ detection rays. A visualization of the LiDAR detection rays is shown in figure 3.2

Table 3.1: LiDAR sensor parameters

Parameter	Description	Value
N_{ray}	Number of detection rays	180
N_{sector}	Number of detection sectors	20
R_{detect}	Maximum detection distance	150 (m)

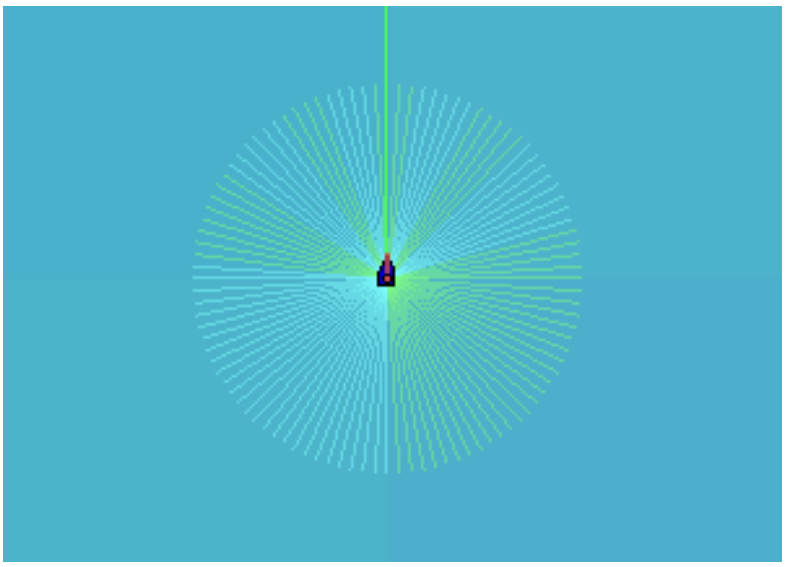


Figure 3.2: Visualization of LiDAR detection rays with $N_{rays} = 100$ and $R_{detect} = 50$. Green lines indicate the direction and range of the rays.

3.2.2 Observation vector

The observation vector used in training the agent consists of two main categories of features: *navigation* and *perception*. The navigation features provide information about the vessel's state relative to the path, in order to perform successful path-following. The perception features utilize LiDAR distance measurements to provide information about the collision risk, facilitating obstacle avoidance. In the following, the process of obtaining the features is explained in more detail.

Navigation features. Table 3.2 gives an overview of the six navigation features and their definition. The first three provide information about the vessel state, while the last three are related to the path and require additional elaboration. Figure 3.3 illustrates the path following concepts used to find the cross-track error, heading error, and look-ahead heading error. The parameterized path can be represented as $\boldsymbol{p}_d(\omega) = [x_d(\omega), y_d(\omega)]^T$, where $x_d(\omega)$ and $y_d(\omega)$ are given in the NED-frame. The cross-track error, ϵ, is defined as the smallest Euclidean distance from the path to the vessel's position, which on the figure is the distance between the ship and $\boldsymbol{p}_d(\bar{\omega})$. The look-ahead point $\mathrm{p}_d\left(\bar{\omega} + \Delta_{LA}\right)$ is the point that lies Δ_{LA} further a long the path from $\boldsymbol{p}_d(\bar{\omega})$, where Δ_{LA} is the constant look-ahead distance. Then, the heading error $\tilde{\psi}$ could be defined as the change in heading needed for the vessel to navigate straight towards the look-ahead point from the current position. Lastly, the look-ahead heading error $\tilde{\psi}_{LA}$ takes the path direction at the look-ahead point, $\gamma_p\left(\bar{\omega} + \Delta_{LA}\right)$, into account to ensure a smoother vessel trajectory. It is defined as the difference between the heading ω and $\gamma_p\left(\bar{\omega} + \Delta_{LA}\right)$. Based on results from previous work by Meyer [25], it is shown that these features provide the necessary information for the agent to intelligently follow a path.

Table 3.2: Navigation features provided to the RL agent

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

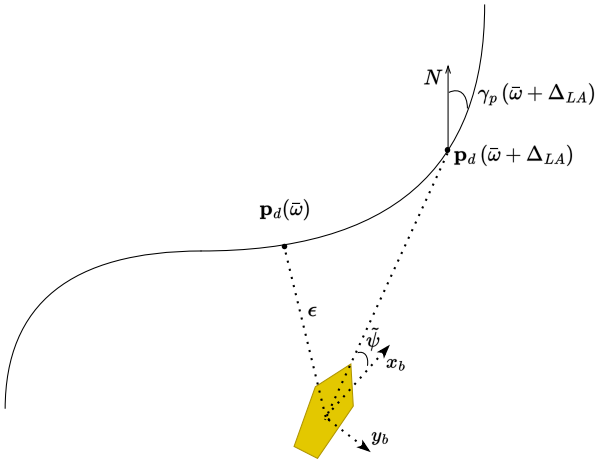


Figure 3.3: Illustration of path following concepts. The RL agent uses the cross-track error ϵ, heading error $\tilde{\psi}$ and look-ahead heading error $\tilde{\psi}_{LA}$ (which is derived from path direction at the look-ahead point $\gamma_p(\bar{\omega}+\Delta_{LA})$) as navigation features.

Perception features. The perception features are based of LiDAR distance measurements from the simulated LiDAR sensor suite. Based on the paper by Larsen et al. [44], the LiDAR measurements are encoded by a pre-trained convolutional neural network (CNN). The CNN is used to predict the CRIs corresponding to the measurements (see subsection 2.1.3 for an explanation of the CRI calculation). Multiple CRIs can exist for each LiDAR scan since each is associated with a distinct nearby obstacle. Based on 180 closeness measurements, the maximum of 12 CRIs are predicted and used as perception features for the RL agent. Further details on the architecture and training of the CNN are found in [44]. Finally, the diagram in Figure 3.4 summarizes how the observation vector is processed through the RL agent and the PSF.

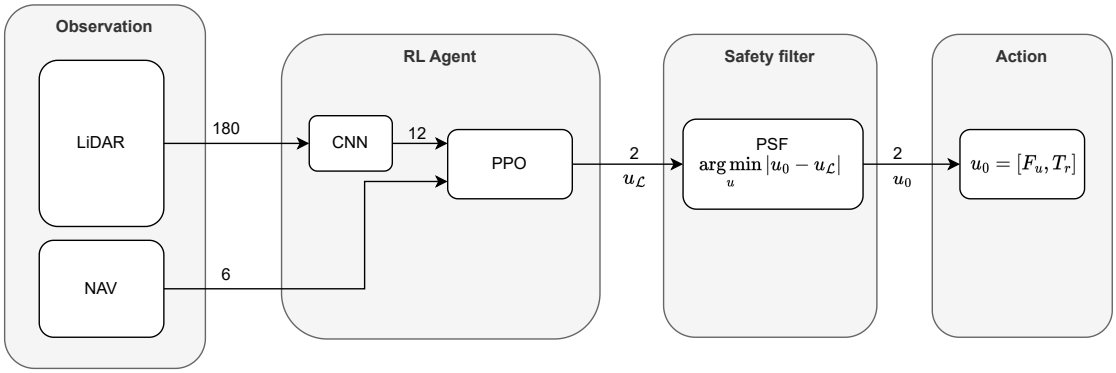


Figure 3.4: RL agent diagram. The observation vector contains both LiDAR and navigation features. While the navigation features are used directly in the PPO algorithm, the LiDAR measurements are processed through a CNN. The PPO outputs an action $u_{\mathcal{L}}$ that is sent through the safety filter. Finally, the safe action u_0 can be executed in the environment.

3.3 Ship model parameters

The model parameters used for the 3-DOF Cybership II model were obtained from [26], and are presented in table 3.3:

Table 3.3: 3-DOF Cybership II model parameters

Parameter	Value	Unit	Parameter	Value	Unit
m	23.8	kg	$Y_{\|r\|r}$	-0.02	$\frac{kg}{m}$
x_g	0.046	m	$Y_{\|v\|r}$	-0.01	$\frac{kg}{m}$
I_z	1.760	$kg \cdot m^2$	$Y_{\|r\|v}$	-0.01	$\frac{kg}{m}$
X_u	-0.7225	$\frac{N \cdot s}{m}$	N_v	0.1052	$\frac{kg \cdot s}{m}$
$X_{\dot{u}}$	-2.0	kg	$N_{\dot{v}}$	0.0	kg
$X_{\|u\|u}$	-1.3274	$\frac{kg}{m}$	$N_{\|v\|v}$	5.0437	$\frac{kg}{m}$
X_{uuu}	-5.8664	$\frac{kg \cdot s}{m^2}$	N_r	-0.5	$\frac{kg \cdot s}{m}$
Y_v	-0.8612	$\frac{N \cdot s}{m}$	$N_{\dot{r}}$	-1.0	kg
$Y_{\dot{v}}$	-10.0	kg	$N_{\|r\|r}$	0.005	$\frac{kg}{m}$
$Y_{\|v\|v}$	-36.2823	$\frac{kg}{m}$	$N_{\|v\|r}$	-0.001	$\frac{kg}{m}$
Y_r	0.1079	$\frac{kg \cdot s}{m}$	$N_{\|r\|v}$	–0.001	$\frac{kg}{m}$
$Y_{\dot{r}}$	0.0	kg			

3.4 Disturbance modelling

The sea current velocity V_c and angle β_c are modeled as slow-varying constrained random walk processes:

$$\begin{aligned} \dot{V}_c &= W_{V_c}, \quad s.t.\ |V_c| \leq V_{c,max}\ ,\ |W_{V_c}| \leq W_{V_c,max} \\ \dot{\beta}_c &= W_{\beta_c}, \quad s.t.\ |\beta_c| \leq \beta_{c,max}\ ,\ |W_{\beta_c}| \leq W_{\beta_c,max} \end{aligned} \tag{3.1}$$

The generalized force disturbances are modeled similarly, with an additional white noise component added to the slowly varying signal:

$$\begin{aligned} \boldsymbol{\tau}_d &= \boldsymbol{\delta}_d + \boldsymbol{W}_{\boldsymbol{\tau}1}, \quad s.t. \ |\boldsymbol{\tau}_d| \leq \boldsymbol{\tau}_{d,max} \ , \ |\boldsymbol{W}_{\boldsymbol{\tau}1}| \leq \boldsymbol{W}_{\boldsymbol{\tau}1,max} \\ \dot{\boldsymbol{\delta}}_d &= \boldsymbol{W}_{\boldsymbol{\tau}2}, \quad s.t. \ |\boldsymbol{W}_{\boldsymbol{\tau}2}| \leq \boldsymbol{W}_{\boldsymbol{\tau}2,max} \end{aligned} \tag{3.2}$$

To ensure a reasonable degree of controllability in the face of disturbances, the maximum current velocity $V_{c,max}$ is set to $\sim 20\%$ of the maximum surge speed u_{max}. Similarly, the maximum surge and sway force disturbances are set to $\sim 20\%$ of the maximum applied surge force $F_{u,max}$, while the maximum yaw moment disturbance is $\sim 10\%$ of $T_{r,max}$.

3.5 Disturbance observer implementation

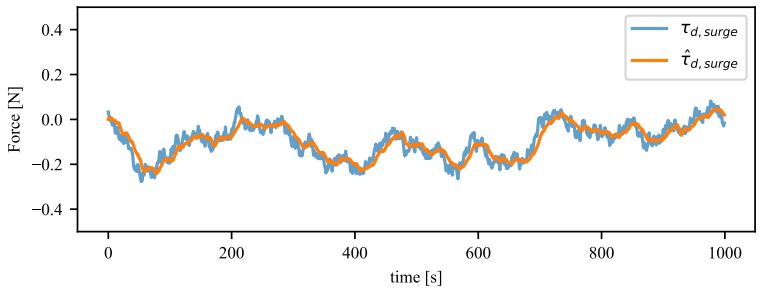
Table 3.4: Environmental disturbance observer parameters

Parameter	Description	Value
Γ_1	Adaptation gain (surge force disturbance)	0.1
Γ_2	Adaptation gain (sway force disturbance)	0.1
Γ_3	Adaptation gain (yaw moment disturbance)	0.08

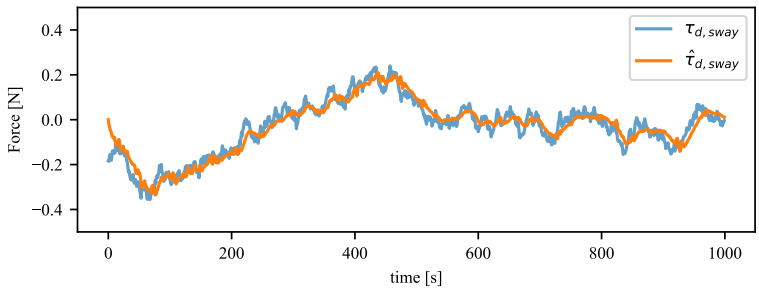
The observer system (2.23) is implemented using a simple forward-Euler scheme, with the parameter values in table 3.4:

$$\begin{aligned} \hat{\boldsymbol{\tau}}_{d,k} &= \boldsymbol{\zeta}_k + \boldsymbol{T}\boldsymbol{\nu}_k \\ \boldsymbol{\zeta}_{k+1} &= \boldsymbol{\zeta}_k - \boldsymbol{T}\boldsymbol{M}^{-1}(\boldsymbol{D}(\boldsymbol{\nu}_k)\boldsymbol{\nu}_k - \boldsymbol{C}(\boldsymbol{\nu}_k)\boldsymbol{\nu}_k + \boldsymbol{B}\boldsymbol{u}_k + \hat{\boldsymbol{\tau}}_{d,k})\Delta T \end{aligned} \tag{3.3}$$

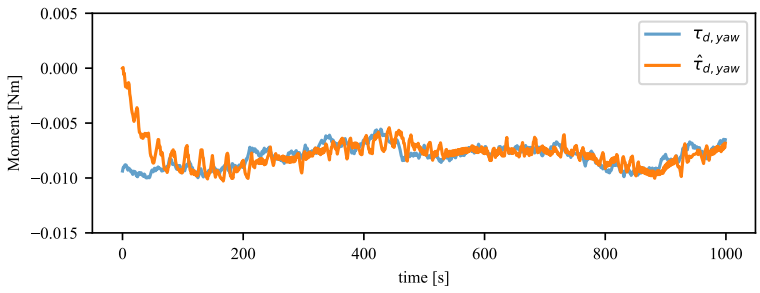
The adaptation gains are chosen relatively small to ensure stability and sufficiently smooth disturbance estimates. Figure 3.5 shows the typical performance of the estimator given randomized environmental disturbance forces generated according to equation (3.2).



(a) Estimate of surge force disturbance $\boldsymbol{\tau}_{d,surge}$



(b) Estimate of sway force disturbance $\boldsymbol{\tau}_{d,sway}$



(c) Estimate of yaw moment disturbance $\boldsymbol{\tau}_{d,yaw}$

Figure 3.5: Estimates of randomly generated environmental disturbances using adaption gains from table 3.4

3.6 Predictive safety filter implementation

3.6.1 Collision Avoidance

In the 2-dimensional **gym-auv** simulation environment, static obstacles are represented as circles with a given position and radius, while other ships are represented as polygonal objects following pre-defined paths. In this work, we assume that the predictive safety filter has access to the position, shape and velocity of ships that must be avoided. In addition, the integrated LiDAR sensor suite is used for static obstacle collision avoidance.

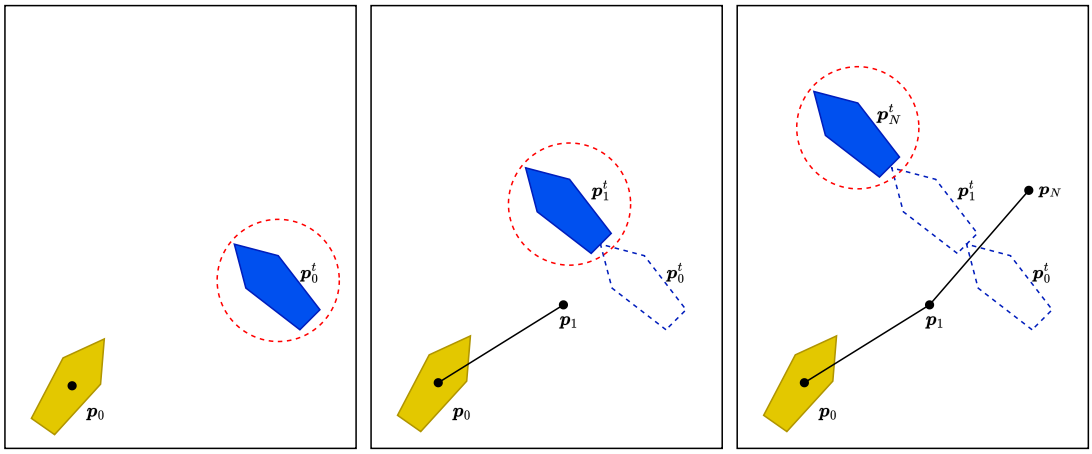
Dynamic obstacle collision avoidance

Each ship is given a circular hazard region centered at the midpoint of the ship, with radius equal to the length of the ship. Denoting the position and velocity of

target ship j at the current iterate as $\boldsymbol{p}_j^t$ and $\boldsymbol{v}_j^t$ respectively, the dynamic obstacle collision avoidance constraint is implemented as:

$$||\boldsymbol{p}_{i|k} - (\boldsymbol{p}_j^t + \boldsymbol{v}_j^t \cdot i \cdot \Delta T)||_2 \geq l_j + d_{safe} \quad \forall\, i \in [0, N],\ j \in [0, N_j] \tag{3.4}$$

where ΔT is the step length, l_j is the length of ship j and N_j is the number of ship obstacles. This formulation assumes that all ship obstacles move with constant velocities over the entire prediction horizon, which does not necessarily match the true trajectories of the obstacles. However, results will show that satisfactory performance is achieved by choosing an appropriately large safety distance d_{safe}. Figure 3.6 visualizes the dynamic obstacle collision avoidance mechanism.



(a) Shooting node 0 (b) Shooting node 1 (c) Terminal node

Figure 3.6: Dynamic obstacle collision avoidance. Blue object indicates the target ship, red dotted circle indicates the approximation of the area covered by the target ship. For each prediction step, the PSF predicts the movement of the target based on the initial velocity, and individual collision avoidance constraints for each shooting node are computed accordingly.

LiDAR-based collision avoidance

To encode collision avoidance constraints from the LiDAR measurements, the $N_{col} \leq \frac{N_{ray}}{N_{sector}}$ closest obstacle detections within each sector are extracted. For each of the extracted ray measurements, the coordinates of the point of detection $\boldsymbol{p}_{detect}$ are calculated by:

$$\boldsymbol{p}_{detect} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d \cdot \cos(\theta + \psi) \\ d \cdot \sin(\theta + \psi) \end{bmatrix} \tag{3.5}$$

where d is the distance to the object measured by the ray, and θ is the angle offset of the ray w.r.t. the heading of the ship. The LiDAR collision avoidance constraint is then defined as:

$$||\boldsymbol{p}_{i|k} - \boldsymbol{p}_{detect,m}||_2 \geq R_{avoid} + d_{safe} \quad \forall\, i \in [0, N],\ m \in [0, N_{col}N_{sector}] \tag{3.6}$$

where R_{avoid} defines the radius of avoidance around each detected point $\boldsymbol{p}_{detect}$, and $N_{col}N_{sector}$ is the total number of included ray detections. The formulation above effectively makes it so that the boundaries of nearby obstacles are approximated as a collection of circles centered at the detection points, with radius equal to R_{avoid}. Safety is thus ensured by choosing a sufficiently high value for R_{avoid}, and having a sufficiently high number N_{col} detected points for each LiDAR sector. Figure 3.7 shows how the LiDAR measurements are used to generate static obstacles.

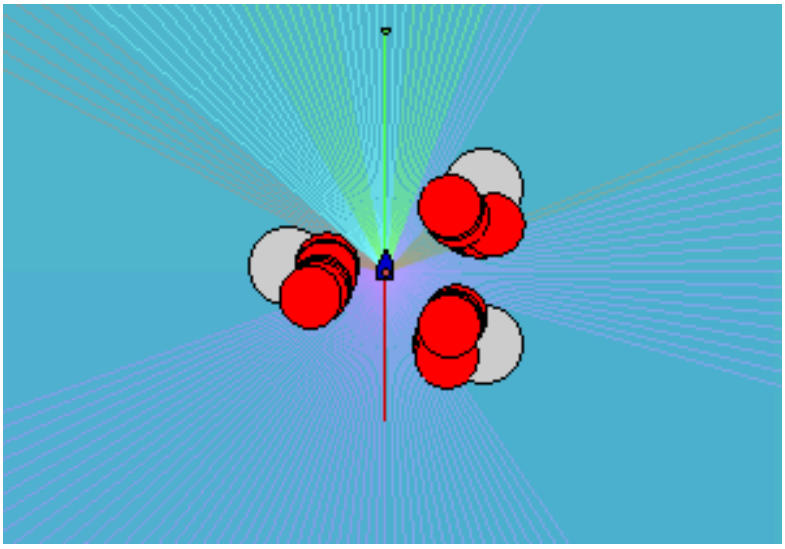


Figure 3.7: Visualization of obstacles constructed for PSF using LiDAR detection. Grey circles indicate obstacles, while red circles indicate avoidance zones computed from the points at which the LiDAR detection rays intersect the obstacles.

3.6.2 Terminal constraint computation

The computation of the matrix $\boldsymbol{P}_f$ which defines the terminal ellipsoidal invariant set and the corresponding control law $\boldsymbol{k}_f(\boldsymbol{x})$ follows the same procedure as used in [19] and [45]. The main idea is to linearize the system and constraints w.r.t. an equilibrium state, which allows us to construct a semi-definite program (SDP) that simultaneously optimizes the matrix $\boldsymbol{P}_f$ and terminal control law $\boldsymbol{k}_f(\boldsymbol{x})$. Hence, we need to transform the system to the linear form:

$$\begin{aligned} \dot{\bar{\boldsymbol{x}}} &= \boldsymbol{A}\bar{\boldsymbol{x}} + \boldsymbol{B}\bar{\boldsymbol{u}} \\ s.t. \quad \boldsymbol{H}\bar{\boldsymbol{x}} &\leq \boldsymbol{h} \\ \boldsymbol{G}\bar{\boldsymbol{u}} &\leq \boldsymbol{g} \end{aligned} \tag{3.7}$$

Where $\bar{\boldsymbol{x}} = \boldsymbol{x} - \boldsymbol{x}_{eq}$ and $\bar{\boldsymbol{u}} = \boldsymbol{u} - \boldsymbol{u}_{eq}$ are the states and inputs shifted with respect to the linearization (equilibrium) points, and $\boldsymbol{A}$ and $\boldsymbol{B}$ are the state and input matrices for the linearized system. The pairs $\{\boldsymbol{H}^{n_h \times n_x}, \boldsymbol{h}^{n_h \times 1}\}$ and $\{\boldsymbol{G}^{n_g \times n_u}, \boldsymbol{g}^{n_g \times 1}\}$ encode the state and input constraints as *linear polytopic constraints*, which means that $\boldsymbol{H}$ and $\boldsymbol{G}$ are real-valued, column independent matrices, while $\boldsymbol{h}$ and $\boldsymbol{g}$ are

real valued, strictly positive column vectors. n_h, n_x, n_g, and n_u are the number of state constraints, states, inputs, and input constraints, respectively.
We choose the equilibrium state equal to the maximum surge thrust and the corresponding constant maximum surge velocity, $\boldsymbol{x}_e = \begin{bmatrix} U_{max} & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$ and $\boldsymbol{u}_e = \begin{bmatrix} F_{u,max} & 0 \end{bmatrix}^T$. The maximum surge velocity is found by solving for the steady state of the decoupled surge equation:

$$\begin{aligned} 0 &= -D_u(U_{max})U_{max} + F_{u,max} \\ \rightarrow &- X_u u_{max} - X_{|u|u}|U_{max}|U_{max} - X_{uuu}U_{max}^3 = F_{u,max} \end{aligned} \tag{3.8}$$

The next step is to linearize the state-space system w.r.t. the equilibrium. To simplify the process, we omit the non-linear damping terms from the system model, because the absolute value function $|\cdot|$ (which appears in the non-linear damping) is non-differentiable at the origin. In practice, since the non-linear damping terms *increase* the inertia of the system, this simplification will not impact the validity of the following results, rather we would expect that using the simplified system will lead to a more conservative estimate of $\boldsymbol{P}_f$ than otherwise. The simplified system model is given by:

$$\begin{aligned} \tilde{\boldsymbol{x}} &= \begin{bmatrix} \tilde{\boldsymbol{\eta}} & \tilde{\boldsymbol{\nu}} \end{bmatrix} \\ \dot{\tilde{\boldsymbol{\eta}}} &= \boldsymbol{R}(\tilde{\psi})\tilde{\boldsymbol{\nu}} \\ \dot{\tilde{\boldsymbol{\nu}}} &= \boldsymbol{M}^{-1}(-\boldsymbol{C}(\tilde{\boldsymbol{\nu}})\tilde{\boldsymbol{\nu}} - \boldsymbol{D}_L\tilde{\boldsymbol{\nu}} + \boldsymbol{u}) \end{aligned} \tag{3.9}$$

where $\boldsymbol{D}_L$ is the linear damping:

$$\boldsymbol{D}_L = \begin{bmatrix} -X_u & 0 & 0 \\ 0 & -Y_v & -Y_r \\ 0 & -N_v & -N_r \end{bmatrix} \tag{3.10}$$

The linearized system is then computed as:

$$\dot{\bar{\boldsymbol{x}}} = \boldsymbol{A}\bar{\boldsymbol{x}} + \boldsymbol{B}\bar{\boldsymbol{u}} \tag{3.11}$$

where:

$$\begin{aligned} \bar{\boldsymbol{x}} &= \tilde{\boldsymbol{x}} - \boldsymbol{x}_e \\ \bar{\boldsymbol{u}} &= \boldsymbol{u} - \boldsymbol{u}_e \\ \boldsymbol{A} &= \begin{bmatrix} \frac{\delta\dot{\tilde{\boldsymbol{\eta}}}}{\delta\tilde{\boldsymbol{\eta}}^T}\big|_{\boldsymbol{x}_e} & \frac{\delta\dot{\tilde{\boldsymbol{\eta}}}}{\delta\tilde{\boldsymbol{\nu}}^T}\big|_{\boldsymbol{x}_e} \\ \boldsymbol{0}^{3\times 3} & \frac{\delta\dot{\tilde{\boldsymbol{\nu}}}}{\delta\tilde{\boldsymbol{\nu}}^T}\big|_{\boldsymbol{x}_e} \end{bmatrix} \\ \boldsymbol{B} &= \begin{bmatrix} \boldsymbol{0}^{3\times 2} \\ \frac{\delta\dot{\tilde{\boldsymbol{\nu}}}}{\delta\tilde{\boldsymbol{u}}^T}\big|_{\boldsymbol{x}_e,\boldsymbol{u}_e} \end{bmatrix} \end{aligned} \tag{3.12}$$

By defining the terminal control law $\boldsymbol{k}_f(\boldsymbol{x})$ as a linear feedback controller and applying it to the linearized system: $\boldsymbol{k}_f(\bar{\boldsymbol{x}}) = \boldsymbol{K}\bar{\boldsymbol{x}}$, the constraint $\bar{\boldsymbol{x}}^T\boldsymbol{P}_f\bar{\boldsymbol{x}} \leq 1$ can be rewritten using the closed-loop Lyapunov equation [19] defined by

$$(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K})^T\boldsymbol{P}_f(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}) \prec 0 \tag{3.13}$$

The final step before constructing the SDP is to transform the constraints to linear polytopic form, i.e.

$$\begin{aligned} \boldsymbol{H}\bar{\boldsymbol{x}} &\leq \boldsymbol{h} \\ \boldsymbol{G}\bar{\boldsymbol{u}} &\leq \boldsymbol{g} \end{aligned} \tag{3.14}$$

Recall the definition of the terminal feasible set:

$$\mathbb{C}_f := \{\boldsymbol{x}_f| \quad ||\boldsymbol{p}_f - \boldsymbol{p}_{N|k}||_2 \leq d_f \ \cap \ \boldsymbol{\nu}_{lb} \leq \boldsymbol{\nu}_f \leq \boldsymbol{\nu}_{ub}\} \tag{2.29 revisited}$$

without loss of generality, we assume $\boldsymbol{p}_{N|k} = 0$, which yields

$$\mathbb{C}_f := \{\boldsymbol{x}_f| \quad ||\boldsymbol{p}_f||_2 \leq d_f \ \cap \ \boldsymbol{\nu}_{lb} \leq \boldsymbol{\nu}_f \leq \boldsymbol{\nu}_{ub}\} \tag{3.15}$$

To satisfy the polytopic form requirement, we approximate the distance constraint $||\boldsymbol{p}_f||_2 \leq d_f$ by imposing that $\boldsymbol{p}_f$ must be inside the largest inscribed square of the circle with radius d_f.

$$\begin{bmatrix} |x_f| \\ |y_f| \end{bmatrix} \leq \begin{bmatrix} \frac{d_f}{\sqrt{2}} \\ \frac{d_f}{\sqrt{2}} \end{bmatrix} \rightarrow ||\boldsymbol{p}_f||_2 \leq d_f \tag{3.16}$$

The construction of the polytopic constraints is now trivial, as the control input constraints and the remaining state constraints are simple bound constraints.
The largest constrained ellipsoidal set $\{\bar{\boldsymbol{x}}| \ \bar{\boldsymbol{x}}^T\boldsymbol{P}_f\bar{\boldsymbol{x}} \leq 1\} \in \mathbb{C}_f$, can now be computed by solving the following SDP ([19], [45])

$$\begin{aligned} &\min_{\boldsymbol{E},\boldsymbol{Y}} - logdet(\boldsymbol{E}) \\ s.t. \quad &\boldsymbol{E} \succeq 0 \\ &\begin{bmatrix} ([\boldsymbol{h}]_i - [\boldsymbol{H}]_i\boldsymbol{x}_e)^2 & [\boldsymbol{H}]_i\boldsymbol{E} \\ \boldsymbol{E}[\boldsymbol{H}]_i^T & \boldsymbol{E} \end{bmatrix} \succeq 0 \quad \forall\, i \in [1, n_h] \\ &\begin{bmatrix} ([\boldsymbol{g}]_j - [\boldsymbol{G}]_j\boldsymbol{u}_e)^2 & [\boldsymbol{G}]_j\boldsymbol{E} \\ \boldsymbol{E}[\boldsymbol{G}]_j^T & \boldsymbol{E} \end{bmatrix} \succeq 0 \quad \forall\, j \in [1, n_g] \\ &\begin{bmatrix} \boldsymbol{E} & \boldsymbol{E}\boldsymbol{A}^T + \boldsymbol{Y}^T\boldsymbol{B}^T \\ \boldsymbol{A}\boldsymbol{E} + \boldsymbol{B}\boldsymbol{Y} & \boldsymbol{E} \end{bmatrix} \succeq 0 \end{aligned} \tag{3.17}$$

where $\boldsymbol{E} = \boldsymbol{P}_f^{-1}$ and $\boldsymbol{Y} = \boldsymbol{K}\boldsymbol{E}$. From the computed optimal $\boldsymbol{P}_f$ we can extract $\boldsymbol{P}_{f_\nu}$, which is inserted in the terminal velocity constraint (8) of (2.35). The semi-definite program in (3.17) was solved using the convex optimization python software package **CVXPY** [46].

3.6.3 Predictive safety filter parameters

From equation (2.35), we define:

$$\boldsymbol{u}_{lb} = \begin{bmatrix} F_{u,lb} \\ T_{r,lb} \end{bmatrix}, \quad \boldsymbol{u}_{ub} = \begin{bmatrix} F_{u,ub} \\ T_{r,ub} \end{bmatrix} \tag{3.18}$$

The cost matrix $\boldsymbol{W}$ is defined as:

$$\boldsymbol{W} = \begin{bmatrix} \frac{\gamma_{F_u}}{(F_{u,ub}-F_{u,lb})^2} & 0 \\ 0 & \frac{\gamma_{T_r}}{(T_{r,ub}-T_{r,lb})^2} \end{bmatrix} \tag{3.19}$$

where γ_{F_u} and γ_{T_r} are weighting constants, while the denominators ensure that the input signals are normalized according to their respective operating ranges. The parameters used in the implementation of the predictive safety filter is shown in table 3.5:

Table 3.5: Predictive safety filter parameters

Parameter	Description	Value
N	Number of shooting nodes	50
ΔT	Discretization step	0.5 (s)
$F_{u,lb}$	Minimum surge force	-0.2 (N)
$F_{u,ub}$	Maximum surge force	2 (N)
$T_{r,lb}$	Minimum yaw moment	-0.15 (Nm)
$T_{r,ub}$	Maximum yaw moment	0.15 (Nm)
N_{col}	Number of LiDAR COLAV detection points per sector	5
R_{avoid}	LiDAR COLAV detection point avoidance radius	8 (m)
d_{safe}	Minimum safe distance to hazards	5 (m)
γ_{F_u}	Surge force modification cost	1
γ_{T_r}	Yaw moment modification cost	0.01

With 50 shooting nodes and a discretization step of $0.5s$, the length of the prediction horizon is $T_f = 50 \cdot 0.5s = 25s$, which enables the safety filter to predict possible hazardous situations far in advance. Because $\gamma_{F_u} > \gamma_{T_r}$, the PSF is penalized less for applying modifications to the yaw moment (causing the ship to turn) as opposed to decreasing the surge force (causing the ship to slow down). As a consequence, the ship is more likely to steer away from potential hazards instead of slowing down to avoid them, which encourages forward progress.
The predictive safety filter is created with the **acados** nonlinear optimal control software [47], using a sequential-quadratic-programming real-time-iteration scheme (SQP-RTI), and the internal QP-solver HPIPM [48]. The model equations (3) and (4) in 2.35 are automatically discretized by **acados** using an implicit Runge-Kutta order 4 (IRK4) scheme. State constraints and collision avoidance constraints are implemented as soft constraints to guarantee feasibility. Using the described implementation and software, running on a laptop with an AMD Ryzen 7 4700U processor, the average computation time of a PSF OCP iteration is around 5 milliseconds, which is sufficiently fast for real-time application.

3.7 PSF integration for digital twins

In order to demonstrate the possibility of utilizing the PSF in a digital twin, a version of the PSF was adapted to be compatible with a model of the milliAmpere ferry [49]. A group in the course "Experts in Teamwork" used this to showcase

the PSF in a realistic virtual reality (VR) environment. Figure 3.8 presents two scenarios where the PSF is activated. The ships with the red markings are the obstacles used in the PSF to calculate a safe trajectory. Although this demonstration did not use an RL agent for navigation and instead utilized inputs from a joystick controller, it illustrates the PSF functionality in a more visually realistic simulation environment. While it was outside the scope of this thesis to explore the use of PSF in digital twins more thoroughly, we note that applying an RL agent in this framework could present interesting possibilities for future work.

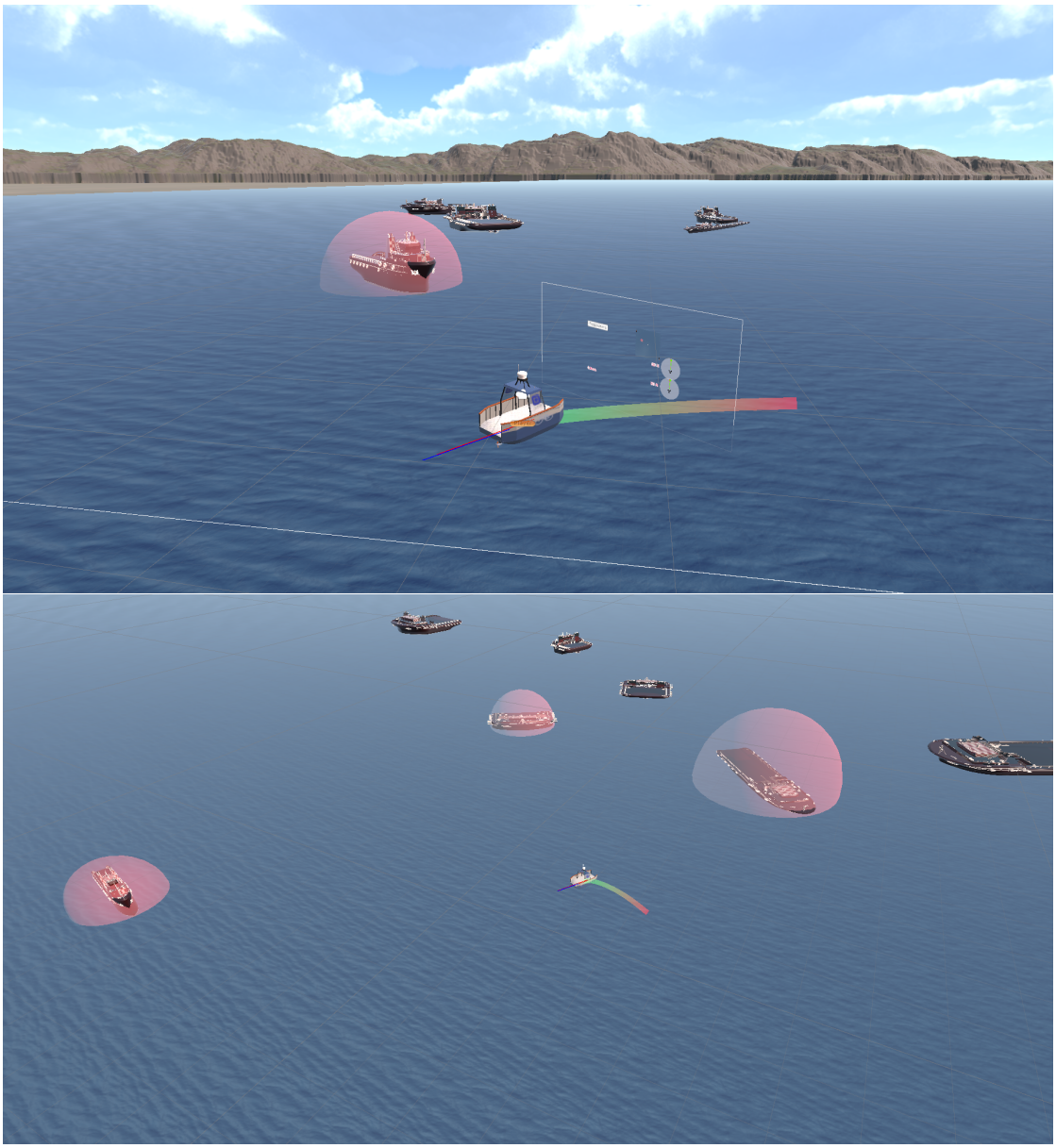


Figure 3.8: Illustration of a PSF implemented on a digital twin. The predicted safe path is visualized in green (short-term prediction) and red (long-term prediction).

3.8 Reward function

The reward function employed is derived from the collision avoidance reward function proposed by Meyer [25], with an additional term included for safety violations in the PSF. The reward function includes the main terms: r_{path}, r_{colav} and r_{PSF}, which respectively provides rewards for path following, collision avoidance, and safety violations. Additionally, there are two constant terms r_{exists} and $r_{collision}$, which is the living and collision penalty. A more detailed elaboration on the path and collision avoidance reward can be found in [25].

Path reward. The path reward has both a velocity-based and a CTE-based reward. The velocity-based reward is chosen in order to reward speed close to the maximum vessel speed U_{max}, while the look-ahead heading term, $\tilde{\psi}$ is small. The CTE-based reward penalizes large cross-track errors, with the expression going toward zero. In order for the path reward not to be zero when the cross-track error is large, the constant term γ_r is included.

$$r_{\text{path}}^{(\text{t})} = \underbrace{\left(\frac{u^{(t)}}{U_{\max}} \cos \bar{\psi}^{(t)} + \gamma_r\right)}_{\text{Velocity-based reward}} \underbrace{\left(\exp\left(-\gamma_\epsilon \left|\epsilon^{(t)}\right|\right) + \gamma_r\right)}_{\text{CTE-based reward}} - \gamma_r^2 \tag{3.20}$$

Collision avoidance reward. The collision avoidance reward penalizes the vessel for being close to obstacles heading towards it. It uses the LiDAR sensor measurements, where d_i is the i^{th} distance sensor measurement, θ_i is the vessel-relative angle of the corresponding sensor ray, and v_y^i is the y-component of the i^{th} velocity measurement. The final expression, which accounts for both static and dynamic obstacles, is the following weighted average.

$$r_{\text{colav}}^{(t)} = -\frac{\sum_{i=1}^{N} \frac{1}{1+\gamma_\theta|\theta_i|} \exp\left(\max\left(0, v_y^i\right) - \gamma_x d_i\right)}{\sum_{i=1}^{N} \frac{1}{1+\gamma_\theta|\theta_i|}} \tag{3.21}$$

Safety violation reward. In order to avoid the agent to rely on the PSF, a negative reward is added for actions that violate the constraints in the PSF. The weighting factor γ_{PSF} decides how much the agent is penalized for being corrected by the PSF. It considers the proposed RL input $\mathbf{u}_{\mathcal{L}}$, the PSF-corrected input $\mathbf{u}_0$, and the maximum possible input $\mathbf{u}_{\max}$.

$$\begin{aligned} r_{PSF}^{(t)} &= -\gamma_{PSF} \| \frac{\mathbf{u}_{\mathcal{L}} - \mathbf{u}_0}{\mathbf{u}_{\max}} \|_1 \\ &= -\gamma_{PSF} \left(\left| \frac{F_u - F_{u,PSF}}{F_{u,\max}} \right| + \left| \frac{T_r - T_{r,PSF}}{T_{r,\max}} \right| \right) \end{aligned} \tag{3.22}$$

Complete reward function. The final expression includes the two constant terms for living and collision penalty, r_{exists} and $r_{collision}$. Additionally, the parameter λ is provided to regulate the trade-off between the path and collision avoidance reward.

$$r = \begin{cases} r_{\text{collision}} , & \text{if collision} \\ (1-\lambda) r_{\text{path}} + \lambda r_{\text{colav}} + r_{PSF} + r_{\text{exists}} , & \text{otherwise} \end{cases} \tag{3.23}$$

Table 3.6 shows the parameters used in the reward function. The majority of them were selected based on previous work in the **gym-auv** environment [44]. The exception was the new parameter γ_{PSF}, which was chosen after some experimentation. A suitable value was found to ensure a large penalty for safety violations to encourage safe actions early on in the training. Subsequently, after safe behavior is learned, the agent could optimize for other terms in the reward function.

Table 3.6: Reward function parameters

Parameter	Interpretation	Value
γ_e	Cross-track error scaling	5
γ_θ	Sensor angle scaling	10.0
γ_x	Obstacle distance scaling	0.1
γ_{PSF}	Safety violation scaling	5
γ_r	Constant in path reward	1
λ	Objective trade-off coefficient	0.05
$r_{\text{collision}}$	Collision penalty	-1000
r_{exists}	Living penalty	-1.0

3.9 Test Cases

To assess the performance of the PPO + PSF agent, four test cases were defined with different levels of difficulty and complexity. For test cases 1 and 2, disturbances are not included. Removing disturbances allow us to verify the implementation of the PSF, and study the impact of the PSF on the learning rate and behavior of the agent in a controlled setting. For test cases 3 and 4, environmental disturbances are added to increase the difficulty and realism of the training and test scenarios. In the training environment, each episode continues until one of three termination criteria was satisfied: the distance to the goal location was less than 5 meters, the path progress exceeded 99%, or the episode reached the maximum limit of 5000 timesteps. A description of each test case follows.

3.9.1 Case 1: Predescribed path with stationary obstacles

The first case considers a randomly generated path with stationary obstacles. Each obstacle is generated with random size and position according to the pre-set parameters $\mu_{\text{r, stat}}$ and σ_d. Table 3.7 shows the chosen parameters for the scenario and Figure 3.9a shows an example of a generated scenario. This scenario is a good starting point to evaluate how the inclusion of the PSF changes the performance in a basic obstacle avoidance setting compared to the standard PPO algorithm, which already has demonstrated good results in similar scenarios [25].

Table 3.7: Parameters for generating case 1

Parameter	Description	Initialization
$N_{o,stat}$	Number of static obstacles	8
N_w	Number of path waypoints	2
L_p	Path length	500
$\mu_{\text{r, stat}}$	Mean static obstacle radius	30
σ_d	Obstacle displacement distance standard deviation	100

3.9.2 Case 2: Predescribed path with stationary and moving obstacles

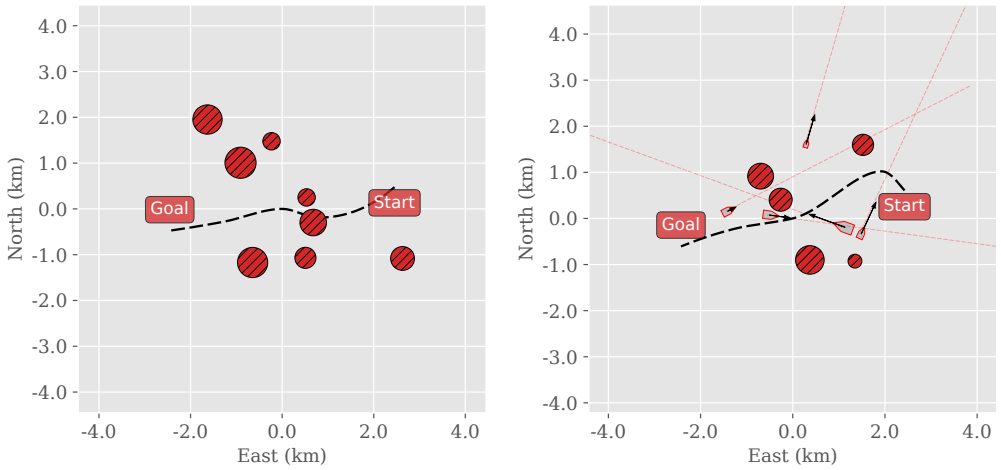
The second case includes moving obstacles in order to simulate ships in a real marine environment. The moving obstacles are spawned similarly to the static obstacles and follow linear trajectories. Table 3.8 shows the chosen parameters and Figure 3.9b shows an example of a generated scenario.

Table 3.8: Parameters for generating case 2

Parameter	Description	Initialization
$N_{o,stat}$	Number of static obstacles	5
$N_{o,dyn}$	Number of dynamic obstacles	5
N_w	Number of path waypoints	2
L_p	Path length	500
$\mu_{\text{r, stat}}$	Mean static obstacle radius	25
$\mu_{\text{r, dyn}}$	Mean moving obstacle radius	15
σ_d	Obstacle displacement distance standard deviation	100

3.9.3 Case 3: Predescribed path with stationary and moving obstacles and disturbances

In test case 3, randomized environmental disturbances are added to the simulations in order to assess the robustness of the predictive safety filter under more realistic and challenging conditions. The ocean current and disturbance forces are generated according to equation 3.1. Both static and dynamic obstacles are included and generated using the same parameters as in case 2. The environmental disturbances are measured using the observer described in section 3.5, and the estimates are included in both the predictive safety filter and the observation vector of the reinforcement learning agent.

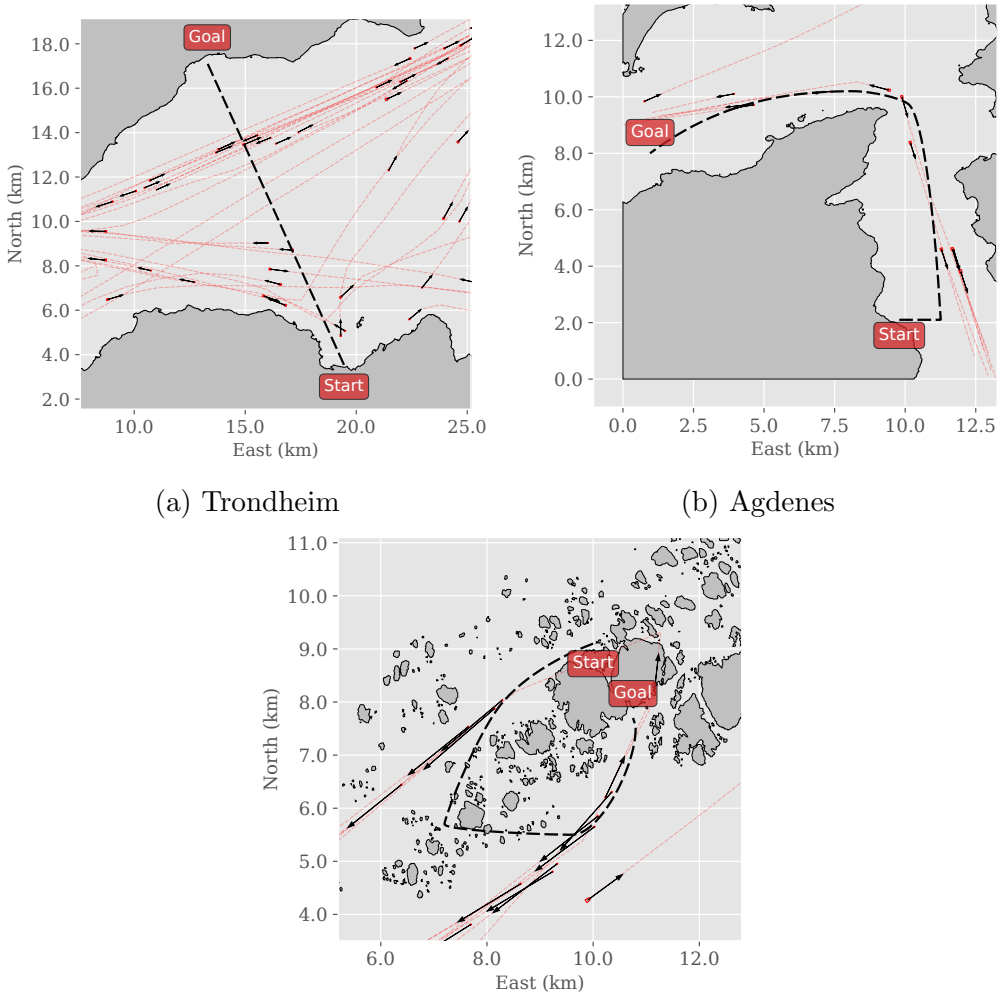


(a) Scenario generated in case 1 (b) Scenario generated in case 2

Figure 3.9: Sample of two randomly generated scenarios for test cases 1 and 2. Red circles indicate static obstacles, which are randomized both in terms of position and size. The dotted black curve indicates the path, which might be obstructed by obstacles. If this is the case, the agent should find the minimum necessary deviation from the path. In the right-hand figure, red polygons indicate target ships, while the red lines show their trajectories. Target ships are allowed to pass through static obstacles and each other because otherwise, prohibitively complex randomization procedures would be necessary.

3.9.4 Case 4: Realistic environment

Finally, the fourth case evaluates the algorithm's performance in more realistic marine environments. These environments were developed by Meyer [25] and include terrain data from the Trondheim fjord and AIS tracking data from vessels in the area. There are in total three challenging environments that require a different set of skills to navigate. The **Trondheim** scenario requires the vessel to follow a straight path to cross the fjord while avoiding traffic from multiple crossing ships. In the **Agdenes** scenario, the vessel has to blend in with two-way traffic in order to avoid collisions in a narrow area at the entrance of the Trondheimsfjord. Lastly, for the **Sorbuoya** scenario, the vessel has to navigate through hundreds of small islands to get to the goal, requiring proficient static obstacle avoidance. Each scenario is generated with a random sample of ships from an AIS database, such that the vessel will face a variety of different traffic situations. The disturbances from Case 3 were also included in this scenario, to further improve the realism. A snapshot of the different scenarios with ship trajectories can be seen in Figure 4.8. These scenarios will only be used for testing the agents trained in case 3 in order to verify the agent's ability to generalize to novel environments.



(a) Trondheim

(b) Agdenes

(c) Sorbuoya

Figure 3.10: Snapshots from the realistic environments used in Case 4. The Trondheim scenario has a relatively open path, but features a significant amount of crossing traffic. The Agdenes scenario tests the agents' capabilities of avoiding parallel traffic, while the Sorbuoya scenario is scattered with small islands that must be avoided.

3.10 Evaluation

The evaluation aims to compare the standard PPO algorithm with the PPO + PSF algorithm to demonstrate how the inclusion of the PSF changes the learning process and behavior of the agent. The assessment of the agent's performance was two-fold. First, the performance during training in the environment was evaluated. Then, testing results were examined to compare the final performance of the two agents.

Training performance. To evaluate the training performance, we first considered the number of collisions during training. The standard PPO agent is expected to have several collisions early on, but later go towards zero when a good policy is learned. On the other hand, the PPO + PSF agent is expected to have zero collisions during the entire training process since the PSF should be able to correct the actions of any suboptimal policy in order to avoid collisions. Therefore, by comparing the number of collisions, we should be able to differentiate the two algorithms. It is also a good indicator that the PSF ensures safe behavior. Additionally, statistics on reward and cross-track error during training are used to evaluate the overall performance and to understand how the inclusion of a PSF influences the learning process.

Test performance. The trained agents are tested on 100 randomly generated scenarios to evaluate their final performance. This sample size was chosen as it provides statistically significant results while keeping computational time manageable. We decided to use slightly different performance metrics for the test results. While the average reward and cross-track error are useful for observing the learning progress and overall performance in terms of reward, these metrics become less relevant in a practical setting. What is more critical then, is whether the agent reaches the goal within a reasonable time and without any collisions. Therefore, we used the average path progress and time consumption as performance indicators, in addition to collision avoidance. While the path progress and collision avoidance are straightforward to calculate, the metric for time consumption requires further explanation. The minimum possible time was set as the path length divided by the top speed of the vessel, which corresponds to 100%. Note that this is usually not possible to achieve in practice. The maximum time was set to the highest number of timesteps allowed in a scenario before the episode was terminated, which was set to 5000 in all cases, corresponding to 0%. All episodes with a collision were excluded from the time consumption average.

In addition to statistical data collected during training and testing, figures illustrating the agents' behavior in various scenarios are also provided. These can assist in making more qualitative assessments of the performance.

4 Results and Discussions

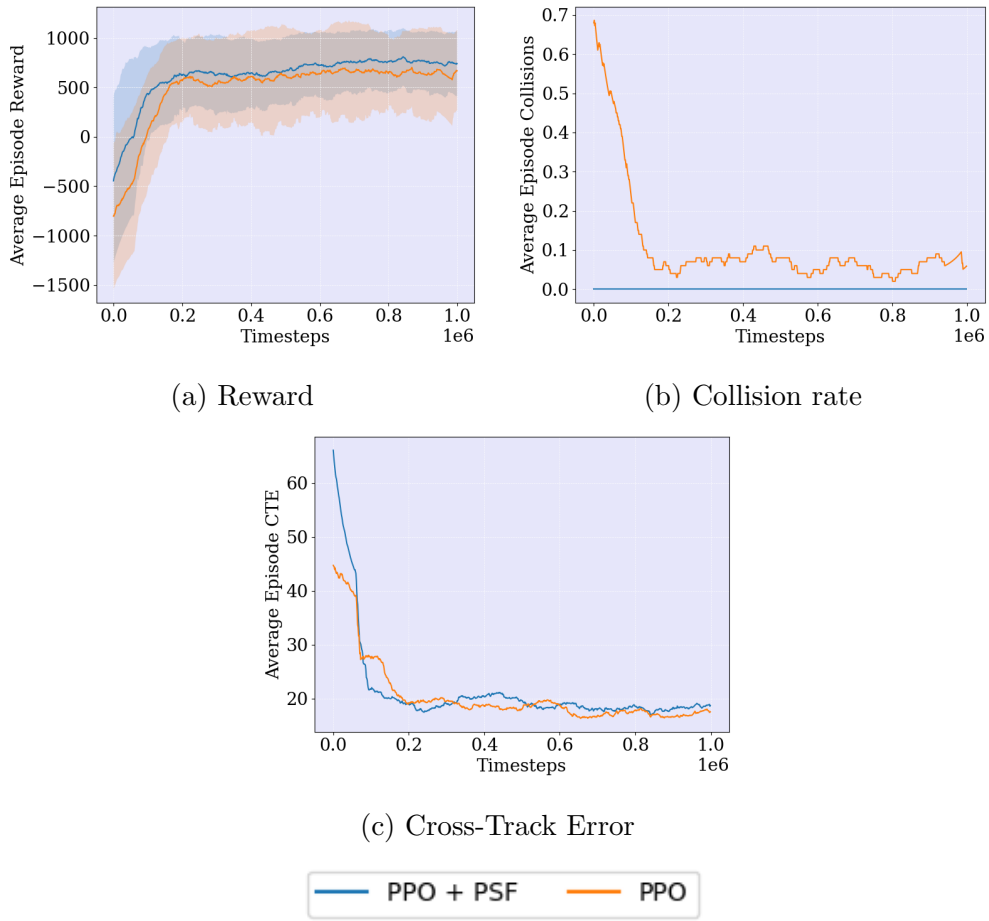
In this chapter, the results of the simulation experiments are presented and discussed, with each of the 4 test cases covered separately. Rolling averages of the agents' total reward, cross-track error and collision rate during the training phases are plotted individually. The average episode progress, average collision avoidance rate and average time consumption in the test phase are condensed into radar charts, to better facilitate a comparison of the total performance of the standard PPO agents versus the PPO + PSF agents. In addition, sample trajectories from the various test environments are visualized and discussed.

4.1 Training results

4.1.1 Case 1: Predescribed path with stationary obstacles

Each agent was trained for 1 million timesteps, which corresponded to around 850 episodes, with the exact number depending on the time spent to complete each scenario. As can be seen in Figure 4.1, no collisions occurred for the PPO + PSF agent in case 1. The PPO agent conversely experienced a high collision rate early on, but eventually reached a rate near zero. The reward for the PPO + PSF agent is slightly higher throughout the training, which is partly because of the absence of collisions. Additionally, the cross-track error is slightly lower for the PPO agent. This is expected since the PSF requires the agent to be at a minimum constant distance to every obstacle, which in some cases would mean it has to stay further away from the path.

A comparison between the agent trajectories at different stages of the training can be seen in Figure 4.2. These agents were trained on random scenarios in case 1 for different durations, and tested on a sample scenario to compare their performance. After 10.000 timesteps the agent performs quite poorly, with the PPO agent crashing at an early stage. The PSF saves the PPO + PSF agent from crashing similarly by modifying the action to perform a sharp right turn before the obstacle is reached. Notice that even though the agent does not reach the goal exactly, the scenario finishes since the path progress is above 99 %, which is one of the termination conditions. Gradually as the agents learn, we observe that the PPO agent becomes better at collision avoidance and that both agents achieve a lower cross-track error. After 400.000 time steps, both agents follow a close to optimal trajectory for this specific scenario. Interestingly, the PPO + PSF agent seems to converge to the optimal trajectory faster than the PPO agent. While this is less evident in the cross-track error plot in Figure 4.1c, it could suggest that the absence of collisions early on in the training accelerates the learning process.



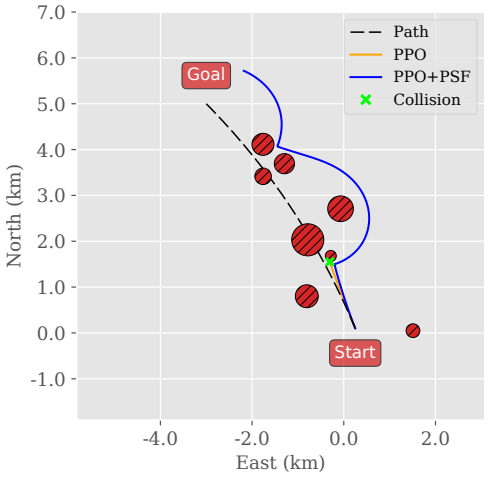
(a) Reward

(b) Collision rate

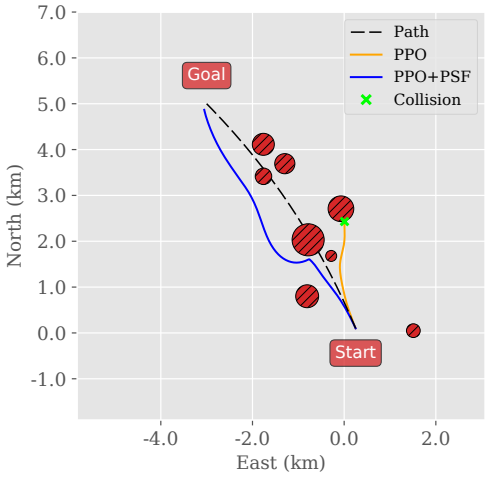
(c) Cross-Track Error

Figure 4.1: Average reward, collisions, and cross-track error during training in case 1, smoothed with a rolling average over 100 episodes. The collision rate is zero for the PPO + PSF agent during the entire training period. The difference in collision rate is especially striking during the first 200.000 time-steps of training, where the standard PPO agent crashes in about 40% of the episodes

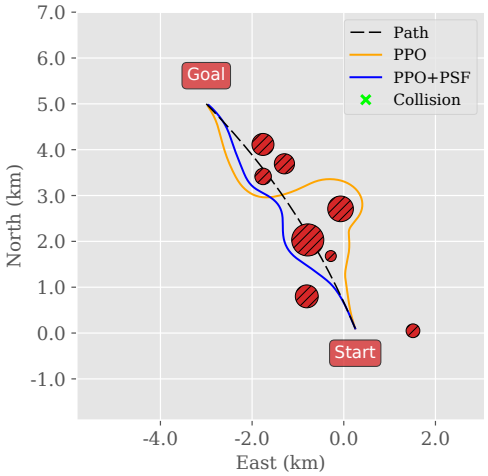
Furthermore, Figure 4.3 illustrates instances when the PSF is activated during training. The green dots visualize where the agents' intended action is regarded as unsafe, and therefore corrected by the PSF. Because the PSF avoids situations that otherwise likely would become collisions, the episode length for the PPO + PSF agent is significantly higher in the early stages of training compared with the pure PPO agent. Additionally, within a single episode, the agent can encounter multiple situations that would have resulted in crashes without the PSF. This allows the agent to experience more unsafe states within a shorter amount of time. As a result, the PPO + PSF agent is able to learn more effectively from a smaller number of training episodes.



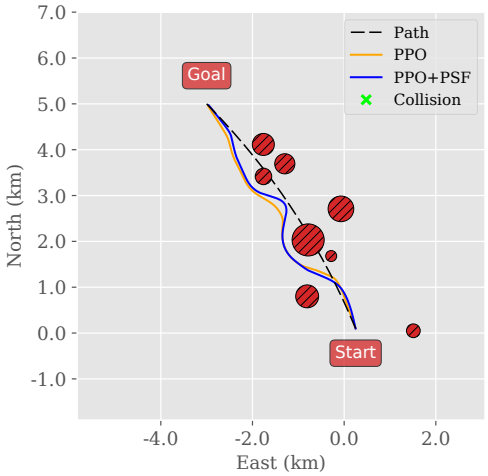
(a) Agent trained for 10.000 timesteps.



(b) Agent trained for 50.000 timesteps

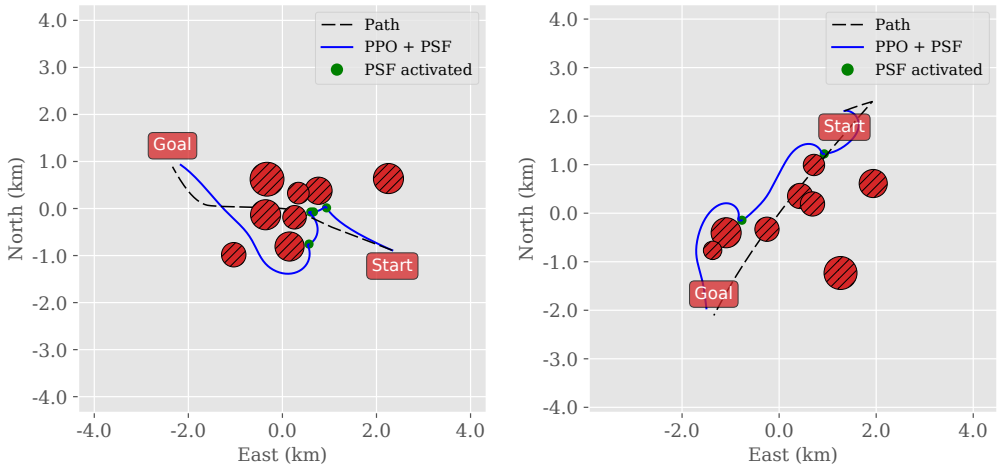


(c) Agent trained for 100.000 timesteps.



(d) Agent trained for 400.000 timesteps.

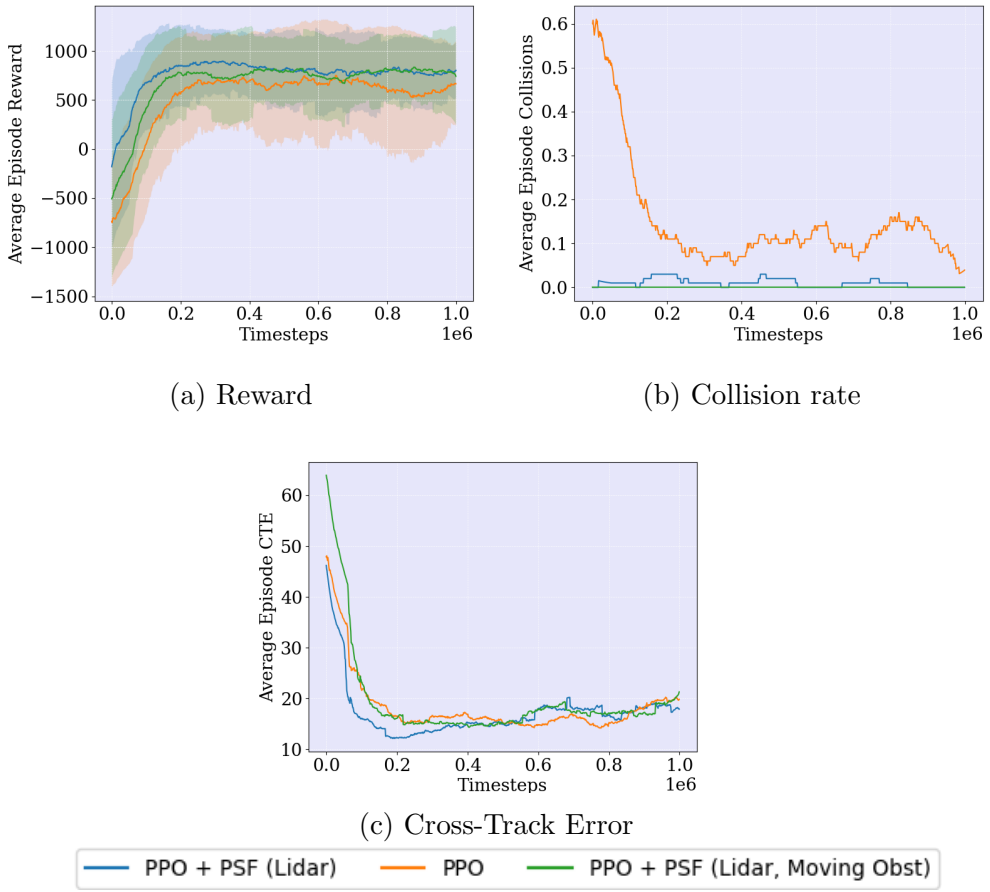
Figure 4.2: Comparison between the PPO and the PPO + PSF agent at different stages of training. Notice that the PPO + PSF agent converges faster to an optimal trajectory for this specific scenario.



(a) Scenario 1 (b) Scenario 2

Figure 4.3: PSF corrections in two scenarios for an agent trained for 50.000 timesteps. The PSF prevents a collision multiple times to ensure that the agent reaches the goal destination. Because the PSF only optimizes with respect to finding the minimum perturbation to the proposed agent input, the PSF does not modify the control input until the last time-step before collision is unavoidable. Therefore the agent is close to the obstacles before control input modifications occur.

4.1.2 Case 2: Predescribed path with stationary and moving obstacles



(a) Reward

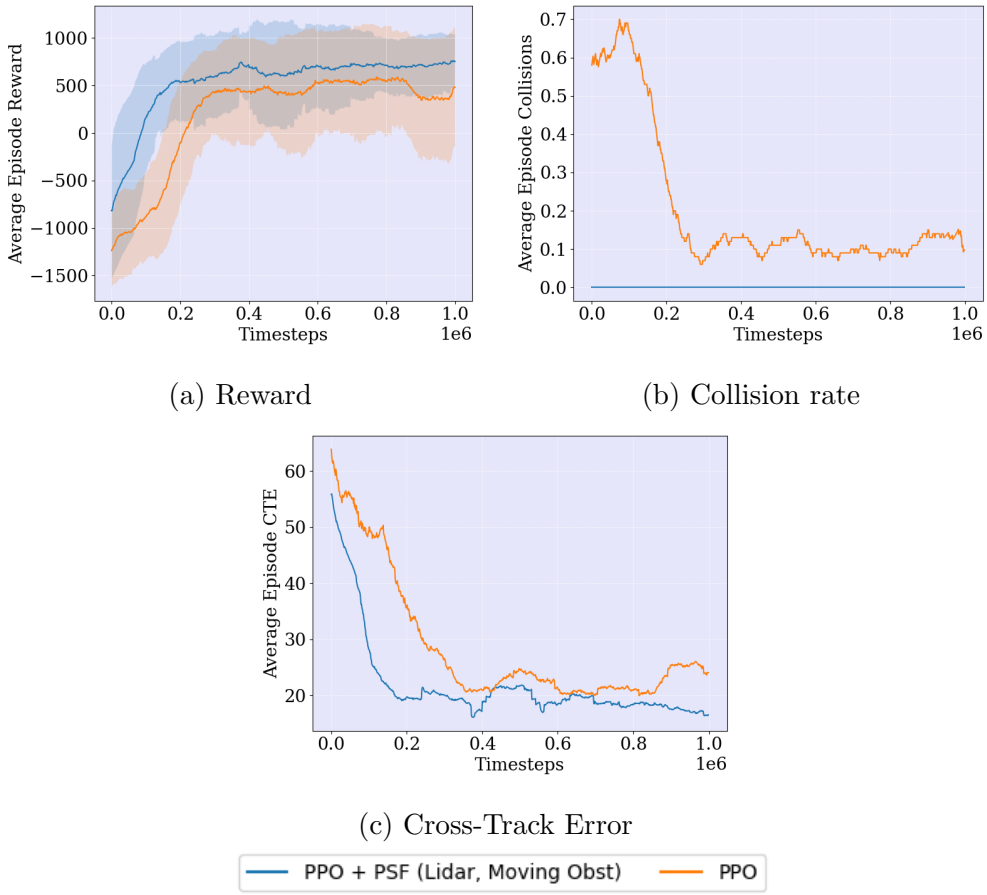
(b) Collision rate

(c) Cross-Track Error

Figure 4.4: Average reward, collisions, and cross-track error during training in case 2, smoothed with a rolling average over 100 episodes. Only the PPO + PSF with information about the moving obstacles has a collision rate of zero during the entire training period.

In case 2 we observe that the PSF using only LiDAR for collision avoidance no longer managed to prevent all collisions during training, as shown in figure 4.4. However, when enhanced with explicit moving obstacle collision avoidance, the PSF is able to predict the trajectories of the other ships, which in turn leads to better decisions for avoiding unsafe situations. The approach proved to be effective, resulting in zero collisions during training. In contrast, the standard PPO agent struggled more, with a higher collision rate than in case 1, especially during the later stages of the training, which can be seen from figure 4.4b.

4.1.3 Case 3: Predescribed path with stationary and moving obstacles, and environmental disturbances



(a) Reward

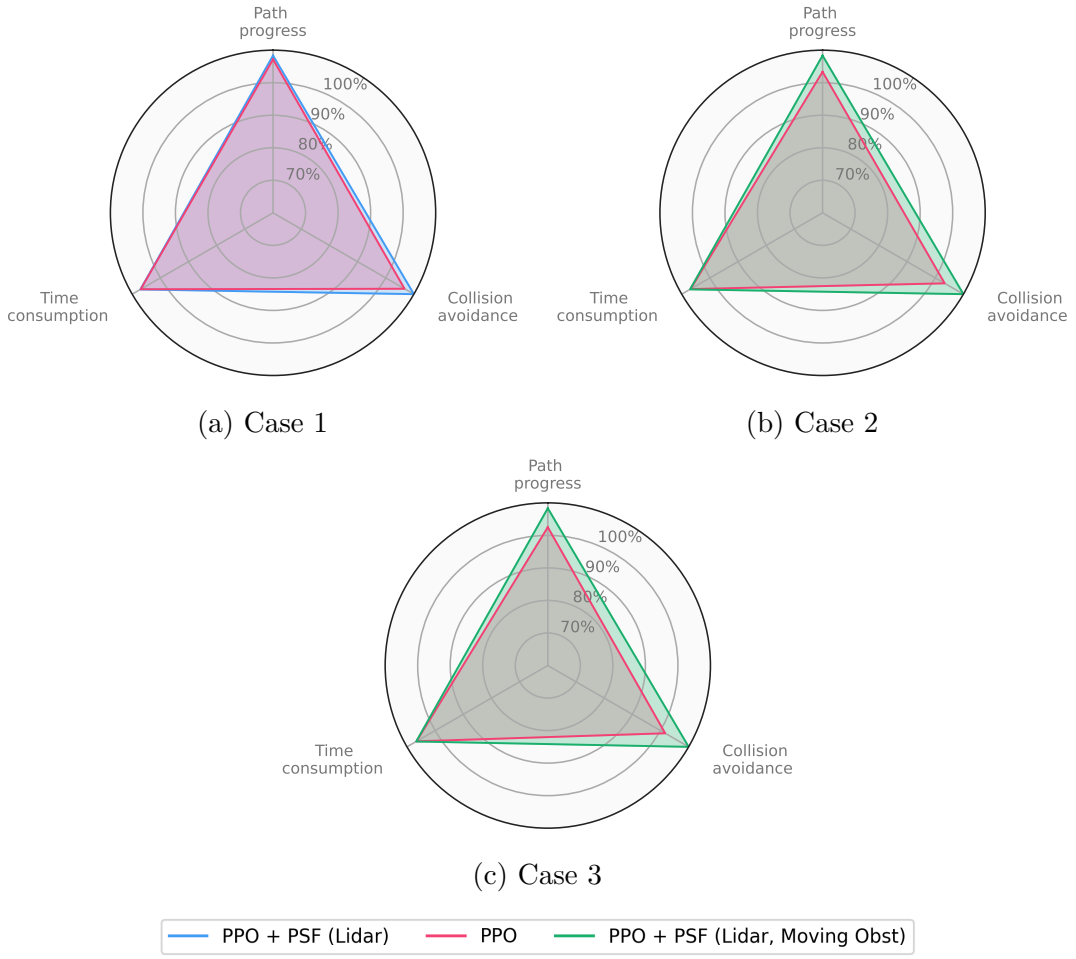
(b) Collision rate

(c) Cross-Track Error

Figure 4.5: Average reward, collisions, and cross-track error during training in case 3, smoothed with a rolling average over 100 episodes. The added disturbances lower the overall performance of the agents, but they still converge to a satisfactory level

To account for the added disturbances in this case, the observation vector was augmented with the estimates from the disturbance observer described in section 3.5. The training results in Figure 4.5 show that the PPO agent took a longer time before reaching a low collision rate, compared to previous cases. As anticipated, the cross-track error is slightly higher, while the total reward is slightly lower. During the initial 200,000 timesteps of training, the collision rate for the standard PPO agent actually increased before it began to learn more effective collision avoidance behavior. Generally, the plots show more fluctuations due to the added disturbances.

4.2 Test results



(a) Case 1

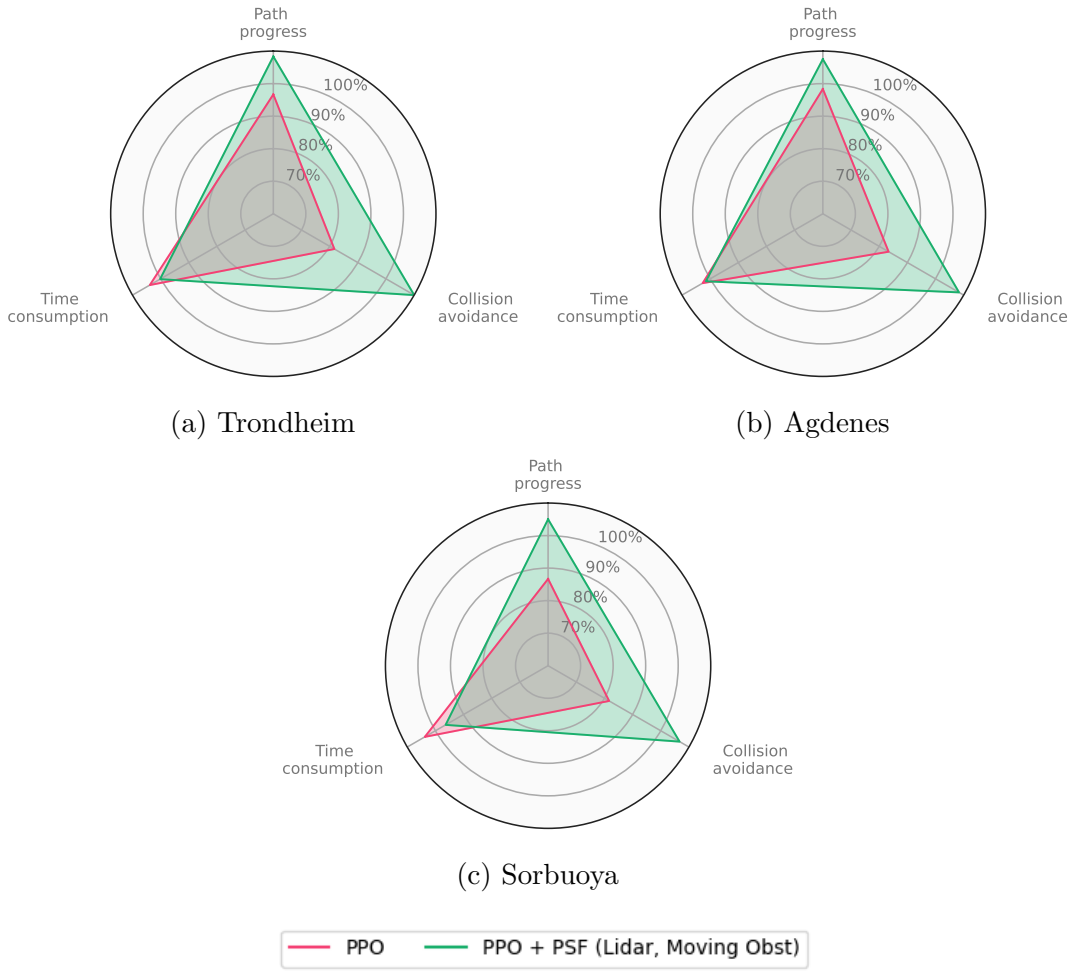
(b) Case 2

(c) Case 3

Figure 4.6: Test results for Case 1, 2 and 3. In Case 2 and 3 the PSF was modified to have access to information about the moving obstacles in addition to the LiDAR obstacle detection. The PPO + PSF agent has a perfect score on path progress and collision avoidance in all three cases.

After each agent was trained for 1 million timesteps, they were run for an additional 100 episodes to evaluate the test performance. The results are visualized in radar charts in Figure 4.6. The PPO + PSF agents maintained perfect collision avoidance for all 3 cases, successfully completing every episode. The standard PPO agent did not achieve the same level of performance, scoring 98%, 96% and 95% respectively on case 1, 2 and 3 on collision avoidance. As expected, the PPO agent generally struggled more in the scenarios with moving obstacles, while the inclusion of the PSF improved the agent's moving obstacle collision avoidance capabilities. The addition of disturbances in case 3 did not lower the performance significantly, suggesting that the environmental disturbances were efficiently handled by the observer. Finally, the time consumption is similar for the agents in all three cases.

4.2.1 Case 4: Realistic environments



(a) Trondheim

(b) Agdenes

(c) Sorbuoya

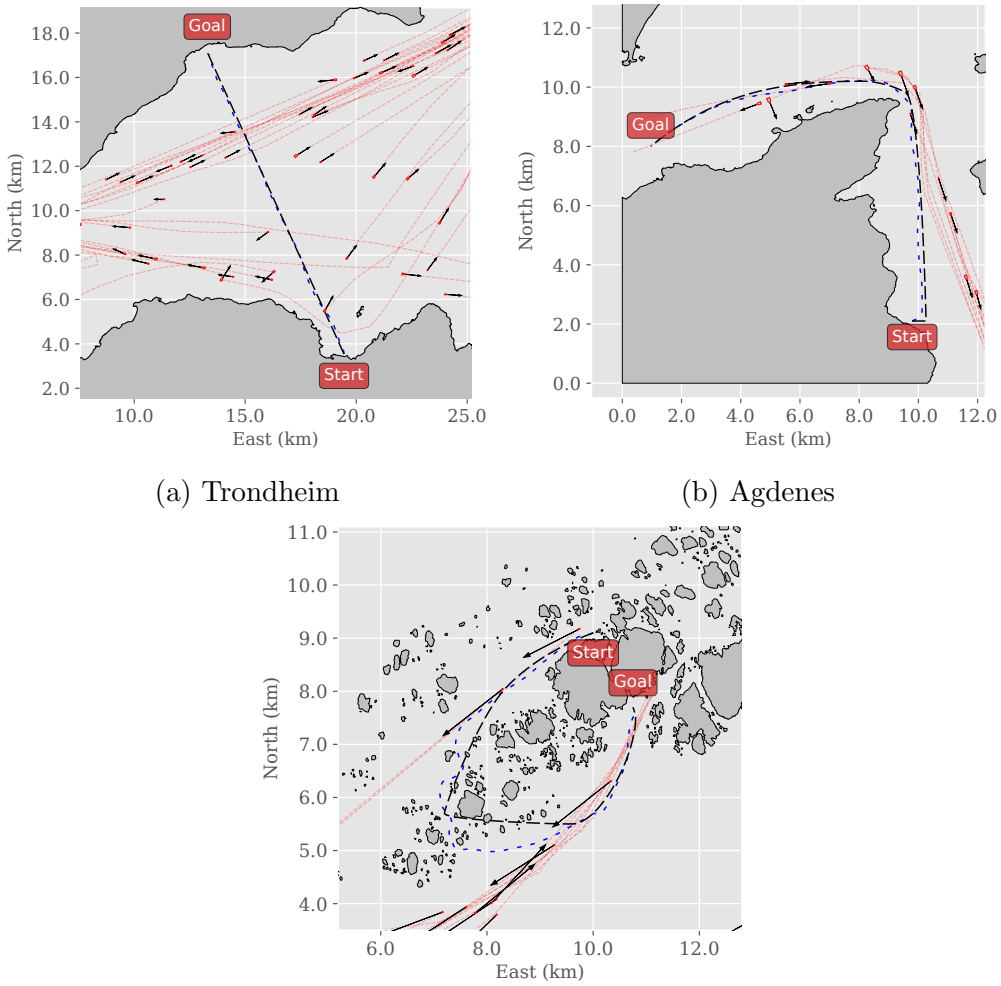
Figure 4.7: Test results for the Trondheim, Agdenes and Sorbuoya scenario. The PPO + PSF agent performs better on path progress and collision avoidance in all three cases, but it has a slightly higher time consumption.

Case 4 is the most challenging test case, with three realistic environments (Trondheim, Agdenes, Sørbuøya) of increasing difficulty. Prior to testing in these environments, the agents were trained for 2.000.000 time-steps on randomized scenarios generated according to case 3, with the PPO + PSF agent using both LiDAR-based collision avoidance and explicit moving obstacle collision avoidance. As seen in Figure 4.7, the agents in general perform worse in these environments. However, the discrepancy between the agents across the various performance metrics is also much more noticeable. Furthermore, in Figure 4.8, the path taken by the PPO + PSF agent during one episode is plotted for all three of the realistic environments. Observe that the Sorbuoya environment is the only case where the agent has to significantly deviate from the path in order to reach the goal. In the two other scenarios, the agent mostly adheres to the path, making maneuvers only when a ship gets too close.

For the Trondheim environment, PPO + PSF has a 0% collision rate compared to 17% for the standard PPO agent, at the expense of a slightly higher average time consumption. Looking at the average episode progress, we see that PPO + PSF indeed manages to successfully reach the goal more often. The PPO + PSF agent used slightly more time, which was caused by the PSF in occasionally choosing a longer route to avoid collisions in episodes with crossing traffic.

In the Agdenes environment, the PPO + PSF agent again performed significantly better than standard PPO. However, a single collision was registered, meaning that the predictive safety filter was not able to fully guarantee the safety of the agent in this environment. Looking at the scenario where the agent collided, it seems that the particular configuration of target ships, and the path chosen by the agent, led to a situation where multiple ships were simultaneously approaching it from different directions, making it extremely difficult to avoid collision. Having a longer prediction horizon for the PSF might have enabled it to recognize the potential hazard at an earlier time. It should be mentioned however that even though the trajectories of the target ships are taken from real AIS data, they themselves do not try to avoid collision in any way.

The Sorbuoya environment is the most difficult to navigate, which is reflected in the results. Again the PPO + PSF agent has much lower collision rate, but still there were two registered collisions across the 100 episodes. Additionally, the standard PPO agent has a significantly shorter average episode duration. This is a result of a few instances similar to the one shown in Figure 4.9 where the episode was terminated after the maximum number of timesteps was reached. Because the Sorbuoya environment consists of many densely packed islands, it is possible for the agent to get stuck in virtual dead ends, where the islands blocking the forward path are clustered in such a way that the predictive safety filter cannot find a canal that is wide enough to pass through considering the pre-specified minimum safety distance.



(a) Trondheim

(b) Agdenes

(c) Sorbuoya

Figure 4.8: PPO + PSF agent trajectories in the real-world scenarios. The blue dotted line is the path taken by agent and the red dotted lines are the path taken by the moving obstacles. The agent stays close to the desired path in all scenarios except for Sorbuoya, where it has to deviate in order to avoid collision with islets.

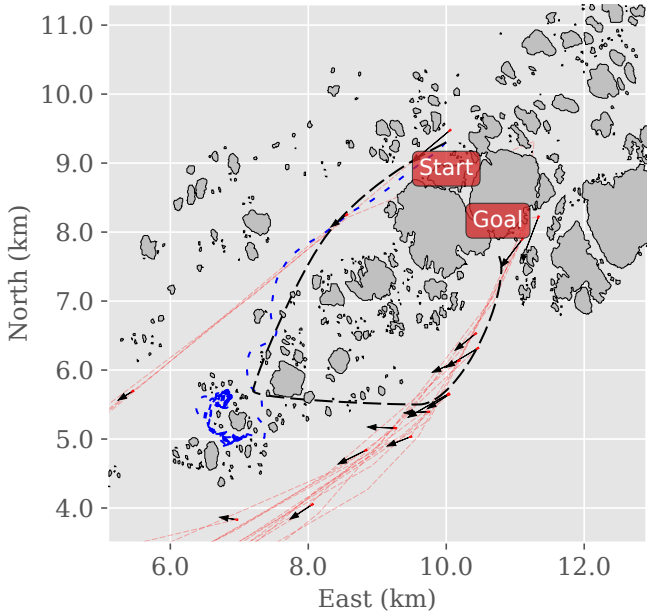


Figure 4.9: Example undesired agent path in Sorbuoya scenario. The PPO + PSF agent does not manage to reach the goal because it gets stuck in between the cluster of small islands, and can not find a path through without safety violations.

4.3 Limitations

The test results are based on simulations of the 3-DOF ship dynamics model described in chapter 2. Important factors such as measurement noise, model inaccuracy and sensor failure were not considered in this report, and these must be accounted for if physical experiments are to be conducted. The training scenarios for the agents were randomized, however the degree of variation could be increased in order to produce agents better capable of navigating real-world scenarios. Additionally, the target ships used in the simulations were modeled as naive actors, not capable of performing collision avoidance themselves. In a real setting, all ships would cooperate in order to maximize safety. Finally, the Convention on the International Regulations for Preventing Collisions at Sea (COLREGS) [50] was not considered in this work. A fully functional autonomous agent should be able to comply with COLREGS, however the specific COLREGS rules remain quite ambiguous and are not straightforward to implement algorithmically, especially in situations with a high number of ships involved [51]. Despite this, previous work on implementing COLREGS in the **gym-auv** framework has been conducted by Heiberg [28] with promising results. It was not considered mainly because it would require a more sophisticated implementation of the PSF algorithm, which is beyond the scope of this study.

5 Conclusion and Future Work

We have presented a solution for safe ASV control, combining reinforcement learning with a predictive safety filter. The method allows an RL agent to propose actions, with the PSF performing corrections in order to guarantee constraint satisfaction, which is vital for safety-critical systems. We demonstrated how the inclusion of a PSF significantly reduces the number of collisions during testing and training of a state-of-the-art RL algorithm in various complex scenarios, some including environmental disturbances. Furthermore, with regard to the research questions proposed for the thesis, the main conclusions can be summarized as follows:

Can a PSF/RL scheme, capable of real-time collision avoidance and safety verification, be successfully realized within an ASV simulation environment? Based on the simulation results, we are confident that the predictive safety filter is a promising strategy for ensuring safety in learning-based ship navigation and control. The PSF successfully managed to keep the RL agent from colliding during all training and test episodes for each of the randomly generated test cases (test case 1 to 3). In addition, the PSF-enhanced agent performed just as good as the baseline, requiring fewer training episodes to converge to a satisfactory performance level. Re-engineering the PPO reward function to accommodate the predictive safety filter was relatively straightforward, requiring only an additional PSF-activation penalty, and performing simple tuning. This indicates that the introduction of the PSF is not prohibitive with regard to the additional time spent re-designing the RL algorithm itself. Using state-of-the-art non-linear model predictive control software, an average OCP solver runtime of less than 10 milliseconds was achieved, which is comfortably within the requirements of real-time application. In addition, the fast runtime of the optimization solver meant that the PSF-enhanced agent needed only slightly more time to complete the same number of simulation time steps compared with the standard PPO agent.

What effect does predictive safety filtering have on the performance and learning of reinforcement learning agents designed for autonomous surface vessels? As expected, the predictive safety filter had the most impact in the initial phase of training. During the first 100.000 time-steps of training the standard PPO agent often registered a collision rate above 50%. By using the PSF, the learning agent is able to stay alive and collect much more experience from the initial episodes. This is advantageous not only from a safety perspective, but also in terms of learning efficiency. In miniature-scale physical experiments the cost of collision is not necessarily high, however keeping the agent from colliding for as long as possible can save a significant amount of time and labor that would otherwise be necessary

to reset the episode and environment after a collision. For fully trained agents (trained for 1 to 2 million time-steps), the PPO + PSF agents behaved similarly to the standard PPO agents in the randomly generated environments, with close to the same average cross-track error, and choosing similar paths most of the time.

Are predictive safety filters a feasible approach to incorporate reinforcement learning in real-world marine environments? Both agents performed worse in the realistic scenarios with terrain data, although the PSF-enhanced agent still performed significantly better than the standard PPO agent. The collision avoidance rate, and consequently the rate of successful episodes, was significantly higher for PPO + PSF compared with standard PPO. Still, across a total of 300 simulations in the realistic scenarios, there were three instances where the predictive safety filter was not able to prevent collision. Identifying the exact reasons for the collisions is difficult, owing to the level of complexity in these environments. However, the fact that collisions happened even with the PSF enabled suggests that transferring from generated to real environments still poses numerous challenges, and that sufficient variation in the training phase is imperative to minimizing risk when applying the agents to real environments. The Sorbuoya environment also showed that in some situations, the additional safety margins imposed by the predictive safety filter can hinder the forward progress of the agent, even though a feasible path exists. This indicates that the trade-off between safety and freedom of exploration must be considered carefully when designing the PSF. Despite this, the overall results suggest that predictive safety filters can significantly improve safety when deploying RL-based autonomous vessels in real environments, thereby increasing the viability of reinforcement learning in marine navigation and control.

Future work. In this thesis, the main focus has been on demonstrating how a reinforcement learning agent enhanced with a predictive safety filter can learn to perform efficient ASV navigation and control, while safety is ensured. The long-term goal would be to apply the method on a real vessel, verifying the functionality and effectiveness of the controller. We imagine that possible future work could include:

- **Increased realism in the simulation environment.** In a real setting, factors such as measurement noise, modeling error and sensor failure must be handled rigorously in order to fully ensure safety and efficiency. The variation in the randomly generated environments, in terms of the distribution and complexity of static and dynamic obstacles, can be increased to better prepare the RL agents for the wide range of scenarios they encounter in the real world.
- **Additional randomization for more robust control.** A recent paper by Haarnoja et al. demonstrate how targeted dynamics randomization and perturbations during training can facilitate the sim-to-real transfer for soccer-playing bipedal robots trained with deep RL [52]. Similar strategies could potentially be applied in this simulation environment to get a more robust policy applicable for real-world experimentation. A careful consideration of

the level of randomization would then be necessary. Excessive randomization could lead to a conservative policy, reducing the overall performance. Conversely, insufficient randomization may not address the sim-to-real gap, resulting in poor performance in the real environment compared to simulation.

- **Perform experiments on a down-scaled ship.** A good starting point for experimentation with the controller, would be to acquire a simple test setup. This could include a small model ship with a sensor suite, objects to simulate obstacles and a pool or a designated sea area suitable for experimentation. With this, it would be possible to explore the challenges in transferring a policy learned in simulation, to a real-world system.

- **Multi-agent environments.** Another interesting topic that was not addressed in this work is the interaction and possible cooperation between multiple autonomous vessels. In a realistic setting, all involved ships cooperate to minimize collision risk and maximize efficiency. By deploying multiple agents simultaneously in the same environment, collective behaviors can be studied, and multi-agent strategies for collision avoidance can be developed.

Bibliography

[1] Sampo Kuutti, Richard Bowden, Yaochu Jin, Phil Barber, and Saber Fallah. A survey of deep learning applications to autonomous vehicle control. *IEEE Transactions on Intelligent Transportation Systems*, 22(2):712–733, 2021.

[2] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul F. Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *CoRR*, abs/1606.06565, 2016.

[3] Yewen Gu, Julio Cesar Goez, Mario Guajardo, and Stein W Wallace. Autonomous vessels: state of the art and potential opportunities in logistics. *International Transactions in Operational Research*, 28(4):1706–1739, 2021.

[4] Lutz Kretschmann, Hans-Christoph Burmeister, and Carlos Jahn. Analyzing the economic benefit of unmanned autonomous ships: An exploratory cost-comparison between an autonomous and a conventional bulk carrier. *Research in transportation business & management*, 25:76–86, 2017.

[5] Jiri de Vos, Robert G Hekkenberg, and Osiris A Valdez Banda. The impact of autonomous ships on safety at sea–a statistical analysis. *Reliability Engineering & System Safety*, 210:107558, 2021.

[6] Omer San, Adil Rasheed, and Trond Kvamsdal. Hybrid analysis and modeling, eclecticism, and multifidelity computing toward digital twin revolution. *GAMM-Mitteilungen*, 44(2):e202100007, 2021.

[7] European Maritime Safety Agency. Annual overview of marine casualties and incidents. EMSA Publication, 2022. Available online at `https://www.emsa.europa.eu/newsroom/latest-news/item/4867-annual-overview-of-marine-casualties-and-incidents-2021.htmll`.

[8] Javier Sánchez-Beaskoetxea, Imanol Basterretxea-Iribar, Iranzu Sotés, and María de las Mercedes Maruri Machado. Human error in marine accidents: Is the crew normally to blame? *Maritime Transport Research*, 2:100016, 2021.

[9] Kongsberg. Automatic ferry enters regular service following world-first crossing with passengers onboard, 2020. Accessed: 2023-05-25.

[10] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[11] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR*, abs/1712.01815, 2017.

[12] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[13] Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking Safe Exploration in Deep Reinforcement Learning. 2019.

[14] Martin Pecka and Tomas Svoboda. Safe exploration techniques for reinforcement learning – an overview. In Jan Hodicky, editor, *Modelling and Simulation for Autonomous Systems*, pages 357–375, Cham, 2014. Springer International Publishing.

[15] Lukas Brunke, Melissa Greeff, Adam W. Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P. Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning, 2021.

[16] Alexander Hans, Daniel Schneegass, Anton Schäfer, and Steffen Udluft. Safe exploration for reinforcement learning. pages 143–148, 01 2008.

[17] Teodor Mihai Moldovan and Pieter Abbeel. Safe exploration in markov decision processes. *CoRR*, abs/1205.4810, 2012.

[18] Eitan Altman. Constrained markov decision processes. 1999.

[19] Kim P Wabersich and Melanie N Zeilinger. Safe exploration of nonlinear dynamical systems: A predictive safety filter for reinforcement learning. *arXiv preprint arXiv:1812.05506*, 2018.

[20] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer science & business media, 2013.

[21] Emil H Thyri, Erlend A Basso, Morten Breivik, Kristin Y Pettersen, Roger Skjetne, and Anastasios M Lekkas. Reactive collision avoidance for asvs based on control barrier functions. In *2020 IEEE Conference on Control Technology and Applications (CCTA)*, pages 380–387. IEEE, 2020.

[22] Tor Arne Johansen, Tristan Perez, and Andrea Cristofaro. Ship collision avoidance and colregs compliance using simulation-based control behavior selection with predictive hazard assessment. *IEEE transactions on intelligent transportation systems*, 17(12):3407–3422, 2016.

[23] Espen Eilertsen. High-level action planning for marine vessels using reinforcement learning. Master's thesis, NTNU, 2019.

[24] Jonathas Marcelo Pereira Figueiredo and Rodrigo Pereira Abou Rejaili. Deep reinforcement learning algorithms for ship navigation in restricted waters. *Mecatrone*, 3(1), 2018.

[25] Eivind Meyer. On course towards model-free guidance: A self-learning approach to dynamic collision avoidance for autonomous surface vehicles. Master's thesis, 2020.

[26] Roger Skjetne, Øyvind Smogeli, and Thor I Fossen. Modeling, identification, and adaptive maneuvering of cybership ii: A complete design with experiments. *IFAC Proceedings Volumes*, 37(10):203–208, 2004.

[27] Thor I Fossen. *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2011.

[28] Amalie Heiberg, Thomas Nakken Larsen, Eivind Meyer, Adil Rasheed, Omer San, and Damiano Varagnolo. Risk-based implementation of colregs for autonomous surface vehicles using deep reinforcement learning. *Neural Networks*, 152:17–33, 2022.

[29] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

[30] Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022.

[31] Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning, 2017.

[32] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay, 2018.

[33] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization, 2017.

[34] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[35] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

[36] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

[37] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2019.

[38] Thomas Nakken Larsen, Halvor Ødegård Teigen, Torkel Laache, Damiano Varagnolo, and Adil Rasheed. Comparing deep reinforcement learning algorithms' ability to safely navigate challenging waters. *Frontiers in Robotics and AI*, 8, 2021.

[39] Daniel Menges and Adil Rasheed. An environmental disturbance observer framework for autonomous ships. *arXiv preprint arXiv:2211.08360*, 2022.

[40] Ben Tearle, Kim P. Wabersich, Andrea Carron, and Melanie N. Zeilinger. A predictive safety filter for learning-based racing control. *IEEE Robotics and Automation Letters*, 6(4):7635–7642, 2021.

[41] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016.

[42] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *The Journal of Machine Learning Research*, 22(1):12348–12355, 2021.

[43] Thomas Nakken Larsen, Halvor Ødegård Teigen, Torkel Laache, Damiano Varagnolo, and Adil Rasheed. Comparing deep reinforcement learning algorithms' ability to safely navigate challenging waters. *Frontiers in Robotics and AI*, 8:738113, 2021.

[44] Thomas Nakken Larsen, Hannah Hansen, and Adil Rasheed. Risk-based convolutional perception models for collision avoidance in autonomous marine surface vessels using deep reinforcement learning [manuscript submitted for publication]. 2022.

[45] Vebjørn Malmin, Halvor Ødegaard Teigen, and Adil Rasheed. Reinforcement learning and predictive safety filtering for floating offshore wind turbine control. 2021.

[46] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[47] Robin Verschueren, Gianluca Frison, Dimitris Kouzoupis, Jonathan Frey, Niels van Duijkeren, Andrea Zanelli, Branimir Novoselnik, Thivaharan Albin, Rien Quirynen, and Moritz Diehl. acados—a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation*, 14(1):147–183, 2022.

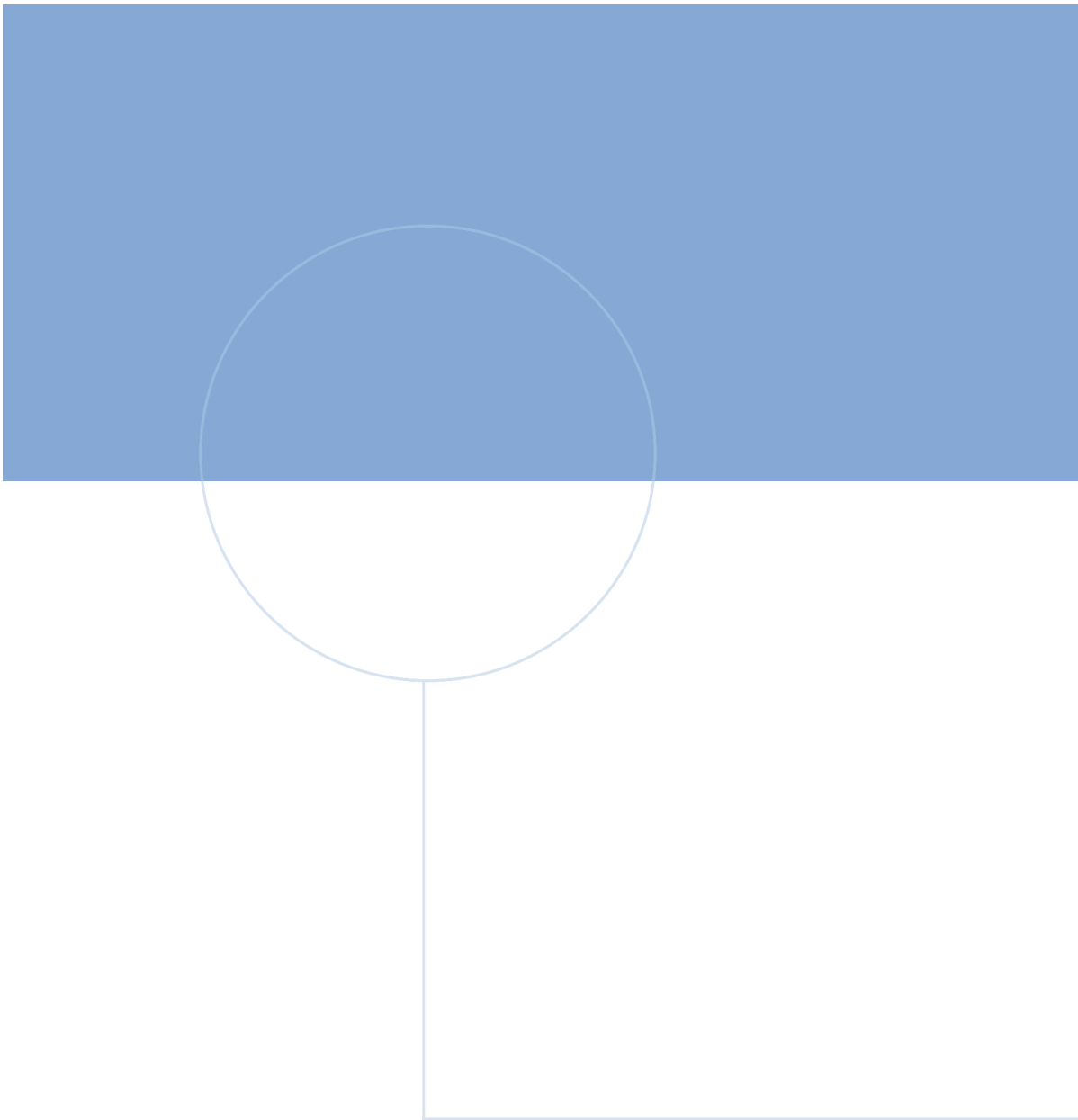
[48] Gianluca Frison and Moritz Diehl. Hpipm: a high-performance quadratic programming framework for model predictive control. *IFAC-PapersOnLine*, 53(2):6563–6569, 2020.

[49] Edmund F Brekke, Egil Eide, Bjørn-Olav H Eriksen, Erik F Wilthil, Morten Breivik, Even Skjellaug, Øystein K Helgesen, Anastasios M. Lekkas, Andreas B Martinsen, Emil H. Thyri, Tobias Torben, Erik Veitch, Ole A Alsos, and Tor Arne Johansen. milliampere: An autonomous ferry prototype. *Journal of Physics: Conference Series*, 2311(1):012029, jul 2022.

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

[51] Krzysztof Wróbel, Mateusz Gil, Yamin Huang, and Ryszard Wawruch. The vagueness of colreg versus collision avoidance techniques—a discussion on the current state and future challenges concerning the operation of autonomous ships. *Sustainability*, 14(24):16516, 2022.

[52] Tuomas Haarnoja, Ben Moran, Guy Lever, Sandy H. Huang, Dhruva Tirumala, Markus Wulfmeier, Jan Humplik, Saran Tunyasuvunakool, Noah Y. Siegel, Roland Hafner, Michael Bloesch, Kristian Hartikainen, Arunkumar Byravan, Leonard Hasenclever, Yuval Tassa, Fereshteh Sadeghi, Nathan Batchelor, Federico Casarini, Stefano Saliceti, Charles Game, Neil Sreendra, Kushal Patel, Marlon Gwira, Andrea Huber, Nicole Hurley, Francesco Nori, Raia Hadsell, and Nicolas Heess. Learning agile soccer skills for a bipedal robot with deep reinforcement learning, 2023.



NTNU

Norwegian University of
Science and Technology