Master's thesis

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

Hanna Waage Hjelmeland

Safety validation of marine collision avoidance systems using Adaptive Stress Testing

Master's thesis in Cybernetics and Robotics
Supervisor: Anastasios M. Lekkas
Co-supervisor: Ole Jacob Mengshoel
June 2022

NTNU
Norwegian University of Science and Technology

Hanna Waage Hjelmeland

Safety validation of marine collision avoidance systems using Adaptive Stress Testing

Master’s thesis in Cybernetics and Robotics
Supervisor: Anastasios M. Lekkas
Co-supervisor: Ole Jacob Mengshoel
June 2022

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

NTNU

Norwegian University of
Science and Technology

Safety validation of marine collision avoidance systems using Adaptive Stress Testing

Hanna Waage Hjelmeland

Master thesis
Supervisor: Anastasios Lekkas
Co-supervisor: Ole Jacob Mengshoel
Industrial co-supervisors: Bjørn-Olav Holtung Eriksen and Øyvind Smogeli
Submission date: June 5, 2022

Department of Engineering Cybernetics
Norwegian University of Science and Technology

Abstract

Maritime Autonomous Surface Ships (MASS) have the potential to contribute to a more flexible urban mobility system with reduced emissions of greenhouse gasses. To enable marine autonomy, the systems have to be thoroughly assessed with regard to safety. Because autonomous systems are comprised of several layers of complex systems which perform reasoning and decision making, traditional safety validation methods become insufficient to test the system. To address this problem, recent research has called for new methods to perform intelligent simulation-based safety validation of MASS.

Several methods have been proposed to automatically identify challenging scenarios for MASS, to reduce the number of necessary test scenarios and perform sufficiently exhaustive testing. However, the author has not found any methods for use in marine autonomy, which address how the simulation of the scenario evolves. However, such methods have been applied to autonomous vehicles and aircraft systems by applying methods such as Adaptive Stress Testing (AST). AST is a simulation-based method which uses reinforcement learning to perform searches for failures in simulations of the system under test. Failures are found by injecting a sequence of disturbances and gradually learning the most efficient and likely ways to disturb the system into failure. This work proposes the use of AST as a step in the safety validation process of MASS. To demonstrate the method, AST is applied to two different Collision Avoidance (COLAV) strategies in simulations of the autonomous passenger ferry milliAmpere.

AST constitutes a highly flexible method, as it can be adjusted and adapted to the domain and the specific purpose. Several such adaptations are proposed in the thesis, to increase the relevance of the method to the maritime domain. The results demonstrate the potential of AST in the safety validation process of MASS, as many interesting failures are identified that uncover potential aspects of the different COLAV systems which can be subject to improvement. Further research should seek to improve the implementation of the method and to combine AST with scenario generation methods.

Sammendrag

Maritime Autonome Overflate-Skip (MAOS) har potensiale til å bidra til en mer fleksibel urban mobilitetsløsning med reduserte utslipp av drivhusgasser. For å muliggjøre maritim autonomi må systemene gjennomgå grundig sikkerhetsvalidering. Fordi autonome systemer består av flere lag med komplekse systemer som utfører vurdering og beslutningstaking, er ikke tradisjonelle metoder for sikkerhetsvalidering tilstrekkelig for å teste systemene. For å adressere dette problemet har nylig forskning etterlyst nye metoder for å utføre intelligent simuleringsbasert sikkerhetsvalidering av MAOS.

Det er foreslått flere metoder for å automatisk identifisere scenarier som er utfordrende for MAOS å navigere i, for å redusere antall nødvendige testscenarier og utføre tilstrekkelig fullstendig testing. Forfatteren har imidlertid ikke funnet noen metoder for bruk i maritim autonomi, som tar for seg måten simuleringen av scenariet utvikler seg. Slike metoder har blitt brukt på autonome kjøretøy og flysystemer ved bruk av metoder som f.eks Adaptive Stress Testing AST. AST er en simuleringsbasert metode som bruker forsterkende læring til å utføre søk etter feil i simuleringer av systemet som testes. Feil blir funnet ved å injisere en sekvens av forstyrrelser og gradvis lære den mest effektive og sannsynlige måten å forstyrre systemet til å svikte. Dette arbeidet foreslår bruk av AST som et trinn i sikkerhetsvalideringsprosessen til MAOS. For å demonstrere metoden brukes AST i tester av to forskjellige kollisjonsunngåelses-strategier i simuleringer av den autonome passassjerfergen milliAmpere.

AST utgjør en svært fleksibel metode, da den kan justeres og tilpasses domenet og det spesifikke formålet. Oppgaven foreslår flere slike tilpasninger for å forbedre metoderelevansen for det maritime domenet. Resultatene viser potensialet til AST i sikkerhetsvalideringsprosessen til MAOS, ved at mange interessante feil blir identifisert, hvilke avdekker potensielle aspekter ved de forskjellige kollisjonsunngåelse-systemene hvor det er forbedringspotensiale. Videre forskning bør utføres for å forbedre metoden og for å kombinere AST med scenario-genereringsmetoder.

Preface and Acknowledgements

This master thesis was written in the spring of 2022 at the Norwegian University of Science and Technology (NTNU) at the Department of Engineering Cybernetics. The thesis is written in an industrial collaboration with Zeabuz, a Norwegian company that springs out of NTNU (ZEABUZ, 2021).

The reader is expected to have basic knowledge about data science, mathematics, physics and control theory. The theory behind the methods used in the thesis will be shortly introduced. The figures are made by the author, mainly by use of the illustration tool Lucidchart, unless specified otherwise.

The thesis aims to implement the AST method in simulations of MASS. The implementation of the AST method relies on the open-source AST Python Toolbox (Koren, Ma, et al., 2021) and the Python milliAmpere Collision Avoidance (COLAV) simulator developed by Bjørn-Olav Holtung Eriksen at Zeabuz. Computer equipment used to perform the experiments was provided by the Department of Engineering Cybernetics and Zeabuz.

The thesis has been written under the supervision of prof. Anastasios M. Lekkas, who has provided me with insights into both the field of marine autonomy and deep learning applications. Having such an engaged and optimistic supervisor has been a blessing, and the thesis would not have been the same without his help. I have also been lucky to receive supervision from prof. Ole J. Mengshoel from the institute of computer science and informatics, who has provided great assistance in the use of the AST method, which he was a part of developing. Furthermore, I have to thank my two industrial co-supervisors from Zeabuz, namely Bjørn-Olav H. Eriksen and Øyvind Smogeli, who have provided great industry and field-related insights. I also want to thank the rest of the team at Zeabuz for allowing me to work there part-time while writing the thesis, which has accelerated my learning process in the field of maritime autonomy. I am very grateful for this opportunity, as well as the opportunity to continue the work on maritime autonomy as I start working at Zeabuz full in the fall of 2022.

I also want to thank all of the people who have supported me in the work of this thesis, as well as throughout the entire master's degree. I would not have been able to deliver my master's degree today, had it not been for the support of my family, whom I have always been able to phone home to when the work and exams were pressing. Also, I want to thank my friends, fellow students and cohabitants in our collective at Loftet, who have made the years in Trondheim fantastic. I also want to appreciate NTNU and the Norwegian welfare model, which in combination have provided me with the opportunity to acquire affordable education of great quality. Writing this, I feel truly humbled and moved by all the people and systems in place which has enabled me to deliver this thesis today. Lastly, I want to thank my boyfriend Alexander for all his support and kind words, and for making me laugh even when I was at my most stressed.

Hanna Waage Hjelmeland June 5, 2022

Contents

List of Figures

Acronyms

MDP Markov Decision Process. 1, 13, 22, 23, 40, 43, 93

RL Reinforcement Learning. 1, 5, 11–17, 21–25, 38, 40, 44, 67, 70, 72, 77, 79, 81–85, 90, 91

DRL Deep Reinforcement Learning. 1, 9, 17, 20, 23, 40, 64, 67, 70, 72, 93

TRPO Trust Region Policy Optimization. 1, 17, 20, 40, 93, 94

PPO Proximal Policy Optimization. 1, 93

AST Adaptive Stress Testing. i, iii, viii, ix, 1, 4, 5, 7, 9, 10, 21–26, 37–47, 49, 50, 52, 55–57, 59, 64, 65, 67, 68, 70, 72, 76, 78–80, 82, 84, 85, 87, 88, 90–94

SUT System Under Test. 1, 5, 22, 24, 25, 50, 92

DAST Differential Adaptive Stress Testing. 1

MCTS Monte Carlo Tree Search. 1, 23, 40

LOS Line Of Sight. 1, 28, 29, 31, 32, 35, 42, 53, 54, 56

COLAV Collision Avoidance. i, viii, 1, 5, 6, 8–10, 30, 31, 35, 37, 39–44, 47–49, 51, 52, 56, 59–64, 74, 79, 88, 90, 93

SP-VP Single Path Velocity Planner. viii, 1, 9, 30, 37, 41–44, 47, 48, 51, 52, 67–69, 73–75, 78, 79, 81

GAE Generalized Advantage Estimation. 1, 17–19, 39, 40

USV Unmanned Surface Vehicle. 1

ASV Autonomous Surface Vehicle. 1

SCS Safety Critical Systems. 1

ANN Artificial Neural Networks. 1

NN Neural Networks. 1, 11, 12, 17, 40

ACAS X Airborne Collision Avoidance Systems. 1, 5

TCAS Traffic Collision Avoidance Systems. 1, 5

DOF Degrees of Freedom. 1, 26–28, 41, 45, 53

NED North East Down. 1, 26, 28, 32

PD Proportional–Derivative. 1

PID Proportional–Integral–Derivative. 1, 41

COLREGs Convention on the International Regulations for Preventing Collisions at Sea. 1, 8–10, 31–35, 37, 44, 47, 48, 52, 56–61, 64, 79–82, 86, 88–91, 93, 94

NTNU Norwegian University of Science and Technology. v, 1

COM Center Of Mass. 1, 26, 46

DTW Dynamic Time Warping. 1, 35, 36, 38, 69, 79

ML Machine Learning. 1, 11, 12, 91

DL Deep Learning. 1

KL Kullback-Leibler. 1, 20, 21

TD Temporal Difference. 1, 19

SGD Stochastic Gradient Descent. 1

MPC Model Predictive Control. 1, 9, 10, 31, 35, 37, 52, 53, 55–57, 59–64, 80, 81, 83, 88, 90

DP Dynamic Positioning. 1, 41

GBDT Grammar Based Decision Trees. 1, 37, 79

MASS Maritime Autonomous Surface Ships. i, v, 1, 5, 6, 9, 93

MAOS Maritime Autonome Overflate-Skip. iii, 1

HIL Hardware in the Loop. 1, 7

SIL Software in the Loop. 1, 7

MIL Model in the Loop. 1, 7

AI Artificial Intelligence. 1, 7

FAA Federal Aviation Administration. 1, 5

RSS Responsibility-Sensitive Safety. 1, 25, 47, 48

ROC Region of Collision. 1, 30

HPR High Penalty Region. 1, 30

LPR Low Penalty Region. 1, 30

SITAW Situational Awareness. 1, 6, 79

SPA Sense-Plan-Act. 1, 6

AIS Automatic Identification System. 1, 8, 92

LOA Level of Autonomy. 1, 5, 6

Chapter 1

Introduction

1.1 Motivation

The primary motivation behind this thesis is the emerging need for thorough simulation-based safety assessment methods which are required to produce a trustworthy and robust autonomous system, which again can enable a mobility transition towards smarter, emission-free solutions designed for increasingly densely populated cities.

Climate change is widely acknowledged as one of the main problems faced by the world's population, and it will likely continue to pose a challenge for future generations. Strategies to prevent substantial consequences of climate change are addressed in different manners throughout the world's communities. In 2020, The European Union approved the European Green Deal, constituting a set of policy initiatives to make the EU carbon neutral by 2050. The deal also poses a set of ambitious short term goals for 2030, with a 55% net decrease in greenhouse gas emissions compared to 1990 levels. In 2021, this ambition was adopted into a law regulation, making it a legal obligation for the EU to reach the specified goals (EU, 2022).

While the world is battling the issues related to climate change, the continuing urbanisation of the world's cities is putting a toll on the urban mobility sector. Population growth is mainly occurring in urban areas, and this trend is likely to continue (Satterthwaite, 2009). In 2018, approximately 55% of the world's population was situated in urban areas and this is expected to grow to 68% by 2050 (UN, 2018). Even though the urbanisation is not the main source of growth in greenhouse gas emissions (Satterthwaite, 2009), it is putting a toll on the mobility sector of the densely populated areas as it causes congestion, noise pollution and a need to expand mobility infrastructure (Hildermeier et al., 2014).

As the need for more intelligent mobility solutions in densely populated areas increases, there is also an urgent need to reduce the greenhouse gas emissions from this sector. The mobility sector constitutes a significant part of the world's emissions, as urban mobility accounts for 40% of all CO2 emissions from road transport and up to 70% of other pollutants from transport (Tsavachidis et al., 2022). The European Green Deal states the vision to reduce transport-related greenhouse gas emissions by 90% within 2050. Tsavachidis et al. (2022) states that

"The EU's decarbonisation objectives cannot be realised without a sustainable urban mobility transition" (Tsavachidis et al. (2022))

Thus, there is a need for innovation in both the technology and the structure of the urban mobility to reduce greenhouse gas emissions and to accommodate for the continuing urbanisation. The need for more innovative mobility solutions has caused a gained interest and rapid increase in the research of autonomous vehicles, with applications in many fields and industries such as automobiles, aviation and agriculture (Faisal et al., 2021). In recent years, autonomy has also become prominent in the marine industry. Examples include Kongsbergs YARA Birkeland, which is set out to be the world's first zero-emission autonomous container ship, with the potential to replace 40 000 journeys of diesel-powered trucks (*Yara Birkeland | Yara International* 2021), shown in Figure 1.1(a). In Amsterdam, researchers have developed the *Roboat*, an autonomous boat that can carry up to five people, collect waste, deliver goods and provide on-demand infrastructure in the Amsterdam canals (*One autonomous taxi, please* 2021), see Figure 1.1(b).

(a) The YARA Birkeland zero-emission autonomous ship. Image obtained from *Yara Birkeland | Yara International* (2021).

(b) The Roboat, an autonomous boat set to sea in Amsterdam in October 2021. Image obtained from *One autonomous taxi, please* (2021).

Both these projects exemplify innovations which utilise a resource that for years has been underutilised: the waterways. Due to the historical significance of waterways, many large cities are situated close to the sea or along rivers. Moreover, about 40% of the world's population lives near a coast (Reddy et al., 2019). Waterways are today merely seen as an obstacle to urban mobility as transportation across them requires expensive and resource-demanding solutions such as bridges or manned vessels. To further utilise the waterways for urban mobility, research have been conducted to enable autonomous passenger ferries for use in cities. Reddy et al. (2019) emphasise the promising aspects of using autonomy to revitalise the urban ferries, as it can reduce operational costs and thereby increase the economic viability. In addition, autonomous waterborne transportation systems can also improve traffic capacity and efficiency on the waterways Wei et al. (2021). By using autonomous ferries, cities can thus potentially reduce both the environmental and economic costs of expanding infrastructure by building bridges while simultaneously increasing urban mobility. A prototype of such an autonomous ferry called *milliAmpere* was constructed at NTNU as part of a pilot project called Autoferry, which has been further developed for commercial use by the start-up company Zeabuz. A simulator of the milliAmpere vessel, shown in Section 1.1, is used as a case study in this thesis.

(a) Zeabuz ferry design example. Image obtained from (ZEABUZ, 2021)

(b) The milliAmpere. Image obtained from (*Autoferry - NTNU* 2021).

Arguably, autonomous zero-emission marine vehicles have the potential to both enable new urban mobility solutions and reduce emissions. However, the technology that enables autonomy is both new and complex, and autonomous solutions have yet to gain traction in public opinion. Research on people's attitudes toward autonomous automobiles shows e.g. that there are substantially more people completely hostile to autonomous cars than those who are totally in favour (Hudson et al., 2019). Due to the lack of transparency in the decision making in these systems, there is a rising suspicion to the increasing implementation of them (Ebert et al., 2019). Even though many believe that autonomy will provide safer solutions by eliminating human errors, autonomous technology has to prove itself as highly robust and significantly outperforming human operators. Ebert et al. (2019) state:

> "To build trust, we need a level of quality at least one order of magnitude higher than human-operated systems." (Ebert et al. (2019))

A system intended for autonomy often consists of many complex modules, comprised by elements such as *machine learning*, computer vision or intricate reasoning techniques. In these systems, the actual decision making is often hidden or hard to grasp for humans. Safety guarantees are therefore hard to obtain as it is often impossible to prove the technology for all possible combinations of input states. Thus, tools for thorough safety assessment must be in place, especially when autonomy is deployed in safety-critical systems such as moving vessels, particularly those designed to transport humans. (Ahvenjärvi, 2016) argues that the most important aspect of the development of fully autonomous ships is the safety aspect.

Traditional safety validation methods are considered by many to be unfit to cover the complexity of these new autonomous systems. Ebert et al. (2019) state that

> "To achieve dependability and trust [of autonomous systems], we need dedicated, intelligent validating techniques that cover, for instance, dynamic changes and learning." (Ebert et al. (2019))

Currently, there are no official safety certification strategies designed for autonomous vehicles, although it has been called for by industry and researchers (Koopman et al., 2017). In e.g.

the automobile industry, the current safety standards only regulate automobile safety-related components without consideration of driving intelligence in completing driving tasks (Feng et al., 2021). Koopman et al. (2017) state that such a strategy must address the cross-disciplinary concerns of safety engineering, hardware reliability, software validation, robotics, security, testing, human-computer interaction, social acceptance, and a viable legal framework. There are, however, ongoing processes in place to establish strategies to ensure proper regulation of autonomous ships, as e.g. described in Ringbom (2019) and IMO (2021).

Because it is hard to perform extensive tests of autonomous systems, it is becoming evident to many autonomy developers that an integral part of the safety validation process of autonomous systems must include simulation-based testing (Pedersen et al., 2020). Important actors in the development of autonomous vehicles such as the American company Waymo, state in safety reports that simulation-based testing is essential for the development and safety validation of their cars (Waymo, 2021). Moreover, as autonomy technology matures and express less faulty behaviour, simulations are especially important in the process of obtaining information about the failure modes of the system, as failures are hard and cost demanding to provoke in real-world testing. The international classification company DNV, which develops policies for safety certification for ships, state in a report from 2018 that

> "The verification procedures for the control systems [of autonomous vessels] can be based on physical and on simulator-based verification for software intensive systems, but since requirements for automatic fault tolerance increase, simulator-based verification would be more efficient in proving a large set of different fault scenarios." (Vartdal et al. (2018))

The former statements indicate the need for simulation-based safety validation methods that tackle the complexity of the autonomous systems, especially methods for identifying failure modes. One such method is AST, a simulation-based method which applies machine learning to find ways of bringing the system under test to failure and return the most likely failure scenarios. In this thesis, AST is applied to identify potential failure modes of the autonomous passenger ferry prototype milliAmpere.

In summary, this thesis is motivated by the potential of zero-emission autonomous marine vehicles in reducing emissions and enabling smart urban mobility for densely populated cities. By illustrating how the simulation-based safety validation method AST adds value to the safety analysis of the milliAmpere ferry, the thesis attempts to contribute to the development of robust safety validation methods for autonomous marine vessels, which is a crucial step in enabling the implementation and commercialisation of such vessels.

1.2 Background and related work

This section presents an overview of the background concepts built upon by the thesis, as well as related work. The concept of AST is accounted for together with some background information on autonomous systems and the safety validation process of such systems.

1.2.1 Adaptive Stress Testing

AST was first proposed in 2015 as a simulation-based testing method to obtain the most likely sequence of states that lead to failure of a system (Lee, Kochenderfer, Mengshoel, Brat, et al., 2015). The method formulates the problem of bringing the system to failure as an Reinforcement Learning (RL) problem and uses RL methods to optimise for failures. The system applies disturbances to the system it is set out to test, referred to as the System Under Test (SUT), and learns how to most effectively disturb the system such that a failure occurs in an iterative manner.

The method was applied to a prototype of a new COLAV system for aircrafts called Airborne Collision Avoidance Systems (ACAS X) proposed by the Federal Aviation Administration (FAA), which was set out to replace the at the time dominating aircraft COLAV system called Traffic Collision Avoidance Systems (TCAS). ACAS X was intended to bring several improvements to the aircraft COLAV system and reduce both risk of collisions and the number of unnecessary alerts (Lee, Kochenderfer, Mengshoel, Brat, et al., 2015). AST was used to perform differential studies of TCAS and ACAS X, which contributed to the acceptance of ACAS X to replace TCAS in 2018 (Lee, Mengshoel, et al., 2020).

Since the method was proposed, it has been extended and applied to test other domains and technologies. AST only requires a simulator of the SUT, and some basic interfacing functionality, which makes the method highly flexible such as information about whether or not a failure has been detected. Systems that have been tested using AST include the driver model of a car approaching a crosswalk with pedestrians (Koren, Alsaif, et al., 2018; Corso, Du, et al., 2019; Koren and Kochenderfer, 2019; Koren and Kochenderfer, 2020), a trajectory planning system of an aircraft (Robert J. Moss et al., 2020), a financial environment in search of fraud (*Adaptive Stress Testing for Adversarial Learning in a Financial Environment* 2021) and a neural net controller for an aircraft model (Julian et al., 2020).

The theoretical foundation and the workings of AST are elaborated on in Chapter 2 and Chapter 3.

1.2.2 Autonomous systems

The term *autonomous*, directly translated from Greek to mean *self-governing*, is a widely used term that is sometimes used inaccurately. Many of these systems are better described as hybrids, with some autonomous functionality and some functionality which is governed by operators. To distinguish between such systems and how close they are to being fully autonomous, the concept of Level of Autonomy (LOA) has been introduced into the different fields where autonomy is applied. UK (2018) describe the LOA for MASS as in Table 1.1.

Level	Name	Description
0	Manned	Vessel/craft is controlled by operators aboard
1	Operated	Under Operated control all cognitive functionality is controlled by the human operator. The operator has direct contact with the Unmanned Vessel over e.g., continuous radio and/or cable. The operator makes all decisions, directs and controls all vehicle and mission functions.
2	Directed	Under Directed control some degree of reasoning and ability to respond is implemented into the Unmanned Vessel. It may sense the environment, report its state and suggest one or several actions. It may also suggest possible actions to the operator, such as e.g. prompting the operator for information or decisions. However, the authority to make decisions is with the operator. The Unmanned Vessel will act only if commanded and/or permitted to do so.
3	Delegated	The Unmanned Vessel is now authorised to execute some functions. It may sense environment, report its state and define actions and report its intention. The operator has the option to object to (veto) intentions declared by the Unmanned Vessel during a certain time, after which the Unmanned Vessel will act. The initiative emanates from the Unmanned Vessel and decision-making is shared between the operator and the Unmanned Vessel.
4	Monitored	The Unmanned Vessel will sense environment and report its state. The Unmanned Vessel defines actions, decides, acts and reports its action. The operator may monitor the events.
5	Autonomous	The Unmanned Vessel will sense environment, define possible actions, decide and act. The Unmanned Vessel is afforded a maximum degree of independence and selfdetermination within the context of the system capabilities and limitations. Autonomous functions are invoked by the on-board systems at occasions decided by the same, without notifying any external units or operators.

Table 1.1: LOA for MASS.

The system's ability to reason about its environment and make well-informed decisions has to increase in order for the LOA to increase. For autonomous marine vehicles, LOA 3-5 requires the vessel to be able to both sense the environment and be able to reason about it, create a plan of actions, and execute the actions. The structure of these operations has been thoroughly described in different software architectures, resulting in three main architecture paradigms. The first software paradigm for autonomous systems was the Sense-Plan-Act (SPA), which was later replaced by the *subsumption* paradigm, and subsequently the *hybrid deliberative/reactive* paradigm, which is the current software architecture paradigm. The former paradigms are not elaborated on but are described further in Gat (1998). The hybrid deliberative/reactive paradigm includes architectures which are either deliberative, consisting of long-term high-level planning, or reactive, which is based on directly utilizing sensory information for short-term low-level commands (Pirjanian et al., 2000).

Autonomous vehicles or vessels which are supposed to navigate in environments consisting of other elements such as pedestrians or other vessels, need a COLAV system to keep from collisions. COLAV systems can be both deliberate or reactive; deliberate by incorporating aggregated so-called Situational Awareness (SITAW) information to plan longer-term COLAV strategies, or reactive by considering current sensory information to employ short-term motion planning (Eriksen et al., 2019). A complete COLAV system may be a hybrid of both a deliberate and a reactive COLAV system, utilizing the computational efficiency of the short-term COLAV for e.g. unexpected events and the high-level features of the long-term COLAV. This thesis performs experiments using two deliberate COLAV algorithms for MASS, which perform path planning in separate ways.

1.2.3 Safety validation of autonomous systems

Safety validation is the process of ensuring the correct and safe operation of a system operating in an environment (Corso, Robert J. Moss, et al., 2021). The term *validation* is sometimes confused with the related term *verification*, and different definitions of the two exist. For the scope of this thesis, the two terms are defined as in Corso, Robert J. Moss, et al. (2021). Validation is thus referred to as the process of making sure the product works right, by performing e.g. physical tests or numerical simulations. In contrast, verification refers to a more static kind of testing which ensures that the product is the right product for its purpose by testing of the design to make sure it complies with regulations and specifications.

Methods for verification of autonomous systems do exist, such as formal verification by the use of automated theorem proving, as described in Foster et al. (2020). However, these methods rely on thorough mathematical models of the systems which can be considered in all possible scenarios. Thus, they typically scale poorly to large problems or problems where a sufficiently describing model of the system and its environment cannot be obtained (Corso, Robert J. Moss, et al., 2021). Examples of such systems are *black-box* systems, where the system's internal workings are hidden or too complex to model, or systems designed to interact with an environment that cannot be modelled due to stochasticity, such as real-world applications. Therefore, the use of safety validation by rigorous system testing is typically relied on for systems of higher complexity dedicated for use in stochastic, real-world scenarios.

Traditional methods for safety validation of autonomous systems include fault injection, functionality based testing, Software in the Loop (SIL), Model in the Loop (MIL), Hardware in the Loop (HIL) and brute force (Ebert et al., 2019). The methods will not be elaborated on, but further descriptions can be found in Ebert et al. (2019), who further describe four categories of validation techniques for autonomous systems, depicted with examples in Table 1.2:

Validation handling	Automatic	- Simulation environments: MIL, HIL, SIL	- Simulation environments MIL, SIL - Brute-force usage in real world, while running realistic scenarios - Intelligent validation: e.g., congnitive testing, AI testing
	Manual	- Function test - Fault injection - Negative requirements: misuse, abuse, confuse cases - FMEA, FTA - Simulation environments: MIL, HIL, SIL	- Experiments and empirical test strategies - Simulation environments MIL, SIL - Brute-force usage in real world, while running realistic scenarios - Specific quantity requirements, e.g., penetration, testing and usability
		White box	black-box
		Validation strategy	

Table 1.2: Validation approaches for autonomous vehicles as described in Ebert et al. (2019).

The work of this thesis falls under the category of automated black-box testing, as highlighted in Table 1.2, as the Artificial Intelligence (AI) based testing approach AST is applied.

The problem of testing autonomous systems can furthermore be divided into three parts, namely:

- Test scenario generation: How are the initial conditions such as initial positions, velocities and the path reference of the vessels chosen?
- Episode evolution: How do the scenarios evolve? How do the system elements, such as adversary vessels, behave throughout the scenario? Are there any disturbances applied, and if so, how are they generated?
- Test scenario evaluation: How is the scenario evaluated? Which measures are used to quantify the system behaviour in the scenario?

where the terminology is adapted from Pedersen et al. (2020). A specific *scenario* describes the details of the test situation, such as initial positions, velocities and waypoints, and an *episode* refers to a single simulation in the particular scenario. The term *adversary vessel* describe other vessels within the environment, and the term *own ship* will further be used to describe the ship under test.

For the specific application of autonomy in marine vessels, a comprehensive simulation-based testing system is described in Pedersen et al. (2020). The paper describes different elements required for thorough simulation-based testing of autonomous navigation systems, and argues for the use of both traditional methods and systematic simulation-based testing. Based on a proposal of DNV to apply a digital twin for testing, the article describes a comprehensive prototype of a test system. The paper calls for methods to perform automatic generation of challenging scenarios based on some evaluation criteria, as this will limit the search space in which the system has to be tested to provide sufficient confidence in the system. Similar scenario generation techniques have been developed for autonomous automobile systems, as described in e.g. Althoff et al. (2018); Tang et al. (2021). Since the publication of the paper, several scenario generation techniques have been proposed and applied to marine vessels, with corresponding scenario evaluation methods. The evaluation methods are often based on measures which quantify the own ship's ability to avoid risk, comply with the mission and comply with the Convention on the International Regulations for Preventing Collisions at Sea (COLREGs), which is a set of navigation rules for marine vessels that will be elaborated on later in the thesis. Some of the recent proposals of scenario generation methods are:

- Torben et al. (2022) proposed an automatic simulation-based testing method for autonomous ships, by using formal logic to specify a set of requirements to test against, using COLREGs compliance, a safety measure and mission compliance. A Gaussian Process (GP) model was used to predict the performance of the vessel over the entire parameter space. The system incrementally runs new simulations until the entire parameter space of the test case is covered to the desired confidence level, or until a case which falsifies the requirement is identified.
- Bolbot et al. (2021) proposed an automatic scenario generation method for marine COLAV systems which uses a geometrical risk measure to evaluate scenarios, and sampling techniques to generate encounter scenarios.
- Bakdi et al. (2021) proposed an automatic scenario generation model for testing autonomous ships with a big data approach using Automatic Identification System (AIS) data, a common data source in maritime transportation where ships exchange traffic information.

- Porres et al. (2020) proposed an automatic scenario generation system for ship COLAV systems using a scenario generator that generates scenarios with one adversary vessel, determining the initial positions, velocities and one waypoint for each vessel to follow. The problem of generating test cases is optimised with Deep Reinforcement Learning (DRL) for scenarios where it is challenging for the own ship to comply with the COLREGs.

While all of the aforementioned research propose scenario generation methods, only a couple of them address ways to affect the episode evolution. The method proposed in e.g. Porres et al. (2020) only provides scenario descriptions with an own ship and one adversary vessel, with initial positions, velocities and one waypoint for each of the vessels to follow. In Torben et al. (2022), the episode evolution is addressed in some of the test cases, by introducing a predetermined number of changes in the course of the adversary vessel, which was varied between the scenarios to search for collision.

The work in this thesis addresses only the episode evolution, as AST is implemented to affect the behaviour of the adversary vessels throughout the episode. Thus, it differs from the aforementioned methods as the adversary vessels are not set up to either follow straight-line paths or change course at some predefined time. By the use of AST, the adversary vessels in this work perform continuous navigation manoeuvres throughout the episode, which allows for simulations with very diverse adversary behaviour. Using AST, we attempt to find several episodes where own ship collides with the adversary, to obtain information not only about scenarios in which the vessel possibly *can* collide but several descriptions of *how*. For the COLAV systems of marine vessels, the author has yet to find proposed methods which address the evolution of the episode in this manner. Furthermore, the author considers this kind of method to also be a necessity in a complete validation process of autonomous marine vehicles, as information about failure modes is important and hard to obtain in real-life tests as they pose significant cost and potential damage to the physical system.

The test scenarios used in this work are chosen manually as situations in which the own ship and the adversary vessel cross or pass each other and where there is a risk of collision. It is left for further research or commercial efforts to combine test scenario generation methods with the evolution-based method implemented here.

1.3 Research Questions

The work on this thesis builds upon work conducted in the previous semester, presented in (Hjelmeland, 2021), where the AST method was applied and adjusted to the maritime setting to test the COLAV system of the milliAmpere ferry called SP-VP. The work in this thesis aims to answer the research questions defined as follows:

RQ1 : Can the AST implementation be further adjusted to the MASS domain and the specific case study of the SP-VP, by e.g. incorporating domain-based knowledge such as the COLREGs, to obtain failures which uncover critical aspects of the SP-VP COLAV system?

RQ2 : Can AST be applied to find failure modes in a more dynamic COLAV system such as a Model Predictive Control (MPC) based controller, with COLREGs compliance?

RQ3 : Can AST be applied to MASS systems to find failure modes in scenarios with multiple adversary vessels?

1.4 Contributions

The thesis is divided into two parts, with the following main contributions:

Part 1:

- Several augmentations are proposed to the problem formulation presented in Hjelmeland (2021), to obtain failures where the ferry is more involved in the cause of the collision. The results are promising, as failures are identified where the cause of collision can be credited to e.g. errors in the ferry's estimation of the adversary.
- A clustering technique called soft-Dynamic Time Warping k-means clustering is applied to categorise the results based on the shapes of the adversary movement. The use of clustering aids the analysis when comparing the results from the different problem variations.

Part 2:

- A MPC based COLAV approach with COLREGs compliance is implemented in the Zeabuz COLAV simulator to enable testing of a COLAV system which performs COLREGs compliant manoeuvres to avoid collision.
- AST is applied to the MPC approach using the problem variation which yielded the best results in part 1. Three test scenarios are chosen, which all relate to a specific rule of the COLREGs.AST is set up to optimise for collisions between the ferry and the adversary vessel where the ferry violates the COLREGs prior to collision. The results are promising, as AST identifies various episodes where the MPC controller makes questionable decisions which lead to a collision.
- The test scenarios are extended to include multiple adversary vessels. AST simulations are performed where AST is set up to control all vessels. The results show that AST can uncover more failure modes by the use of several adversaries, and the failures show interesting scenarios where AST uses the different adversaries in cooperation to induce COLREGs violations in the ferry and cause collisions.

Furthermore, the results are thoroughly discussed, and suggestions for possible improvements and further research are provided. Combined results from work done in Hjelmeland (2021) and this thesis resulted in a paper currently being reviewed for submission to the 14th IFAC Conference on Control Applications in Marine Systems, Robotics and Vehicles, presented in Hjelmeland et al. (2022).

Chapter 2

Theoretical background

This background chapter presents the theory behind the methods applied in the thesis. The chapter builds upon the background presented in (Hjelmeland, 2021), as well as basic knowledge about mathematics, physics and computer science. Some of the topics described here are elaborated on further in (Hjelmeland, 2021), but these chapters will provide a summary of the most important details and elaborate on topics that are in focus in this work.

2.1 Machine Learning and Deep Learning

Machine Learning (ML) has revolutionised methods of performing tasks in many fields, such as object detection and classification in images, speech processing and translation as well as search optimisation (LeCun et al., 2015). Machine learning has been defined in numerous ways. Tom M. Mitchell gave a fitting definition of the concept:

> A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. (Mitchell, 1997)

ML methods are often divided into three main categories: *supervised*, *unsupervised* and *RL*. Supervised machine learning comprises machine learning methods where training is performed in a supervised manner, using a training set consisting of training examples together with the corresponding desired network output. The neural network, which will be further described in this section, is a typical example of a supervised machine learning framework. Unsupervised methods are methods where the training data is not labelled with a specific output, but the ML method has to categorise the data using patterns it discovers. An example of an unsupervised method is the concept of *clustering*, where data is divided into clusters using some sort of similarity measure. Clustering will be elaborated on in Section 2.7. RL methods are methods where the ML model learns not only by looking at data, but by examining and possibly navigating through an environment and obtaining rewards based on its behaviour. RL will be further discussed in Section 2.2.

A typical supervised machine learning structure is the Neural Networks (NN), which constitutes a mathematical imitation of the neural network of the brain. The NN consists of layers of weighted *nodes*, as illustrated in the example network topology shown in Figure 2.1. The nodes receive input from the nodes of the former layer and output a nonlinear conversion of the input, in a similar fashion to the brain's neurons. The NN is typically trained using backpropagation methods, a group of algorithms used to adjust the network weights. The NN is trained in iterations by running training samples through the network and comparing the output to a ground truth. This comparison is made using a predefined loss function, which results in a loss value used to adjust the values of the network weights through backpropagation. Backpropagation methods change the network weights in a direction and magnitude proportional to the negative gradient of the loss function, and they do so efficiently by calculating the derivatives using the chain rule, one layer at a time. In this way, intermediate calculations are avoided through dynamic programming, which allows for fast and stable training of NNs.

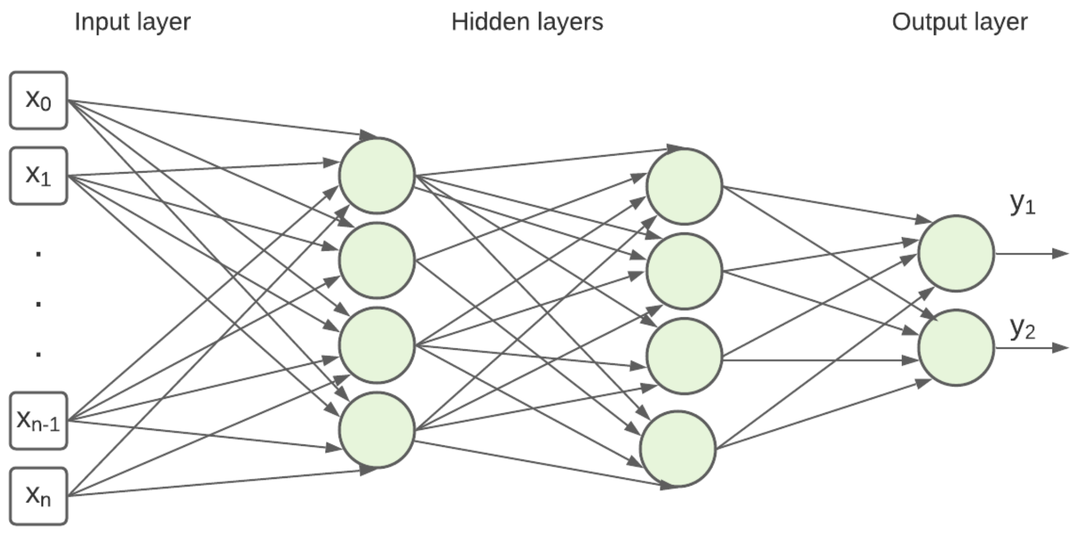


Figure 2.1: Example neural network topology with n inputs, two outputs and two hidden layers of size 4.

Deep learning is a subset of machine learning, where NNs are applied and consist of several layers (LeCun et al., 2015). The different layers are trained to detect various features at varying levels of abstraction of the data it is supposed to process. Take a multi-layer NN used in object detection in images, for instance: Gradually throughout training, first layer in the NN may become trained to detect edges in the image. The second layer becomes able to recognize arrangements of these edges, while the third layer is able to assemble these arrangements into recognizable parts of the object it is detecting (LeCun et al., 2015). The more layers, the more levels of abstraction. This makes multi-layer NNs suitable for complex tasks where different features of the problem must be identified and assembled for recognition. However, increasing the number of trainable parameters in the network comes with a price. The available data becomes more sparse when increasing the dimension of the NN, as summarized in Richard M. Bellmans term *curse of dimensionality* (Richard, 1957).

2.2 Reinforcement learning

RL is a ML framework where an agent navigates in an environment and collects rewards as a result of its behaviour and interaction with the environment. The objective in RL is to learn a behavioural strategy, a so-called *policy*, which gives the maximum total reward - not only in the following steps, but in a long-term perspective. Thus, the RL agent has to figure out the smart way to behave to gain short term rewards and a way to behave such that the long-term rewards are maximised.

RL mimics many of the human learning processes. For instance, we learn that even though eating a bunch of candy gives an instant high reward, eating a lot of it over a long time is a bad idea, as the long-term reward is negative. Through progressively worsened shape and stomach aches, we learn that candy consumption should be restricted to Saturday nights. In this way, we update our behaviour as a consequence of the feedback from our environment to balance short term rewards with long-term ones. The development of RL was highly inspired by the field of psychology and today the fields interact and learn from each other. Research of the human brain supports the existence of an RL-like mechanism involved in human decision-making processes (Niv, 2009) and RL-inspired psychological theories of optimising long-term returns have contributed to explanations of behaviour in humans and animals that were not previously well understood (Sutton et al., 2018).

2.2.1 Markov Decision Process (MDP)

An RL problem is mathematically formulated as a finite MDP (Sutton et al., 2018). A finite MDP consists of the tuple (X, U, P_u, R_u), specifying the workings of the RL agent $\mathcal{A}$ and its environment $\mathcal{E}$:

- X is the finite set of states the agent can be in, i. e. the *state space.*
- U is the finite set of actions the agent can take, i. e. the *action space.*
- $P_u(x, x')$=$P(X_{t+1} = x'|X_t = x, U_t = u)$ is the *transition function*, which specify the probability of a possible next state x' given a state-action pair (x, u).
- $R_u(x, x') = R(X_t = x, X_{t+1} = x', U_t = u)$ is the *reward function*, which specify the reward the agent obtains as a consequence of transitioning from state x to state x' due to the action u.

The RL problem is structured as an iterative interaction between the agent $\mathcal{A}$ and the environment $\mathcal{E}$, as depicted in Figure 2.2. The agent performs actions according to its current policy, denoted π, and observes the consequences of the action which include the reward r_{t+1} and the next state x_{t+1}. Both the reward and the next state are evaluated by the agent and used to update its policy π, from which the next action u_t is drawn.

2.2.2 Value functions

What is good behaviour? To learn optimal ways to behave, the RL agent needs some sort of metric to quantify the value of being in different states and taking possible actions. The agent receives rewards which can indicate how good or bad it is to be in the current state, but as the agent is supposed to maximise total reward, the value of the current state is also dependent on the possible rewards it can lead to later. The same goes for the value of an action in a given

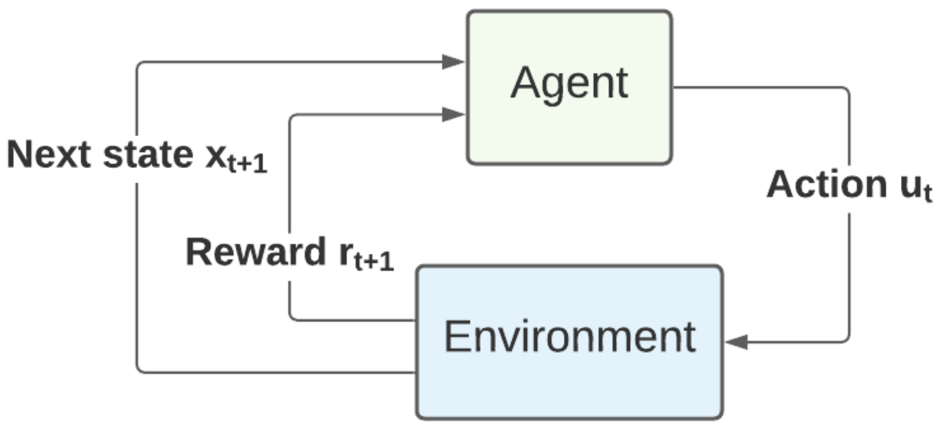


Figure 2.2: The structure of an RL problem. The agent acts according to its current policy and receives feedback from the environment, which it then uses to update its way of behaviour.

state; the value of the action is determined by the immediate reward it leads to, as well as the subsequent rewards the agent can obtain later as a consequence of the action. The two concepts indicated here are called the *value function* V^π and the *action-value* function Q^π, and they play an important role in most RL schemes.

The two value functions provide a metric for the expected sum of total rewards, starting in the current state x and following the current policy π thereafter. The value functions are mathematically formulated as:

$$V^\pi(x) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k}|X_t = x\right] \tag{2.1}$$

$$Q^\pi(x,u) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k}|X_t = x, U_t = u\right] \tag{2.2}$$

where r_{t+k} is the reward received at time step $t+k$ and the constant γ is a *discount factor* between $[0,1]$ that discounts rewards further into the future. The difference between V^π and Q^π is the inclusion of the effect of the action u in the action-value function Q^π. In the value function V^π, it is assumed that the agent will continue following the policy π and take the action given by $\pi(x)$, while the action-value function is able to evaluate the value of all possible actions from state x. The value functions equal one another when the action evaluated in the action-value function is the one chosen from the policy:

$$V^\pi(x) = Q^\pi(x, \pi(x)) \tag{2.3}$$

It is possible to use the difference between the value function and the action-value function to our benefit, as the numerical difference between them gives a measure of how good an action is, in comparison to sticking to the policy. This measure is named the *advantage function* A^π and is given as:

$$A^{\pi}(x,u) = Q^{\pi}(x,u) - V(x) \tag{2.4}$$

Thus, $A^{\pi}(x,u) > 0$ will indicate that it is advantageous to go for another action than the one provided by the current policy π. Naturally, this measure gives a good indication of how well the agent is utilizing its possible actions and it can thus be a good metric to use in RL training.

The total reward of the full trajectory is dependent on which states the agent will visit before the episode is over. The trajectory of the RL agent is often stochastic and thus unknown, but an estimate can be made based on the transition probabilities by weighing future rewards by the probability of the agent visiting them. The value functions can then be decomposed into a term describing the reward at the current step, and a recursive term where the value at the future steps are weighted by their probabilities, known as the *Bellman equations* (Sutton et al., 2018):

$$\begin{aligned} V^{\pi}(x) &= R_{\pi(x)}(x) + \gamma \sum_{x'} P_u(x'|x,\pi(x)) V^{\pi}(x') \\ Q^{\pi}(x,u) &= R_u(x) + \gamma \sum_{x'} P_u(x'|x,u) Q^{\pi}(x',\pi(x')) \end{aligned}$$

The optimal policy π^* is the policy which maximises total rewards. The value functions then become the *optimal value functions*, given as

$$\begin{aligned} V^*(x) &= \max_{\pi} \mathbb{E}_{\pi} \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | X_t = x \right] \\ Q^*(x,u) &= \max_{\pi} \mathbb{E}_{\pi} \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | X_t = x, U_t = u \right] \end{aligned}$$

where, from Equation (2.3), it follows that

$$V^*(x) = Q^*(x,\pi^*(x)) = \max_u Q^*(x,u)$$

2.2.3 Exploration vs exploitation

One of the challenges of RL is the trade-off between exploration and exploitation (Kaelbling et al., 1996). The trade off is based on the question of whether to exploit the rewards that the agent has already obtained by exclusively following the current policy, or to explore new paths with some level of randomness. Adding a level of exploration to the agent is usually constructive, as it prevents the agent from overseeing possibly smarter solutions by locking itself to the current policy.

2.2.4 RL algorithms

As the field of RL has grown, many RL algorithms have been proposed to enhance performance and tackle problems of different nature. The algorithms differ in many ways, including the details of the problems they can be applied to, such as

- Discrete or continuous action space. An example of a discrete action space is e.g. navigating in a grid such as in a maze, where the only allowed actions are up, down, left and right. An example of a continuous action space is found in control problems, where the actions can correspond to the actuation of different parts.
- Discrete or continuous state space. Discrete state spaces occur in problems where the possible states are distinct, such as in the game of chess. A continuous state space may be found in problems of e.g. real-world applications, such as navigation.

The algorithms also vary greatly in how they are constructed, and what measure they apply to learn better behaviour. Examples of this are:

- The value function or metric used to update the policy. In so-called *action-value* methods, one of the value functions is learned, and actions are chosen based on the value function estimate. In *policy-based* methods, the policy is learned and updated directly, with or without the help of a value function.
- Whether or not the policy used to generate the training data is the same policy as the one that is optimised. *Off-policy* methods use a proxy policy to generate the training data and use it to optimise another, optimal, policy. In *On-policy* methods, the policy that is followed during data collection is the same as the one being optimised (Sutton et al., 2018).

2.2.5 Policy Gradient Methods

Policy gradient methods are a group of policy-based methods where the policy π is parameterised by a parameter vector $\boldsymbol{\theta}$, and the parameters are updated through steps which approximate gradient ascent:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \widehat{\nabla J(\boldsymbol{\theta})} \tag{2.5}$$

where α is the step size and $\widehat{\nabla J(\boldsymbol{\theta})}$ is the stochastic estimate of the policy gradient (Sutton et al., 2018). This approach differs from other RL methods as the actions of the RL agent are not retrieved from a value function, but drawn directly from the learned parameterised policy.

Policy gradient methods often work well in problems where the action space is continuous, which is the case for the problems in this work. These methods also provide stronger convergence guarantees than action-value methods due to the smooth updates of the policy (Sutton et al., 2018).

The policy gradient methods are on-policy, meaning that the policy is the same in data collection and optimisation. When always following the current policy, measures must be made to ensure sufficient exploration, as consistently following the current policy would lead to no exploration. This issue is overcome by requiring that the policy is never deterministic, by adding some form of stochasticity to the policy output (Sutton et al., 2018). Although this ensures exploration, policy gradient methods typically become gradually less exploratory as the gradient step encourages exploiting. This can lead to sub-optimal convergence, which policy gradient methods are known for due to the non-convex nature of the optimisation problem (Bhandari et al., 2020).

The policy can be parameterised in any way as long as a policy gradient can be obtained, i. e. as long as the policy is differentiable with respect to the parameter vector $\boldsymbol{\theta}$. The DRL methods applied in this work use policies parameterised by NNs with the policy parameters $\boldsymbol{\theta}$ as the NN weights.

2.3 Deep Reinforcement Learning

In DRL, multi-layer NNs are combined with RL to tackle problems that previous RL methods could not overcome due to scalability issues for high dimensional problems. DRL is a powerful tool when dealing with complex, high dimensional function approximation and thus an efficient way of overcoming the curse of dimensionality (Arulkumaran et al., 2017).

This section presents the policy gradient method Trust Region Policy Optimization (TRPO), which can be used together with a multi-layer NN parameterised policy, constituting a DRL framework. Subsection 2.3.1 introduces some theoretical background which TRPO builds upon.

2.3.1 Background

Actor-critic methods

Actor-critic methods describe RL methods which use a separate value function of choice to learn a policy (Sutton et al., 2018). The name *actor-critic* is fitting and refers to the policy as the actor and the value function as the critic, as the value function "criticises" the action selection of the policy. The actor then uses the feedback from the critic to update its policy, by positively reinforcing the probability of valuable actions and decreasing the probability of less valuable actions. A general illustration of the actor-critic structure is depicted in Figure 2.3.

In the case of TRPO, which is an actor-critic method, the most common value function to use is the advantage function A_t, see Equation (2.4).

Generalized Advantage Estimation (GAE)

For policy gradient methods to work, a satisfactory estimate of the policy gradient $\widehat{\nabla J(\boldsymbol{\theta})}$ must be provided. The policy gradient is found by differentiating some measure which yields the expectation of the cumulative rewards, with respect to the parameter vector $\boldsymbol{\theta}$:

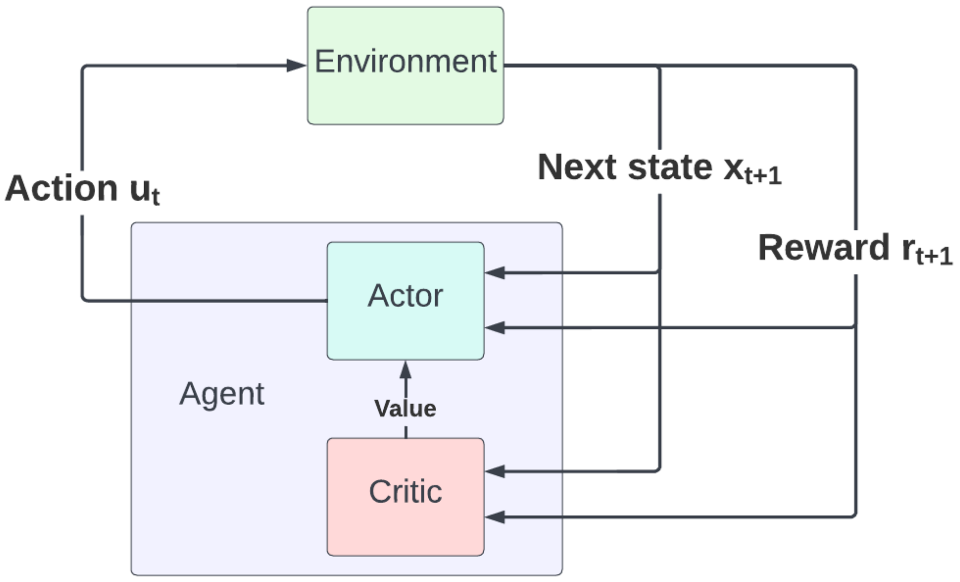


Figure 2.3: Architecture of actor-critic methods.

$$\nabla J(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathbb{E}\left[\sum_{t=0}^{\infty} r_t\right] \tag{2.6}$$

A general expression for this policy gradient is:

$$\nabla J(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=0}^{\infty} \boldsymbol{\Psi}_t \nabla_{\boldsymbol{\theta}} \log \pi_\theta(\boldsymbol{u}_t|\boldsymbol{x}_t)\right] \tag{2.7}$$

where $\boldsymbol{\Psi}_t$ may be expressed in several ways, using e. g. approximations of the total reward or the state-action value function (Schulman, Moritz, et al., 2018). A popular choice of $\boldsymbol{\Psi}_t$ is the advantage function A_t, described in Equation (2.4). The advantage function is unknown and must be estimated as $\widehat{A}_t$. GAE is an effective technique to estimate the advantage function. Using the advantage function is intuitively a good choice of measure, as it quantifies how well the policy π is performing compared to other possible actions. Thus, computing the gradient via the advantage function will point the parameters in the direction of improving the policy by comparing it to other options (Schulman, Moritz, et al., 2018). Learning this relative measure of how good an action is compared to another is also intuitively easier than learning the full value of the action (Arulkumaran et al., 2017).

In GAE, a value function estimator is introduced. The estimator is discounted by a factor γ, which introduces a bias, but reduces variance. This yields the advantage function estimator A^γ and the following policy gradient approximation:

$$\widehat{\nabla J(\boldsymbol{\theta})} = \mathbb{E}\left[\sum_{t=0}^{\infty} A^\gamma(x_t, u_t) \nabla_{\boldsymbol{\theta}} \log \pi_\theta(\boldsymbol{u}_t|\boldsymbol{x}_t)\right] \tag{2.8}$$

Where the expectation value in practice is found by averaging over a finite batch of samples. An estimate of the value function is used to compute the advantage function using the expression:

$$\delta_t^V = R_t + \gamma V(x_{t+1}) - V(x). \tag{2.9}$$

This expression is called the *TD residual* of the value function, as it is also used as an error measure in Temporal Difference (TD) methods. It becomes an unbiased estimator of the advantage function A_t if the value function is correct, i.e., if $V = V^\pi$ (Schulman, Moritz, et al., 2018). However, when $V \neq V^\pi$, the estimator becomes biased due to the term $\gamma V(x_{t+1})$. To overcome the bias of the estimator, it is expanded as a telescoping sum of k terms as

$$\widehat{A}_t^k = \delta_t^V + \gamma\delta_{t+1}^V + \gamma^2\delta_{t+2}^V + ... + \gamma^{k-1}\delta_{t+k-1}^V, \tag{2.10}$$

the intermediate value function terms disappear as the sum expands, and the expression can be reformulated as

$$\widehat{A}_t^k = -V(x_t) + r_t + \gamma r_{t+1} + ... + \gamma^{k-1} r_{t+k-1} + \gamma^k V(x_{t+k}) = \sum_{l=0}^{k-1} \gamma^l \delta_{t+l}^V \tag{2.11}$$

when $k \to \infty$, the bias becomes smaller as the last value function term of future states becomes heavily discounted. The GAE estimator is given as a λ-weighted sum of these k-step estimators:

$$\widehat{A}_t^{GAE(\gamma,\lambda)} = (1-\lambda)(\widehat{A}_t^{(1)} + \lambda\widehat{A}_t^{(2)} + \lambda^2\widehat{A}_t^{(3)} + ...) = \sum_{l=0}^{\infty} (\lambda\gamma)^l \delta_{t+l}^V \tag{2.12}$$

The full GAE gradient approximation thus becomes:

$$\widehat{\nabla J(\boldsymbol{\theta})} = \mathbb{E}\left[\sum_{t=0}^{\infty}\sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V \nabla_{\boldsymbol{\theta}} \log \pi_\theta(\boldsymbol{u}_t|\boldsymbol{x}_t)\right] \tag{2.13}$$

.

GAE is used as the gradient approximation method in this work.

Importance Sampling

A well known statistical problem is the problem of estimating the expected value of a probability distribution. In the case where it is possible and simple to draw samples from the distribution, the problem also becomes simple - the expectation is found by averaging over a set of random samples, as in traditional Monte Carlo sampling techniques. *Importance sampling* is useful when it is hard or impossible to draw samples from the probability distribution of interest, such as in cases where we do not have access to samples from the distribution at all. The importance sampling technique performs a mathematical trick to allow estimating the distribution by using samples from another known distribution. Let X be a random variable distributed according to $p(x)$, and $f(x)$ be a function of X. The expectation of $f(x)$ is given as:

$$\mu_f = \mathbb{E}_p[f(X)] = \int f(x)p(x)dx \tag{2.14}$$

which can be reformulated as

$$\mu_f = \int f(x)\frac{p(x)}{q(x)}q(x)dx = \mathbb{E}_q[f(X)w(X)] = \tag{2.15}$$

where $q(x)$ is a known distribution. The expression becomes the expected value of $f(x)w(x)$ over distribution $q(x)$, where $w(x) = \frac{p(x)}{q(x)}$ is referred to as the importance weight (Tokdar et al., 2010).

The Kullback-Leibler (KL) divergence

The KL divergence is an important statistical measure of the difference between two probability distributions $p(x)$ and $q(x)$. It is denoted $D_{KL}(p(x)||q(x))$ and defined as:

$$D_{KL}(p(x)||q(x)) = \mathbb{E}_{x\sim p}[\log\frac{p(x)}{q(x)}] = \int p(x)\log\frac{p(x)}{q(x)}dx \tag{2.16}$$

(Kullback et al., 1951).

TRPO uses the three techniques presented in this section. The complete TRPO algorithm will be elaborated on in the following section.

2.3.2 TRPO

TRPO was proposed in 2015 by Schulman, Levine, et al. (2015), and has since then become a central algorithm in DRL (Arulkumaran et al., 2017).

For every policy update, the algorithm seeks to maximise an approximation of the expected total discounted reward $\zeta(\pi)$ of the new policy π, given its advantage over an old policy $\pi_{\boldsymbol{\theta}_{old}}$. $\zeta(\pi)$ is given as the expected sum of rewards when following policy $\pi_{\boldsymbol{\theta}_{old}}$ plus the advantage of following the new policy π:

$$\zeta(\pi_{\boldsymbol{\theta}}) = \zeta(\pi_{\boldsymbol{\theta}_{old}}) + \mathbb{E}_{(x,u)\sim\pi_{\boldsymbol{\theta}}}\left[\sum_{t=0}^{\infty}\gamma A_{\pi_{\boldsymbol{\theta}_{old}}}(x_t, u_t)\right] \tag{2.17}$$

The expression in Equation (2.17) is hard to optimise directly, which is why the TRPO algorithm introduces a local approximation to $\zeta(\pi)$ given as:

$$L_{\pi_{\boldsymbol{\theta}_{old}}}(\pi_{\boldsymbol{\theta}}) = \zeta(\pi_{\boldsymbol{\theta}_{old}}) + \mathbb{E}_{x\sim\rho_{\pi_{\boldsymbol{\theta}_{old}}}, u\sim\pi_{\boldsymbol{\theta}}}\left[A_{\pi_{\boldsymbol{\theta}_{old}}}(x,u)\right]. \tag{2.18}$$

where $\rho_{\pi_{\boldsymbol{\theta}_{old}}}(x)$ denotes the discounted visitation frequencies of the old policy. The state visitations are thus assumed to be similar between the new and the old policy, which requires the policy updates to be sufficiently small. As the term $\zeta(\pi_{\boldsymbol{\theta}_{old}})$ is not dependent on $\boldsymbol{\theta}$, it can be omitted from the optimization problem. The issue of the remaining term is that we want to evaluate actions drawn from the new policy π, before the policy has been updated. This is where importance sampling comes into play: action samples from the old policy $\pi_{\boldsymbol{\theta}_{old}}$ can be used instead, as long as the importance weight factor discussed in Section 2.3.1 is included in the optimization objective. The final objective becomes:

$$\mathbb{E}_{x\sim\rho_{\pi_{\boldsymbol{\theta}_{old}}}, u\sim\pi_{\boldsymbol{\theta}_{old}}}\left[\frac{\pi_{\boldsymbol{\theta}}(u_t|x_t)}{\pi_{\boldsymbol{\theta}_{old}}(u_t|x_t)}A_{\pi_{\boldsymbol{\theta}_{old}}}(x,u)\right]. \tag{2.19}$$

Moreover, a so-called *trust region* constraint is put on the policy update using an upper bound on the KL divergence between the old and the new policy, to ensure that the policy updates are sufficiently small. In practice, a heuristic approximation of the average KL divergence is used, due to the complexity of finding the KL divergence for the whole state space. The resulting optimization problem formulation then becomes:

$$\max_{\boldsymbol{\theta}} \; \widehat{\mathbb{E}}_t\left[\frac{\pi_{\boldsymbol{\theta}}(u_t|x_t)}{\pi_{\boldsymbol{\theta}_{old}}(u_t|x_t)}\widehat{A}_{\pi_{\boldsymbol{\theta}_{old}}}(x,u)\right] \tag{2.20}$$

$$\text{s. t. } \widehat{\mathbb{E}}_t\left[D_{KL}(\pi_{\boldsymbol{\theta}}(\circ|x_t)||\pi_{\boldsymbol{\theta}_{old}}(\circ|x_t))\right] \leq \delta \tag{2.21}$$

where $\widehat{\mathbb{E}}_t$ denotes the stochastic estimate found over a batch of trajectories. This optimization problem formulation is quite elegant and guarantees monotonic improvement of the policy s. t. $\zeta(\pi_1) \leq \zeta(\pi_2) \leq \zeta(\pi_3)...$) (Schulman, Levine, et al., 2015) while ensuring sufficiently small policy updates. See (Schulman, Levine, et al., 2015) for further details of the method.

2.4 Adaptive Stress Testing

AST was proposed in 2015 (Lee, Kochenderfer, Mengshoel, Brat, et al., 2015) as a method to perform falsification by applying RL to find the most likely paths to failure. It is a simulation-based method that, given a simulator of the system of interest, simulates batches of trajectories and gradually learns how to uncover failure modes of the system.

2.4.1 The AST formulation

AST seeks to find the most likely sequence of disturbance actions $\boldsymbol{u}$ that lead the system into failure. Hence, the optimisation problem is mathematically formulated as:

$$\max_{\boldsymbol{u}_0,\ldots,\boldsymbol{u}_{end}} \prod_{t=0}^{t_{end}-1} p(\boldsymbol{u}_t|\boldsymbol{x}_t) \qquad (2.22)$$
$$\text{subject to } \boldsymbol{x}_{t_{end}} \in E$$

where $E \subset X$ is the set of failure states, referred to as the *event space*, and t_{end} is the final time step of the episode.

The AST problem is thus a constrained optimisation problem that maximizes the probability of the disturbance sequence, subject to the constraint that failure occurs in the final state $\boldsymbol{x}_{t_{end}}$.

AST solves the optimisation problem by formulating the process of bringing the system to failure as an MDP and solving it by applying an RL solver. The MDP is formulated as follows:

- The state $\boldsymbol{x}$ of the MDP is the state of the simulator.
- The agent observes the state $\boldsymbol{x}$ and chooses action $\boldsymbol{u}$.
- The transition to the next state is given by the transition behaviour of the simulator which is the combined behaviour of $\mathcal{M}$ and $\mathcal{E}$.
- The reward function is set to optimize Equation (2.22).

(Lee, Mengshoel, et al., 2020). The resulting RL structure differs from traditional RL setups. In most simulated RL problems, the agent navigates in the simulator environment. In the AST case, the simulator environment and the SUT are joined together to constitute the environment of the AST agent. The AST structure is presented in Figure 2.4.

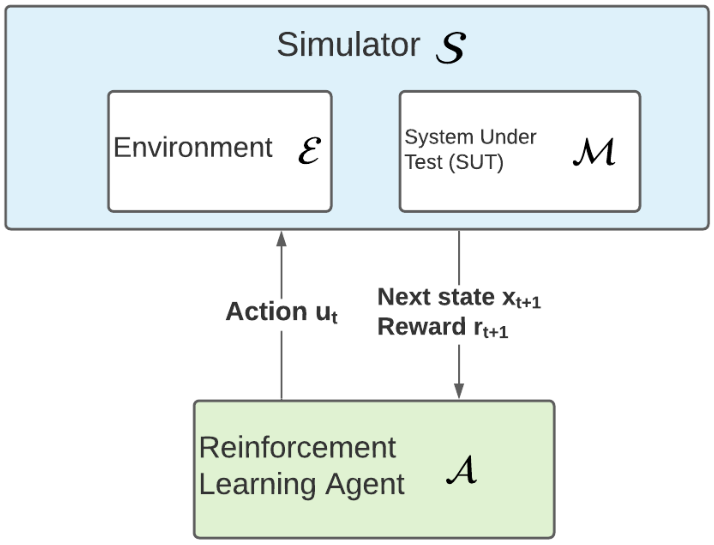

Simulator $\mathcal{S}$
Environment $\mathcal{E}$
System Under Test (SUT) $\mathcal{M}$
Action u_t
Next state x_{t+1}
Reward r_{t+1}
Reinforcement Learning Agent $\mathcal{A}$

Figure 2.4: The AST structure, inspired by (Lee, Mengshoel, et al., 2020)

The MDP reward function R of the RL agent is formulated in a way which maximizes Equation (2.22). There are many formulations of the MDP reward function in the AST literature.

Koren, Corso, et al. (2020) proposed a general reward function formulation, that presents three distinct reward cases:

$$R = \begin{cases} 0, & \text{if } \boldsymbol{x} \in E \\ -\alpha - \beta f(\boldsymbol{x}), & \text{if } \boldsymbol{x} \notin E, t \geq t_{end} \\ -g(\boldsymbol{u}) - \eta h(\boldsymbol{x}), & \text{if } \boldsymbol{x} \notin E, t < t_{end} \end{cases} \tag{2.23}$$

The first reward case in Equation (2.23) is active when a failure mode is obtained, i. e., when $\boldsymbol{x} \in E$. In this case, the agent obtains a zero-reward. The second case presents the times when the simulation has run until the end without obtaining any failure, in which case a large negative reward is distributed according to the user-defined constants α and β, and an optional heuristic $f(\boldsymbol{x})$. The heuristic can be added if there exists a measure that can guide the agent toward the goal faster. Adding such a heuristic is known as *reward shaping* (Ng et al., 1999). While reward shaping has the potential to speed up the RL process, it is important not to implement heuristics which can cause positive reward cycles. These cycles can potentially lead the agent to develop sub-optimal policies away from the goal as it is incentivised to rather collect intermediate rewards (Ng et al., 1999). If the heuristic is constructed so that there are no positive reward cycles, it is called a *potential-based* reward shaping function. Potential-based reward shaping functions are shown to preserve the optimal policy while guiding the agent faster towards the goal (Ng et al., 1999).

The last reward case is active during the episode, i.e., when $t < t_{end}$ and no failure had been obtained. A reward is given according to $g(\boldsymbol{u})$, a function which is recommended by Koren, Corso, et al. (2020) to be proportional to the logarithm of the likelihood of the action such that the reward function optimizes Equation (2.22). The additional penalty $\eta h(\boldsymbol{x})$ constitutes an optional training heuristic given at each time step.

2.4.2 Method augmentations

Since the formulation of the AST method, it has been extended with augmentations and additional logic to overcome certain common issues. Augmentations have mainly been made to the MDP reward function and the choice of RL solver. A handful of augmentations which is implemented in this thesis are presented.

Choice of RL solver

In the first implementation of AST, Monte Carlo Tree Search (MCTS) was used as the RL solver. However, MDPs are solvable in many means, and in later work, the RL solver has been replaced by other solvers. Koren, Alsaif, et al. (2018) showed that the use of a DRL solver gave better results in the automobile case. In this work, a DRL solver is implemented similarly to Koren, Alsaif, et al. (2018). The solving method is further elaborated on in Section 2.1 and Chapter 3.

Reward augmentations

The work in this thesis implements a reward function similar to the one proposed in Koren, Alsaif, et al. (2018) as a baseline reward function:

$$R = \begin{cases} 0 & \text{if } \boldsymbol{x} \in E \\ -\alpha - \beta \mathrm{D}, & \text{if } \boldsymbol{x} \notin E, t \geq t_{end} \\ -\log(1 + M(\boldsymbol{u}, \boldsymbol{\mu_u} | \boldsymbol{x})), & \text{if } \boldsymbol{x} \notin E, t < t_{end} \end{cases} \tag{2.24}$$

In the case of failure, i. e., when $\boldsymbol{x}$ is in the event space E, the RL agent obtains a zero-reward. In the second case, a large negative reward is distributed according to α, β and the distance heuristic D, which is the distance between the ferry and the adversary. The distance heuristic penalizes the agent additionally if the distance between the adversary and the ferry is large in the end. This heuristic guides the agent towards cases where the two vessels end up in close proximity. Distance heuristics are shown to be potential based reward functions, and thus it is guaranteed that the optimal policy is preserved while adding the heuristic (Robert J Moss et al., 2021).

The last case portrays the reward distributed at every time step. The agent receives a reward using the Mahalanobis distance $M(\boldsymbol{u}, \boldsymbol{\mu_u} | \boldsymbol{x})$ (Mahalanobis, 1936), which is the statistical distance between the action $\boldsymbol{u}$ and the mean of the action distribution $\boldsymbol{\mu_u}$ given as

$$D_M(\boldsymbol{u}) = \sqrt{(\boldsymbol{u} - \boldsymbol{\mu_u})^T \mathbf{S}^{-1} (\boldsymbol{u} - \boldsymbol{\mu_u})}$$

where $\mathbf{S}^{-1}$ is the inverse covariance matrix of the action distribution. This part of the reward function is a heuristic measure of the probability of the AST action, which makes the AST method optimize for the most likely action sequences. The logarithm is taken to optimize the product of the probabilities in Equation (2.22).

Previous work which has applied the same or similar reward functions has addressed a couple of common drawbacks with it, such as:

- The identified failures are often unavoidable. In the case of the automobile approaching a pedestrian, the majority of the initially found failures showed scenarios where the pedestrian accelerated and ran straight into the car.
- When the RL agent discovers a failure, it tends to rediscover the same or similar failures.

These drawbacks decrease the relevance of the identified failure modes to the developers of the system, who naturally seek a diverse set of failures where the behaviour of the SUT is, to a larger extent, the cause of the failure. The literature suggests that the drawbacks occur due to the RL solver converging to sub-optimal solutions (Corso, Du, et al., 2019) and because of the reward for failure is so high combined with the fact that rewards distributed during the episode does not aid exploration, forcing the agent to converge to already found failures (Koren and Kochenderfer, 2020). These drawbacks were also seen in the work which led up to this thesis, where the adversary vessel expressed irrational and aggressive behaviour by heading straight toward the ferry (Hjelmeland, 2021). Corso, Du, et al. (2019) suggest two reward augmentations designed to aid the search and obtain a more diverse set of failures which are not unavoidable. Both approaches are implemented in this thesis and are elaborated on in the following sections.

Optimizing for improper behaviour

To cope with the AST tendency to discover unavoidable failures, Corso, Du, et al. (2019) suggest optimizing for failures where the SUT is to a larger extent participatory in the lead-up to the failure. The proposed way to accomplish this is to optimize for failures where there SUT expresses improper behaviour during the simulation. To do so, two augmentations are made to the AST problem formulation: the event space E is altered to only register a collision as a failure if the SUT has expressed a sufficient amount of improper behaviour, measured in the fraction of improper time steps during the simulation. Furthermore, the distance heuristic in the reward function Equation (2.24) is also replaced to optimize for improper time steps, based on the assumption that a larger fraction of improper time steps lead to more failures where improper behaviour is expressed. Corso, Du, et al. (2019) considers a time step improper if the behaviour of the SUT in the time step is non-compliant with Responsibility-Sensitive Safety (RSS), which is a set of rules for responsible driving behaviour.

Let N_{imp} denote the number of improper time steps, N the total number of steps, $f_{imp} = \frac{N_{imp}}{N}$ the fraction of improper time steps, and $f_{prop} = 1 - f_{imp}$ the fraction of proper time steps. Let E_{imp} be the altered event space and ω denote the sequence of AST actions, i.e., $\omega = \{\boldsymbol{u}_{ast_0}, \boldsymbol{u}_{ast_1}, ..., \boldsymbol{u}_{ast_{end}}\}$, then:

$$E_{imp} = \{\omega \mid \boldsymbol{x}_{t_{end}} \in E \land f_{imp}(\omega) > f_{thresh}\}. \tag{2.25}$$

The resulting reward function is [1]:

$$R = \begin{cases} 0, & \text{if } \boldsymbol{x} \in E_{imp} \\ -\alpha - \beta f_{prop}, & \text{if } \boldsymbol{x} \notin E_{imp}, t \geq t_{end} \\ -\log(1 + M(\boldsymbol{u}, \boldsymbol{\mu_u} | \boldsymbol{x})), & \text{if } \boldsymbol{x} \notin E_{imp}, t < t_{end} \end{cases} \tag{2.26}$$

Dissimilarity measure

The issue of the AST method is that the obtained failure modes are highly similar due to the RL solver becoming trapped in a local optimum. The nature of the AST optimisation makes it prone to this effect due to the high reward for failure modes and little guidance towards the failure mode during simulation. Thus, the problem can be described as a hard exploration problem as the rewards do not aid exploration (Koren and Kochenderfer, 2020).

Solutions have been proposed to deal with this issue in the AST literature. This thesis implements a solution proposed by Corso, Du, et al. (2019), where a dissimilarity measure was implemented to reward more dissimilar results. The implementation successfully aided the search for failure modes in simulations of an automobile approaching a pedestrian, resulting in a more diverse set of failures.

[1]Corso, Du, et al. (2019) uses f_{imp} as the heuristic. It is assumed in this work that the intended measure is f_{prop}, as penalizing a higher fraction of improper time steps would lead to *less* improper time steps. Might be (a) a misunderstanding of the definition of f_{imp} or (b) a typo of Corso, Du, et al. (2019).

The dissimilarity measure is implemented as part of the reward function, by distributing more rewards to the trajectories that are dissimilar to the ones already obtained. To measure the dissimilarity, a quantification of the similarity between the trajectories is proposed, where the trajectories are divided into segments and the Center Of Mass (COM)s of the trajectory segments are compared. Details of the method implementation are further elaborated on in Section 3.4.

2.4.3 AST Software

Several open-source software frameworks have been developed, including a Python toolbox (Koren, Ma, et al., 2021) and a Julia framework (NASA, 2022). The AST Python Toolbox is applied in this thesis.

2.5 Marine vessel dynamics and control

The simulator used in this thesis portrays simplified 3-Degrees of Freedom (DOF) horizontal plane models of marine vessels, which will be elaborated on in this chapter.

2.5.1 Dynamics of marine vessel

Let an inertial frame be approximated by the earth-fixed reference frame {e} called North East Down (NED). A marine vessel is fully described by a 6-DOF model with the state-space representation

$$\boldsymbol{\eta} = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{bmatrix}$$

where (x, y, z) represents *surge, sway* and *heave*, describing the body position in 3D space (Fossen, 2011). (ϕ, θ, ψ) are the orientation angles *roll, pitch* and *yaw*, see Figure 2.5 for an illustration.

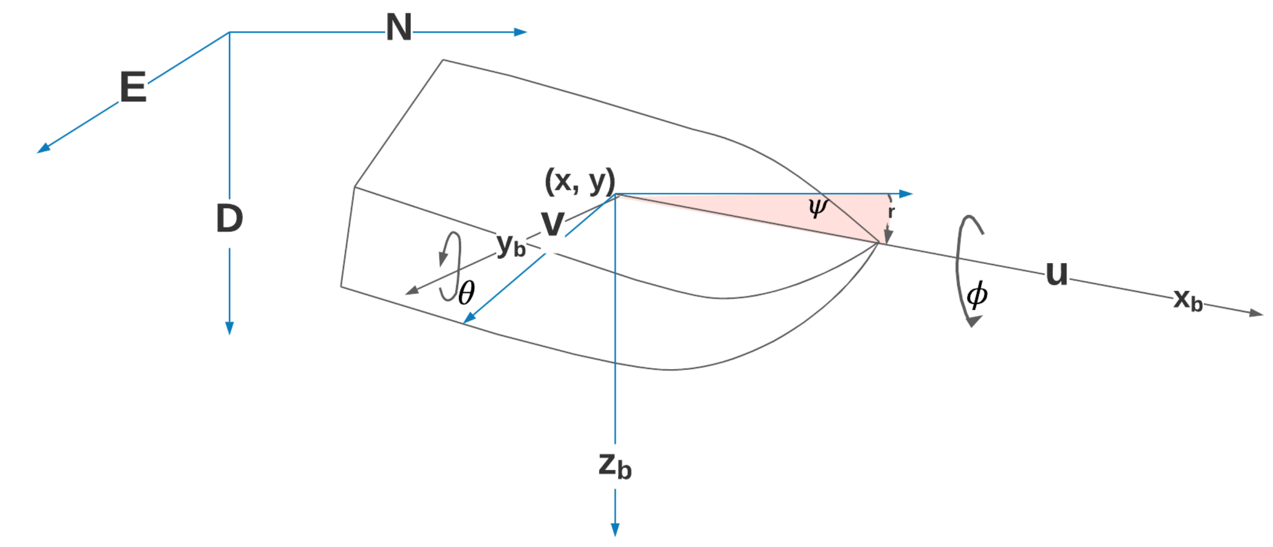


Figure 2.5: Vessel with state parameters, inspired by (Fossen, 2011).

in Figure 2.5, x_b, y_b and z_b makes up the body reference frame of the vessel. As a simplification, one often assumes that the movement in the z-direction is arbitrarily small. The *heave* parameter z is then removed from η together with the vertical angles (ϕ, θ), yielding the 3-DOF state vector

$$\boldsymbol{\eta} = \begin{bmatrix} x \\ y \\ \psi \end{bmatrix}$$

The velocity vector for the 3-DOF model is given as

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

where (u, v, r) is called *surge speed*, *sway speed* and *yaw rate*, see Figure 2.5.

As can be deduced by inspection of Figure 2.5, the following relation between η and ν can be obtained:

$$\dot{\boldsymbol{\eta}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} u\cos(\psi) - v\sin(\psi) \\ u\sin(\psi) + v\cos(\psi) \\ r \end{bmatrix} = \boldsymbol{R}_{z,\psi}\nu$$

where

$$\boldsymbol{R}_{z,\psi} = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

When the assumption of no heave motion is combined with the assumption that there are no external disturbances to the vessel such as wind, ocean currents and waves, the 3-DOF vessel dynamics can be formulated as:

$$\begin{aligned} \dot{\boldsymbol{\eta}} &= \boldsymbol{R}_{z,\psi}\nu \\ \boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} \ + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} &= \boldsymbol{\tau} \end{aligned} \tag{2.27}$$

Here, $\boldsymbol{\tau}$ is the vector of forces and torque acting on the vessel, which can include e. g. disturbances and control actuation. $\boldsymbol{D(\nu)}$ is the hydrodynamic damping matrix, $\boldsymbol{M}$ is the mass matrix and $\boldsymbol{C(\nu)}$ is the Coriolis and centripetal matrix. The matrices $\boldsymbol{M}$ and $\boldsymbol{C(\nu)}$ represents the combination of the dynamics of the rigid body and the dynamics due to the hydrodynamic effect of added mass, denoted RB and A respectively:

$$\begin{aligned} \boldsymbol{M} &= \boldsymbol{M}_{RB} + \boldsymbol{M}_A \\ \boldsymbol{C}(\boldsymbol{\nu}) &= \boldsymbol{C}(\boldsymbol{\nu})_{RB} + \boldsymbol{C}(\boldsymbol{\nu})_A \end{aligned}$$

Moreover, the matrix function $\boldsymbol{D(\nu)}$ represents the linear and nonlinear damping terms, denoted L and NL respectively:

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

2.5.2 Path-following and the Line Of Sight (LOS) guidance principle

Guidance is a method to obtain state references in order to make a vessel follow a certain trajectory. One form of guidance system is *path-following* for straight-line paths, which can be implemented in several ways. Fossen (2011) describes a guidance principle for this purpose called look-ahead based steering, which will be elaborated on and used in the simulations in this thesis.

Consider a straight line path given by the to waypoint wp_k and wp_{k+1}. Let the course angle χ denote the angle of the velocity vector $\boldsymbol{U}$ of the vessel relative to the NED frame, given as the vessel heading ψ plus an additional angle β. In lookahead-based steering, the desired course angle χ_d is given in two parts, consisting of α_k, called the *path tangential angle* plus an angle referred to as the *velocity-path* relative angle, denoted χ_r. The complete expression is then given as:

$$\begin{aligned} \chi_d &= \alpha_k + \chi_r && (2.28) \\ &= \arctan2(y_{\boldsymbol{wp}_{k+1}} - y_{\boldsymbol{wp}_k}, x_{\boldsymbol{wp}_{k+1}} - x_{\boldsymbol{wp}_k}) + \arctan2(-e, L_{LA}) && (2.29) \end{aligned}$$

where α_k is the path angle relative to the NED frame, e is the cross-track error between the vessel position and the path and L_{LA} is the distance of the LOS vector $\boldsymbol{L}$ projection onto the path. The parameters used in the calculation of the desired course angle are depicted in Figure 2.6.

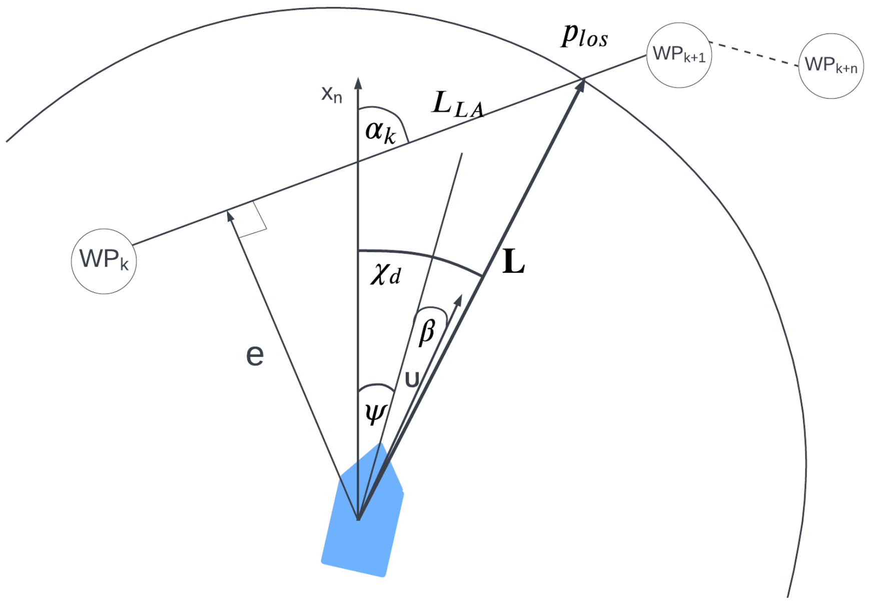


Figure 2.6: Parameters used in lookahead based steering, inspired by (Fossen, 2011).

To make the vessel follow several consecutive paths with different velocity profiles, a LOS controller can be initialized with a sequence of waypoints which chronologically describe the desired paths. The path to the next waypoint is then activated when the vessel is within a predefined switching distance L_{WP} of the current waypoint.

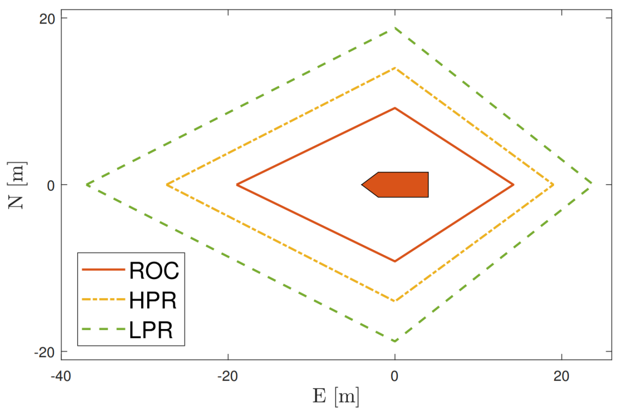
When the desired course angle has been obtained, a path-following LOS controller can be used to align the ship with the desired course. Different control principles are described in Fossen (2011), but will not be elaborated on here. The LOS based controller implemented in this thesis will be further described in Chapter 3.

2.6 Collision Avoidance

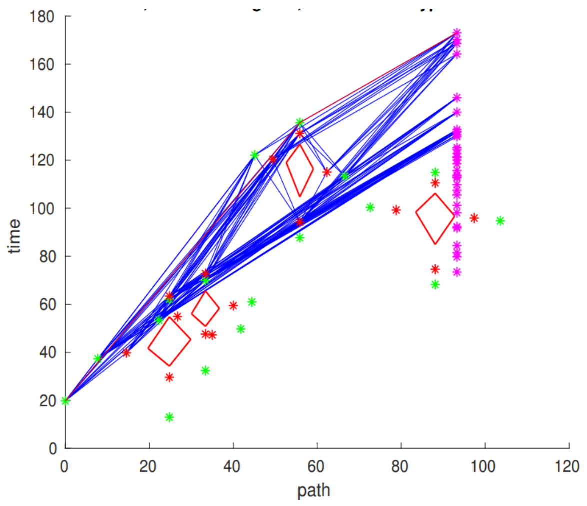
Autonomous vehicles need collision avoidance systems when navigating amidst obstacles such as other vehicles or land. In this thesis, a simulator of the milliAmpere ferry which is simplified and focused for testing the collision avoidance system is used, in which the vessel is simulated in traffic with other vehicles. The milliAmpere prototype is described thoroughly in Brekke et al. (2022).

2.6.1 SP-VP

The collision avoidance system implemented on the milliAmpere ferry is based on the SP-VP COLAV approach, which is a path-velocity decomposition method proposed in (Thyri et al., 2020). The method projects all obstacles into a path-time space and builds a search tree spanning the path-time space of different velocity profiles along a path to a predefined waypoint. A cost dependent on the speed and closeness to obstacles is applied to each edge in the search tree, and the optimal trajectory is found using Dijkstra's algorithm. Three regions are constructed around the obstacle, namely the Region of Collision (ROC), High Penalty Region (HPR) and the Low Penalty Region (LPR), defining different levels of cost, as described in (Uttisrud, 2019) and illustrated in Section 2.6.1. An optimal velocity profile along the path is then selected based on the cost attached to every node in the graph, as illustrated in Section 2.6.1. The optimal path-time trajectory is transformed into a time-expanded Euclidean space and tracked by the ferry using a Dynamic Positioning (DP) controller. Being constrained to following a nominal path, the SP-VP algorithm is only able to control the velocity of the vessel along this path, making it unable to make course changes to avoid collisions (Hjelmeland et al., 2022). Furthermore, the algorithm is parameterized to only allow for a speed within a certain range, which leads to stopping being the preferred action when the ferry is faced with a collision situation.



(a) The SP-VP obstacle representation with ROC, HPR and LPR illustrated. Figure obtained from (Uttisrud, 2019).



(b) Possible velocity profiles along a predefined path in the path-time space. The polygons indicate the path-time representation of the obstacles. The optimal velocity profile is highlighted in red. Figure obtained from (Thyri, 2019).

The SP-VP method is a robust and simple approach to collision avoidance as it produces predictable and intuitive trajectories. As a trade-off of the simplicity of the SP-VP approach, the method is not able to handle situations where e. g. an obstacle is driving head on towards the vessel as it is unable to find a path that does not lead to a collision. In cases like these, the algorithm breaks down due to the problem becoming infeasible.

2.6.2 MPC based COLAV with COLREGs compliance

MPC is a family of methods that perform *closed-loop* optimal control, which chooses control input by predicting its effect on the system in the future. The best possible control input is chosen by forecasting how the system will develop and choosing the first control input of the input sequence that leads to the best outcome (Rawlings et al., 2017). The concept of MPC is thoroughly described in the literature and will not be elaborated on here, see e.g. (Rawlings et al., 2017) for further descriptions.

MPC based COLAV approaches have been studied and applied in many fields, such as automotive vehicles (Ammour et al., 2021) and air crafts (Bousson, 2008). The MPC method is a suited fit for the COLAV problem, especially if there exists a sufficiently good model of the system. If so, the MPC system is able to predict into the future and possibly foresee collisions and thus avoid hazardous situations. In the marine setting, a perfect COLAV system is both able to avoid collision *and* to do so in compliance with the COLREGs. The COLREGs is an international framework designed by the International Maritime Organization, which proposes an extensive set of rules to be followed by ships in order to avoid collisions. A COLREGs compliant method is a method which avoids collisions, but preferably in ways described as proper behaviour by the COLREGs.

An MPC based COLAV method for ship collision avoidance with COLREGs compliance was first described in Johansen et al. (2016). Since then, the method has been extended and proposed in different manners, e.g. in Hagen et al. (2018). Their approach constitutes a simulation-based MPC method where a relatively small subset of control behaviours are evaluated at every step. It is assumed that the ship has a guidance controller which provides references for the ship course angle ψ_d and propulsion command u_d. The MPC COLAV framework then works as an extension of the guidance system by proposing modifications, denoted ψ_m and u_m, to these references in order to perform COLAV and to comply with the COLREGs. Johansen et al. (2016) propose a set of possible control modifications:

- $\psi_m \in$ {-90°, -75°, -60°, -45°, -30°, -15°, 0°, 0°, 15°, 30°, 45°, 60°, 75°, 90°}
- $u_m \in$ {1, 0.5, 0, -1}, i.e., {keep speed, slow down, stop and full backward propulsion}

which can be adjusted to the domain-specific purpose. The more extensive the set is, the better performance of the system.

The fact that the modifications are placed on top of the existing guidance system makes the system both able to follow a predetermined path while performing COLAV, and further implementations for path following is thus redundant. The control behaviour is kept constant for the entire simulation horizon, which is a beneficial detail of this method as it both makes sense, but also reduced the need for computation at each simulation time step.

The cost function to be minimized in the MPC problem consists of three terms: the cost associated with potential collisions, the cost of violating the COLREGs, and the cost of altering the controller reference provided by the LOS controller. To evaluate the cost of a situation with respect to adversary vessel i at time t in control behaviour scenario k, these metrics are used:

- The distance $d_i^k(t)$ between the vessel and the adversary vessel

- The velocity vector of the vessel, denoted $\boldsymbol{v}_0^k(t)$
- The velocity of the adversary vessel, denoted $\boldsymbol{v}_i^k(t)$
- The angle between the velocity vectors, denoted $\theta_i \in [-\pi, \pi]$
- The unit vector in the LOS direction from the vessel to the adversary vessel, denoted $\boldsymbol{L}_i^k(t)$.
- The bearing angle of the LOS direction vector, denoted θ_{L_i}
- The angle between own ship velocity vector and the LOS direction vector, denoted $\theta_{v,l}$

All vectors $\boldsymbol{v}_0^k(t), \boldsymbol{v}_i^k(t)$ and $\boldsymbol{L}_i^k(t)$ are 2-dimensional and describe velocities and relative positions in the NED-frame. The metrics are illustrated in Figure 2.8.

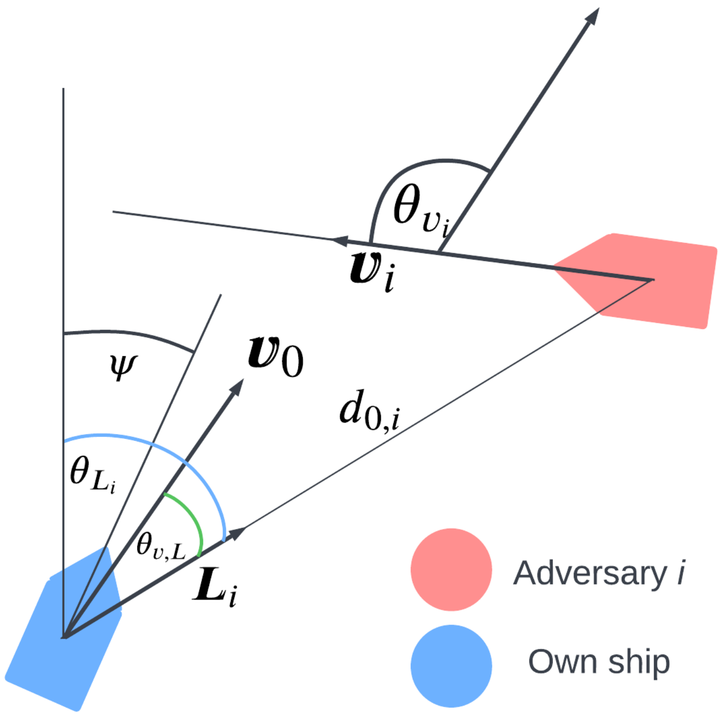


Figure 2.8: Illustration of metrics used in COLREGs violation conditions, based on an illustration in Hagen et al. (2018). The scenario superscripts k are left out for simplification.

The cost of potential collisions is given in a classical risk assessment manner. In standard risk analysis, the risk of an unwanted event is given as the product of the probability of the event occurring, and the consequence of the event occurring. The cost related to collision is given in a similar manner in the definition by Johansen et al. (2016), namely as the product between a *collision risk factor* $\mathcal{R}_i^k(t)$ and the *collision cost* $\mathcal{C}_i^k(t)$. The collision cost thus aids the vessel by not only alerting the system when there is a potential collision coming up, but also by prioritizing the potential risks. This is important as there may be scenarios where the vessel might have to choose between two collisions, which this measure would help with. The two components are based on the kinetic energy of the two vessels, the distance between them, given as

$$\mathcal{C}_i^k(t) = K_i^{coll}|\boldsymbol{v}_0(t) - \boldsymbol{v}_i(t)|^2 \tag{2.30}$$

$$\mathcal{R}_i^k(t) = \begin{cases} \frac{1}{|t-t_0|^p}(\frac{d_i^{safe}}{d_{0,i}^k(t)})^q & \text{if } d_{0,i}^k(t) \leq d_i^{safe} \\ 0 & \text{otherwise} \end{cases} \tag{2.31}$$

where K_i^{coll} is a tunable parameter that weight the risk of collision and d_{safe} is a user-specified safety distance which quantifies the smallest distance of which an adversary vessel is considered a risk. The measure $|t - t_0|^p$ discounts the risk based on how far into the time horizon it occurs, as the uncertainty increases further into the prediction.

The COLREGs cost is distributed if the vessel is currently in a situation which corresponds to a COLREGs violation. Johansen et al. (2016) present five definitions describing the relation between own ship and adversary i:

- **CLOSE**: The adversary vessel i is said to be close to own ship if $d_i^k(t) < d_i^{close}$.

- **STARBOARD**: Adversary i is said to be starboard of own ship if θ_{L_i} is bigger than the heading of own ship ψ.

- **OVERTAKEN**: Own ship is said to be overtaken by adversary i if **CLOSE** is satisfied, $|\theta_i| < 68.5°$ and $||\boldsymbol{v}_i^k(t)|| > ||\boldsymbol{v}_0^k(t)||$.

- **HEAD-ON**: Adversary i is said to be head on of own ship if **CLOSE** is satisfied, the adversary velocity is bigger than some close-to-zero constant ϵ, i.e., $||\boldsymbol{v}_i^k(t)|| > \epsilon$, $|\theta_i| > 157.5°$ and $|\theta_{v,L}| < \phi_{ahead}$, where ϵ and ϕ_{ahead} are user-specified.

- **CROSSED**: Adversary i is said to be crossed by own ship if **CLOSE** is satisfied and $|\theta_i| > 68.5°$

The **CLOSE** and **STARBOARD** conditions are quite intuitive. However, the remaining three conditions can be somewhat hard to visualize. Figure 2.9 illustrates scenarios where these conditions, which are based on the angle between the velocity vectors, are satisfied. The scenarios displayed show own ship (blue) with a velocity of $\boldsymbol{v}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, together with the adversary (red) and examples of the corresponding adversary velocities which lead to the satisfaction of the condition. Note that the angles between the velocities are independent of the position of the vessels, which e.g. makes the **CROSSED**-condition true when the adversary is positioned both on the port and starboard side and makes the **OVERTAKING**-condition true for overtaking by own ship and by the adversary.

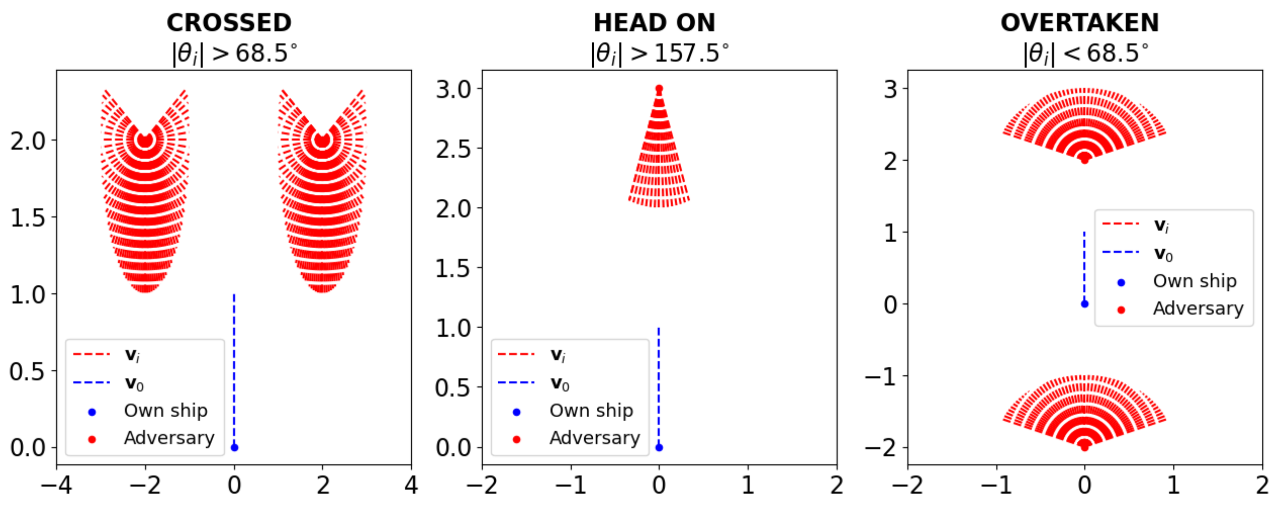


Figure 2.9: Illustration of possible velocity profiles in the relational conditions, based on definitions in Johansen et al. (2016)

These relational definitions make up the components of the logical expression that determine if there is a COLREGs violation when applying the control behaviour of scenario k, at time step t and with respect to adversary i. Let $\mu_i^k(t)$ be a binary indicator of a COLREGs violation. The approach focuses mainly on rules 13-15 of the COLREGs, given by IMO (1972) as:

Rule 13: Overtaking. Any vessel overtaking any other shall keep out of the way of the vessel being overtaken. A vessel shall be deemed to be overtaking when coming up with another vessel from a direction more than 22.5 degrees abaft her beam.

Rule 14: Head-on situation. When two power-driven vessels are meeting on nearly reciprocal courses so as to involve risk for collision, then alter course to starboard so that each pass on the port side of each other.

Rule 15: Crossing situation. When two power-driven vessels are crossing so as to involve risk of collision, the vessel which has the other on her own starboard side shall keep out of the way.

Johansen et al. (2016) defines the indicator as

$$\begin{aligned}\mu_i^k(t) =& \text{RULE 14} \vee \text{RULE 15}\\ \text{RULE 14} =& \textbf{CLOSE} \wedge \textbf{STARBOARD} \wedge \textbf{HEAD-ON}\\ \text{RULE 15} =& \textbf{CLOSE} \wedge \textbf{STARBOARD}\\ & \wedge \textbf{CROSSED} \wedge \text{NOT } \textbf{OVERTAKEN}\end{aligned} \tag{2.32}$$

These rule definitions incorporate Rule 13, which states that the overtaking vessel shall keep out of the way (Johansen et al., 2016).

The total cost associated with each scenario k is then defined as the maximum cost obtained when applying the scenario control behaviour $[u_m, \psi_m]$ and keeping it throughout the prediction horizon, defined as:

$$\mathcal{H}^k(t_0) = \max_i \max_{t \in D(t_0)} (\mathcal{C}_i^k(t)\mathcal{R}_i^k(t) + \kappa\mu_i^k(t)) + f(u_m, \psi_m) + g(u_m, \psi_m) \tag{2.33}$$

where t_0 is the starting point of the prediction and $D(t_0)$ is the finite set of future time steps $D(t_0) = \{t_0, t_0 + \Delta_{mpc}, ..., t_0 + T_{mpc}\}$, where Δ_{mpc} is the MPC prediction step size and T_{mpc} is the MPC time horizon. κ_i is a tuning parameter, $g(u_m, \psi_m)$ is a grounding term that represents a penalty that should be defined based on electronic map data and possibly ship sensor data. Moreover, $f(u_m, \psi_m)$ adds cost according to:

$$f(u_m, \psi_m) = K_{u_m}(1 - u_m) + K_{\psi_m}\psi_m^2 + \Delta_{u_m}(u_m - u_{last}) + \Delta_{\psi_m}(\psi_m - \psi_{last}) \tag{2.34}$$

where K_{u_m} and K_{ψ_m} are tunable parameters that puts a cost on diverging from the reference given by the LOS controller, and Δ_{u_m} and Δ_{ψ_m} are functions designed to put higher costs on big changes from the last control reference denoted $[u_{m_{last}}, \psi_{m_{last}}]$. (Johansen et al., 2016) suggest that K_{ψ_m} and Δ_{ψ_m} should be asymmetrical, yielding a higher cost on modifications that suggest turning to port side, in compliance with the COLREGs rule 14, 15 and 17 which all state that COLAV manoeuvres should be performed by turning to starboard side.

The MPC implementation in this thesis is based on the work of Johansen et al. (2016) and Hagen et al. (2018) and is elaborated on in Section 3.5.

2.7 Clustering

Clustering is an unsupervised machine learning method where data is partitioned into homogeneous groups using some form of similarity measure (Likas et al., 2003). Clustering is applied in this thesis to categorize the resulting trajectories of the adversary to obtain a summary of patterns in the adversary behaviour.

2.7.1 Soft-Dynamic Time Warping (DTW) K-means Clustering

The K-means clustering technique seeks to divide a set of data points of arbitrary dimension into K clusters by minimizing the within-cluster sum of squares (Hartigan et al., 1979). The method can be applied to time series, but as the standard distance measure is the euclidean distance, potential time shifts and lags pose a problem. Methods like DTW can be applied to align the time series before measuring their distance. DTW was proposed in (Sakoe et al., 1978), and constitutes a method of aligning temporal series of data subject to deformations or varying speeds as illustrated in Figure 2.10 (Müller, 2007). It is well known and applied to many problems in fields such as speech recognition. The alignment is done by defining a cost measure, typically using the distance between the series, and finding a minimum of that cost under a set of requirements. See (Sakoe et al., 1978) for further details.

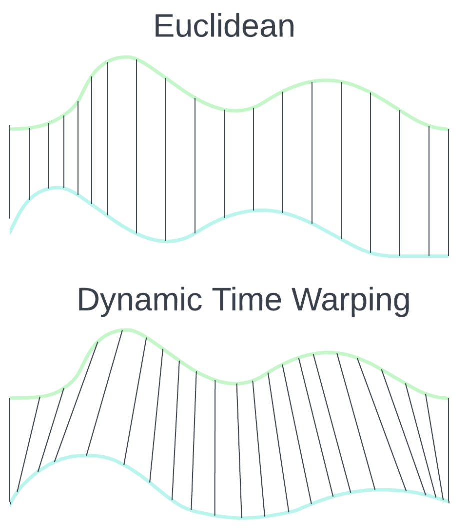


Figure 2.10: Illustration of Dynamic Time Warping matching of time series, based on Müller (2007).

The DTW loss is not differentiable in nature, and therefore soft-DTW was proposed by Cuturi et al. (2017), offering a differentiable and less time demanding extension of the DTW. The soft-DTW has outperformed traditional DTW method baselines.

Chapter 3

Methodology

This master thesis is divided into two parts. The first part presents an in-depth study of the SP-VP case which was studied in Hjelmeland (2021). The second part presents further experiments with the AST method applied to a new case, where the ferry is implemented with a COLREGs-compliant COLAV system based on MPC.

3.1 Clustering

An AST simulation can generate large sets of data on both trajectories that end in failure, as well as trajectories that do not. In order to limit the time needed to analyze all this information, methods to categorize the data can be implemented to present it in a tractable manner.

Lee, Kochenderfer, Mengshoel, and Silbermann (2018) presented the use of Grammar Based Decision Trees (GBDT) to cluster and categorize the trajectories found in airborne systems. GBDT is a decision tree with splits on logical expressions. This was beneficial for categorizing the data sets from these airborne systems, as the aim was to categorize the trajectories on using not only the temporal and spatial information of the aircraft trajectories, but also the timing of the alerts given to the simulated pilot and the actions taken by the pilot. With GBDT, the trajectories could be put into categories which describe the patterns in the trajectory sequences, such as e.g. a category of near mid-air collisions was the sequence of messages for the pilot were correctly issued, but a delay in pilot response time caused the messages to bring the aircrafts closer together, as found in Lee, Kochenderfer, Mengshoel, and Silbermann (2018). The GBDT method both provided categorizing of the results, but it also aided in explaining the categorizations as the description of every category could easily be retrieved by the conjunction of the splits that led to the category leaf node.

The failure trajectories found in this work contain less concrete actions such as e.g. issuing a message for the pilot. One could argue that the COLAV command to reduce the speed issued by the SP-VP system, or the COLREGs violations indicated in the MPC COLAV system could be used for this purpose. However, it was considered more beneficial to use the shape of the adversary trajectory for categorization of the resulting scenarios, as this seemed to be the crucial detail to determine whether or not there would be a collision and why the collision occurred. Moreover, the actions of the AST agent mainly influenced the movement of the adversary, which indicate that categorizing the adversary trajectories would also categorize the policy patterns had by the agent throughout the simulations.

In order to cluster the adversary trajectories, research was done on clustering of multidimensional time series. The soft-DTW k-means clustering method described in Section 2.7 was found as a recommended method for such tasks, especially when the objective is to find similar behavioural patterns, but where the speed of the vessel manoeuvres could vary.

Some experiments were done to find the optimal number of clusters, k, for the purpose. Determining the value of k is a complete field of its own where methods have been proposed to do so automatically, using methods such as the *elbow method* described in e.g. Bholowalia (2014), where k is chosen such that adding another cluster won't add much value to the cluster model. However, experiments where the k-value was chosen manually gave satisfactory results with $k = 4$ clusters. The resulting clusters showed a balanced trade-off between generalization and highlighting specific patterns. It was also desirable to generate the same number of clusters for the different problem formulations to enable an efficient comparison of the results. Therefore, further engineering to optimize the k-value was deemed redundant and not an appropriate use of time. Still, exploring this option could have brought value to the results and should be explored in further research.

3.2 AST implementation

AST is implemented in this work by applying the Python AST Toolbox, an AST framework developed and maintained by Stanford Intelligent Systems Lab (Koren, Ma, et al., 2021). The toolbox is a comprehensive framework consisting of a large network of neatly designed modules, which contains all necessary elements to run training of an AST agent. The toolbox implements RL methods from the open-source RL framework Garage (*garage — garage v2020.09.0rc2-dev documentation* 2020), which again is built using Tensorflow, an open-source software platform for machine learning. Although the toolbox is comprehensive, it comes with a set of examples and is thoroughly documented, which simplifies the familiarization of it. Unfortunately, dependency issues have made the latest version unstable and in need of a specific Docker image to be able to run. This made the implementation somewhat challenging, but once it was able to run, the toolbox offered easy implementation and flexible interfacing with the simulator used for the experiments.

3.2.1 The AST wrapper

The AST method creates a wrapper around the system simulator to perform batches of simulations to train the RL agent. In systems where the simulator state is hidden or inaccessible, it is possible to implement the AST method in an *open-loop* manner, where the RL agent runs full simulation episodes without being able to observe the system output. In the cases presented in this thesis, AST is implemented in *closed-loop* mode, as the simulator state is accessed and fed back to the RL agent for every AST step. The AST wrapper requires some interfacing functions in the simulator in order to run simulation batches and obtain information about the simulator state:

- `reset(`$\boldsymbol{x}_0$`)`: reset the simulator to the fixed initial state $\boldsymbol{x}_0$ and return the initial observation

$\boldsymbol{x}_0$

- `step_simulation(`$\boldsymbol{u}$`)`: Take 1 AST step by stepping the simulator $\frac{\Delta_A}{\Delta_S}$ steps with action $\boldsymbol{u}$, where Δ_A is the AST step size and Δ_S is the simulator step size, both in seconds. Return the state $\boldsymbol{x}_{j+1}$, where j denotes the current AST step.

- `is_terminal()`: return true if the simulation has finished, i.e., if the number of episode steps, denoted j, equals the maximum number of AST steps N_{ast}, or if $\boldsymbol{x}_{j+1} \in E$, i.e., if a failure is detected

A simplified overview of the AST algorithm, implemented through the AST Python Toolbox and interfacing with the Zeabuz COLAV simulator, is presented in Algorithm 1:

Algorithm 1 AST algorithm

```
1: Initialize AST agent with policy $\boldsymbol{\pi_{\theta_0}}$
2: Initialize value function estimate $V_0$
3: for k = 0, 1, ... to k = $N_{epoch}$ do
4:     $S_e \leftarrow 0$
5:     while $S_e < bs$ do
6:         $\boldsymbol{x_0} \leftarrow$ reset($\boldsymbol{x}_0$)
7:         terminal $\leftarrow False$
8:         $j \leftarrow 0$
9:         while not terminal do
10:            $\boldsymbol{x}_{j+1} \leftarrow$ step_simulation($\boldsymbol{u}$)
11:            terminal $\leftarrow$ is_terminal()
12:            Obtain step reward $R(\boldsymbol{x}_{j+1}, \boldsymbol{u})$
13:            $\boldsymbol{x_j} \leftarrow \boldsymbol{x}_{j+1}$
14:            $j \leftarrow j + 1$
15:        end while
16:        $S_e \leftarrow S_e + j$
17:    end while
18:    Compute batch rewards $\widehat{R}_k$
19:    $\widehat{A}_k \leftarrow$ GAE($V_k$)
20:    Compute policy gradient estimate $\widehat{\nabla J(\boldsymbol{\theta}_k)}$ from $\widehat{A}_k$ and $\boldsymbol{\pi_{\theta_k}}$
21:    $\boldsymbol{\pi_{\theta_{k+1}}} \leftarrow TRPO(\widehat{\nabla J(\boldsymbol{\theta}_k)})$
22:    Update value function estimate $V_k$ used in GAE by regression on $\widehat{R}_k$
23: end for
```

where N_{epoch} is the number of epochs, S_e is the number of AST steps in the epoch and bs is the *bath size*. The hyperparameters for each experiment configuration are specified in Section 3.4 and Section 3.5.

Additional interfacing was done to enable experiment features such as the training heuristic or optimizing for improper time steps, where the AST agent also obtained information about the number of improper time steps and the adversary risk measure.

3.2.2 DRL Implementation

In order to optimize the AST MDP, an RL solver must be implemented. When AST was first proposed by Lee, Kochenderfer, Mengshoel, Brat, et al. (2015), the method was implemented using MCTS as the RL solver. In the subsequent literature, other RL solvers have been proposed and consequently the AdaptiveStressTesting Toolbox comes with a variety of solvers. In this work, a DRL solver is chosen as in Koren, Alsaif, et al. (2018), where the DRL solver was implemented as a policy gradient method using TRPO as the policy stepping method with GAE to estimate the policy gradient.

The policy of the RL agent is implemented as a NN, and the policy parameter vector $\boldsymbol{\theta}$ makes up the NN weights. The policy maps the RL state input vector, consisting of the position of the ferry and the adversary, to the mean of a Gaussian distribution with a state-independently parameterized standard deviation. The AST action $\boldsymbol{u}_{ast}$ is then drawn from the resulting distributions. The process is illustrated in Figure 3.1 for the Part 1 case, further described in Section 3.4.

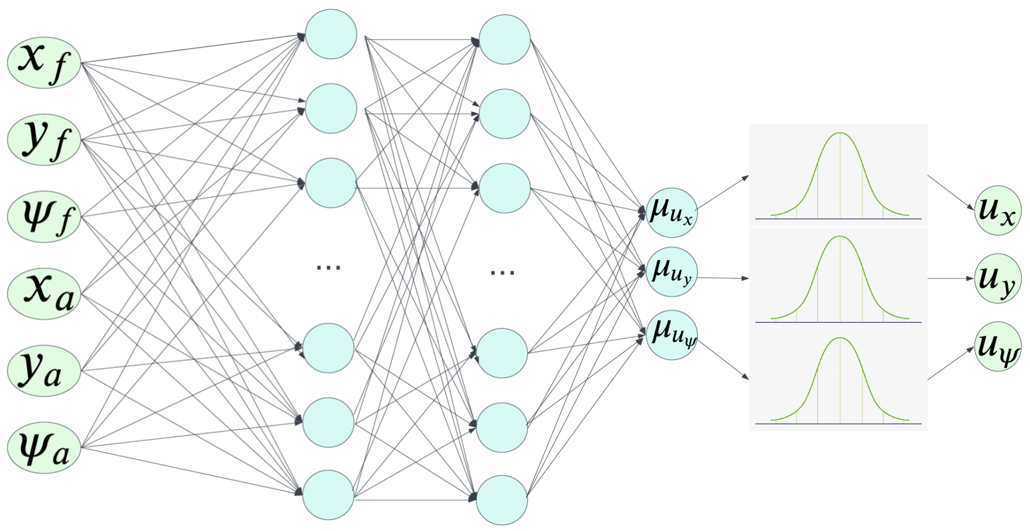


Figure 3.1: Illustration of the DRL network implementation, as it is implemented in Part 1. The state of the RL agent, consisting of the ferry and the adversary positions, is fed into the parameterized policy NN. The action vector $\boldsymbol{u}$ is drawn from Gaussian distributions with the NN output as mean.

The stochasticity introduced in the last steps, where the action is drawn from a probability distribution, makes the method exploratory by nature. Still, the level of randomness typically decreases throughout the training as the TRPO update rule makes it exploit discovered rewards. This makes the TRPO algorithm prone to becoming stuck in local optima.

3.3 Zeabuz COLAV simulator

The simulator used for the AST experiments performed in this thesis is retrieved from Zeabuz and referred to as the Zeabuz COLAV simulator. It is constructed to test the COLAV system of the milliAmpere ferry in a simplified, non-full-scale manner to reduce the complexity and run time needed to test different COLAV schemes and bring focus to the workings of the COLAV system only. Thus, some simulator functionality is simplified or approximated with simple models, such as adversary vessel behaviour or sensory information. The simulator is written in Python, which makes integration with the Python AST ToolBox simple. Integration methods of different complexity are available in the simulator, but simple Euler integration was used as this gave satisfying results.

The Zeabuz COLAV simulator allows for simulations using two different vessel classes: a milliAmpere vessel class, which is constructed according to the full 3-DOF dynamics described in Equation (3.13), with the kinematic parameters of the milliAmpere vessel. Several controllers are also implemented and easily applied to the vessel models. A simple sensory module is used in the simulations, which ables the ferry to be aware of the position and velocity of adversary vessels. In order to simulate approximate milliAmpere behaviour, the milliAmpere vessel model is equipped with a SP-VP controller which again uses a simplified Dynamic Positioning (DP) controller to position the vessel. The DP system takes in the state and velocity reference from the SP-VP system and uses the Proportional–Integral–Derivative (PID) principle to provide a control input $\boldsymbol{\tau}$ needed to reach the reference:

$$\begin{aligned} \boldsymbol{\tau}_p &= -K_p\boldsymbol{\eta}_e = -K_p(\boldsymbol{\eta} - \boldsymbol{\eta}_{ref}) \\ \boldsymbol{\tau}_i &= -K_i \int \boldsymbol{\eta}_e \\ \boldsymbol{\tau}_d &= -K_d\dot{\boldsymbol{\eta}}_e = -K_d(\boldsymbol{R}_{z,\psi}\boldsymbol{\nu} - \dot{\boldsymbol{\eta}}_{ref}) \\ \boldsymbol{\tau}_{pid} &= \boldsymbol{R}_{z,\psi}^T(\boldsymbol{\tau}_p + \boldsymbol{\tau}_i + \boldsymbol{\tau}_d). \end{aligned} \tag{3.1}$$

The DP controller also accounts for the damping effect of the vessel. The full control input is then described by Equation (3.2):

$$\begin{aligned} \tilde{\boldsymbol{D}}(\boldsymbol{\nu}) &= \tilde{\boldsymbol{D}}_L + \tilde{\boldsymbol{D}}_{NL}(\boldsymbol{\nu}) \\ \boldsymbol{\tau} &= \tilde{\boldsymbol{D}}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{\tau}_{pid} \end{aligned} \tag{3.2}$$

where $\tilde{\boldsymbol{D}}_L$ and $\tilde{\boldsymbol{D}}_{NL}(\boldsymbol{\nu})$ are diagonal matrices which approximate the linear and nonlinear damping terms of the milliAmpere dynamic.

The second vessel class represents a simplified first-order vessel model based on the kinematic equation

$$\begin{aligned} \dot{\boldsymbol{\eta}} &= \boldsymbol{R}_{z,\psi}\boldsymbol{\nu} \\ \dot{\boldsymbol{\nu}} &= -\frac{1}{\boldsymbol{T}}\boldsymbol{\tau} \end{aligned} \tag{3.3}$$

where $\boldsymbol{\tau}$ is given as the error between the current velocity and the reference velocity, i.e., $\boldsymbol{\tau} = \boldsymbol{\nu}_e = \boldsymbol{\nu} - \boldsymbol{\nu}_{ref}$. The reference velocity is given by a simple speed-heading proportional controller providing the reference as

$$\boldsymbol{\nu}_{ref} = \begin{bmatrix} u_{ref} \\ 0 \\ K_{\psi}(\psi - \psi_{ref}) \end{bmatrix} \tag{3.4}$$

where the vector of references $\boldsymbol{\Gamma} = [u_{ref}, \psi_{ref}]$ is either set initially as a constant reference for the vessel to follow, or is updated during the simulation by e.g. a LOS guidance controller as described in Section 2.5.2.

In Part 1 of the thesis, described in Section 3.4, the milliAmpere model will be used to simulate approximate milliAmpere behaviour. The simulations do, however, not need to include a milliAmpere model and can just as well be run with only first-order vessels, which will be the case in Part 2 of the thesis, described in Section 3.5.

The Zeabuz COLAV simulator also inhibits a useful plotting module where the resulting simulation output can be animated. The plotting module also allows for simulation metrics such as SP-VP details to be animated as well. Examples from a simulation with the ferry and one adversary vessel with and without the SP-VP details are shown in Figure 3.2 and Figure 3.3.

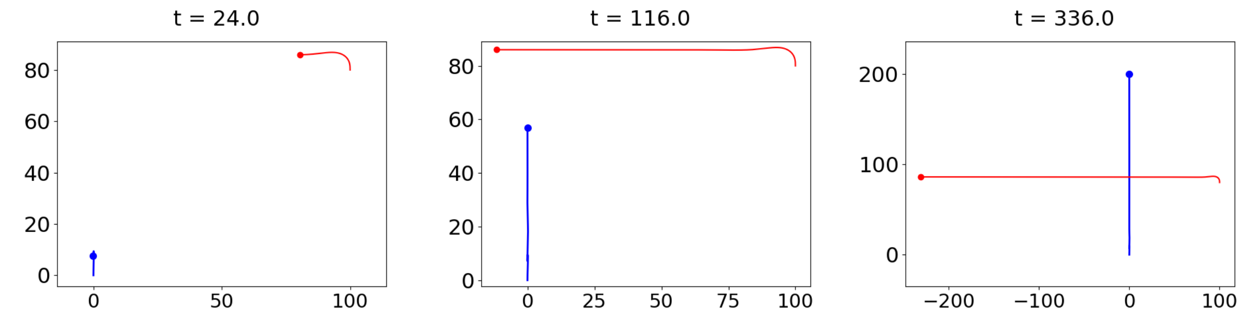


Figure 3.2: Snapshots from animated simulation with the ferry and one adversary vessel.

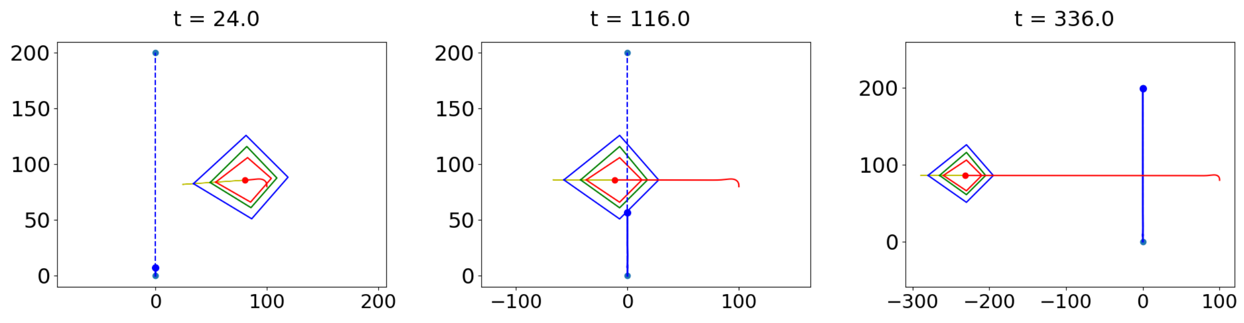


Figure 3.3: Snapshots from animated simulation with the ferry and one adversary vessel, with the SP-VP regions illustrated around the adversary.

As the AST method has the potential to uncover hundreds, sometimes thousands, of failure trajectories, analyzing them one by one in these animations is cumbersome and inefficient. To aid the analysis of the results, an additional plotting module was constructed. The module plots the entire trajectory of both the ferry and the potential adversaries, with lines that indicate positions for simultaneous time steps. Information about the heading of the vessels was also included, in addition to the ferry's velocity and position estimate of the adversary. The plotting module will be illustrated in both Chapter 3 and Chapter 4, see e.g. Figure 3.5 for an example plot.

3.4 Part 1: The SP-VP case

The first part of this thesis is spent on further experiments with the case study of the SP-VP system, which was first examined in (Hjelmeland, 2021).

3.4.1 Previous methodology

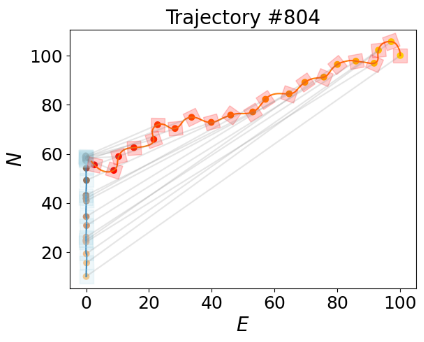
Experiments were conducted on the SP-VP system in Hjelmeland (2021), using the same COLAV simulator as used in this thesis. An initial AST problem formulation was presented, with an MDP reward function implemented as described in Section 2.4.2. The reward function, repeated here for simplicity, is:

$$R = \begin{cases} 0 & \text{if } \boldsymbol{x} \in E \\ -\alpha - \beta \mathrm{D}, & \text{if } \boldsymbol{x} \notin E, t \geq t_{end} \\ -\log(1 + M(\boldsymbol{u}, \boldsymbol{\mu_u} | \boldsymbol{x})), & \text{if } \boldsymbol{x} \notin E, t < t_{end} \end{cases} \tag{3.5}$$

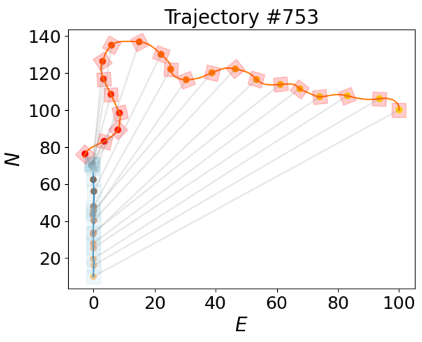
The covariance matrix used in the Mahalanobis distance in the reward function was initialized to the identity matrix and stayed unaltered as this gave reasonable results. An identity covariance matrix indicates that the standard deviation of all variables is equal to 1, and the Mahalanobis distance thus equals the square of the dot product of the action vector with itself. The mean action $\boldsymbol{\mu_u}$ is set to zero to penalize high action values, as this is deemed unlikely.

The results of Hjelmeland (2021) were promising as the AST method did identify many failure modes. However, the majority of identified failure modes showed scenarios where the adversary behaved very aggressive, as it drove straight into the ferry, as depicted in Figure 3.4(a). A training heuristic was introduced to improve the model of the probability of the adversary behaviour, which is used in the reward function to make AST find the most likely failures. A risk measure was introduced, based on the proximity of the adversary to the ferry, and whether or not the adversary was heading straight towards the ferry. If the AST action applied at a time step t resulted in an increased risk measure at time $t + 1$, a penalty would be issued. The intention of the heuristic was based on the hypothesis that it would make the AST method optimize for action sequences where the adversary did not expose risk-increasing behaviour and thus act less aggressive. The resulting failure modes did show an effect by applying the heuristic, as the adversary spent less time heading straight toward the ferry. However, in the majority of the failures, the adversary exposed very aggressive behaviour in the last couple of time steps, by e.g., suddenly turning and accelerating into a collision with the ferry, as in the example depicted in Figure 3.4(b). The results were interpreted as such that the high reward for collision made it worth exposing some aggressive behaviour in the last time steps, as the penalty for this would be insignificant compared to the reward of collision. Because of these results, it was considered to be less constructive to perform more work on the probability model of the adversary, but instead use approaches from the literature which had shown success in the problem of unavoidable failures, like the ones described in Section 2.4.2. The first part of this thesis is thus dedicated to the implementation of these approaches.

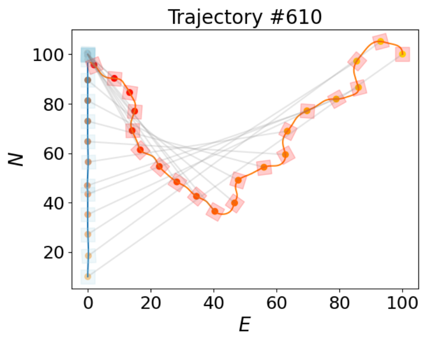
There were also certain aspects of the way the simulations were conducted that were considered to be unnecessary or in disfavour of obtaining the relevant failure modes, such as the fact that failures were detected *after* the ferry had reached and stopped in the final waypoint in $(x_f, y_f) = (100, 0)$, as depicted in Figure 3.4(c). These failures are considered irrelevant as the ferry is standing still, and an augmentation was made to omit cases like these by ignoring a failure if this was the case. To increase the potential ways the collision could occur, the waypoint issued to the ferry was also increased to $(x_f, y_f) = (200, 0)$ and the initial position of the adversary was altered from $(x_a, y_a) = (100, 100)$ to $(80, 100)$. In addition, it was decided to terminate the simulation if the ferry reached the final waypoint $(x_f, y_f) = (200, 0)$, to avoid simulating many cases where the adversary continued to search without the possibility of obtaining failure. The number of epochs, N_{epochs} was also decreased from 50 to 30 in this work compared to Hjelmeland (2021), as the RL agent often converged before it reached 30 epochs and a reduction of epochs would reduce the total simulation time dramatically.



(a) Baseline



(b) With heuristic



(c) Baseline, adversary attacks after the ferry has stopped.

The initial problem formulation of Hjelmeland (2021) is presented in this thesis as a baseline problem formulation, with the aforementioned augmentations, as it is constructive to have a baseline version of the system to compare the results of further problem variations to, and because the said system is further investigated in this work by restricting the action space of the RL agent.

3.4.2 New experiments

This thesis presents new experiments using AST to test the SP-VP COLAV system, for further testing of the SP-VP system as well as further adaptation of the AST method to the maritime domain. The test scenario is a little altered compared to the one presented in Hjelmeland (2021) as described in Section 3.4.1. The scenario chosen for the tests is a crossing scenario, with one adversary vessel passing from the starboard side of the ferry. This scenario was chosen as it constitutes a situation where the COLREGs rules are clear, as it is said explicitly in Rule 15 that the vessel which has the other vessel on its starboard side shall keep out of the way. The scenario is depicted in Figure 3.5.

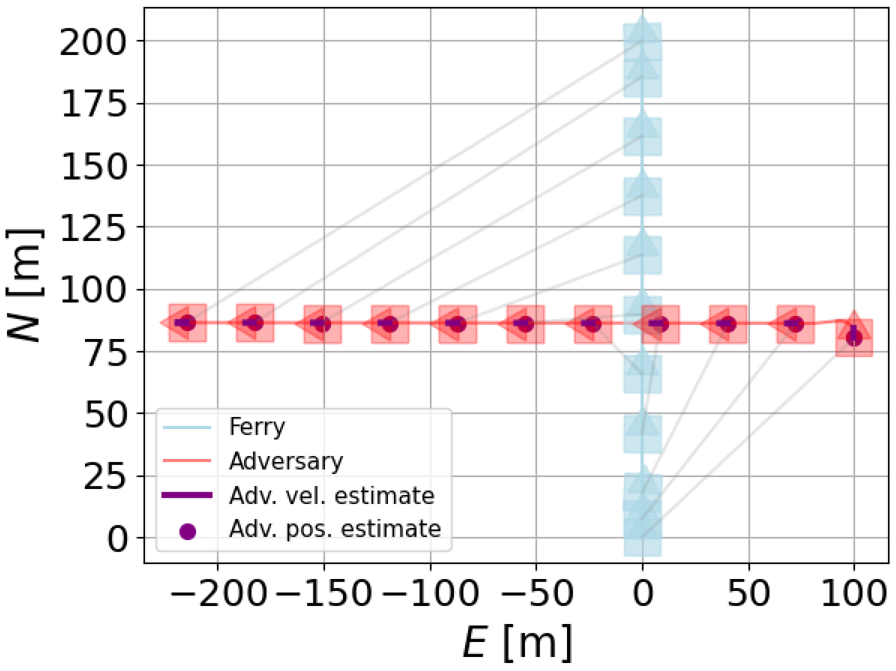


Figure 3.5: Chosen test scenario, illustrated by the additionally created plotting module described in Section 3.3. The grey lines connecting the vessels indicate positions for simultaneous time steps.

The AST agent applies actuation in the adversary dynamics described in Equation (3.3), such that they become:

$$\begin{aligned} \dot{\boldsymbol{\eta}} &= \boldsymbol{R}_{z,\psi}\boldsymbol{\nu} \\ \dot{\boldsymbol{\nu}} &= -\frac{1}{\boldsymbol{T}}\boldsymbol{\tau} + \boldsymbol{u}_{ast} \end{aligned} \tag{3.6}$$

where $\boldsymbol{u}_{ast}$ is the vector of AST actions, where each component affect a DOF of the adversary:

$$\boldsymbol{u}_{ast} = \begin{bmatrix} u_x \\ u_y \\ u_\psi \end{bmatrix} \tag{3.7}$$

The AST problem formulation is augmented in different ways in a total of five different experiment variations which are all attempts to omprove the relevance of the failure modes to system developers. The five different problem variations are:

1: The baseline variation

2: The dissimilarity measure

3: Optimizing for improper behaviour

4: Optimizing for improper behaviour with estimation noise

5: Optimizing for improper behaviour with estimation time delay

Variation **2** introduces the use of the dissimilarity measure described in Section 2.4.2 by augmenting the reward function R. In the last three variations, optimization for improper behaviour is implemented as described in Section 2.4.2 which includes alterations of the AST problem formulation on a more fundamental level, by augmenting both the reward function and the definition of failure. Finally, the simulation scenario is tweaked in the last two variations to allow for errors in the ferry's tracking system, making the estimates of the adversary prone to both estimation noise and time delays. Additionally, for every problem variation, the experiment is conducted three times with different restrictions on the AST agent action space for the adversary movement. This is done to search for failure modes where the adversary is less able to do abrupt movements, as seen in the results of (Hjelmeland, 2021).

First, the different problem variations are described. Then, the concept of restricting the adversary action space is elaborated on. Lastly, an overview of the AST hyperparameters is presented.

3.4.3 Variation 1: The baseline

This baseline system was implemented and presented in (Hjelmeland, 2021) and (Hjelmeland et al., 2022), but is presented again here as a baseline system is constructive for comparison and due to the fact that further experiments were performed with a restricted AST action space. The variation is described in Section 2.4.2 and Section 3.4.1.

3.4.4 Variation 2: The dissimilarity measure

The dissimilarity measure presented by Koren and Kochenderfer (2020) and discussed in Section 2.4.2 is implemented as follows: whenever a failure trajectory ρ_s is obtained, the trajectory is compared to a subset of the trajectories already obtained, $\boldsymbol{\rho} = \{\rho_0, \rho_1, \rho_2, ..., \rho_n\}$. The trajectories may be of different lengths and therefore the trajectories are segmented into n segments. A sequence of COMs of the segments is then computed, denoted $\boldsymbol{c} = \{c_0, c_1, ..., c_n\}$, to provide representative points of the segments, which in turn are used to compare the trajectories. The COM of a segment is denoted $(\bar{x}, \bar{y})$ and is calculated according to Liu et al. (2012):p. 3:

$$(\bar{x}, \bar{y}) = \left(\frac{\sum_{i=1}^{n-1}(x_{i+1}^2 - x_i^2)}{2\sum_{i=1}^{n-1}(x_{i+1} - x_i)}, \frac{\sum_{i=1}^{n-1}(y_{i+1}^2 - y_i^2)}{2\sum_{i=1}^{n-1}(y_{i+1} - y_i)} \right) \tag{3.8}$$

Let the dissimilarity measure between two trajectories ρ_1 and ρ_2 be denoted $D(\rho_1, \rho_2)$. The dissimilarity is then computed as the distance between the COMs of the trajectories, computed as the distance between them:

$$D(\rho_1, \rho_2) = \frac{1}{n} \sum_{i=1}^{n} \|c_1^i - c_2^i\| \tag{3.9}$$

The resulting reward function becomes:

$$R = \begin{cases} \frac{\gamma}{\mu} \sum_{i=1}^{\mu} D(\rho_s, \rho_i) & \text{if } x \in E \\ -\alpha - \beta \text{dist}(p_f, p_a), & \text{if } x \notin E, t \geq t_{end} \\ -\log(1 + M(u, \mu_u | x)), & \text{if } x \notin E, t < t_{end} \end{cases}$$

where γ is a user-specified constant gain and μ is the length of $\boldsymbol{\rho}$ defined as $\min(k, k')$ where k is user-specified and describes the maximum number of top failure trajectories ordered by reward, and k' is the number of failure trajectories obtained (Koren and Kochenderfer, 2020). The parameters used in the dissimilarity measure are illustrated for the trajectory of a vessel in Figure 3.6.

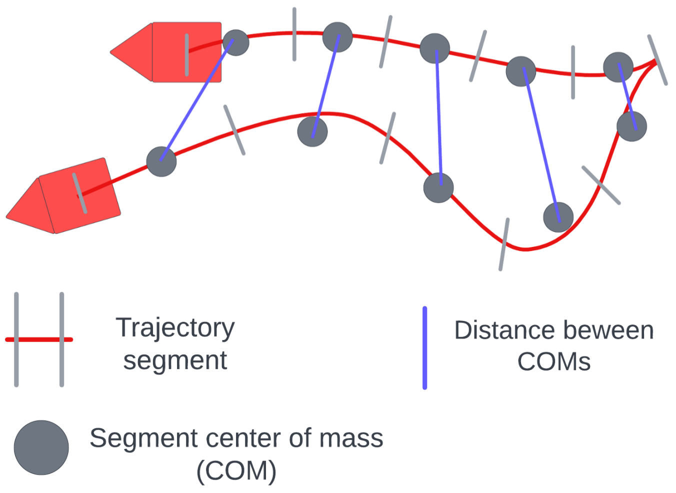


Figure 3.6: Illustration of parameters used in dissimilarity measure using 5 segments.

3.4.5 Variation 3: Optimizing for improper behaviour

In the results of (Hjelmeland, 2021), the AST agent found failure scenarios where the adversary attacked the ferry in an aggressive manner. The SP-VP and most COLAV systems are not designed to avoid aggressive, deliberate attacks of other vessels, and thus, these unavoidable failures are deemed to be of little value to system designers. As described in Section 2.4.2, unavoidable failures such as these are often found and converged to by the AST agent.

To deal with the unavoidable failure problem, solutions proposed in the AST literature were evaluated. As the maritime COLAV case considered in this thesis is in many ways similar to the automobile-pedestrian COLAV system described in Koren and Kochenderfer (2020), a similar method to their solution was chosen. The solution consists of an augmentation to the reward function and the definition of the failure space, as elaborated on in Section 2.4.2.

The solution proposed by Koren and Kochenderfer (2020) requires a definition of improper behaviour, to which the rules described in RSS were applied. As the COLREGs can be said to be the marine equivalent of the RSS, a definition based on the COLREGs was applied. In the simulation scenario, the adversary is crossing from the starboard side of the ferry. According to Rule 15 of the COLREGs, the proper behaviour in crossing cases is that the vessel which has the other vessel on its starboard side is to give way by either lowering its speed to let the vessel pass or by navigating to pass astern the crossing vessel. As the SP-VP system is only able to adjust the vessel speed, the only option to behave properly according to the COLREGs is to lower the vessel speed to a sufficient level and let the adversary pass. Improper behaviour is then recognized if the ferry has a speed above some predefined limit while in close proximity to the adversary, while the adversary is crossing. These conditions are formulated as the logical expression :

$$\begin{aligned} S &:= U > U_{max} \\ P &:= D < D_{min} \\ A &:= \theta \in [\delta_{min}, \delta_{max}] \end{aligned} \tag{3.10}$$

where S denotes a violation of the speed condition, i.e., when the ferry speed U is above a limit value U_{max}. The proximity condition is denoted P and evaluates to true if the proximity D is less than a minimum proximity D_{min}. Condition A is true if the angle from the ferry to the adversary θ is within the angle sector defined by $[\delta_{min}, \delta_{max}]$, ensuring that the adversary is crossing ford the starboard beam of the ferry and thus that the ferry is still in a crossing situation.

The parameters of the improper conditions were set as follows:

Parameter	Value
U_{max}	0.2 m/s
D_{min}	30 m
δ_{min}	0 rad
δ_{max}	$\frac{\pi}{2}$ rad

A time step is then deemed improper if the following logical expression evaluates to true:

$$improper := S \wedge P \wedge A \tag{3.11}$$

Note that this approach is an oversimplified implementation of COLREGs Rule 15, which is highly adapted to the simulation scenario. A more comprehensive COLREGs compliance framework is implemented in Section 3.5.

3.4.6 Variation 4 & 5: Optimizing for improper behaviour with estimation noise and time delay

In the COLAV simulator, the milliAmpere control system uses a tracking module which simulates sensory information about obstacles and adversary vessels. It is possible to introduce noise in the tracker measurements to investigate the effects of sensory errors on the COLAV system, both in the position and velocity estimate of potential adversaries.

Let the ferry's estimate of the adversary be denoted $\tilde{\boldsymbol{q}_a}$ estimate noise be denoted $\boldsymbol{w_n}$. The tracker estimate is then given as:

$$\tilde{\boldsymbol{q}_a} = \boldsymbol{q_a} + \boldsymbol{w_n} = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix} + \begin{bmatrix} w_x \\ w_y \\ w_{\dot{x}} \\ w_{\dot{y}} \end{bmatrix} \tag{3.12}$$

where the noise is updated at every time step by the noise vector $\boldsymbol{u_n}$. The noise dynamics are described by the first-order model:

$$\dot{\boldsymbol{w}}_{\boldsymbol{n}} = \frac{1}{K}\boldsymbol{w_n} + \frac{1}{K}\boldsymbol{u_n}$$

where K is a predefined constant.

As sensory errors are a possible cause of malfunction of the COLAV system, this was introduced into the system as in AST experiments to identify failures where the estimation errors lead the system to fail. The system with sensory errors was combined with optimizing for improper behaviour to focus on sensory errors that lead the system to behave improperly and omit failures where the adversary deliberately attacks the ferry. The AST agent is able to induce noise in both the position and velocity estimate of the adversary by controlling the values of (u_p, u_v) in the noise vector in Equation (3.12):

$$\boldsymbol{u_n} = \begin{bmatrix} u_p \\ u_p \\ u_v \\ u_v \end{bmatrix}.$$

Note that the added noise in position and velocity is the same in x and y to simplify the action space of the AST agent.

A time delay of T seconds is also introduced in the ferry's estimates of the adversary as an action of the AST agent. The time delay is implemented by withholding new measurements and feeding the ferry with old measurements corresponding to the given time delay. The time delay at time step t is saturated by the previous time delay as measurements cannot be delayed further back in time than the last measurement, expressed as:

$$T_t = \min(T_t,\ T_{t-1} + \Delta_S).$$

The complete vector of AST actions becomes:

$$\boldsymbol{u}_{ast} = \begin{bmatrix} u_x \\ u_y \\ u_\psi \\ u_p \\ u_v \\ T \end{bmatrix}.$$

Note that the aforementioned action space restrictions only apply to the AST agent actions which affect the motion of the vessel. The noise- and time delay action space was experimented with until effects were seen in the estimates of the adversary. The resulting action space configurations for the noise and time delay actions were:

$$\begin{aligned} u_{p_{max}} &= 2 \\ u_{v_{max}} &= 2 \\ T_{min} &= 0, T_{max} = 15s \end{aligned}$$

The covariance matrix of the Mahalanobis distance function in the reward function can no longer stay the identity as the noise- and time delay actions are of much larger magnitude than the motion-related actions. If not altered, the AST agent quickly reduced the noise and time delay actions to similar values to that of the motion actions, to reduce the penalty on the magnitude of the actions. The covariance matrix was set to

$$\mathbf{S} = \begin{bmatrix} \sigma_x^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_y^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_\psi^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_p^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_v^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_T^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10\,000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10\,000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100\,000 \end{bmatrix}$$

where σ_i^2 denotes the variance of variable i. This covariance matrix indicates that the standard deviations of the distributions of u_p, u_v and T, respectively σ_p, σ_v is 100 and $\sigma_T = 1000$, which might seem a bit high, but is set in resemblance to the standard deviation of the motion actions which are also set with standard deviations of 10 - 100 times their own maximum value.

3.4.7 Restricting the adversary action space

The Python AST toolbox allows for infinitely big action spaces. However, in this SUT it was considered appropriate to restrict the action space both to reduce the size of the action space and to omit actions that lead to entirely unrealistic behaviour of the adversary. The actions are restricted to the action space U,

$$
\begin{aligned}
&\boldsymbol{u}_{ast} \in U, \ U = [-\boldsymbol{u_{max}}, \boldsymbol{u_{max}}] \\
&\boldsymbol{u_{max}} = \begin{bmatrix} u_{max} \\ u_{max} \\ u_{max} \end{bmatrix},
\end{aligned}
\tag{3.13}
$$

where u_{max} is a user-specified constant which, in this test case, is set to low magnitude values as the adversary is highly affected by the actuation due to the first-order dynamics. In Hjelmeland (2021), the actions were restricted to $u_{max} = 0.1$. This action space configuration resulted in a movement that seemed realistic, but the resulting failure modes showed adversary behaviour that was either very aggressive or where the movement of the adversary varied abruptly. The experiments in this thesis are therefore carried out using three action space configurations:

$$
\begin{aligned}
U_{0.1} &: u_{max} = 0.1 \\
U_{0.05} &: u_{max} = 0.05 \\
U_{0.01} &: u_{max} = 0.01
\end{aligned}
\tag{3.14}
$$

These action space configurations were chosen to obtain three different experiment cases with different levels of adversary aggressiveness and abrupt movement. As mentioned, the configuration $U_{0.1}$ gives an identical action space to the one in Hjelmeland (2021). With the action space configuration $U_{0.05}$, the adversary is able to perform some aggressive manoeuvres by heading straight towards the ferry, but it is less able to turn and move abruptly. With the configuration $U_{0.01}$, the adversary is only able to adjust its speed and course by a small amount, which led to scenarios where it performs a realistic and highly predictable crossing.

3.4.8 Discussion on the SP-VP method

It is important to emphasize an augmentation made for the adaption of the SP-VP system to the experiments conducted in this thesis as well as in Hjelmeland (2021): in the full milliAmpere system, the SP-VP module is included as a part of a more comprehensive state-machine system. The SP-VP system is designed to *fail-fast*, meaning that when the conditions of a scenario are such that the SP-VP system finds no feasible solutions, it fails, reports the failure back to the state machine and the state machine delegates control to other modules to e.g. make the vessel stop. As an approximate behaviour of this system, a modification was made to the SP-VP system not to crash when the situation is infeasible, but to issue a waypoint at the current position and with zero velocity to make it stop. This was done because simulations including the entire state-machine system would decrease the focus of the COLAV system by introducing several other modules which potentially could fail and cause a collision, making the test less modularized. However, the tests could be performed with the system in its entire form for more wholesome integration tests with a focus on COLAV. The behaviour seen in the experiments is thus an incomplete representation of how the milliAmpere ferry will behave in such situations.

3.4.9 AST hyperparameters

The hyperparameters for the simulator and the AST simulations used for the experiments of Part 1 are described in Section 3.4.9.

A collision is registered if the vessels are in proximity of d_{coll} meters. The simulation end time t_{end}, the number of AST steps N_{ast} and the batch size bs was altered when the action space of the agent was restricted by $u_{max} = 0.01$. This is because the adversary moved in a straightforward manner with few detours, which reduced the time necessary to simulate a full crossing scenario. This again reduced the number of AST steps necessary, which reduced the batch size needed to perform the same number of trajectories. See Section 3.2 for definitions of the parameters.

u_{max}	Hyperparameter	Value
All configurations	N_{epoch}	30
	$t0$	0 s
	d_{coll}	10 m
	Δ_S	0.1 s
	Δ_A	4 s
0.1, 0.05	t_{end}	400 s
	N_{ast}	100
	bs	20 000
0.01	t_{end}	200 s
	f_{thresh}	2%
	N_{ast}	50
	bs	10 000

3.5 Part 2: The COLREGS-compliant MPC COLAV case

The SP-VP COLAV system trades flexibility for robustness. In most cases with an approaching adversary, the system will prompt the ferry to stop and wait for the potentially dangerous situation to pass, instead of performing evasive manoeuvres such as changing course. This behaviour is desirable in commercial use as it limits the possible ways that the system can be the cause of failure. In order to test the AST method on a more dynamic COLAV system, a guidance algorithm was implemented using an MPC approach. This implementation is inspired by the work of Johansen et al. (2016) and Hagen et al. (2018), where an MPC layer is applied on top of the guidance system, with the ability to modify course and speed to avoid collision in a COLREGs-compliant manner while following a predetermined path.

The MPC system is tested in three scenarios where the COLREGs rules clearly state the proper behaviour. In order to challenge the system further, more adversary vessels are added in further experiments. The motivation for these experiments is to investigate how AST uncovers failures in a more dynamic system than the SP-VP, and to see the effect of multiple adversaries. Multiple adversaries could have been implemented in the SP-VP system as well, but wasn't as it was deemed unlikely that it would uncover more special situations as the SP-VP is so conservative, and also because the workings and potential of AST in multiple adversary scenarios would be better illustrated in simulations with a more reactive COLAV system.

3.5.1 Vessel implementations

All vessels, including the ferry, are in this part implemented as first-order vessels. This was due to time limitations which made it hard to test and tune the MPC configuration to work sufficiently for the milliAmpere model, combined with the fact that Hagen et al. (2018) use a simplified kinematic vessel model which in simulations gave minor result differences to a full 3-DOF vessel model. The dynamics of the first-order vessels are given by Equation (3.3).

For the adversaries, the speed-heading reference vector $\mathbf{\Gamma}_{ref} = [u_{ref}, \psi_{ref}]$ is set initially and kept constant throughout the episode, resulting in straight-forward behaviour.

For the ferry, the speed-heading reference vector $\mathbf{\Gamma}_{ref} = [u_{ref}, \psi_{ref}]$ is updated every Δ_{ref} seconds by the LOS guidance controller and subsequently modified by the MPC controller. The modified speed-heading reference, given as $\mathbf{\Gamma}_{ref_m} = [u_m u_{ref}, \psi_{ref} + \psi_m]$, is then fed to the speed heading controller, which gives the actuator input $\boldsymbol{\tau}$ in an identical manner as in the adversary case, as the velocity error $\boldsymbol{\nu}_e = \boldsymbol{\nu}_f - \boldsymbol{\nu}_{f_{ref}}$. The two controller schemes are visualized in Figure 3.7.

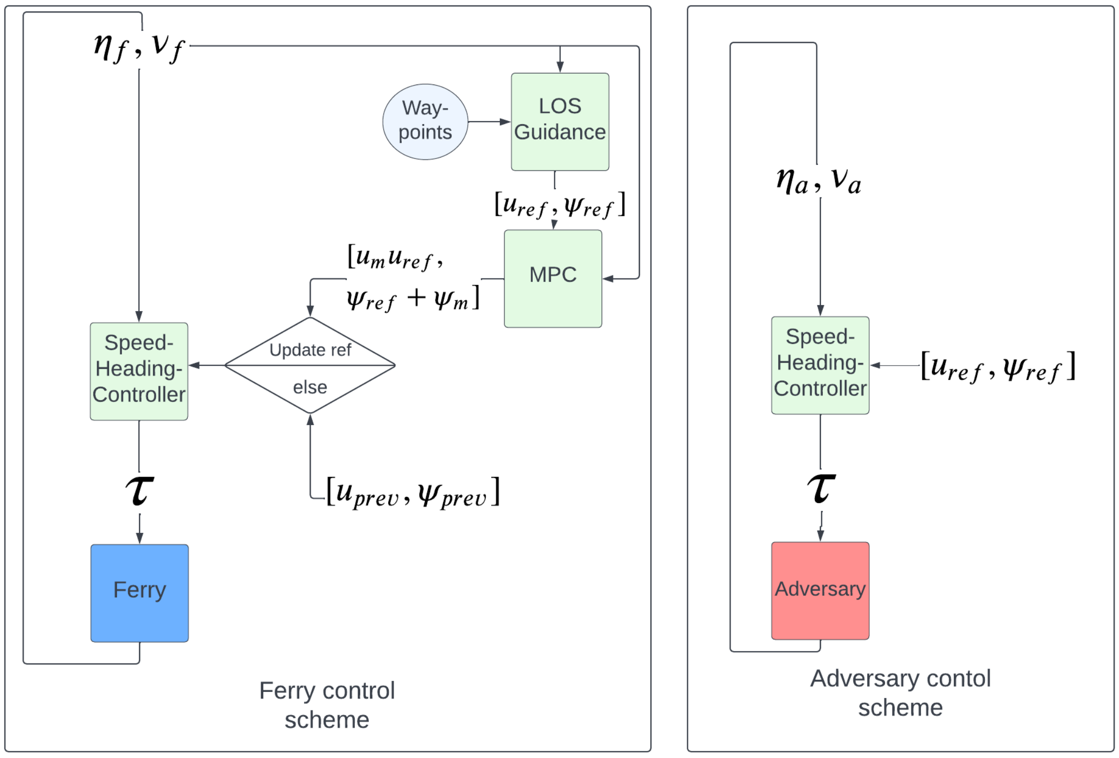


Figure 3.7: Illustrated control schemes for the ferry and potential adversaries in simulations in Part 2.

The decision block in the ferry control scheme in Figure 3.7 illustrates the if-statement of whether or not to update the reference vector. The *Update ref*-statement is true if the time step is such that it is time for a new reference update according to the refresh rate of $\frac{1}{\Delta_{ref}}$ Hz. The latest reference is kept for all time steps in-between updates.

3.5.2 LOS guidance implementation

The LOS guidance controller used in the ferry model is described in Table 3.1. See Section 2.5.2 for further description of the variables.

Parameter	Value
L_{LA}	30 m
L_{WP}	20 m

Table 3.1: LOS guidance controller parameters.

The initial position η_f and velocity ν_f of the ferry is, in all scenarios:

$$\boldsymbol{\eta}_f = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{\nu}_f = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.15}$$

meaning that the ferry starts still in the origin. The LOS controller is set up with four waypoints meant to be followed in sequential order:

$$\begin{aligned} &\mathbf{wp}_1 : \{N : 50, E : 0, u : 1\} \\ &\mathbf{wp}_2 : \{N : 100, E : 0, u : 1\} \\ &\mathbf{wp}_3 : \{N : 115.5, E : 0, u : 0\} \\ &\mathbf{wp}_4 : \{N : 130, E : 0, u : 0\}. \end{aligned}$$

The four waypoints are chosen as a result of experiments to make the ferry keep its nominal speed of 1 m/s for most of the trajectory until it slows down, in the end, to reach $(100, 0)$ in a smooth manner. As the waypoint switching distance L_{WP} is set to 20, $\mathbf{wp}_3$ is activated when the ferry reaches $(80, 0)$ which makes the ferry slow down in the last 20m. $\mathbf{wp}_4$ is set to ensure that the speed remains zero also after the ferry has switched from $\mathbf{wp}_3$. The resulting behaviour is shown in snapshots from simulations in Figure 3.8:

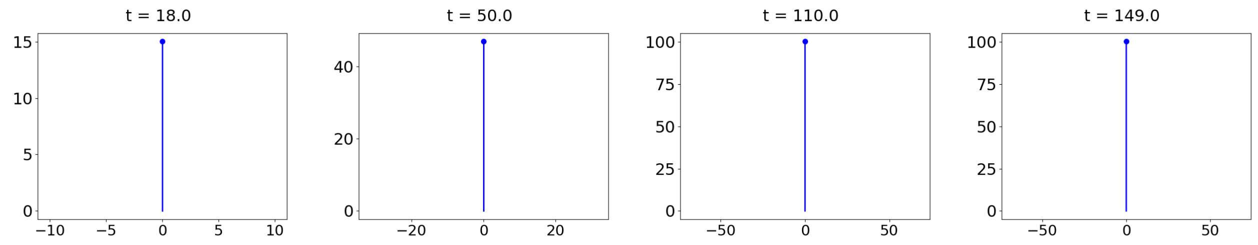


Figure 3.8: Snapshots from simulations in ideal scenario with no adversaries. The ferry stops in a smooth manner under the instructions of the LOS controller, in the desired endpoint $(N = 100, E = 0)$ and stands still until the end of the simulation.

3.5.3 MPC implementation

The possible set of control reference modifications are implemented as in Hagen et al. (2018), with:

- $\psi_m \in$ {-90°, -75°, -60°, -45°, -30°, -15°, 0°, 0°, 15°, 30°, 45°, 60°, 75°, 90°}
- $u_m \in$ {1, 0.5, 0}, i.e., {keep speed, slow down, stop}

which constitutes the same set of control behaviours as in Johansen et al. (2016), only without the speed factor $u_m = -1$, which corresponds to full backward propulsion. This configuration results in a total of 39 scenarios to be evaluated at every MPC update.

The cost function is implemented identically to the one described in Johansen et al. (2016), given in Equation (2.33), except that the grounding function $g(u_m, \psi_m)$ was omitted for simplification. The update interval Δ_{ref} was proposed to be around 5s by Johansen et al. (2016). This implementation uses an update interval of 2 seconds, as the vessels move quite fast and are in close proximity.

The MPC controller needs a model of the environment to make predictions. In this implementation, the model is given by the vessel models with their corresponding controllers as described in Section 3.5.1. The prediction is simply made by running the full simulator from the current time step until the end of the prediction horizon T_{mpc}. This means that in the ideal case without AST actions applied, the MPC predictions are exact. However, in the AST simulations, the MPC predictions do not include the effect of the AST actions on the adversary movements, which makes the implementation more similar to the real world and as the predictions do not necessarily match the exact behaviour of the vessels. However, due to the computational complexity of predicting the outcome of all the possible control behaviours, measures are made to simplify the predictions, including:

- The time step of the predictions is not the same as the simulator time step Δ_S, but set to Δ_{mpc}.
- The evaluation is not performed at every time step, but at an evaluation interval given by Δ_{eval}.

The distance measure d_i to adversary i is implemented as the distance from the ferry to the closest point on a sphere of radius r_a around the adversary position, representing a safety region around the adversary. Further, the simulation scenario is set up with a start time t_0 and final time t_{end}. The parameter values are described in Table 3.2.

Parameter	Value
ϵ	0.2 m/s
ϕ_{ahead}	$\frac{\pi}{4}$
t_0	0 s
t_{end}	150 s
r_a	5 m
Δ_{ref}	2 s
T_{mpc}	30 s
Δ_{mpc}	2 s
Δ_{eval}	2 s
t_0	0 s
t_{end}	150 s

Table 3.2: MPC parameters and hyperparameters for simulations in Part 2

See Section 2.6 for further descriptions of the parameters. The remaining parameters of the MPC implementation, such as the cost function parameters, are described in the following section, as two different MPC configurations are implemented.

3.5.4 MPC configurations

To illustrate how AST can be helpful for comparing different controller configurations, two different MPC configurations are implemented. The first configuration is a conservative one with a high cost for both COLREGs violations and collision. It also has a small cost of changing the speed of the vessel to easily allow it to slow down or stop the vessel when risk is prominent. The second configuration constitutes a more aggressive COLAV approach. The MPC controller is given a lower cost of collision and COLREGs violation and a very high cost of deviating from the speed and heading suggested by the LOS controller.

All adversary vessels are treated the same way, for simplicity. This eliminates the need for the subscript i in the constant definitions. As suggested by Johansen et al. (2016), K_{ψ_m} and Δ_{ψ_m} is defined in an asymmetrical manner, as:

$$K_{\psi_m} = \begin{cases} 5K_0 & \text{if } \psi_m < 0 \\ K_0 & \text{if } \psi_m \geq 0 \end{cases} \tag{3.16}$$

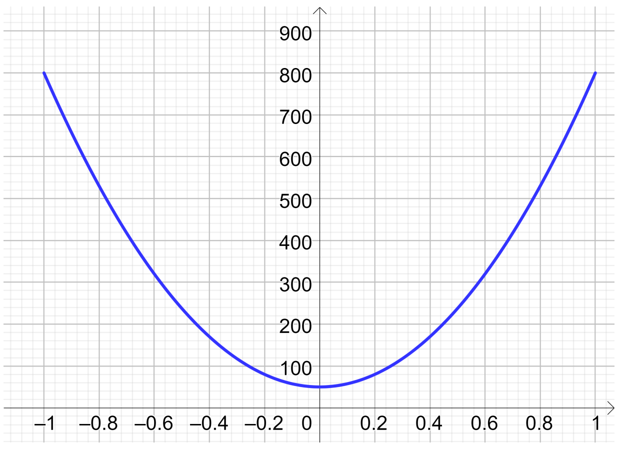
yielding a higher penalty of a change to the port side. The same is true for Δ_{ψ_m}. Let the change of reference angle compared to the last be denoted $\delta_\psi = \psi_m - \psi_{m_{last}}$. Δ_{ψ_m} is then implemented as:

$$\Delta_{\psi_m} = \begin{cases} 0.005(10 + 15\delta_\psi^2)) & \text{if } \delta_\psi < 0 \\ 0.0005(10 + 15\delta_\psi^2)) & \text{if } \delta_\psi \geq 0 \end{cases} \tag{3.17}$$

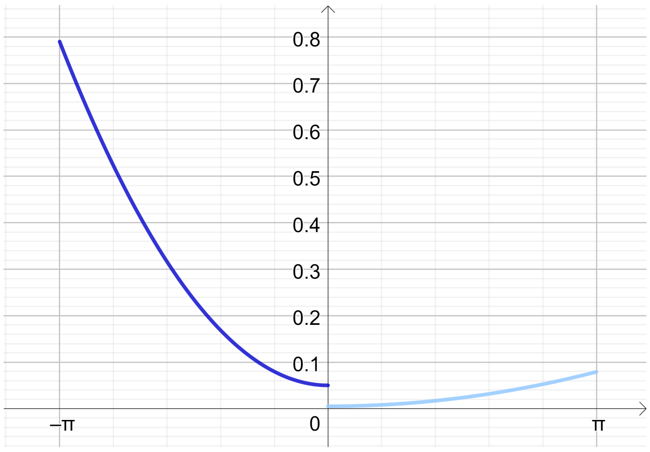
Δ_{u_m} is defined as a function of the change in velocity $\delta_{u_m} = u_m - u_{m_{last}}$:

$$\Delta_{u_m} = 50 + 750\delta_{u_m}^2 \tag{3.18}$$

The functions Δ_{u_m} and Δ_{ψ_m} are illustrated in Figure 3.9(a) and Section 3.5.4, respectively.



(a) Δ_{u_m} as a function of δ_{u_m}



(b) Δ_{ψ_m} as a function of Δ_{ψ_m}

The remaining MPC parameters used in the configurations are described in Table 3.3.

Configuration	Parameter	Value
Both	q	4
	p	1
	d_{safe}	50 m
	ϵ	0.2 m/s
Conservative	K_{coll}	10
	κ	1000
	K_{u_m}	0.005
	K_0	500
Aggressive	K_{coll}	1
	κ	100
	K_{u_m}	50000
	K_0	500

Table 3.3: Parameters used in the different MPC configurations.

The parameters were found by tuning in experiments. They might not constitute optimal MPC implementations but gave satisfying results for the purpose. See Section 2.6 for further descriptions of the functions and parameters.

The performance of the two controllers in the scenario cases with ideal conditions, i.e., without AST intervention, is presented in the following sections.

3.5.5 COLREGs testing scenarios

Three scenarios were chosen for testing, which all express a situation related to the COLREGs rules:

Scenario 1: A crossing situation where an adversary vessel is passing from the starboard side of the ferry, as in the case study of part 1, Section 3.4. According to COLREGs rule 15, the ferry becomes the *give-way* vessel as it has the adversary on its starboard side. This means that the ferry is obliged to take action, if possible, to avoid crossing in front of the adversary. Implemented with reference vector $\mathbf{\Gamma} = [1, -\frac{\pi}{2}]$ and adversary initial position and velocity:

$$\boldsymbol{\eta}_a = \begin{bmatrix} 50 \\ 50 \\ -\frac{\pi}{2} \end{bmatrix}, \ \boldsymbol{\nu}_a = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Scenario 2: A head on situation, where an adversary vessel starts north of the vessel and heads straight towards it. Rule 14 of the COLREGs dictates that the proper behaviour in such a situation is for both vessels to alter their course to the starboard side. Implemented with reference vector $\mathbf{\Gamma} = [1, \pi]$ and adversary initial position and velocity:

$$\boldsymbol{\eta}_a = \begin{bmatrix} 100 \\ 0 \\ \pi \end{bmatrix}, \ \boldsymbol{\nu}_a = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Scenario 3: An overtaking situation, where an adversary vessel starts North of the ferry and heads in the same direction, but with a lower speed than that of the ferry. COLREGs Rule 13 instructs the ferry to stay out of the way of the adversary as it becomes the overtaking vessel. Implemented with reference vector $\mathbf{\Gamma} = [0.5, 0]$ and adversary initial position and velocity:

$$\boldsymbol{\eta}_a = \begin{bmatrix} 20 \\ 0 \\ 0 \end{bmatrix}, \ \boldsymbol{\nu}_a = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

The scenarios are illustrated in Figure 3.10.

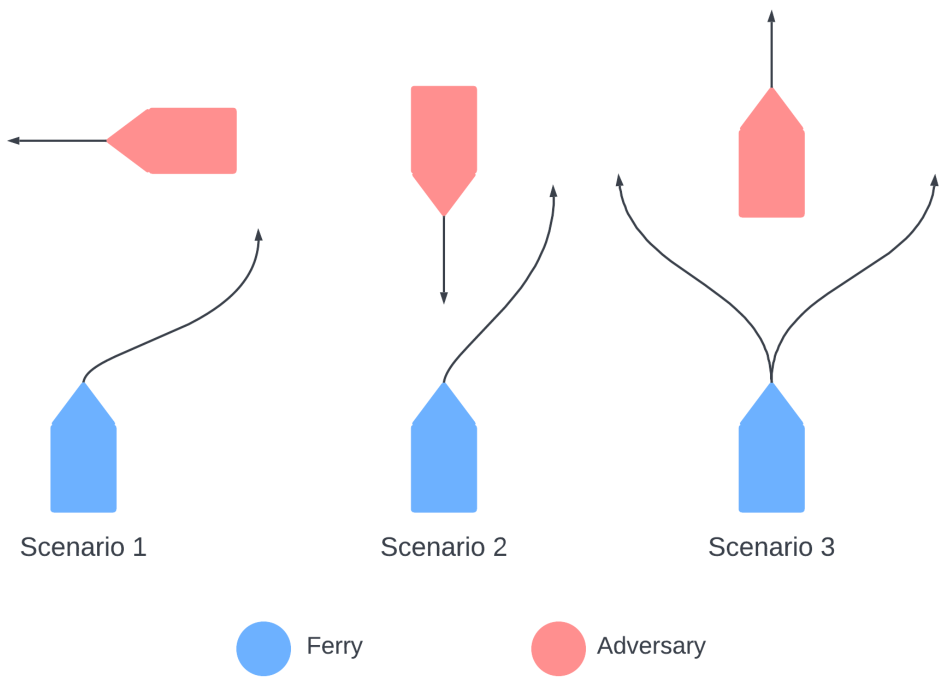


Figure 3.10: The three COLREGs based test scenarios used in AST simulations with correct COLREGs-compliant behaviours.

3.5.6 Simulations, scenario 1: Crossing

Snapshots from simulations of the scenario are depicted in Figure 3.11 and Figure 3.12. The conservative MPC COLAV system handles the crossing scenario by both turning to starboard side and stopping while the adversary passes. It continues to follow the path when the risk of doing so is sufficiently low and it is no longer violating the COLREGs when turning port. The aggressive configuration does not lower its speed but proceeds to pass astern the adversary, in compliance with COLREGs. Both behaviours comply with COLREGs rule 15.

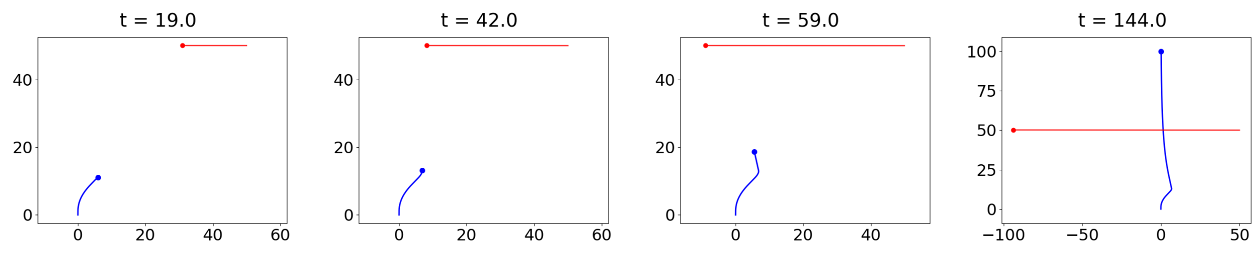


Figure 3.11: Snapshots from simulations with conservative configured MPC COLAV system in crossing scenario.

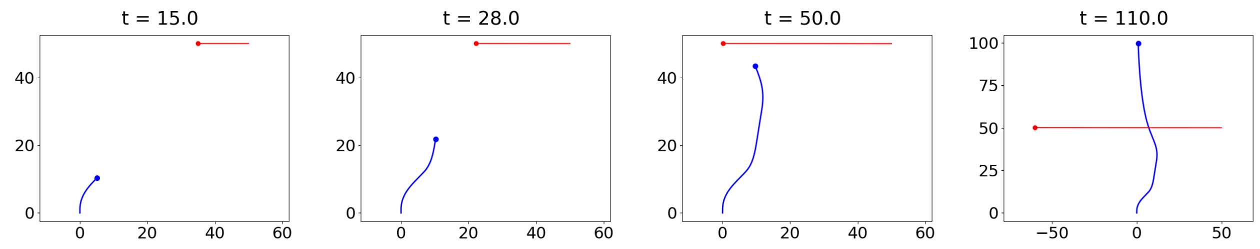


Figure 3.12: Snapshots from simulations with aggressive configured MPC COLAV system in crossing scenario.

3.5.7 Simulations, scenario 2: Head on

Snapshots from simulations of the scenario are depicted in Figure 3.13 and Figure 3.14. In the head on situation, the conservative MPC COLAV system handles the increasing risk of the approaching vessel by navigating to starboard side in compliance with COLREGs. It proceeds to stop for a while as it lets the adversary pass before it proceeds to follow the path. As in the crossing scenario, the aggressive configuration does not lower its speed, but passes the adversary vessel on the starboard side and efficiently returns to follow the path.

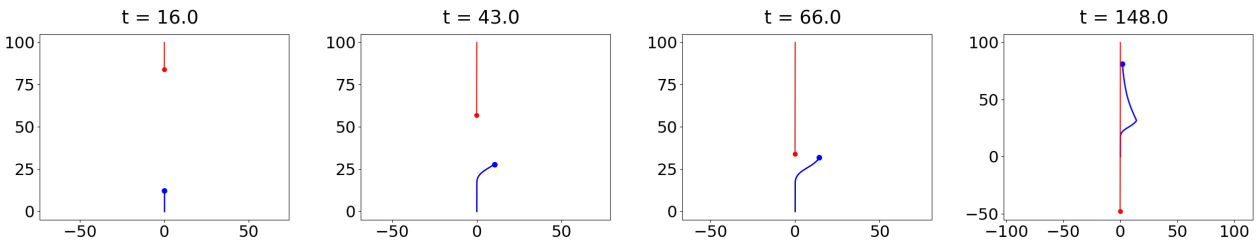


Figure 3.13: Snapshots from simulations with conservative configured MPC COLAV system in head on scenario.

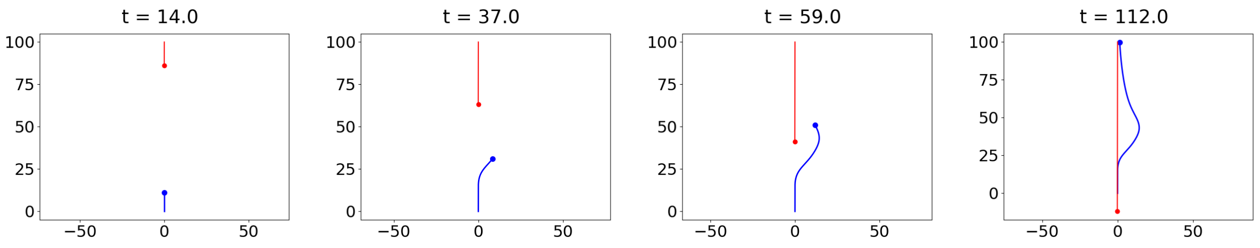


Figure 3.14: Snapshots from simulations with aggressive configured MPC COLAV system in head on scenario.

3.5.8 Simulations, scenario 3: Overtaking

Snapshots from simulations of the scenario are depicted in Figure 3.15 and Figure 3.16. The conservative MPC handles the overtaking by lowering its speed and keeping a safe distance to the adversary. The aggressive MPC passes the vessel by a manoeuvre to the starboard side, and efficiently returns to the path.

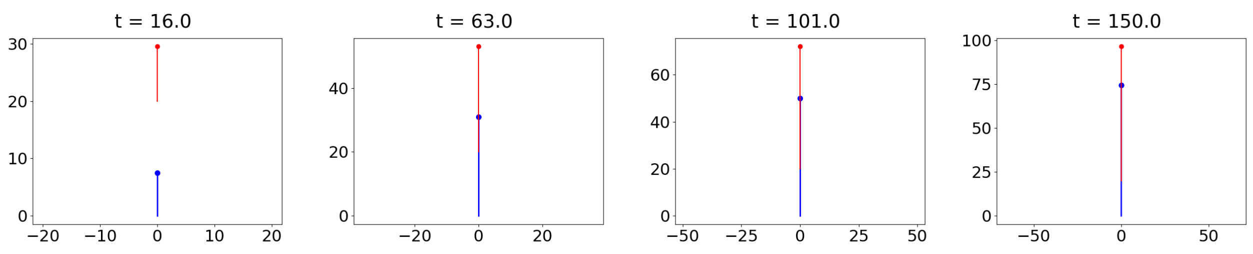


Figure 3.15: Snapshots from simulations with conservative configured MPC COLAV system in overtaking scenario.

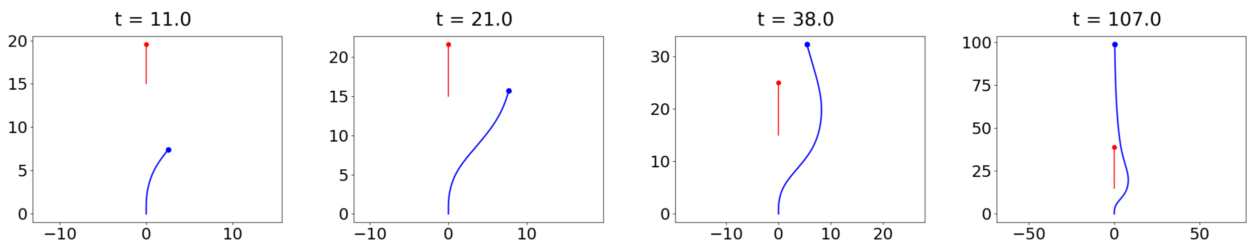
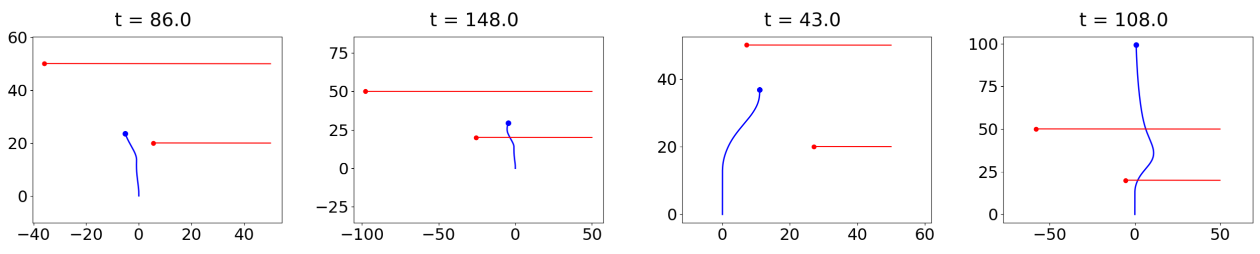


Figure 3.16: Snapshots from simulations with aggressive configured MPC COLAV system in overtaking scenario.

3.5.9 Multiple adversary scenarios

The three scenarios were expanded to include multiple adversaries to challenge the MPC system further and possibly uncover failures caused by e.g. conflicting COLREGs violations. Note that the plots of these sections show both the conservative and the aggressive MPC configuration behaviours.

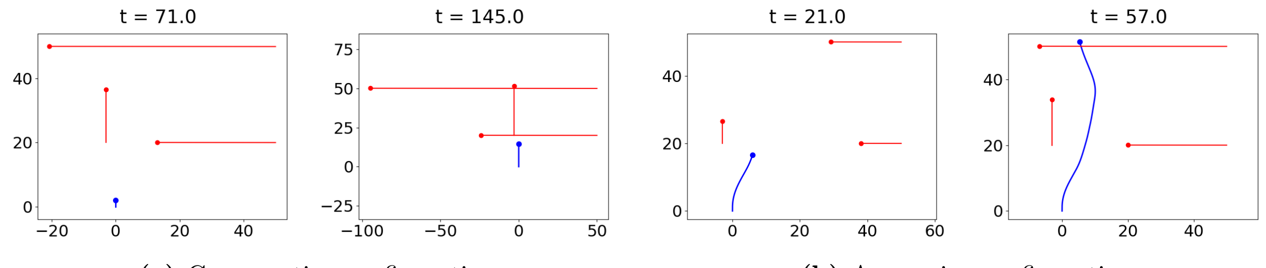
Scenario 1 was expanded to include two adversaries, where the second adversary was set up to perform a crossing, but closer to the ferry to investigate how this would affect the vessel as it attempts to perform the proper manoeuvre to the starboard side of the crossing vessels. See Figure 3.17 for simulation snapshots of the scenario. The conservative configuration ends up with a manoeuvre to the port side of the closest crossing vessel, and quite close to it too. There seem to be some conflicting directions provided by the MPC, as it initially attempts to cross port of the closest adversary, but stops as it awaits the passing of the second adversary. When the closest adversary gets even closer, it is forced to evade to the port side. The scenario poses less challenge to the aggressive configuration, which



(a) Conservative configuration **(b)** Aggressive configuration

Figure 3.17: Snapshots from simulations with aggressive configured MPC COLAV system in crossing scenario with two adversaries.

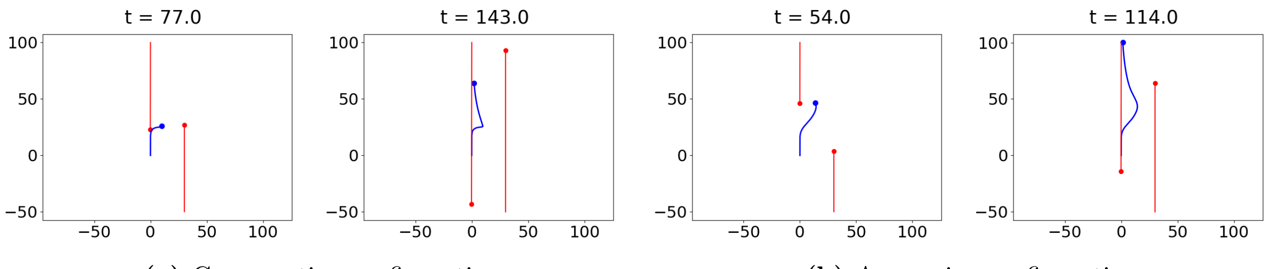
The scenario was further expanded to include a total of three adversary vessels, by adding a vessel to be overtaken, see simulation snapshots in Figure 3.18. The conservative configuration solved the scenario by stopping for a long time initially and proceeding when the crossing vessels had passed and the distance to the vessel to be overtaken was sufficient. The vessel to be overtaken seemed to make less challenge for the aggressive configuration, except that the starboard manoeuvre is performed for a little longer duration.



(a) Conservative configuration **(b)** Aggressive configuration

Figure 3.18: Snapshots from simulations with aggressive configured MPC COLAV system in the crossing scenario with three adversaries.

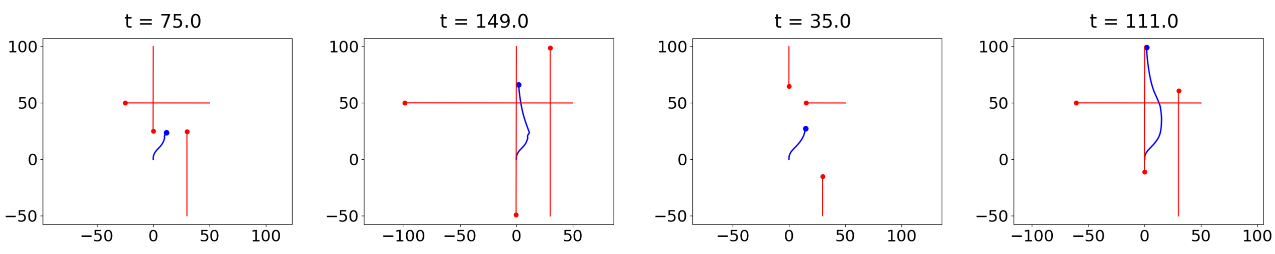
Scenario 2 was expanded to include two adversaries by adding a vessel which approached the ferry from astern. This was done in order to challenge the starboard manoeuvre that the ferry is supposed to perform with regard to the head-on vessel. The two configurations solve the challenge in similar manners, by evading the head-on vessel while keeping clear of the overtaking vessel. Simulations from the two-vessel scenario are shown in Figure 3.19.



(a) Conservative configuration **(b)** Aggressive configuration

Figure 3.19: Snapshots from simulations with aggressive configured MPC COLAV system in head on scenario with two adversaries.

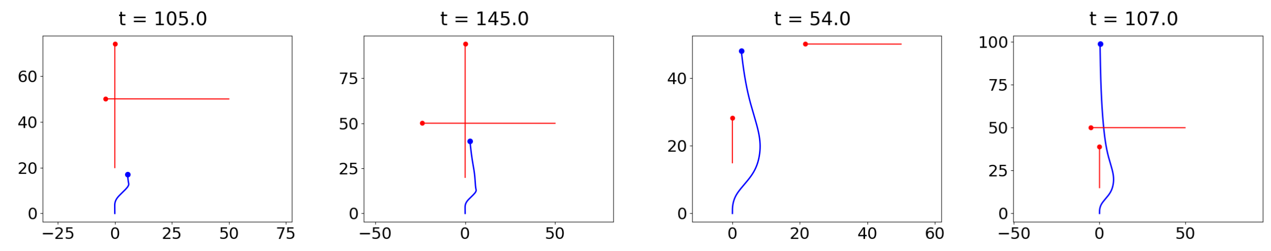
The scenario was further extended to include a crossing vessel, which made the conservative MPC perform a manoeuvre to starboard side initially and await there until all the vessels had passed. The aggressive configuration was able to manoeuvre on the starboard side of the crossing vessel while avoiding the head-on and overtaking vessel, see simulations in Figure 3.20.



(a) Conservative configuration **(b)** Aggressive configuration

Figure 3.20: Snapshots from simulations with aggressive configured MPC COLAV system in head on scenario with three adversaries.

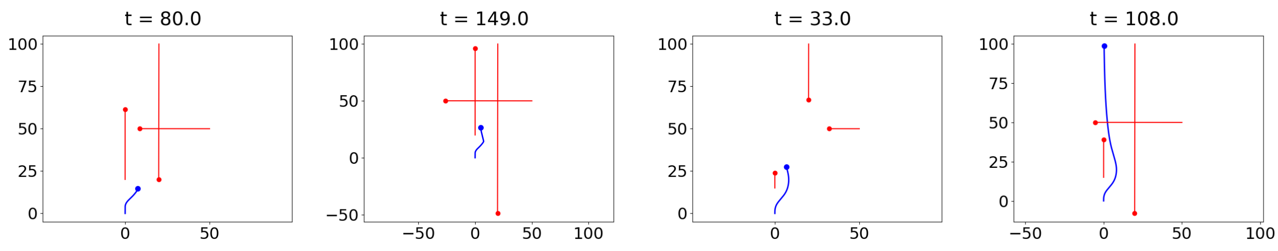
Scenario 3 was extended to include a crossing adversary to challenge the overtaking manoeuvre. The aggressive configuration solved this by passing in front of the crossing vessel, as the overtaking manoeuvre had led it to have speed and head toward the port side when the crossing becomes active. The conservative MPC stops and waits for the crossing to pass, see Figure 3.21.



(a) Conservative configuration **(b)** Aggressive configuration

Figure 3.21: Snapshots from simulations with aggressive configured MPC COLAV system in the overtaking scenario with two adversaries.

The last adversary in scenario 3 was set up to induce a head-on situation amid of the overtaking manoeuvre. The conservative configuration solved this by stopping in the middle of the trajectories of the head-on and the vessel to be overtaken. The aggressive configuration performed the overtaking manoeuvre so fast that the head-on vessel did not pose a challenge, see Figure 3.22



(a) Conservative configuration **(b)** Aggressive configuration

Figure 3.22: Snapshots from simulations with aggressive configured MPC COLAV system in the overtaking scenario with three adversaries.

3.5.10 Discussion on the MPC method

There are signs of a problem described in Hagen et al. (2018), where the COLREGs manoeuvres are interrupted while performing them due to the decrease of collision risk. This does, in some of the cases, make the ferry behave somewhat abrupt and little in regard to Rule 17 of the COLREGs, which urges ships to act in a predictable manner unless there is a prominent need to take drastic action. Hagen et al. (2018) solve this problem by introducing a cost of interrupting COLREGs compliant manoeuvres. However, as the implementation of the MPC COLAV system was sought to be more illustrative than an in-depth safety study of the MPC based methods, it was decided to keep the implementation simple and understandable. Introducing more terms in the cost function could also make the system harder to debug during development.

3.5.11 AST implementation

In order to identify collisions where the MPC COLAV system violated the COLREGs, the AST problem is formulated to optimize for improper behaviour, as described in variation 3 in Section 3.4. A time step is deemed improper if the binary indicator μ_i is true for any of of the adversary vessels, which indicates that either Rule 14 or Rule 15 is violated, and the trajectory is considered a failure if the fraction of improper time steps, f_{imp} is larger than the threshold f_{thresh}, as described in Section 3.4.

The AST agent is able to control the adversary vessels in an under-actuated manner, by adding disturbances to the surge speed and yaw rate only. The disturbance in sway speed was dropped to limit the action space of the agent to simplify the DRL network when applying multiple adversaries. The dynamics of the adversary vessels then become:

$$\begin{aligned} \dot{\boldsymbol{\eta}} &= \boldsymbol{R}_{z,\psi}\boldsymbol{\nu} \\ \dot{\boldsymbol{\nu}} &= -\frac{1}{\boldsymbol{T}}\boldsymbol{\tau} + \boldsymbol{u} \end{aligned} \tag{3.19}$$

with the disturbance vector $\boldsymbol{u}$ given as:

$$\boldsymbol{u} = \begin{bmatrix} u_u \\ 0 \\ u_\psi \end{bmatrix} \tag{3.20}$$

and the vector of AST actions $\boldsymbol{u}_{ast}$ defined as

$$\boldsymbol{u}_{ast} = \begin{bmatrix} u_u \\ u_\psi \end{bmatrix} \tag{3.21}$$

where u_u is the AST disturbance in surge speed and u_ψ the AST disturbance in yaw rate. The action space of the agent was restricted to $U_{0.05}$, which is the middle value of the action space configurations from Part 1, resulting in realistic adversary trajectories. In the multiple adversary case, the vector of AST actions is expanded to

$$\boldsymbol{u}_{ast} = \begin{bmatrix} u_{u,1} \\ u_{\psi,1} \\ u_{u,2} \\ u_{\psi,2} \\ \cdots \\ u_{u,n} \\ u_{\psi,n} \end{bmatrix} \tag{3.22}$$

such that the AST agent provides a surge and heading disturbance for all n adversaries.

The AST hyperparameters applied in Part 2 are specified in Table 3.4. Note that the collision distance is reduced from Part 1, as the vessels are thought to be smaller in Part 2.

Hyperparameter	Value
bs	5 000
N_{epoch}	20
t_0	0 s
t_{end}	150 s
d_{coll}	5 m
Δ_S	0.1 s
Δ_A	3 s
f_{thresh}	10%
N_{ast}	50

Table 3.4: AST hyperparameters, part 2.

3.5.12 Computer specs and simulation duration

The experiments were conducted on two computers: a Dell OptiPlex 7060 SFF, with Core i5, 8 GB RAM and 256GB SSD provided by NTNU, and a Dell XPS 15 9510, with Core i7, 16 GB RAM and 512 GB SSD provided by Zeabuz. The duration of one AST simulation was in the middle of 4 to 5 hours, in both parts. Via bash scripts, many simulations were set up to run in sequence, often overnight.

Chapter 4

Results and discussion

In this chapter, the results from the two parts are presented together with a discussion of the findings. The advantages and disadvantages of the different approaches are discussed based on the effect they have on the resulting failure modes as well as the learning process of the RL agent. The results of the clustering approach are shown in the results from Part 1.

4.1 Part 1: The SP-VP case

This chapter presents the results from further experiments with the SP-VP system, which builds upon the work of Hjelmeland (2021). The results from Part 1 are meant to illustrate the differences in the failure modes obtained in the problem variations described in Section 3.4, as well as the effect of the problem variation on the learning process of the AST agent. In the cases where it is relevant, the clusters produced with the method described in Section 3.1 are portrayed. Example failure trajectories are also shown, sometimes together with snapshots from simulations that illustrate the SP-VP representation of the adversary to illustrate the SP-VP point of view of the scenario. Three DRL statistics are also presented when relevant, which illustrate the progress of the training of the DRL policy of the AST agent. The discussion of the results is done in the presentation of them, as well as in the general discussion in Section 4.1.6.

Table 4.1 presents an overview of the failure statistics from the experiments conducted in Part 1. The table describes the number of failures found in each scenario, denoted **F**, with the corresponding action-space configuration U, the total number of trajectories **T** and the percentage of the trajectories. The total number of simulated trajectories varies between the experiments, as it typically increases when failure is found frequently, as this shortens the episodes and gives room for more trajectories in the batch.

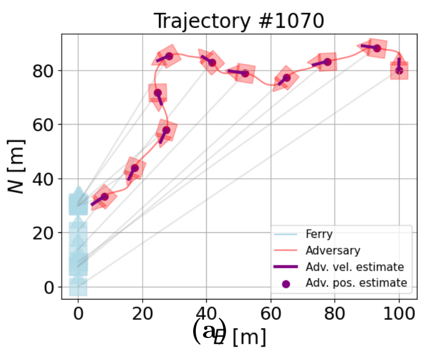
Problem variation	U	**F**	**T**	%
Baseline	0.1	1691	8307	20.4%
	0.05	290	8427	3.4%
	0.01	0	6000	0%
Dissimilarity	0.1	677	8102	8.4%
	0.05	822	7993	10.2%
Improper behaviour	0.1	364	7418	4.9%
	0.05	9	6765	0.1%
	0.01	0	6000	0%
Improper behaviour & estimation noise	0.1	220	6954	3.2%
	0.05	1910	8881	21.5%
	0.01	10145	11160	90.9%
Improper behaviour & time delay	0.1	737	7918	9.3%
	0.05	244	7478	3.3%
	0.01	0	6000	0%

Table 4.1: Failure statistics from experiments in Part 1. Column **F** denotes the number of failures, **T** the total number of trajectories in the AST simulation and % the failure percentage.

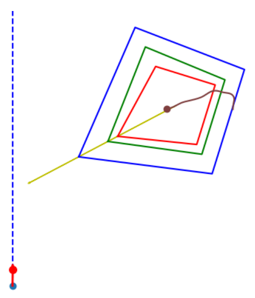
4.1.1 Variation 1: Baseline

Failure modes were found in AST simulations for the baseline variation with the adversary action-space configurations $U_{0.1}$ and $U_{0.05}$, but none with $U_{0.01}$. Even though the simulation scenario was tweaked compared to Hjelmeland (2021), the general tendency of the failure modes is the same. The adversary vessel expresses aggressive behaviour as it heads straight towards the ferry to provoke a collision.

An example of a failure trajectory from simulations with $U_{0.1}$ is depicted in Figure 4.1(a), together with snapshots from the corresponding simulation showing the SP-VP representation of the adversary. The ferry stands still for the majority of the simulation, except for small windows where it perceives the adversary to be heading in the northwest direction. Still, it is evident that the adversary is responsible for the collision, as it eventually heads straight for the ferry while the ferry is standing still.



(a)


(b) Start of trajectory 1070.

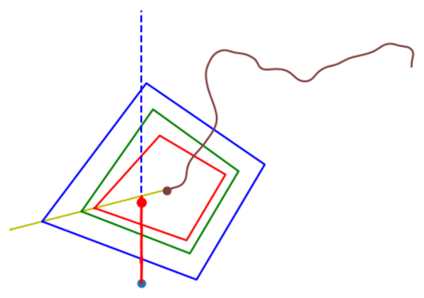
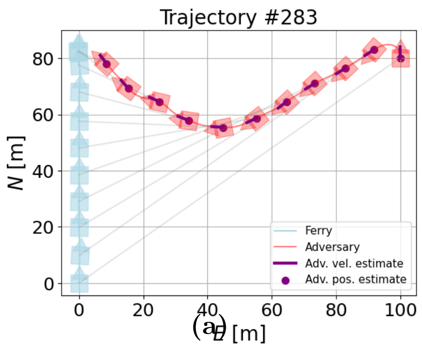
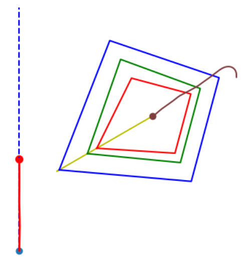

(c) Near the end of trajectory 1070.

Figure 4.1: Example failure trajectory together with snapshots from simulations showing the SP-VP representation of the adversary.

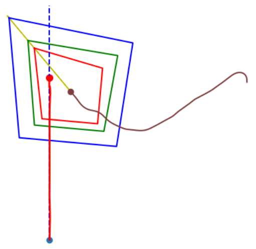
In the results from the simulations with the action-space configuration $U_{0.05}$, there is, as expected, less abrupt and irrational behaviour of the adversary. The behaviour is, however, still quite aggressive, as it heads straight toward the ferry in all the failure modes obtained. An example of a failure trajectory from simulations with $U_{0.05}$ is depicted in Figure 4.2(a), together with snapshots from the corresponding simulation with the SP-VP details depicted. In this scenario, the adversary seems to be on a course to pass astern the ferry, which makes the ferry keep its speed straight ahead. The adversary then turns and attacks the ferry from astern, which is again, quite obviously not a collision caused by the SP-VP system.



(a)



(b) Start of trajectory 283.



(c) Near the end of trajectory 283.

Figure 4.2: Example failure trajectory together with snapshots from simulations showing the SP-VP representation of the adversary.

The clusters from the simulations are depicted in Figure 4.3 and Figure 4.4. In all clusters, the aggressive behaviour of the adversary is evident. The effect of the restricted actions space is also well illustrated in the clusters, as the movement of the adversary is much smoother in all cases. In order to reduce the space needed to present all the results and enable easy comparison, the size of the cluster plots is kept small. However, the shape of the clusters is the most important aspect of these plots, which can still be easily extracted from the plots. The red lines in the clusters describe the cluster centroids, also known as the cluster barycenters, which are computed by minimizing the sum of squared DTW distance between the barycenter and the other series of the cluster.

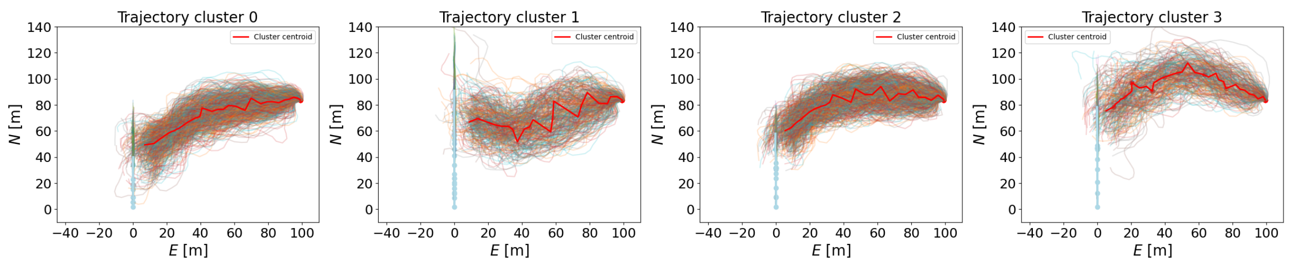


Figure 4.3: Resulting clusters of the obtained failure modes for the baseline variation under the adversary action-space configuration $U_{0.1}$. The red line depicts the cluster centroid.

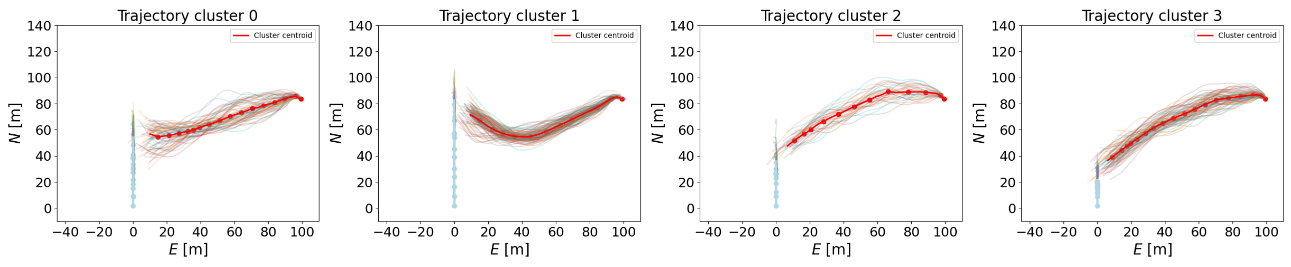
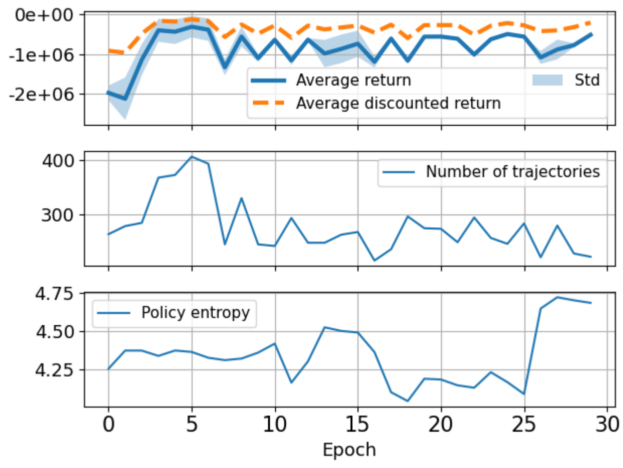


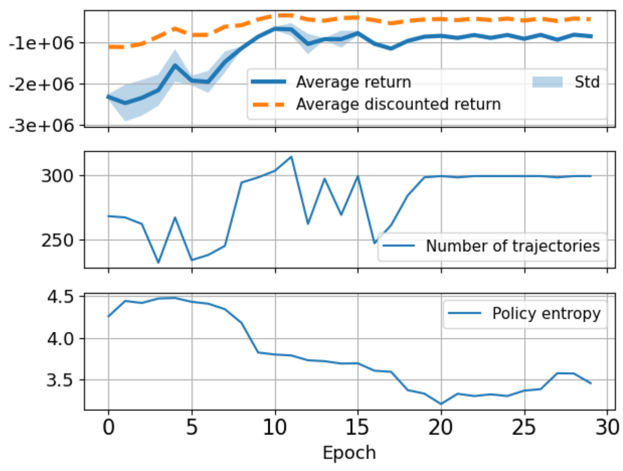
Figure 4.4: Resulting clusters of the obtained failure modes for the baseline variation under the adversary action-space configuration $U_{0.05}$

The RL statistics of the baseline case are shown in Figure 4.5. The first plot shows the average return, both the actual return and the discounted return, together with the standard deviation of returns denoted Std, for all trajectories in the epoch. The average return is typically higher as the final large negative reward in the case of no failure is discounted for the majority of the episode. The second plot presents the number of trajectories run in each epoch, which as previously mentioned, typically increases when failures are found. The bottom plot depicts the entropy of the policy, which indicates the level of randomness the DRL agent picks its actions with. The peaks of the entropy plot typically correspond to the largest standard deviation in the returns, as would be expected as a higher level of stochasticity in the action choice results in a more significant variation of trajectory outcomes and thus rewards.

The plots show that the AST agent in both cases seems to be unable to identify a strategy which results in failure every time, as the average return seems to stabilize. However, there are more signs of convergence in the $U_{0.05}$ case, as the policy entropy decreases throughout the simulation, the average return stabilizes and the number of trajectories flattens. This indicates that the AST agent converged to a sub-optimal strategy in the $U_{0.05}$ case. The fact that the agent in the $U = U_{0.05}$ converges faster than the agent in the $U = U_{0.01}$ case, is likely due to the decreased action-space restriction, as the search space is significantly smaller and thus the alternative ways of bringing the system to failure are fewer. In the $U_{0.01}$ case, there are signs of continued exploration, as the entropy stays high throughout the simulation and increases toward the end, which indicates that the agent has not converged and that the simulation could have been extended to more epochs to see convergence of the AST agent.



(a) $U_{0.1}$



(b) $U_{0.05}$

Figure 4.5: RL statistics over all simulation epochs.

4.1.2 Variation 2: Dissimilarity measure

The resulting failure modes which are identified when applying the dissimilarity measure in the reward function are in fact more diverse compared to the baseline case. In the $U = U_{0.1}$ case, fewer trajectories were found, but the variation between trajectories is higher and the shapes of the trajectories are more creative. In the $U = U_{0.05}$ case, the set of failures is both significantly bigger, and also more diverse than the baseline case.

Four examples of resulting trajectories are depicted in Figure 4.6, which show more peculiar adversary trajectories than the ones found in the baseline variation.

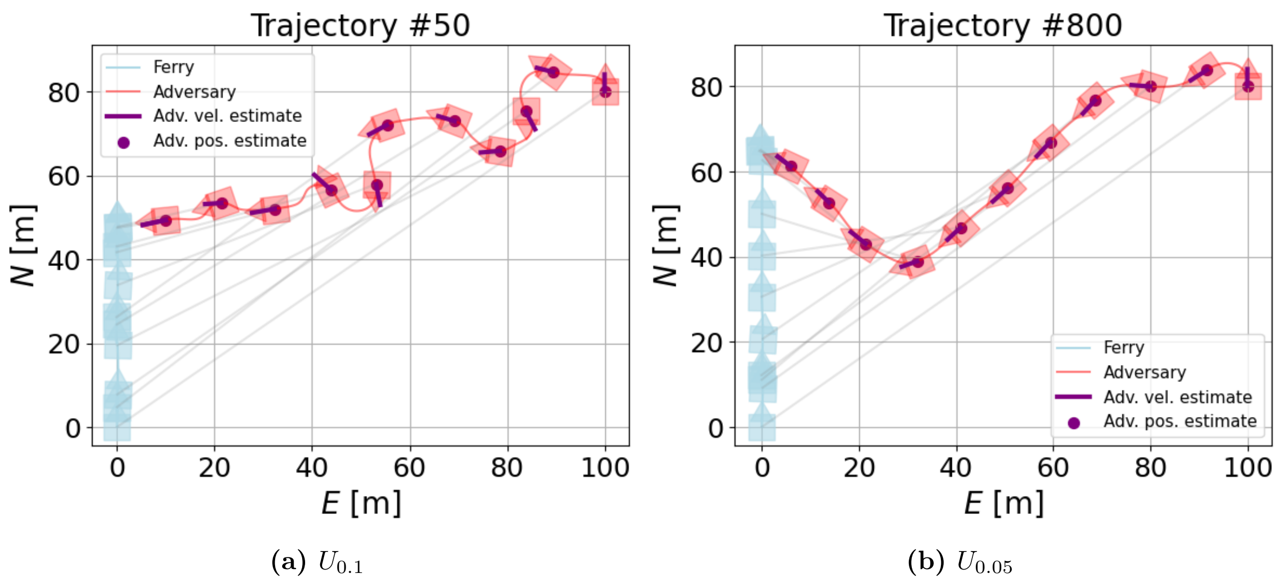


Figure 4.6: Examples of failure trajectories found when applying the dissimilarity measure.

The resulting clusters are shown in Figure 4.7 and Figure 4.8. The effect is evident in the clusters, as new cluster shapes are introduced in both cases. In the clusters for the $U = U_{0.1}$ case, the manually chosen number of clusters k seems like a poor fit as e.g. cluster 1 seem to hide some internal variations. Still, the effect of the dissimilarity measure is evident as the agent obtained trajectories of completely new shapes.

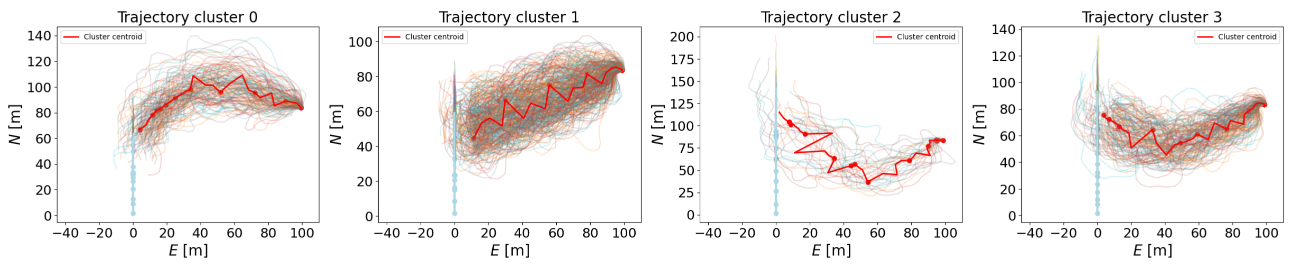


Figure 4.7: Resulting clusters of the obtained failure modes for the dissimilarity variation under the adversary action-space configuration $U_{0.1}$

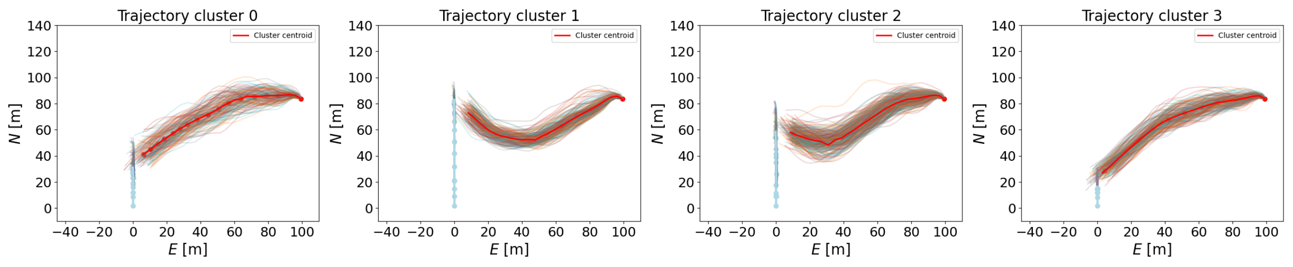
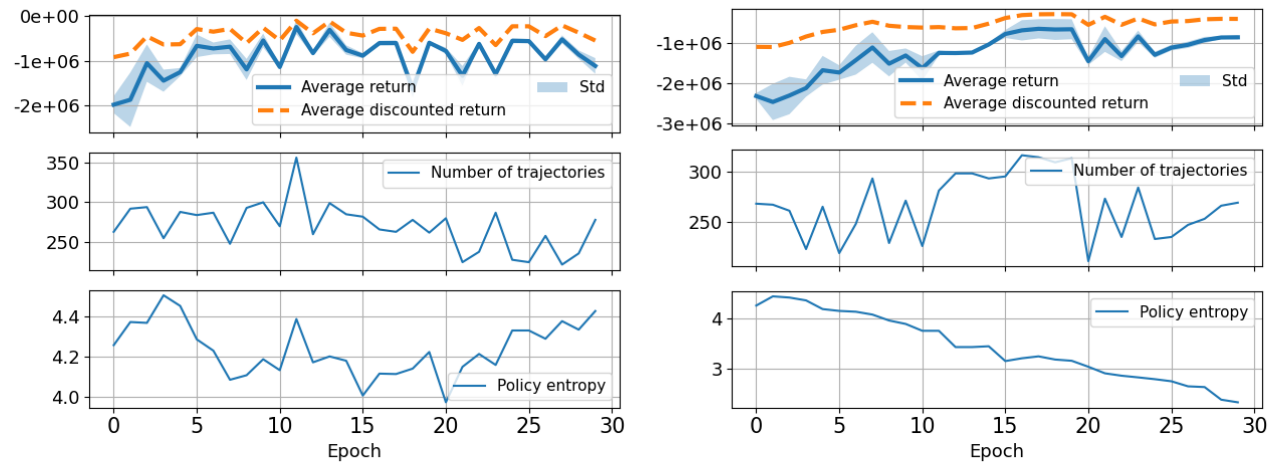


Figure 4.8: Resulting clusters of the obtained failure modes for the dissimilarity variation under the adversary action-space configuration $U_{0.05}$

In the RL statistics, depicted in Figure 4.9, the dissimilarity measure is evident as the average return is higher in both cases, which is expected as the dissimilarity measure gives a positive reward in the case of failure. Other than that, the RL learning process does not seem too affected by the dissimilarity measure, except that the standard deviation on returns is slightly higher and that the convergence tendency in the $U = U_{0.05}$ case is not seen in the return and number of trajectories. However, the policy entropy is lower than seen in the baseline case and decreases more steadily, which may be a sign that the policy is on its way to converge toward the end.



(a) $U_{0.1}$ **(b)** $U_{0.05}$

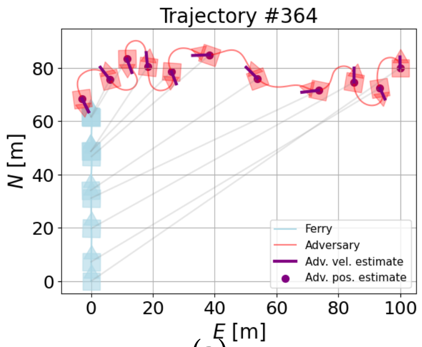
Figure 4.9: RL statistics over all simulation epochs.

It is possible that the dissimilarity measure could be categorized as *over-engineering* of the reward function, as it implements an additional purpose of the AST agent which is not related to the specific goal of the agent. The effect of the measure seems to have a positive effect when obtaining failures in this specific case, however, it could have an unwanted effect in other systems by leading the AST agent away from failure. As it is hard to isolate the effect of the measure, it was decided not to keep the dissimilarity measure in the reward functions for further experiments. It can, of course, be implemented as an additional feature if the results of an AST simulation was highly similar. Other methods exist to ensure more exploration of the DRL agent, such as the go-explore method implemented in (Koren and Kochenderfer, 2020) or altering the hyperparameters of the Gaussian distributions of the policy output.

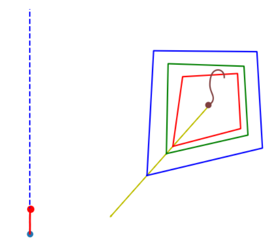
4.1.3 Variation 3: Optimizing for improper behaviour

Optimizing for improper behaviour gave significantly fewer results compared to only optimizing for collision, as 364 failure modes were found in the $U = U_{0.1}$ case and only 9 failures were found in the $U = U_{0.05}$ case.

Examples of failure trajectories are shown in Figure 4.10(a) (a) and Figure 4.11(a) (a), with snapshots from simulations and corresponding SP-VP representation of the adversary. In the $U = U_{0.1}$ case, the adversary learns to trick the ferry into behaviour deemed improper by Equation (3.11) by behaving sufficiently abrupt with turns that indicate that it is headed elsewhere, such that the ferry continues forward and thus has a certain speed while in close proximity of the adversary. In the $U = U_{0.05}$ case, the behaviour is less abrupt, but the tendency is the same: the adversary pretends to be crossing at a distance which is sufficiently large so that the SP-VP system withholds the speed, but turns toward the ferry in the end of the simulation.



(a)


(b) Start of trajectory 364.

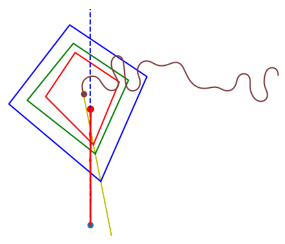
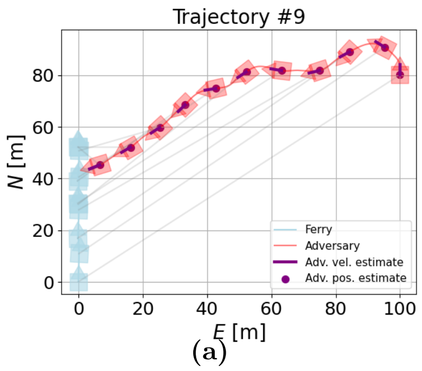
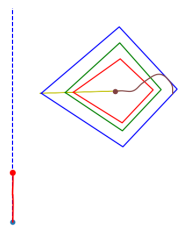

(c) Near the end of trajectory 364.

Figure 4.10: Example failure trajectory together with snapshots from simulations showing the SP-VP representation of the adversary.



(a)


(b) Start of trajectory 9.

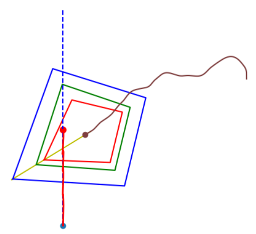

(c) Near the end of trajectory 9.

Figure 4.11: Example failure trajectory together with snapshots from simulations showing the SP-VP representation of the adversary.

The resulting trajectory clusters are shown in Figure 4.12 and Figure 4.13. The abrupt behaviour of the adversary in the $U = U_{0.1}$ case is evident in the clusters, and the strategy of heading in another direction until the end of the simulation is evident in the clusters corresponding to the $U = U_{0.05}$ case.

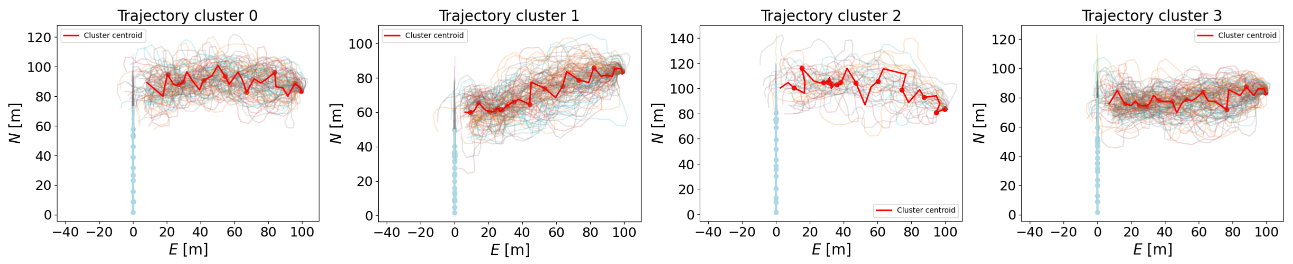


Figure 4.12: Resulting clusters of the obtained failure modes for the improper variation under the adversary action-space configuration $U_{0.1}$

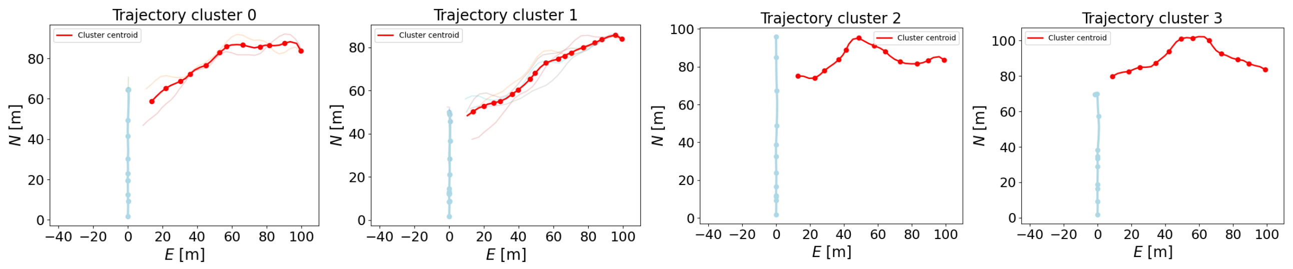
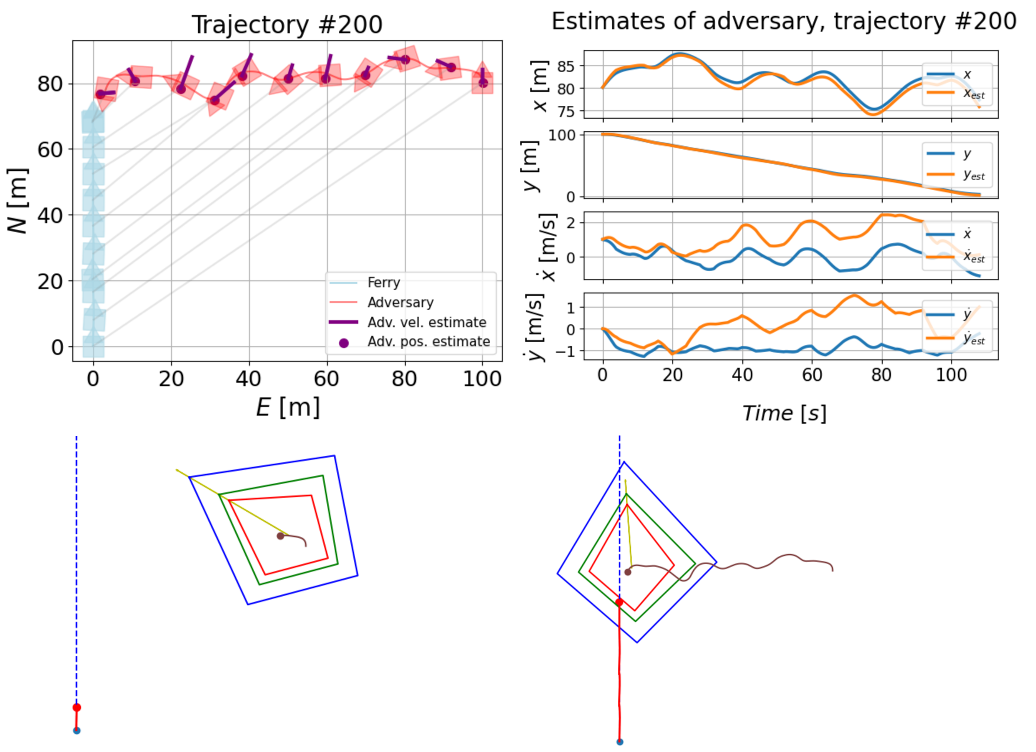


Figure 4.13: Resulting clusters of the obtained failure modes for the improper variation under the adversary action-space configuration $U_{0.05}$

Even though the results show adversary behaviour that can be considered abrupt, irrational or unlikely, they do expose a concept that the SP-VP system and most COLAV systems lack, namely the concept of understanding *intention*. With the simplified simulated sensory module and the SP-VP system, the ferry is able to, at every time step, determine the state of the environment and potentially react to it. However, the intention of the environment elements are typically not accounted for, such as the question of "where do we believe this adversary vessel is going". This is evident in the results of this variation, as the SP-VP system issues the order to increase the speed of the ferry every time it seems that the adversary is headed in another direction. There is no accounting for the adversary behaviour in the previous time steps, which could have led to the conclusion that even though this adversary vessel is behaving abrupt, the overall tendency of its movement implies that it will at some point cross in front of the ferry. In similar real-life situations, a human helmsman would probably have registered this kind of passage as a "drunken driver" and waited for passing. A possible way to implement such reasoning would be to await speeding up again after a stop has occurred, with the condition that the overall risk of collision must decrease for some time before returning to nominal speed. This kind of reasoning could, of course, be made by a system on another layer in the hybrid deliberative/reactive structure and thus not be the responsibility of the COLAV system.

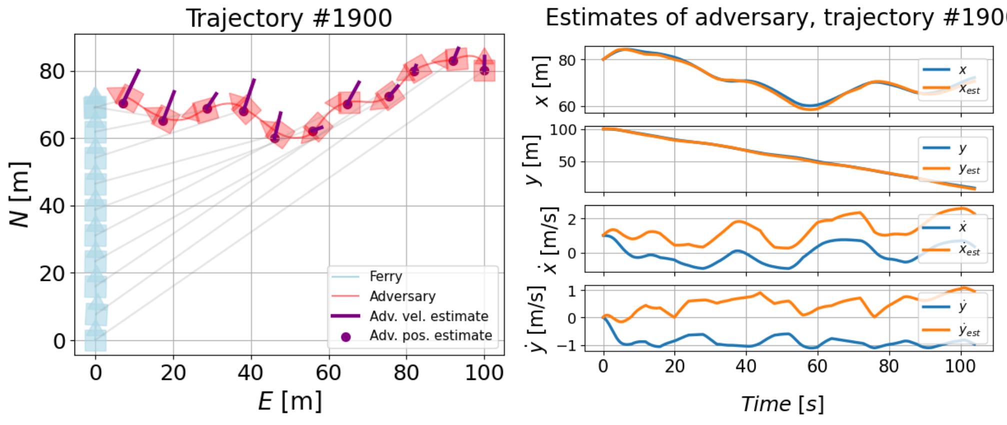
4.1.4 Variation 4: Optimizing for improper behaviour with estimation noise

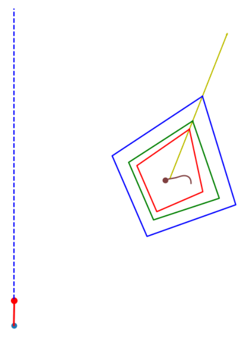
In variation 4, failure modes were obtained under all action-space configurations. Examples of failure trajectories found in the $U = U_{0.1}$ and $U = U_{0.05}$ are depicted in Figure 4.14 and Figure 4.15 respectively, together with plots showing how the adversary estimates evolve, and snapshots from the corresponding simulation with the SP-VP adversary representation. In both action-space configuration cases, the noise is quite significant and highly affects the velocity estimate, which causes the ferry to keep its nominal speed as the SP-VP system figures that the adversary course points north. The results from the two action-space configurations are quite similar, as it is mainly the estimation noise which causes the collisions.



(a) Start of trajectory 200. **(b)** Near the end of trajectory 200.

Figure 4.14: Example trajectory for the $U = U_{0.1}$ case when optimizing for improper behaviour with estimation noise.




(a) Start of trajectory 1900.

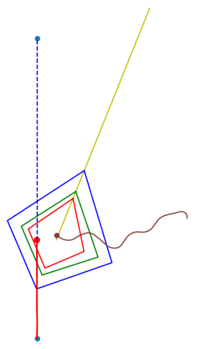
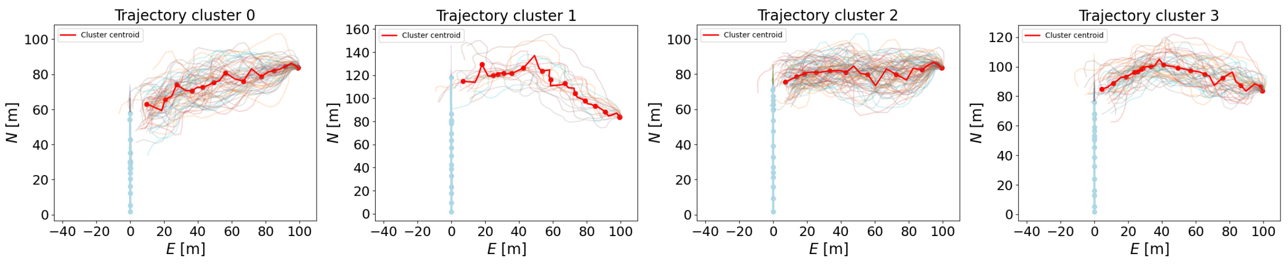
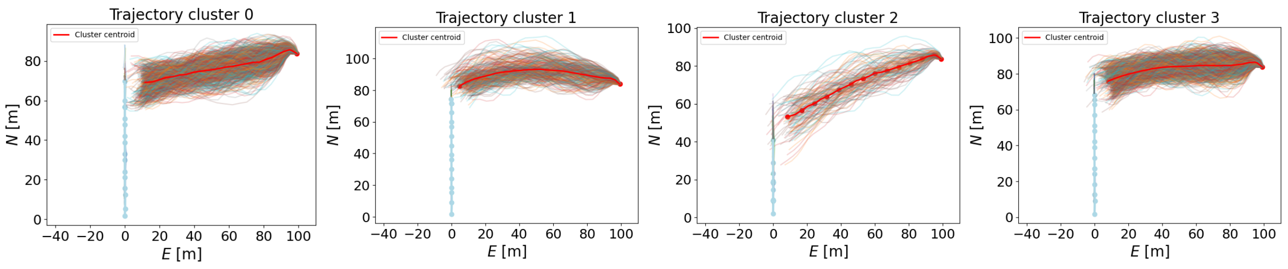

(b) Near the end of trajectory 1900.

Figure 4.15: Example trajectory for the $U = U_{0.05}$ case when optimizing for improper behaviour with estimation noise.

The resulting trajectory clusters for the $U = U_{0.1}$ and $U = U_{0.05}$ cases are shown in Section 4.1.4 and Section 4.1.4. Clusters were not obtained in the $U = U_{0.01}$ case, as all the failure modes were highly similar. The number of failures found in the $U = U_{0.05}$ case is dramatically higher than in the $U = U_{0.1}$ case, which indicates that it is an advantage for the AST agent to not have to search through the action-space corresponding to the movement of the adversary.





An example of a failure trajectory for the $U = U_{0.01}$ case is depicted in Figure 4.18. As in the previous action-space configurations, the noise values are significant throughout the episodes.

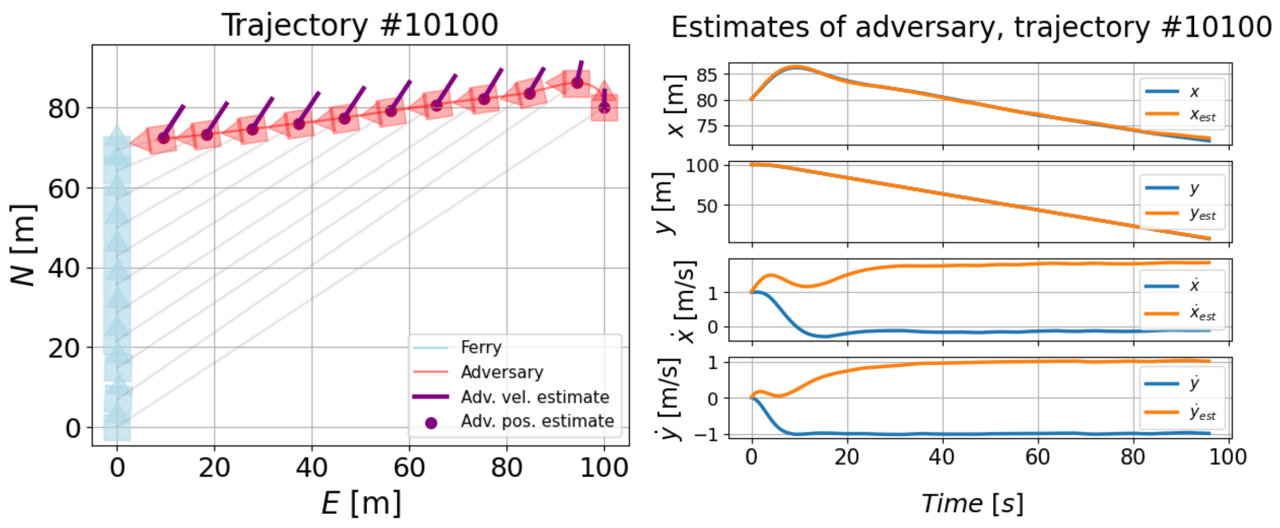


Figure 4.18: Example trajectory for the $U = U_{0.01}$ case when optimizing for improper behaviour with estimation noise.

The estimation noise variation was the only variation where failures were obtained in the $U = U_{0.01}$ case, and also interestingly the case where most failure modes were found. This is likely due to the heavy restriction of the action-space, which causes the agent to focus on exploring the noise values, and quickly converge to high noise levels. The agent is seen to clearly converge in the RL statistics shown in Figure 4.19.

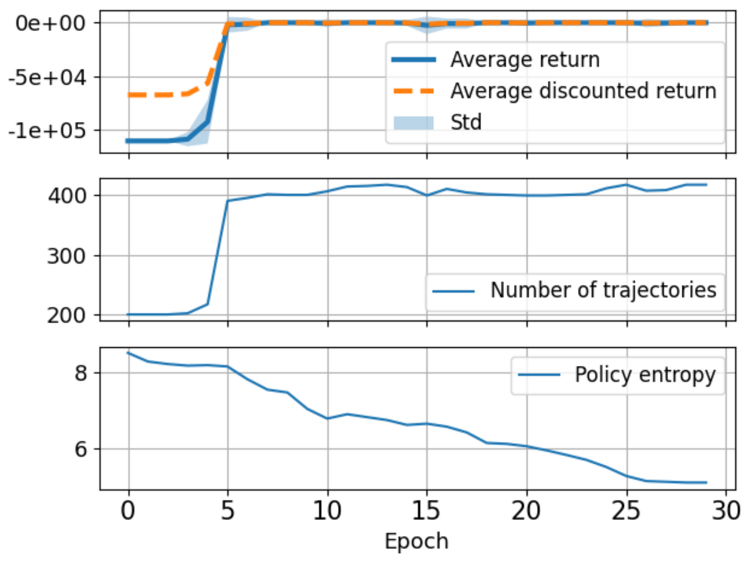


Figure 4.19: RL statistics over all simulation epochs.

4.1.5 Variation 5: Optimizing for improper behaviour with time delay

Failures were found in variation 5 for both the $U = U_{0.1}$ and the $U = U_{0.05}$ cases. Examples of failure modes, with the corresponding adversary estimate development, are shown in Section 4.1.5 and Figure 4.21. The estimate plots show that there is a significant time delay in the system in both cases. Note that the plots can be somewhat misleading in this case, as the estimates plotted typically correspond to vessel position and movements further ahead in the trajectory.

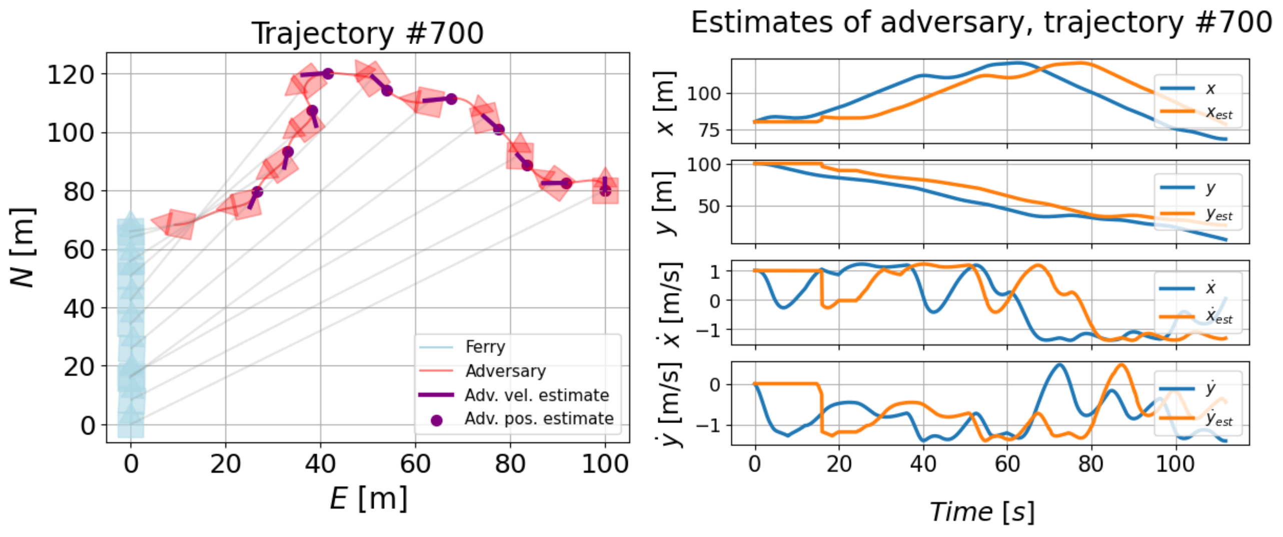


Figure 4.20: Example trajectory for the $U = U_{0.1}$ case when optimizing for improper behaviour with time delay.

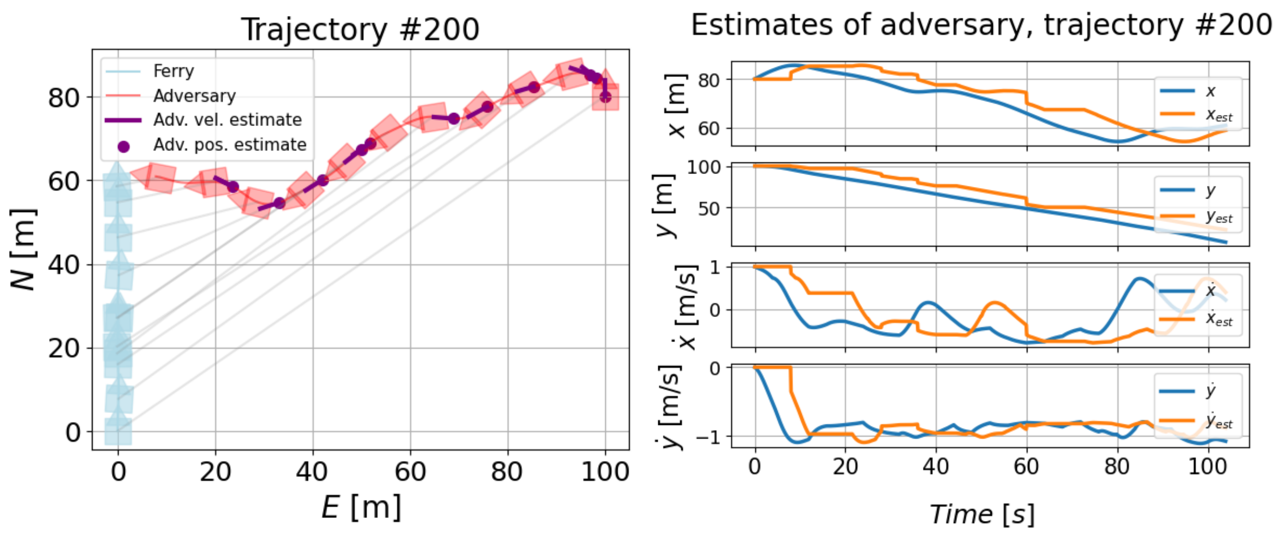


Figure 4.21: Example trajectory for the $U = U_{0.05}$ case when optimizing for improper behaviour with time delay.

As the time delay required to bring the system to failure was so significant, the failure modes are considered somewhat irrelevant. Experiments were performed with lower values of the time delay value T_{max}, but no failures were obtained. The time delay is thus interpreted as a less appropriate way to induce errors in the SP-VP system.

4.1.6 General discussion of results from Part 1

The results from the experiments of part 1, which is an extension of the work presented in Hjelmeland (2021), show that the further extensions of the scenarios and problem augmentations have interesting effects on the failures found by AST.

The restriction of the adversary action-space proved effective, as it resulted in smoother behaviour of the adversary and thus more reasonable and likely failure modes. The dissimilarity measure in variation 2 gave a more diverse set of failures. Still, due to the uncertainty regarding if this is an over-engineering of the reward function or not, the choice was made to omit the measure in further experiments.

Measures were made in variations 3, 4 and 5 to obtain failure modes which were not caused by the aggressive behaviour of the adversary vessel, but by improper behaviour of the ferry under the control of the SP-VP COLAV system. The results show that this way of formulating the AST problem has definite benefits. In the cases with estimation noise or time delays, it was evident that the reason for the collision was, in fact, the estimation noise and the time delay induced in the SITAW system of the ferry. In variation 3, without estimation errors, failures are found where the SP-VP system misunderstands the intention of the adversary and proceeds to keep speed straight ahead, ending in a collision. Even though the adversary still exposes irrational and improbable behaviour, the failures do uncover this lack of grasp of intention. However, as mentioned in the variation 3 results, the concept of understanding intention may not be the responsibility of the COLAV system but could be assigned to another level of the hybrid/deliberative architecture.

The SP-VP system has, through all the experiments in this work in addition to the ones performed in Hjelmeland (2021), undergone thorough testing under different problem formulations, and proven hard to provoke into causing a collision. The system has not been tested in various scenarios, and thus it is outside the scope of this thesis to comment on its overall performance. Still, it can be stated with confidence that the SP-VP system has proved substantially robust to a diverse set of adversary behaviour in the specific crossing scenario used in the experiments.

4.2 Part 2: Experiments with COLREGS-compliant MPC COLAV system

This chapter presents the results from the experiments conducted in Part 2 of the thesis. The results from part 2 are meant to illustrate how the AST can be applied to a more dynamic system with built-in compliance with the COLREGs. The results also show how AST can be used to illustrate the difference in possible failures of two different COLAV configurations, and how the introduction of multiple adversaries affects the failures, the COLREGs violations and the learning process of the AST agent. Example failure modes are shown in the cases where such were obtained, and the RL statistics are shown in the cases where it is relevant. Discussion of the results is done in the presentation of them, as well as in the general discussion in Section 4.2.4.

Clustering is not applied to the results of this part, mainly because the clustering module was centered around the adversary movement. In the SP-VP system, the movement of the ferry was restricted to the path and therefore highly similar, allowing for categorization of the failures using only the trajectory of the adversary. If the DTW method was to be applied to the system of Part 2, it would have to be altered to take the trajectory of the ferry into account, as it varied at least as much as the adversary trajectories, and also the trajectories of the additional adversaries. Moreover, in this case it could have been beneficial to implement a more similar approach to that presented in Lee, Kochenderfer, Mengshoel, and Silbermann (2018), where GBDT were applied, to categorize the failures on e.g. similar sequences of COLREGs violations. Due to lack of time, this was not implemented in this work, but it definitely could have added value to the results and should be subject for further research.

Table 4.2 presents an overview of the failure statistics from the experiments conducted in Part 2. The table describes the number of failures found in each scenario, with the corresponding MPC configuration, total number of trajectories and the percentage of the trajectories. As in Part 1, the total number of trajectories simulated varies between the experiments, as it typically increases when failure is found frequently, as this shortens the episodes and gives room for more trajectories in the batch.

It is evident in the overview that the crossing scenario was the only scenario where failure modes were found with only one adversary, as well as all the scenarios with more adversary vessels. For the head on scenario, two adversary vessels had to be introduced to find failures in the conservative configuration. For the aggressive configuration, failures were not obtained in this scenario until the three-adversary case. In the overtaking-scenario many failures were found for the aggressive configuration, but fewer in the case of the conservative configuration.

Scenario	MPC config	One adversary			Two adversaries			Three adversaries		
		F	**T**	%	**F**	**T**	%	**F**	**T**	%
1: Crossing	Conservative	1115	2705	41.2%	408	2146	19.0%	25	2010	1.2%
	Aggressive	1425	3086	46.2%	108	2201	4.9%	1189	2869	41.4%
2: Head on	Conservative	0	2007	0%	151	2050	7.4%	667	2310	28.9%
	Aggressive	0	2116	0%	0	2135	0%	388	2339	16.6%
3: Overtaking	Conservative	0	2000	0%	22	2010	1.1%	119	2041	5.8%
	Aggressive	0	2010	0%	4838	5398	89.6%	5082	5550	91.6%

Table 4.2: Failure statistics from experiments in Part 2. The column **F** denotes the number of failures, **T** the total number of trajectories in the AST simulation and % the failure percentage.

The following sections will present and discuss examples of failure modes from the cases where failure modes were obtained.

4.2.1 Scenario 1: Crossing

One adversary vessel

In the one adversary vessel case, many failures were found for both MPC configurations. Some example failure modes are depicted in Figure 4.22 and Figure 4.23, together with the time steps that are considered violations of either rule 14, rule 15, or both. The failures are similar between the conservative and the aggressive case, as the adversary heads for the vessel. The ferry initially attempts to perform a COLREGs compliant manoeuvre to the starboard side in all cases, but as the course of the adversary changes to seem to be on its way to pass astern the ferry, the manoeuvre is interrupted and a change to the port side is initiated, which eventually results in a collision. Although the behaviour of the adversary can again be considered aggressive, it is evident that the MPC controller could have solved the scenarios in a better manner by not interrupting its initial COLREGs compliant manoeuvre to the starboard side. As mentioned in Section 3.5.10, this is a concept discussed in Hagen et al. (2018), where a cost was put on the interruption of COLREGs-compliant manoeuvres, a concept which could have resolved these situations.

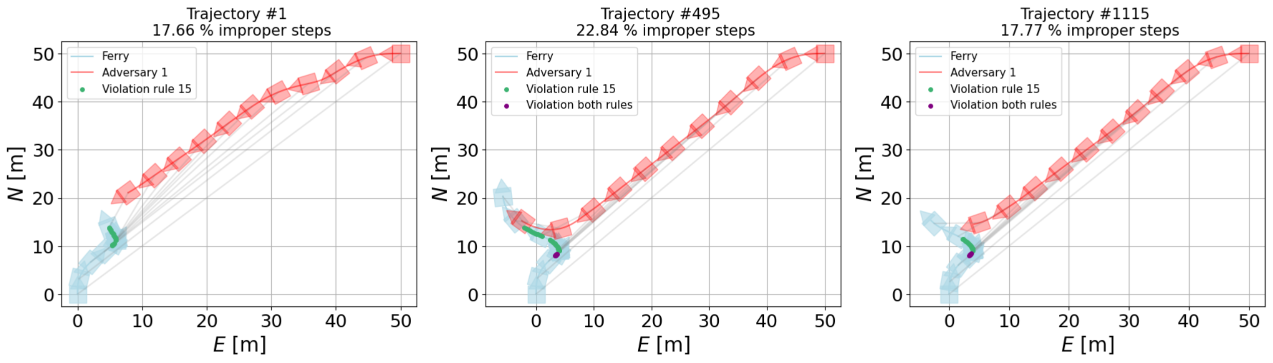


Figure 4.22: Examples of failure modes found with the conservative configuration in the crossing scenario with one adversary vessel.

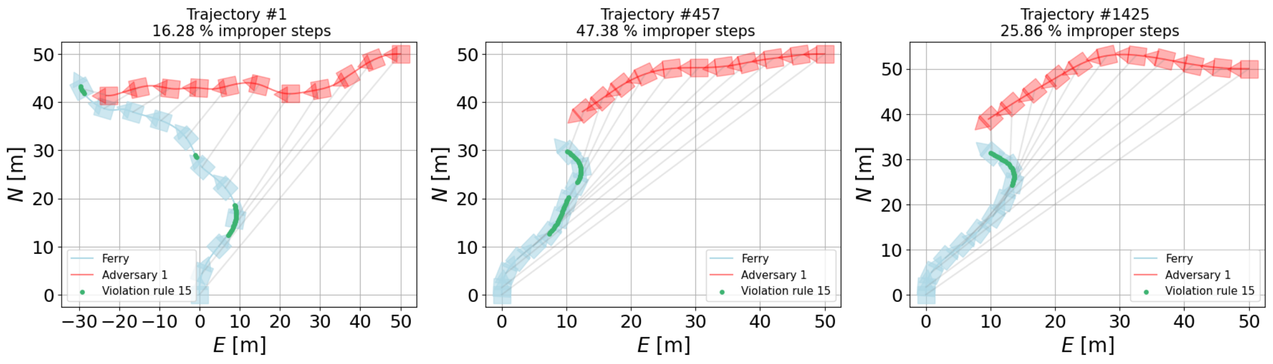
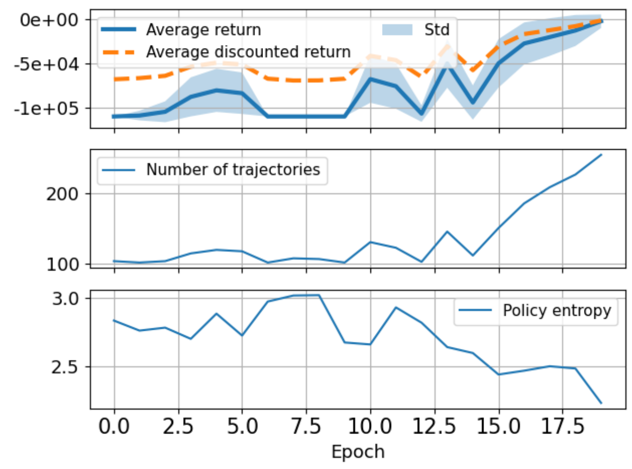
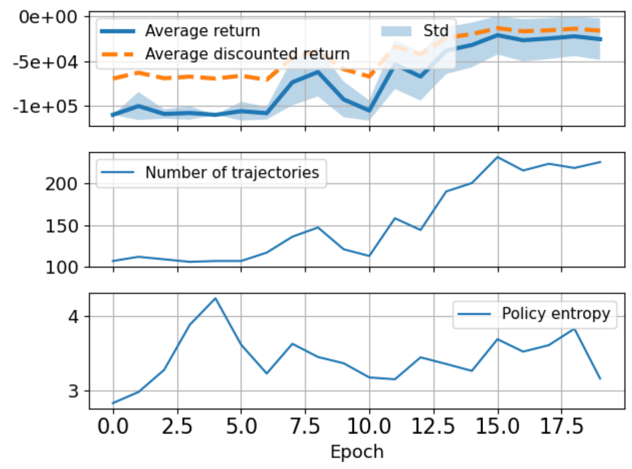


Figure 4.23: Examples of failure modes found with the aggressive configuration in the crossing scenario with one adversary vessel.

The RL statistics show signs of converging to an optimal solution in both cases toward the end, as the average return tend toward zero, see Figure 4.27. The policy entropy is relatively low in the conservative configuration case, which is also evident in the resulting trajectories as they are very similar, with only dissimilarities toward the end of the trajectory. In the aggressive configuration case, the shape of the adversary trajectory varies to a greater extent, which is evident in the policy entropy. The initial value of the policy entropy is in both cases lower than the entropies seen in Part 1, which is likely due to the action-vector only consisting of two elements. The standard deviation on returns are in general a lot higher in Part 2 compared to Part 1, which corresponds to the more dynamic behaviour of the MPC controller compared to the SP-VP controller. This naturally results in a variety of different scenario outcomes, as even a small change in a couple of actions can cause highly different responses in the MPC system. Periods of low standard deviation can thus be interpreted as periods where the adversary behaviours are not affecting the MPC response much, which may correspond to scenarios where the adversaries are headed in directions where they pose no risk to the ferry.



(a) Conservative configuration



(b) Aggressive configuration

Figure 4.24: RL statistics over all simulation epochs.

Two adversary vessels

With two adversary vessels attempting to cross, the COLREGs compliant manoeuvre to the starboard side is challenged as the ferry is less able to turn to the starboard side due to the increased risk of collision with the closest adversary. Failure modes were found for both configurations and examples are shown in Figure 4.25 and Figure 4.26. In the conservative configuration case, collisions are found with both adversaries, while in the aggressive configuration case, the AST agent tends to a strategy of having adversary 2 provoke the ferry into a manoeuvre to port side to enable collision with adversary 1. It is interesting how, in both cases, the two adversaries seem to be cooperating to make the collision happen.

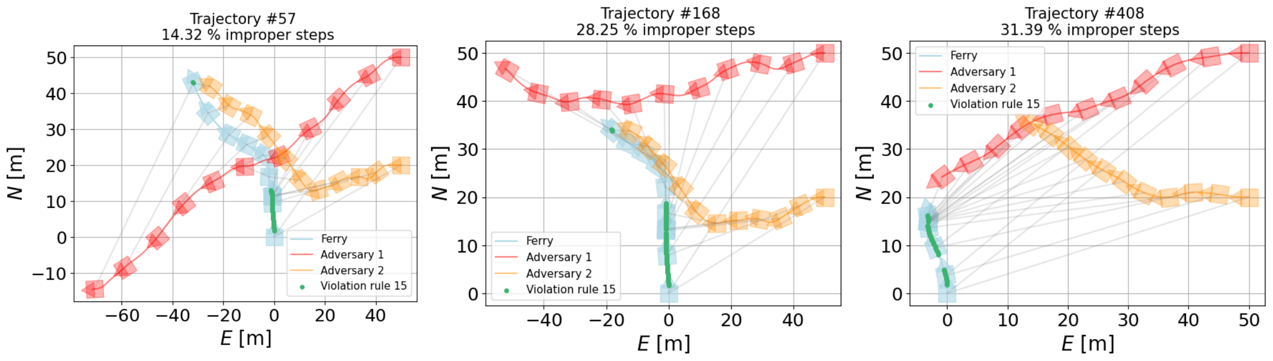


Figure 4.25: Examples of failure modes found with the conservative configuration in the crossing scenario with two adversary vessels.

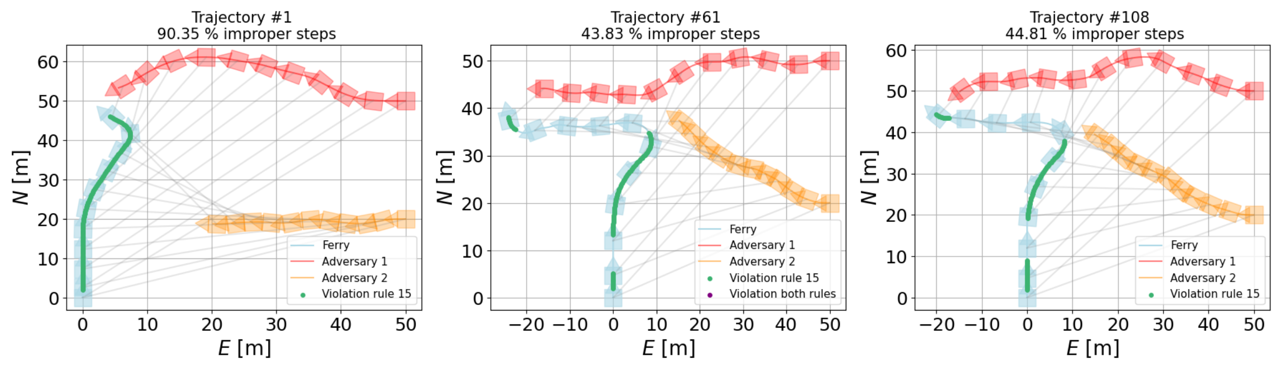
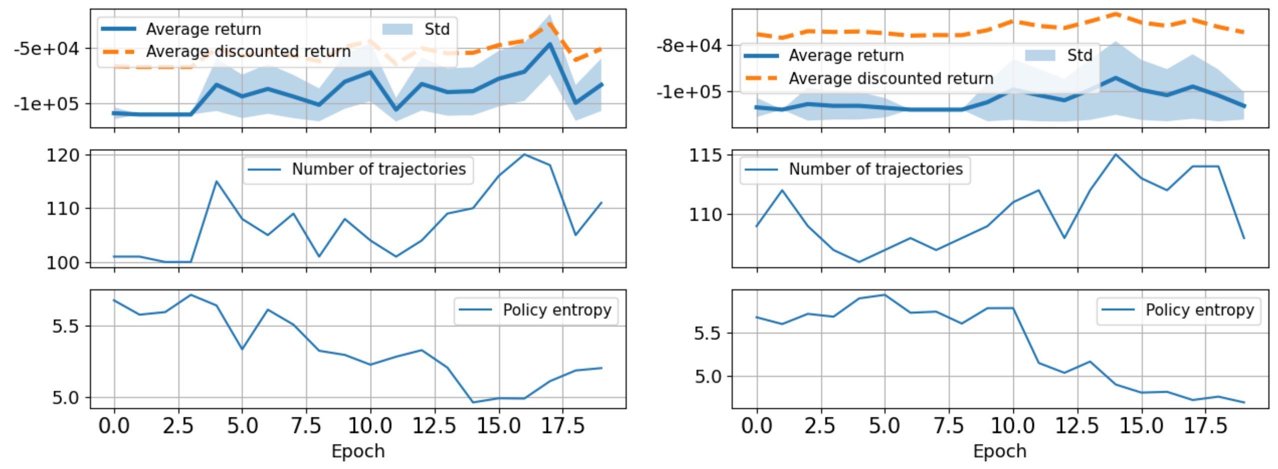


Figure 4.26: Examples of failure modes found with the aggressive configuration in the crossing scenario with two adversary vessels.

Signs of convergence are less evident in these simulations, with a high standard deviation on returns throughout the simulations and little increase in returns, suggesting that the number of epochs should have been increased.



(a) Conservative configuration **(b)** Aggressive configuration

Figure 4.27: RL statistics over all simulation epochs.

Three adversary vessels

When faced with three adversary vessels, where two of the vessels are crossing and one is positioned such that the ferry becomes the overtaking vessel, the failures found are dramatically reduced in the conservative configuration case, while it increases from the two-adversary case for the aggressive MPC. The examples shown in Figure 4.28 and Figure 4.29 can help explain why: the conservative configuration initially stops due to the vessel being overtaken, which makes it less prone to collision with the two crossing vessels as well. Thus, the vessel being overtaken can be said to aid the conservative configuration in this case. For the aggressive configuration, adversary 2 is mostly used to induce a violation of rule 15 and urge the ferry to a trajectory closer to the vessel being overtaken, while the vessel being overtaken provokes either a collision itself, or urges the ferry to either perform a sufficiently large starboard manoeuvre such that adversary 1 can collide with the ferry. Thus, the vessel to be overtaken poses an additional challenge for the aggressive configuration. In all scenarios with the aggressive configuration, a high fraction of improper time steps is seen as it is almost impossible to navigate such a situation without violating any rules unless it chooses to stop, as in the conservative configuration. This is also the first case where a violation of rule 14 is found, as adversary 1 is in some time steps found to be head on of the ferry.

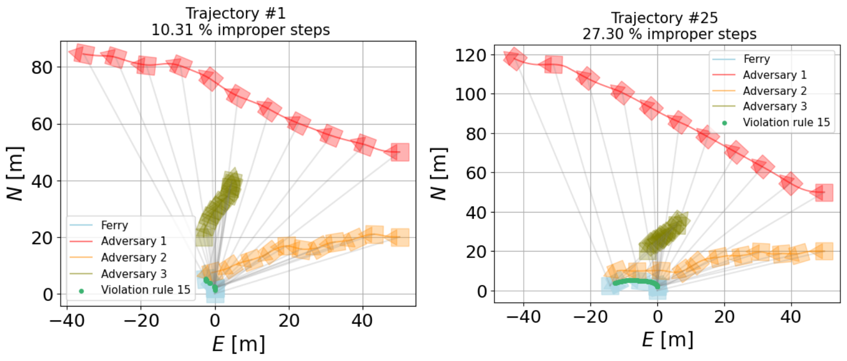


Figure 4.28: Examples of failure modes found with the conservative configuration in the crossing scenario with three adversary vessels.

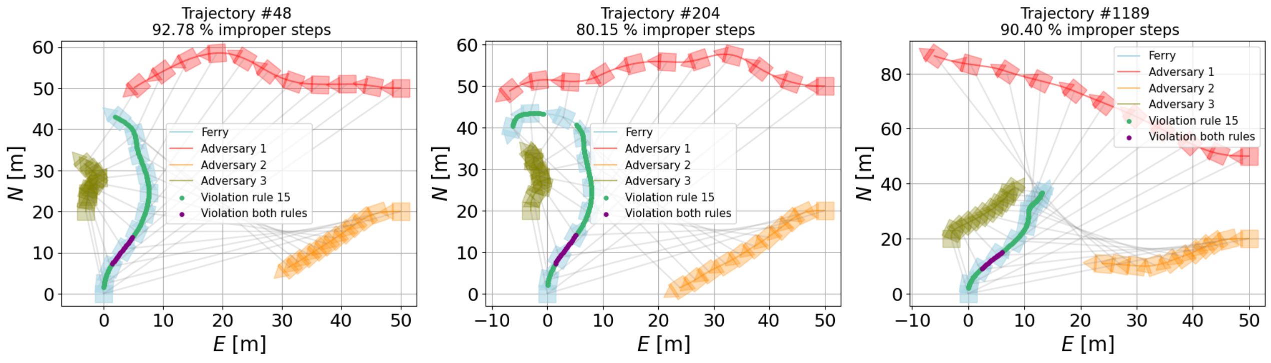
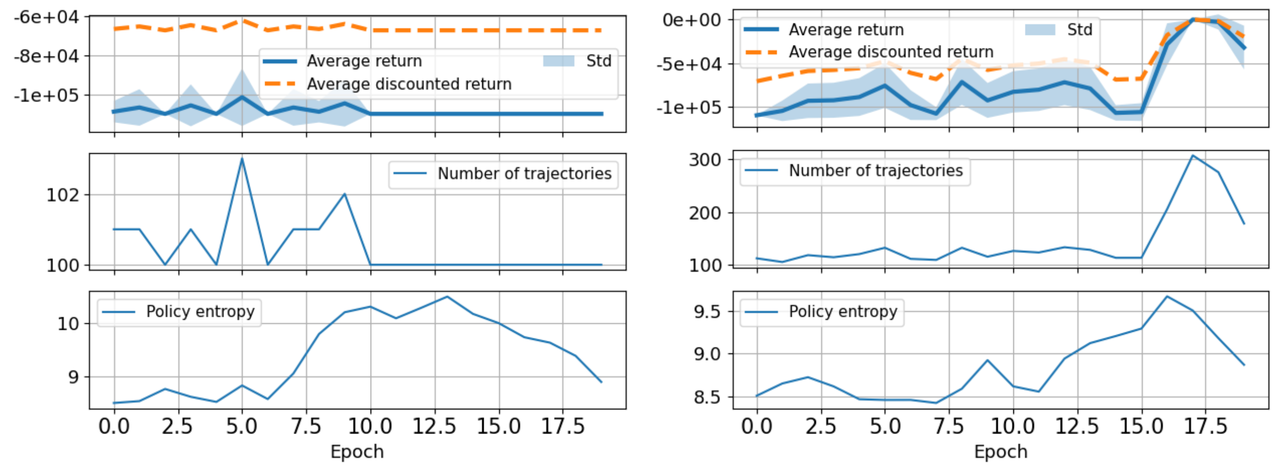


Figure 4.29: Examples of failure modes found with the aggressive configuration in the crossing scenario with three adversary vessels.

The resulting RL statistics depicted in Figure 4.30 show signs of the AST agent getting lost in the conservative configuration case, as it stops to discover more failures after 10 epochs have passed, which will be further discussed in Section 4.3. For the aggressive configuration case, there is a spike in return at the end of the trajectory, which likely corresponds to the part where the agent realized it could use the vessel being overtaking to provoke collisions, as this is a tendency which is seen in all the last failures found in this scenario.



(a) Conservative configuration **(b)** Aggressive configuration

Figure 4.30: RL statistics over all simulation epochs.

4.2.2 Scenario 2: Head on

In the head on scenario, no failures were found in the one-adversary case for neither of the two configurations.

Two adversary vessels

A few failures were found when adding adversary 2, which is set to additionally challenge the starboard manoeuvre of the ferry as it comes in a straight-forward manoeuvre from aster the ferry. Three example failure modes are shown in Figure 4.31. In the start, some failures are found where the ferry collides with the head on-vessel. However, the AST agent tends toward a strategy of having adversary 2 attack the ferry from astern while the head on-vessel only urges the ferry to perform the starboard manoeuvre.

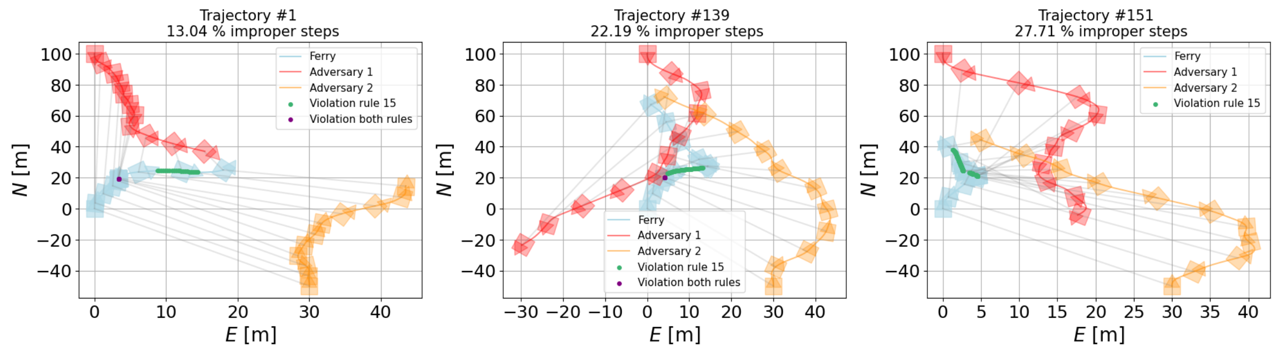


Figure 4.31: Examples of failure modes found with the conservative configuration in the head on scenario with two adversary vessels.

Three adversary vessels

A crossing vessel was added to the scenario in the case with three adversaries. Failures were obtained with both configurations, however, none of the collisions were with the head on adversary. Example failure modes are depicted in Figure 4.32 and Figure 4.33. In all cases, the head on vessel provokes an initial manoeuvre to the starboard side. The crossing adversary, adversary 3, then provokes a manoeuvre to port side, where a collision is often found either with adversary 2 or 3. The aggressive configuration is in general attacked later in the episode than the conservative one, by an astern attack of adversary 2. Violations of both rule 14 and rule 15 are found in both cases, as seen in the examples.

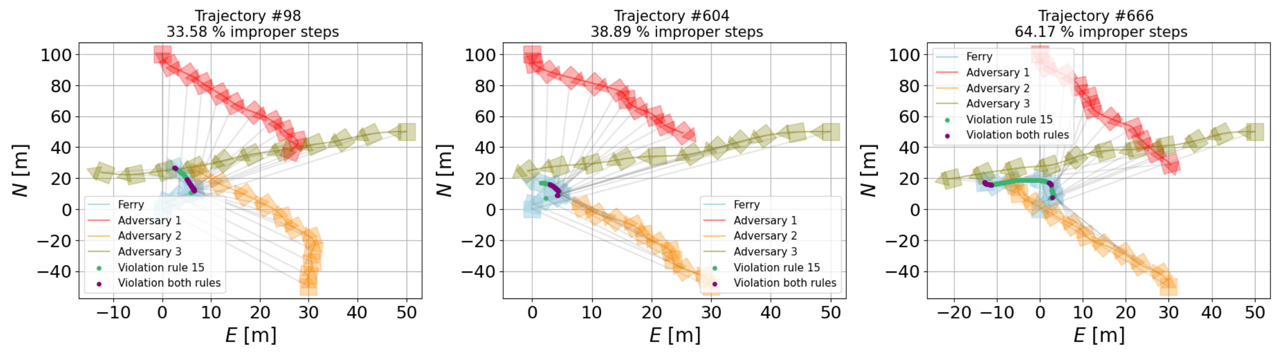


Figure 4.32: Examples of failure modes found with the conservative configuration in the head on scenario with three adversary vessels.

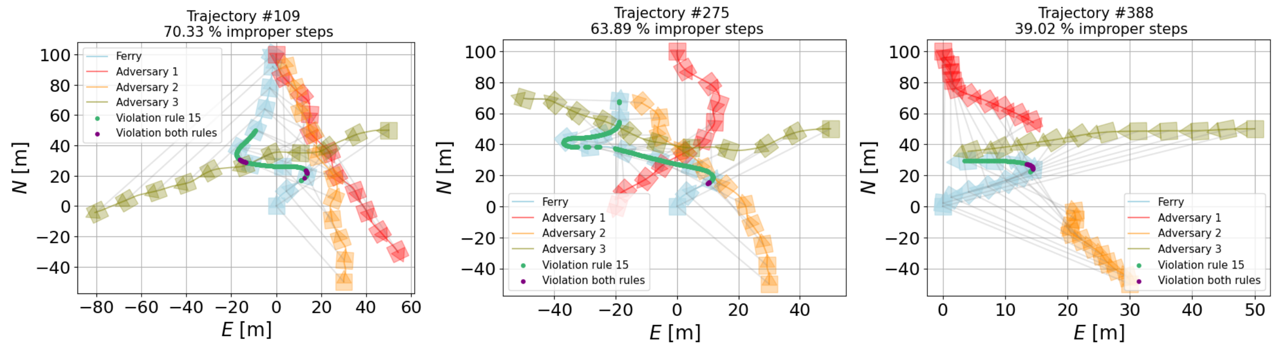


Figure 4.33: Examples of failure modes found with the aggressive configuration in the head on scenario with three adversary vessels.

4.2.3 Scenario 3: Overtaking

In scenario 3, failures were found for the both configurations with both two and three adversary vessels.

Two adversary vessels

Example failure scenarios for the case with two adversaries are shown in Figure 4.34 and Figure 4.35. In this case, the conservative configuration collides only with the additional crossing vessel. It attempts to perform a COLREGs compliant manoeuvre to starboard side but is interrupted by the crossing vessel, and thus, the results are similar to the ones obtained in the crossing scenario, and has little to do with the vessel being overtaken. Under aggressive configuration, however, failures are found where the ferry collides with the vessel it is overtaking, as it is unable to perform a sufficiently large evasive starboard manoeuvre because of the crossing vessel.

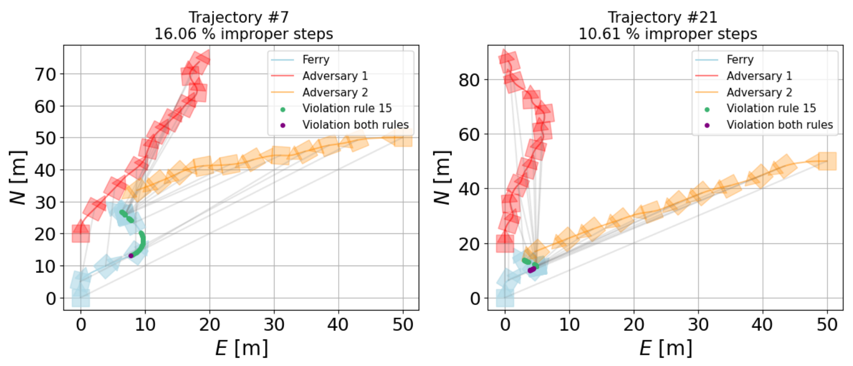


Figure 4.34: Examples of failure modes found with the conservative configuration in the overtaking scenario with two adversary vessels.

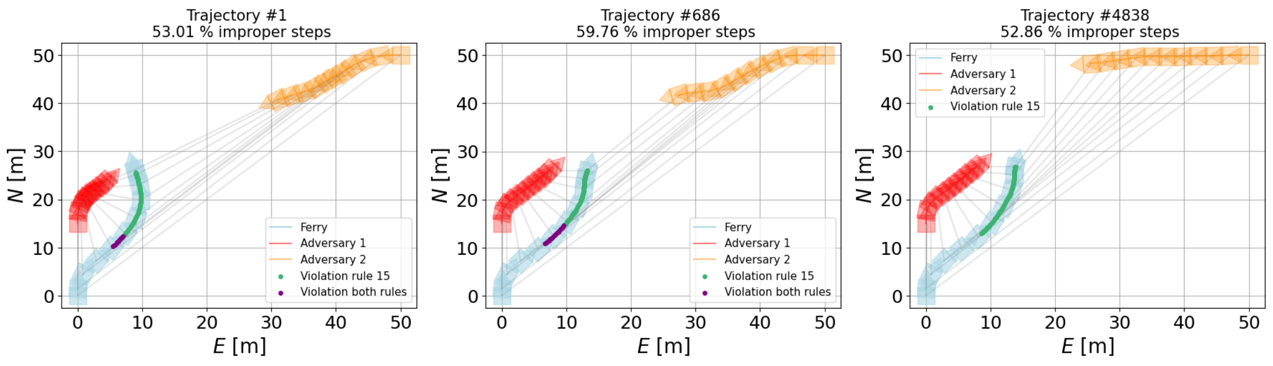


Figure 4.35: Examples of failure modes found with the aggressive configuration in the overtaking scenario with two adversary vessels.

Three adversary vessels

In the three adversary vessel case, failures are obtained for the conservative configuration where it is again the crossing vessel which is responsible for the collision. However, the failures were slightly more diverse in this scenario, as AST can be seen to experiment with the effect of adversary 3 on the failures, as depicted in the example failure modes shown in Figure 4.36. Failures were also found for the aggressive configuration where the ferry collides with the vessel being overtaken, similar to the case with two adversaries. Example failure modes are shown in Figure 4.37. The AST agent seems to figure out that the head on vessel plays little to no role in the matter of creating collisions, and the results are thus highly similar to the two-adversary case.

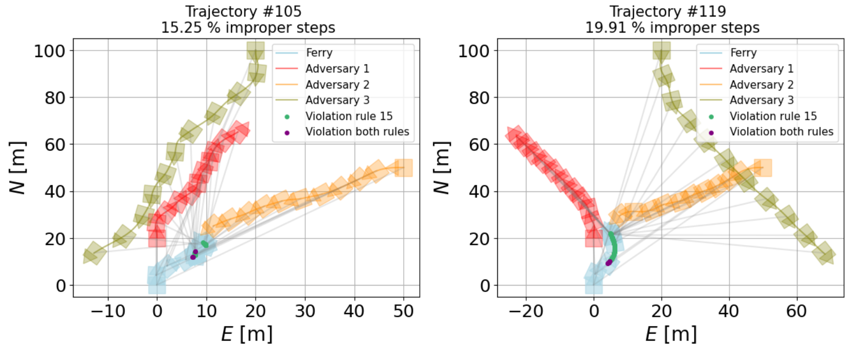


Figure 4.36: Examples of failure modes found with the conservative configuration in the overtaking scenario with three adversary vessels.

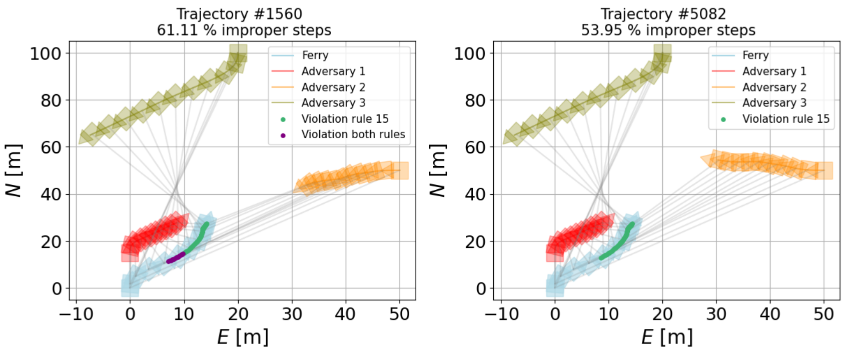


Figure 4.37: Examples of failure modes found with the aggressive configuration in the overtaking scenario with three adversary vessels.

4.2.4 General discussion of results from Part 2

The results from part two illustrate how AST is capable of identifying a diverse set of failures when applied to the MPC COLAV system, which can perform dynamic manoeuvres in compliance with the COLREGs. They also verify the implementation of the problem variation 3 introduced in part 1, which is the only problem variation implemented in this part. Problem formulation 3 introduced optimization for improper behaviour of the ferry, which is evident in most of the results in this part, as the ferry violates the COLREGs prior to collision.

Although the AST successfully identified interesting scenarios, a possible criticism of the problem formulation is that the definition of improper behaviour might not lead to more collisions. As described in Section 3.5, the definition is only based on whether or not the ferry is currently violating the COLREGs, with the violations defined as in Section 2.6. An additional problem is that the COLREGs conditions might not be appropriately defined for all cases. An example of this is shown in Figure 4.38, which shows one of the last trajectories in the head on simulation scenario with two obstacles for the aggressive configuration. In this experiment, the hypothesis was that adding a second adversary approaching from astern would make the head on situation harder to cope with for the ferry, as it would simultaneously have to make way for the overtaking vessel. However, the AST agent converged to make Adversary 2 head to starboard side, as this made the angle between the velocity vectors of the ferry and the adversary greater than 68.5° and thus the situation is deemed as a violation of Rule 15 of the COLREGs. After discovering this failure mode, measures were considered to alter the conditions for **CROSSED** and **STARBOARD**, to either make **CROSSED** false if the ferry had already passed the adversary in the North direction or to limit the angle categorized as starboard to omit the lower part of the starboard side. However, it was found that these augmentations could have unwanted impacts on the violations in other scenarios, such as when the adversary is crossing from an angle in the stern sector of the starboard side, which would not be detected as an overtaking due to the angle between the velocities. Hagen et al. (2018) also describe that the **CLOSE**-condition can be altered to take more information into account, such as the heading of the vessels, which could have improved the system by causing it to omit violations like these. Furthermore, to avoid the cases where optimization for improper behaviour did not lead to collision, the definition of the improper time step could have been implemented with a different evaluation metric than just based on the COLREGs compliance, such as a combination of the COLREGs compliance and a measure of the compliance with safety measures. Several such evaluation techniques have been proposed in research, as listed in (Pedersen et al., 2020). This could also have resulted in more collisions, as a measure of the safety could direct the AST agent to optimize for dangerous scenarios as well as cases with COLREGs-non-compliance.

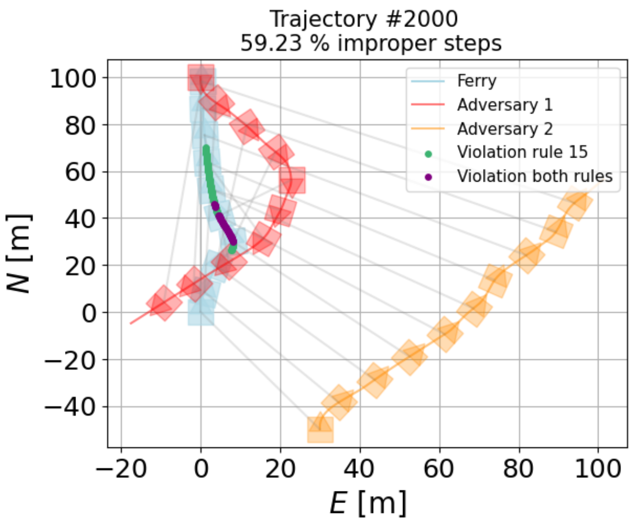


Figure 4.38: Example where optimization for improper behaviour does not lead to collision, and where the COLREGs conditions are not necessarily appropriately defined.

Another possibly problematic aspect of the problem formulation is that there are no conditions on how close in time the COLREGs violation had to be to the collision. Some of the failure modes found depict situations where the improper behaviour takes place long before the collision, as seen in the example illustrated in Figure 4.39(a). Moreover, it can also be criticized that the formulation allowed for violations regarding another adversary vessel than the one involved in the collision, in the multiple adversary scenarios. This led to some episodes where one adversary provoked improper behaviour, and the other one attacked the ferry deliberately, such as in the example portrayed in Figure 4.39(b).

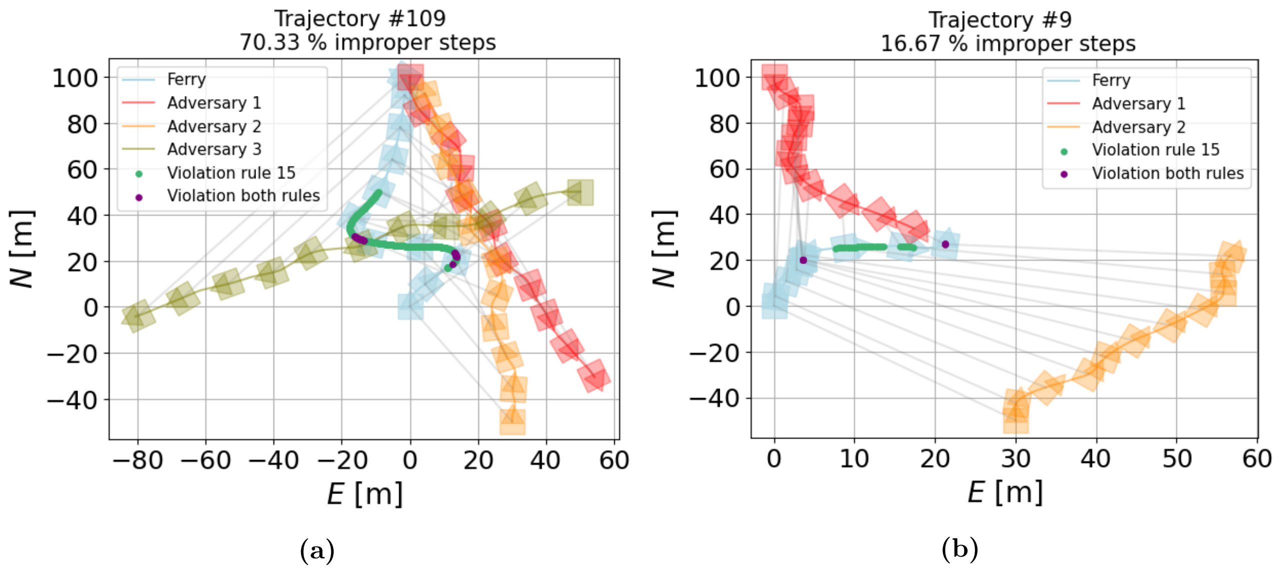


Figure 4.39: Failure examples where the improper behaviour is either registered long prior to the collision **(a)**, or where the improper behaviour is induced by another vessel than the one involved in the collision **(b)**.

Furthermore, as some of the results from multiple adversary scenarios suggest, it may be redundant to have AST control all adversaries. Some of the adversaries could have been implemented with simple straight-forward behaviours to limit the AST action space and thus the search space. There were also no restrictions on the adversary behaviour to avoid the other adversaries, which could lead to unrealistic scenarios as the adversaries proceeded to drive into each other.

Although there are some critical aspects of the implementation, the results are promising and show the potential of AST in use for COLAV systems with COLREGs compliance, as well in scenarios with multiple vessels. Furthermore, the author can vouch for the usefulness of the AST method, as AST simulations helped discover errors in the development and implementation of the MPC controllers.

4.3 General discussion of AST for both cases

In both parts, AST proves itself to constitute an efficient approach to the testing of COLAV systems. However, some aspects of the implementation and methods could have been done differently. A general discussion on the methods and the results obtained in both part 1 and part 2 are presented in this section.

4.3.1 Size of the action space matters: convergence to sub-optimal solutions

In some of the experiments, the AST agent converges to a sub-optimal solution after it had previously discovered several failure modes. This phenomenon took place in both part 1 and part 2, when optimizing for improper behaviour with $u_{max} = 0.1$ and for the crossing scenario with three adversary vessels for the conservative configuration, respectively. Example episodes are shown in Section 4.3.1.

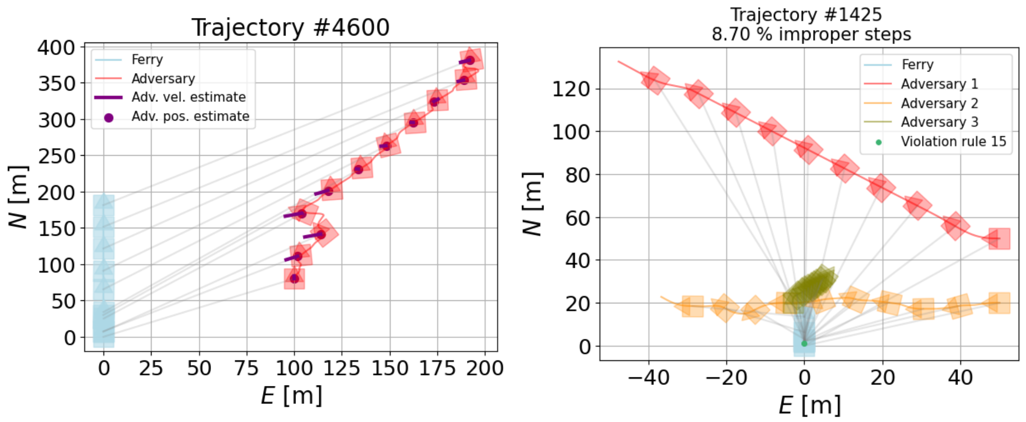
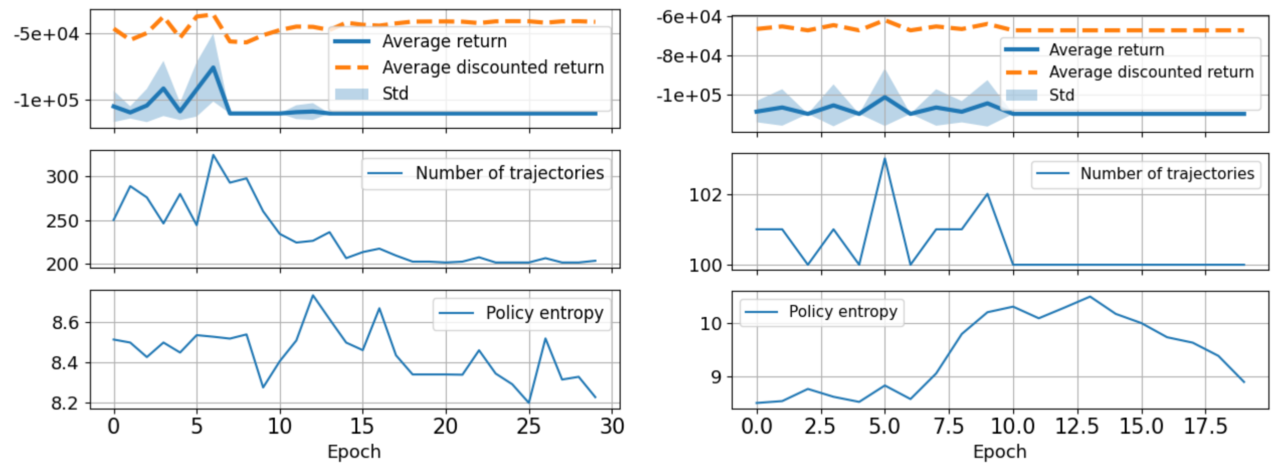


Figure 4.40: Examples of episodes from AST simulations where the AST agent converged to a sub-optimal solution, likely due to the size of the action space.

In both these cases, failures were found within the first epochs of training, and zero were found after this, as illustrated in the RL statistics portrayed in Section 4.3.1.



(a) Noise

(b) Conservative crossing, 3 adversaries

Figure 4.41: RL statistics where the agent converges to a sub-optimal solution even though it has previously identified failures.

After some time, the agent is unable to recreate the conditions that led to failure. In the case from part 1, the agent continues to explore the movement of the adversary in an inconvenient direction with respect to obtaining failures, and in the case from part two, the scenarios either end up with the ferry not violating the COLREGs enough or that it simply stands still.

Issues like these could have been addressed by methods such as Go-Explore, as implemented in Koren and Kochenderfer (2020). Go-Explore is a method for hard-exploration problems proposed in Ecoffet et al. (2021). AST can be categorized as a hard-exploration problem due to the crucial rewards only being distributed in the end, and the fact that the per-step rewards may be leading the agent away from the goal as collision-provoking actions may be improbable. This is true especially if a domain-based heuristic is hard to implement (Koren and Kochenderfer, 2020), and in this case, it has already been up for discussion whether or not the fraction of improper time steps is a good heuristic or not. In Go-Explore, previously visited states are remembered and the ones that were promising are chosen and explored. This concept prevents cases where the agent gradually forgets previously good strategies as the policy is updated (Ecoffet et al., 2021).

4.3.2 Early stopping

The output of the RL training process could have been used to determine whether or not the AST agent had converged, and the simulations could have been stopped there. The concept is known in the ML field as *early stopping*, and it has been proposed for RL problems such as in Even-Dar et al. (2006). Early stopping could have reduced the simulation time in some of the experiments where the AST converged early, and similarly expanded the number of epochs in the cases where the AST agent did not converge, in order to obtain more relevant failures.

4.3.3 Adversary probability model

As AST optimizes for the most likely failure modes of a system, the probability model used in the reward function is an important factor. Although the probability model was attempted altered in Hjelmeland (2021) by adding a heuristic model based on a risk measure to reduce the risk-seeking behaviour of the adversary, this approach was not continued further as the effect was not satisfactory. However, it is possible that further development of the adversary probability model could provide some interesting results. Even though it was seen in Hjelmeland (2021) that the high reward for collision trumped the penalties for improbable behaviour, causing AST to return the most probable of the unavoidable failures, such alterations could have an effect when combined with e.g. optimizing for improper behaviour.

There was also little to no experimenting with the covariance matrix **S** used in the Mahalanobis distance in the reward function, which could have had a positive effect. These values could also have been adjusted when the action space was restricted, which they were not. An improved probability model of the adversary could also have been acquired by the use of e.g. AIS-data. This could be an important upgrade of the method to ensure that the behaviours of the vessels actually relate to the behaviours of real-life vessels.

4.3.4 Further advantages and disadvantages of the AST method

Some advantages and disadvantages of the AST method have already been discussed in the results. In this section, some general advantages and disadvantages are presented:

- A clear disadvantage of the AST method is that it takes a lot of time, and the analysis cannot be performed until the end of the simulation. If the simulation of the SUT takes a lot of time, AST will be very slow unless there's access to great GPU power. This makes it hard to e.g. implement AST with more complex simulators, like the simulators that simulate the full milliAmpere system or evaluate for a long period of time without having very large AST steps. This issue can be avoided by implementing the method in simplified simulators, as is done in this thesis. Still, even though faced with long run time, AST can e.g. be set up to run during nighttime or when the system developers are out of office to provide valuable feedback when developers return.
- An advantage of AST is that the user can learn by the agent's learning, i.e. that the agent discovers methods during the simulations that might not be optimal, but can uncover e.g. important edge cases.
- Another great advantage of the AST method is that it is highly flexible. The heuristic or definition of improper behaviour can be implemented in any way desired, and the goal can be defined and redefined depending on what type of failure mode the system designer is looking for.

Chapter 5

Conclusion and Further Work

This thesis demonstrates the implementation, use and relevance of AST for marine COLAV systems and proposes its use as part of the safety validation process of MASS in general. Two different COLAV approaches are implemented and tested, and failure modes are identified in both cases using AST. Furthermore, different problem formulations are presented to improve the relevance of the results to system developers, in order to obtain failure cases where the ferry is partly or fully responsible for the collision due to improper behaviour. The use of optimisation for improper behaviour by the use of COLREGs violations proved especially constructive and served as the most important contribution of the work. Still, there is room for improvement in the implementation, which should be subject to further research.

The results show how AST is a useful tool for thorough safety testing and how it can easily be implemented and adjusted to the simulator of choice. AST can be used both under the development of a COLAV system, to unveil edge cases that have not been considered, or to uncover errors in the workings of the system. AST also be used after deployment, by running it whenever the system is updated to search for new potential failures in the updates. Either way, AST can add significant value to the safety validation, and the author strongly encourages MASS developers to adopt the method in their safety validation process.

5.1 Further work

Some specific improvements to the AST application to COLAV systems are proposed in the discussion of the results and these should be taken into account if the methods are subject to further research. Moreover, further work should be conducted to investigate how the AST application to MASS systems can be improved, and how the method can and applied in the safety validation pipeline.

The focus of this work has been on demonstrating the AST method in use for different system and altering the optimisation problem to find relevant failures. Due to the time consuming simulations, a prioritization had to be done on whether to focus on this or to go in-depth on the technical details of the AST method by e.g. altering the MDP solver or changing parameters or hyperparameters. However, there may be a lot to gain in exploring these areas, such as changing the DRL solver to use Proximal Policy Optimization (PPO) instead of TRPO, as PPO has shown to outperform TRPO in many areas (Schulman, Wolski, et al., 2017).

There might also be information of value in the scenarios which do not end in failure, especially in the ones that are similar to the failure trajectories. Which details lead the system to fail in one scenario, but not other similar ones? Comparisons like these could unveil important workings of the system. In (Lee, Mengshoel, et al., 2020), the scenarios which did *not* end in failure were also clustered to provide further insights to system developers. By assessing some of the episodes which did not end in failure in e.g. part two, some interesting aspects are seen as they expose cases where the ferry chooses to violate the COLREGs for a long period of time, to avoid collision. This is similar to real case scenarios, as human helmsmen would also seize to obey the COLREGs if there was a vessel heading straight towards it.

Furthermore, AST should be combined with an automatic scenario generation method, as described in Section 1.2. In this way, a test suite of challenging scenarios could be generated, and AST could be applied to optimize the episodes for failure. The author believes that this would constitute an efficient and thorough part of a safety validation system and strongly urges further work to be conducted on the subject.

References

Adaptive Stress Testing for Adversarial Learning in a Financial Environment (July 2021). Number: arXiv:2107.03577 arXiv:2107.03577 [cs, q-fin]. URL: http://arxiv.org/abs/2107.03577 (visited on 05/29/2022).

Ahvenjärvi, Sauli (2016). "The Human Element and Autonomous Ships". en. In: *TransNav, the International Journal on Marine Navigation and Safety of Sea Transportation* 10(3), pp. 517–521. URL: http://www.transnav.eu/Article_The_Human_Element_and_Autonomous_Ahvenj%C3%A4rvi,39,675.html (visited on 05/27/2022).

Althoff, Matthias and Sebastian Lutz (June 2018). "Automatic Generation of Safety-Critical Test Scenarios for Collision Avoidance of Road Vehicles". en. In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE: Changshu, pp. 1326–1333. URL: https://ieeexplore.ieee.org/document/8500374/ (visited on 05/28/2022).

Ammour, M., R. Orjuela, and M. Basset (2021). "Collision avoidance for autonomous vehicle using MPC and time varying Sigmoid safety constraints". en. In: *IFAC-PapersOnLine* 54(10), pp. 39–44. URL: https://linkinghub.elsevier.com/retrieve/pii/S2405896321015408 (visited on 05/13/2022).

Arulkumaran, Kai et al. (Nov. 2017). "Deep Reinforcement Learning: A Brief Survey". In: *IEEE Signal Processing Magazine* 34(6). Conference Name: IEEE Signal Processing Magazine, pp. 26–38.

Autoferry - NTNU (2021). URL: https://www.ntnu.edu/autoferry (visited on 11/02/2021).

Bakdi, Azzeddine, Ingrid Kristine Glad, and Erik Vanem (Dec. 2021). "Testbed Scenario Design Exploiting Traffic Big Data for Autonomous Ship Trials Under Multiple Conflicts With Collision/Grounding Risks and Spatio-Temporal Dependencies". In: *IEEE Transactions on Intelligent Transportation Systems* 22(12). Conference Name: IEEE Transactions on Intelligent Transportation Systems, pp. 7914–7930.

Bhandari, Jalaj and Daniel Russo (Oct. 2020). "Global Optimality Guarantees For Policy Gradient Methods". en. In: *arXiv:1906.01786 [cs, stat]*. arXiv: 1906.01786. URL: http://arxiv.org/abs/1906.01786 (visited on 05/02/2022).

Bholowalia, Purnima (2014). "EBK-Means: A Clustering Technique based on Elbow Method and K-Means in WSN". en. In: *International Journal of Computer Applications* 105(9), p. 8.

Bolbot, Victor and Gerasimos Theotokatos (2021). "Automatically generating collision scenarios for testing ship collision avoidance system using sampling techniques". en. In: p. 4.

Bousson, K. (Oct. 2008). "Model predictive control approach to global air collision avoidance". en. In: *Aircraft Engineering and Aerospace Technology* 80(6), pp. 605–612. URL: https://www.emerald.com/insight/content/doi/10.1108/00022660810911545/full/html (visited on 05/13/2022).

Brekke, Edmund F et al. (2022). "milliAmpere: An Autonomous Ferry Prototype". en. In: p. 15.

Corso, Anthony, Peter Du, et al. (Oct. 2019). “Adaptive Stress Testing with Reward Augmentation for Autonomous Vehicle Validatio”. In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 163–168.

Corso, Anthony, Robert J. Moss, et al. (Oct. 2021). “A Survey of Algorithms for Black-Box Safety Validation of Cyber-Physical Systems”. In: *Journal of Artificial Intelligence Research* 72. arXiv: 2005.02979. URL: `http://arxiv.org/abs/2005.02979` (visited on 10/31/2021).

Cuturi, Marco and Mathieu Blondel (2017). “Soft-DTW: a Differentiable Loss Function for Time-Series”. en. In: *Proceedings of Machine Learning Research* 70. arXiv: 1703.01541. (Visited on 03/11/2022).

Ebert, Christof and Michael Weyrich (Sept. 2019). “Validation of Autonomous Systems”. en. In: *IEEE Software* 36(5), pp. 15–23. URL: `https://ieeexplore.ieee.org/document/8802868/` (visited on 05/27/2022).

Ecoffet, Adrien et al. (Feb. 2021). “Go-Explore: a New Approach for Hard-Exploration Problems”. en. In: Number: arXiv:1901.10995 arXiv:1901.10995 [cs, stat]. URL: `http://arxiv.org/abs/1901.10995` (visited on 05/21/2022).

Eriksen, Bjørn-Olav H. et al. (Oct. 2019). “The branching-course model predictive control algorithm for maritime collision avoidance”. en. In: *Journal of Field Robotics* 36(7), pp. 1222–1249. URL: `https://onlinelibrary.wiley.com/doi/10.1002/rob.21900` (visited on 06/02/2022).

EU (2022). *European Green Deal*. en. URL: `https://www.consilium.europa.eu/en/policies/green-deal/` (visited on 05/27/2022).

Even-Dar, Eyal, Shie Mannor, and Yishay Mansour (2006). “Action Elimination and Stopping Conditions for the Multi-Armed Bandit and Reinforcement Learning Problems”. en. In: p. 27.

Faisal, Asif et al. (Oct. 2021). “Mapping Two Decades of Autonomous Vehicle Research: A Systematic Scientometric Analysis”. en. In: *Journal of Urban Technology* 28(3-4), pp. 45–74. URL: `https://www.tandfonline.com/doi/full/10.1080/10630732.2020.1780868` (visited on 05/27/2022).

Feng, Shuo et al. (Dec. 2021). “Intelligent driving intelligence test for autonomous vehicles with naturalistic and adversarial environment”. en. In: *Nature Communications* 12(1), p. 748. URL: `http://www.nature.com/articles/s41467-021-21007-8` (visited on 05/28/2022).

Fossen, Thor I. (2011). *Handbook of marine craft hydrodynamics and motion control*. Wiley: Chichester, West Sussex.

Foster, Simon, Mario Gleirscher, and Radu Calinescu (Oct. 2020). “Towards Deductive Verification of Control Algorithms for Autonomous Marine Vehicles”. en. In: *2020 25th International Conference on Engineering of Complex Computer Systems (ICECCS)*. arXiv:2006.09233 [cs, eess], pp. 113–118. URL: `http://arxiv.org/abs/2006.09233` (visited on 05/27/2022).

garage — garage v2020.09.0rc2-dev documentation (2020). URL: `https://garage.readthedocs.io/en/latest/` (visited on 06/01/2022).

Gat, Erann (1998). “On Three-Layer Architectures”. en. In: p. 11.

Hagen, I. B. et al. (May 2018). “MPC-based Collision Avoidance Strategy for Existing Marine Vessel Guidance Systems”. en. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE: Brisbane, QLD, pp. 7618–7623. URL: `https://ieeexplore.ieee.org/document/8463182/` (visited on 05/10/2022).

Hartigan, J. A. and M. A. Wong (1979). “Algorithm AS 136: A K-Means Clustering Algorithm”. In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28(1). Publisher: [Wiley, Royal Statistical Society], pp. 100–108. URL: `https://www.jstor.org/stable/2346830` (visited on 03/23/2022).

Hildermeier, Julia and Axel Villareal (July 2014). “Two ways of defining sustainable mobility: Autolib’ and BeMobility”. en. In: *Journal of Environmental Policy & Planning* 16(3), pp. 321–

336. URL: http://www.tandfonline.com/doi/abs/10.1080/1523908X.2014.880336 (visited on 05/27/2022).

Hjelmeland, Hanna W. (Dec. 2021). "Identification of failure modes in autonomous marine vehicles using Adaptive Stress Testing". En. PhD thesis. Trondheim: Norwegian University of Science and Technology.

Hjelmeland, Hanna W. et al. (2022). "Identification of Failure Modes in the Collision Avoidance System of an Autonomous Ferry using Adaptive Stress Testing". en. In: p. 8.

Hudson, John, Marta Orviska, and Jan Hunady (Mar. 2019). "People's attitudes to autonomous vehicles". en. In: *Transportation Research Part A: Policy and Practice* 121, pp. 164–176. URL: https://linkinghub.elsevier.com/retrieve/pii/S0965856416311004 (visited on 05/27/2022).

IMO (1972). *Colregs - International Regulations For Preventing Collisions at Sea.*

IMO (2021). *Autonomous ships: regulatory scoping exercise completed.* URL: https://imopublicsite.azurewebsites.net/en/MediaCentre/PressBriefings/pages/MASSRSE2021.aspx (visited on 05/28/2022).

Johansen, Tor Arne, Tristan Perez, and Andrea Cristofaro (Dec. 2016). "Ship Collision Avoidance and COLREGS Compliance Using Simulation-Based Control Behavior Selection With Predictive Hazard Assessment". In: *IEEE Transactions on Intelligent Transportation Systems* 17(12). Conference Name: IEEE Transactions on Intelligent Transportation Systems, pp. 3407–3422.

Julian, Kyle D., Ritchie Lee, and Mykel J. Kochenderfer (2020). "Validation of Image-Based Neural Network Controllers through Adaptive Stress Testing". In: Publisher: arXiv Version Number: 1. URL: https://arxiv.org/abs/2003.02381 (visited on 05/28/2022).

Kaelbling, L. P., M. L. Littman, and A. W. Moore (May 1996). "Reinforcement Learning: A Survey". en. In: *Journal of Artificial Intelligence Research* 4, pp. 237–285. URL: https://www.jair.org/index.php/jair/article/view/10166 (visited on 05/02/2022).

Koopman, Philip and Michael Wagner (2017). "Autonomous Vehicle Safety: An Interdisciplinary Challenge". en. In: *IEEE Intelligent Transportation Systems Magazine* 9(1), pp. 90–96. URL: http://ieeexplore.ieee.org/document/7823109/ (visited on 05/27/2022).

Koren, Mark, Saud Alsaif, et al. (June 2018). "Adaptive Stress Testing for Autonomous Vehicles". In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. ISSN: 1931-0587, pp. 1–7.

Koren, Mark, Anthony Corso, and Mykel J. Kochenderfer (Apr. 2020). "The Adaptive Stress Testing Formulation". In: *arXiv:2004.04293 [cs, eess, stat]*. arXiv: 2004.04293. URL: http://arxiv.org/abs/2004.04293 (visited on 11/09/2021).

Koren, Mark and Mykel J. Kochenderfer (Oct. 2019). "Efficient Autonomy Validation in Simulation with Adaptive Stress Testing". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 4178–4183.

Koren, Mark and Mykel J. Kochenderfer (Sept. 2020). "Adaptive Stress Testing without Domain Heuristics using Go-Explore". In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–6.

Koren, Mark, Xiaobai Ma, et al. (2021). "AST Toolbox: An Adaptive Stress Testing Framework for Validation of Autonomous Systems". en. In.

Kullback, Solomon and Richard Leibler (1951). "On information and sufficiency". In: *Ann Math Stat.*

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (May 2015). "Deep learning". en. In: *Nature* 521(7553), pp. 436–444. URL: http://www.nature.com/articles/nature14539 (visited on 09/07/2021).

Lee, Ritchie, Mykel J. Kochenderfer, Ole J. Mengshoel, Guillaume P. Brat, et al. (Sept. 2015). "Adaptive stress testing of airborne collision avoidance systems". In: *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*. ISSN: 2155-7209, pp. 6C2–1–6C2–13.

Lee, Ritchie, Mykel J. Kochenderfer, Ole J. Mengshoel, and Joshua Silbermann (Jan. 2018). "Interpretable Categorization of Heterogeneous Time Series Data". In: Number: arXiv:1708.09121 arXiv:1708.09121 [cs]. URL: `http://arxiv.org/abs/1708.09121` (visited on 05/25/2022).

Lee, Ritchie, Ole J. Mengshoel, et al. (Dec. 2020). "Adaptive Stress Testing: Finding Likely Failure Events with Reinforcement Learning". en. In: *Journal of Artificial Intelligence Research* 69, pp. 1165–1201. URL: `https://jair.org/index.php/jair/article/view/12190` (visited on 10/15/2021).

Likas, Aristidis, Nikos Vlassis, and Jakob J. Verbeek (Feb. 2003). "The global k-means clustering algorithm". en. In: *Pattern Recognition* 36(2), pp. 451–461. URL: `https://linkinghub.elsevier.com/retrieve/pii/S0031320302000602` (visited on 03/23/2022).

Liu, Hechen and Markus Schneider (2012). "Similarity measurement of moving object trajectories". en. In: *Proceedings of the Third ACM SIGSPATIAL International Workshop on GeoStreaming - IWGS '12*. ACM Press: Redondo Beach, California, pp. 19–22. URL: `http://dl.acm.org/citation.cfm?doid=2442968.2442971` (visited on 03/22/2022).

Mahalanobis, Prasanta Chandra (1936). "On the generalized distance in statistics". In: *Proceedings of the National Institute of Sciences (Calcutta)* 2.

Mitchell, Tom M. (1997). *Machine Learning*. McGraw-Hill series in computer science. McGraw-Hill: New York.

Moss, Robert J et al. (2021). "Autonomous Vehicle Risk Assessment". en. In: p. 39.

Moss, Robert J. et al. (Oct. 2020). "Adaptive Stress Testing of Trajectory Predictions in Flight Management Systems". In: *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*. ISSN: 2155-7209, pp. 1–10.

Müller, Meinard (2007). *Information Retrieval for Music and Motion*. en. Springer Berlin Heidelberg: Berlin, Heidelberg. URL: `http://link.springer.com/10.1007/978-3-540-74048-3` (visited on 05/25/2022).

NASA (Feb. 2022). *NASA-SW-VnV/AdaStress.jl*. original-date: 2022-02-11T07:05:40Z. URL: `https://github.com/NASA-SW-VnV/AdaStress.jl/blob/a8802eeb2c7890a100ff87470853b7d1acda` `docs/main.md` (visited on 05/28/2022).

Ng, Andrew Y., Daishi Harada, and Stuart Russel (1999). "Policy invariance under reward transformations Theory and application to reward shapin". In.

Niv, Yael (June 2009). "Reinforcement learning in the brain". en. In: *Journal of Mathematical Psychology* 53(3), pp. 139–154. URL: `https://linkinghub.elsevier.com/retrieve/pii/S0022249608001181` (visited on 05/02/2022).

One autonomous taxi, please (2021). en. URL: `https://news.mit.edu/2021/autonomous-taxi-roboats-1027` (visited on 12/08/2021).

Pedersen, Tom Arne et al. (Sept. 2020). "Towards simulation-based verification of autonomous navigation systems". en. In: *Safety Science* 129, p. 104799. URL: `https://linkinghub.elsevier.com/retrieve/pii/S092575352030196X` (visited on 06/01/2022).

Pirjanian, Paolo et al. (Oct. 2000). "CAMPOUT: a control architecture for multirobot planetary outposts". en. In: ed. by Gerard T. McKee and Paul S. Schenker. Boston, MA, pp. 221–230. URL: `http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=927442` (visited on 06/02/2022).

Porres, Ivan, Sepinoud Azimi, and Johan Lilius (Aug. 2020). "Scenario-based Testing of a Ship Collision Avoidance System". In: *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE: Portoroz, Slovenia, pp. 545–552. URL: `https://ieeexplore.ieee.org/document/9226299/` (visited on 05/28/2022).

Rawlings, James Blake, David Q. Mayne, and Moritz Diehl (2017). *Model predictive control: theory, computation, and design*. en. 2nd edition. Nob Hill Publishing: Madison, Wisconsin.

Reddy, Namireddy Praveen et al. (Dec. 2019). "Zero-Emission Autonomous Ferries for Urban Water Transport: Cheaper, Cleaner Alternative to Bridges and Manned Vessels". In: *IEEE*

Electrification Magazine 7(4), pp. 32–45. URL: `https://ieeexplore.ieee.org/document/8917843/` (visited on 05/27/2022).

Richard, M. Bellman (1957). *Dynamic Programming.* Princeton University Press,

Ringbom, Henrik (July 2019). “Regulating Autonomous Ships—Concepts, Challenges and Precedents”. en. In: *Ocean Development & International Law* 50(2-3), pp. 141–169. URL: `https://www.tandfonline.com/doi/full/10.1080/00908320.2019.1582593` (visited on 05/27/2022).

Sakoe, H. and S. Chiba (Feb. 1978). “Dynamic programming algorithm optimization for spoken word recognition”. en. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26(1), pp. 43–49. URL: `http://ieeexplore.ieee.org/document/1163055/` (visited on 03/11/2022).

Satterthwaite, David (Oct. 2009). “The implications of population growth and urbanization for climate change”. In: *Environment and Urbanization* 21(2). Publisher: SAGE Publications Ltd, pp. 545–567. URL: `https://doi.org/10.1177/0956247809344361` (visited on 05/27/2022).

Schulman, John, Sergey Levine, et al. (2015). “Trust Region Policy Optimization”. In: *Proceedings of the 32nd International Conference on Machine Learning (ICML)*. arXiv: 1502.05477. (Visited on 10/15/2021).

Schulman, John, Philipp Moritz, et al. (Oct. 2018). “High-Dimensional Continuous Control Using Generalized Advantage Estimation”. In: *arXiv:1506.02438 [cs]*. arXiv: 1506.02438. URL: `http://arxiv.org/abs/1506.02438` (visited on 11/15/2021).

Schulman, John, Filip Wolski, et al. (Aug. 2017). “Proximal Policy Optimization Algorithms”. en. In: *arXiv:1707.06347 [cs]*. arXiv: 1707.06347. URL: `http://arxiv.org/abs/1707.06347` (visited on 04/30/2022).

Sutton, Richard S. and Andrew G. Barto (2018). *Reinforcement learning: an introduction.* Second edition. Adaptive computation and machine learning series. The MIT Press: Cambridge, Massachusetts.

Tang, Yun et al. (July 2021). “Collision Avoidance Testing for Autonomous Driving Systems on Complete Maps”. In: *2021 IEEE Intelligent Vehicles Symposium (IV)*. IEEE: Nagoya, Japan, pp. 179–185. URL: `https://ieeexplore.ieee.org/document/9575536/` (visited on 05/28/2022).

Thyri, Emil H. (June 2019). “A Path-Velocity Decomposition Approach to Collision Avoidance for Autonomous Passenger Ferries”. PhD thesis. Norwegian University of Science and Technology.

Thyri, Emil H., Morten Breivik, and Anastasios M. Lekkas (2020). “A Path-Velocity Decomposition Approach to Collision Avoidance for Autonomous Passenger Ferries in Confined Waters”. en. In: *IFAC-PapersOnLine* 53(2), pp. 14628–14635. URL: `https://linkinghub.elsevier.com/retrieve/pii/S240589632031884X` (visited on 12/07/2021).

Tokdar, Surya T. and Robert E. Kass (2010). “Importance sampling: a review”. en. In: *WIREs Computational Statistics* 2(1). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/wics.56, pp. 54–60. URL: `https://onlinelibrary.wiley.com/doi/abs/10.1002/wics.56` (visited on 05/03/2022).

Torben, Tobias Rye et al. (Jan. 2022). “Automatic simulation-based testing of autonomous ships using Gaussian processes and temporal logic”. In: *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability.* Publisher: SAGE Publications, p. 1748006X211069277. URL: `https://doi.org/10.1177/1748006X211069277` (visited on 06/01/2022).

Tsavachidis, Maria and Yoann Le Petit (2022). *Re-shaping urban mobility - Key to Europe´s green transition | Elsevier Enhanced Reader.* en. URL: `https://reader.elsevier.com/reader/sd/pii/S2667091722000024?token=DD3CCA24D2B4E39E77B73D92ABCA4D64DD602E2FB5BDB9681CC260 originRegion=eu-west-1&originCreation=20220527121102` (visited on 05/27/2022).

UK, Maritime (2018). *Maritime Autonomous Surface Ships - UK Code of Practice | Maritime UK*. URL: `https://www.maritimeuk.org/media-centre/publications/maritime-autonomous-surface-ships-uk-code-practice/` (visited on 05/27/2022).

UN (2018). *68% of the world population projected to live in urban areas by 2050, says UN | UN DESA | United Nations Department of Economic and Social Affairs*. URL: `https://www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html` (visited on 05/27/2022).

Uttisrud, Anette (2019). "Hybrid collision avoidance for autonomous passenger ferries". PhD thesis.

Vartdal, Bjørn Johan, Rolf Skjong, and Asun Lera St.Clair (Aug. 2018). "REMOTE-CONTROLLED AND AUTONOMOUS SHIPS". In: *DNV GL – Maritime*. ID:1870097.

Waymo (2021). *Safety*. en. URL: `https://waymo.com/safety/` (visited on 05/27/2022).

Wei, Tao et al. (2021). "A Concept of Autonomous Waterborne Transportation Systems and its Simulation". en. In: *IFAC-PapersOnLine* 54(16), pp. 90–96. URL: `https://linkinghub.elsevier.com/retrieve/pii/S2405896321014804` (visited on 05/27/2022).

Yara Birkeland | Yara International (Nov. 2021). en. URL: `https://www.yara.com/news-and-media/press-kits/yara-birkeland-press-kit/` (visited on 12/07/2021).

ZEABUZ (2021). *ZERO EMISSION AUTONOMOUS URBAN AND COASTAL MOBILITY*. en-US. URL: `https://zeabuz.com/` (visited on 12/15/2021).