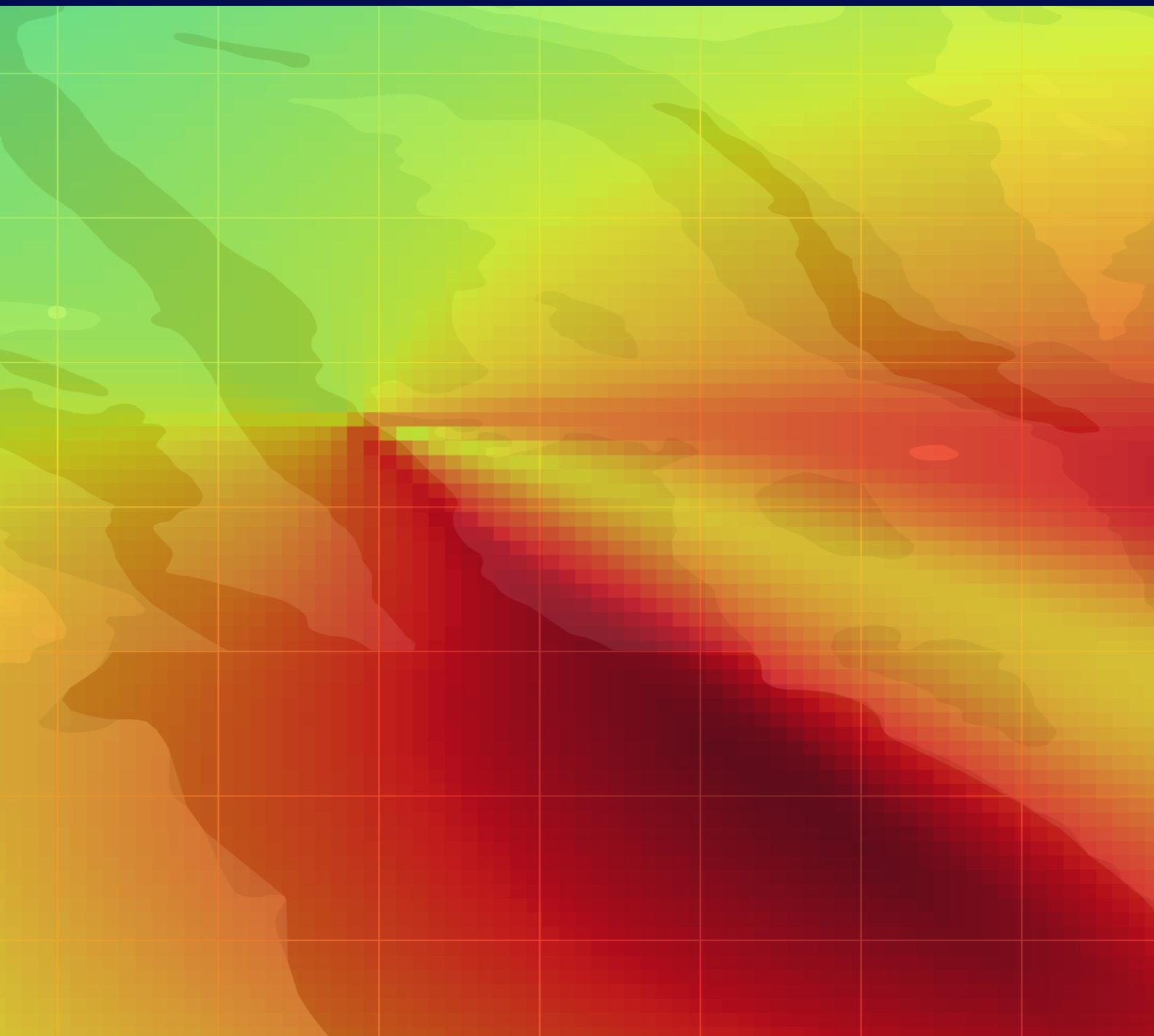
DTU Electro
Department of Electrical and Photonics Engineering

Inverse Reinforcement Learning for Decision Making in Autonomous Ship Navigation

Master Thesis

Lukas Samuel Czekalla (s233561)



Inverse Reinforcement Learning for Decision Making in Autonomous Ship Navigation

Master Thesis
24.06.2025

By
Lukas Samuel Czekalla (s233561)

Main Supervisor
Roberto Galeazzi

Supervisor
Nicholas Hansen

Cover photo: Lukas Samuel Czekalla, 2025
Published by: DTU, Department of Electrical and Photonics Engineering, Ørsteds Plads, Building 343, 2800 Kgs. Lyngby Denmark
electro.dtu.dk

Approval

This thesis has been prepared over five and a half months at the Centre for Collaborative Autonomous Systems, Department of Electrical and Photonics Engineering, at the Technical University of Denmark (DTU), in partial fulfilment for the Master of Science in Autonomous Systems Engineering degree.

It is assumed that the reader has a basic knowledge in the areas of Machine Learning (ML) and Autonomous Ship Navigation (ASN). Knowledge of Reinforcement Learning (RL) is advantageous, but a short introduction to this domain will be provided.

Lukas Samuel Czekalla - s233561

Lukas Czekalla 2025.06.24 17:45:51 +02'00'

Signature

24.06.2025

Date

Abstract

The shipping industry is progressing towards the fully autonomous operation of vessels. However, there are still many challenges in achieving safe and reliable decision making in confined and congested waterways. The present thesis contributes to this research area by investigating the use of Inverse Reinforcement Learning (IRL) for decision making in Autonomous Ship Navigation (ASN). IRL is a promising approach since it not only enables learning of decision making policies but also the inference of reward functions that encode the underlying motives of expert vessel captains. The primary objectives of this thesis were to first translate the publicly available Automatic Identification System (AIS)-data into a format that allowed for the application of IRL methods and to subsequently learn, evaluate and compare policies and reward functions for Goal-Only Navigation (GO-Navigation) and Environment-Aware Navigation (EA-Navigation) derived by different IRL methods.

First, the AIS-data has been analysed and preprocessed using maritime domain knowledge. Based on the insights gained, the available information from the AIS-data has been embedded into a Markov Decision Process (MDP). This entailed the implementation of a 2-Degree of Freedom (DoF) kinematic vessel simulator, the development and evaluation of two observation definitions as well as the development and evaluation of three distinct methods to infer actions from AIS-data. A graph-based observation definition and a Kalman filter-based approach for action inference were chosen.

Three different algorithms - Behaviour Cloning (BC), Adversarial Inverse Reinforcement Learning (AIRL) and Approximate Variational Reward Imitation Learning (AVRIL) - were evaluated on both navigation tasks. BC was used as a baseline and demonstrated decent waypoint-following behaviour and good signs of environmental awareness. It also helped establish that Graph Neural Networks (GNNs) are a superior choice for policy and reward network architectures in the EA-Navigation task. AIRL showed the best result in the GO-Navigation setting with well-performing policies and plausible reward functions. In the EA-Navigation task, the learned policy showed insufficient behaviours, while the reward function was interpretable but only provided limited insights. AVRIL performed well for GO-Navigation but showed inconsistencies in the more complex task. High uncertainties and collapses to a standard normal distribution limited the interpretability of the reward distribution.

Acknowledgements

Nicholas Hansen, Postdoc, Technical University of Denmark
Thank you to Nicholas for proposing this exiting project, the meaningful insights into the maritime world, the joyful technical discussions, the immense help with frequently troubleshooting the docker container and the great feedback on the projects progress.

Roberto Galeazzi, Associate Professor, Technical University of Denmark
Thank you to Roberto for the insightful guidance throughout the project period.

Julius Wirbel, PhD Student, Technical University of Denmark
Thank you to Julius for the cheerful conversations about writing theses, the review of sections of my report and the discussions about technical problems.

Yaqub Prabowo, Postdoc, Technical University of Denmark
Thank you to Yaqub for providing me with the initial tools to load and visualise the AIS-data and frequently supporting in docker container related issues.

Master Students of the Research Group, Technical University of Denmark
Thank you to all master students that worked on their theses next to me in the laboratory. Thank you for making the laboratory an enjoyable place to work during the project. A special mention goes to Gustav and Theo, with whom I was able to share the good and bad experiences of writing a master thesis.

Autonomous Marine Systems Group, Technical University of Denmark
Thank you to the whole Autonomous Marine Systems Group for the opportunity to learn from you and to engage in insightful discussions about my project in the weekly meetings.

Beate, Oskar & Jan
Thank you to my parents and my brother for supporting me during the two years of studies abroad. I am eternally grateful for your support. Danke!

Frederic & Stevan
Thank you to Frederic and Stevan for reviewing parts of this thesis and suggesting valuable improvements to this work.

Contents

Acronyms

A-IRL Adversarial-based Inverse Reinforcement Learning. 16

Adam Adaptive Moment Estimation. 41, 48, 53, 56, 59, 69, 71, 72, 93, 95, 96, 103

AIL Adversarial Imitation Learning. 14–16, 20

AIRL Adversarial Inverse Reinforcement Learning. iii, 4, 6, 7, 20, 47, 51, 66–73, 75, 77, 78, 80–84, 88, 92, 94, 100, 113–115, 117–129

AIS Automatic Identification System. iii, iv, 1–3, 5, 6, 21–23, 25, 28–31, 36–38, 40–42, 45, 54, 57, 79, 80, 98, 100, 127, 128, 138

ASN Autonomous Ship Navigation. ii, iii, 1–6, 8, 48, 69, 93, 127, 130

ASV Autonomous Surface Vehicle. 1

AUV Autonomous Underwater Vehicle. 1

AVRIL Approximate Variational Reward Imitation Learning. iii, 6, 7, 18, 47, 66, 93–97, 102–105, 108, 111–115, 117–125, 127, 129

B-IRL Bayesian Inverse Reinforcement Learning. 16, 18

BC Behaviour Cloning. iii, 5, 6, 14, 15, 47–49, 53–55, 59, 61, 71, 72, 83, 84, 96, 103, 113–115, 117–127, 130

BCE Binary Cross Entropy. 15, 68

BEV Birds-Eye-View. 30, 31

CNN Convolutional Neural Network. 3, 4, 128

COG Course over Ground. 21, 22, 28, 29, 79, 100, 128, 138

COLREG Convention on the International Regulations for Preventing Collisions at Sea. 1–4, 46, 57–59, 63–65, 85–92, 104, 105, 108, 109, 112, 119, 120, 123, 127, 129, 130

CPA Closest Point of Approach. 64

DAC Discriminator Actor-Critic. 20

DAgger Dataset Aggregation. 15

DCPA Distance to Closest Point of Approach. 87

DDPG Deep Deterministic Policy Gradient. 3, 13

DL Deep Learning. 3

DoF Degree of Freedom. iii, 4, 5, 28

DP Dynamical Programming. 11

DQN Deep Q-Network. 3

DTU Technical University of Denmark. ii, 5

EA-Navigation Environment-Aware Navigation. iii, 2, 6, 21, 30, 45, 46, 50, 51, 53, 55, 56, 66, 72, 82–84, 88, 92, 96, 103, 105, 108, 111–113, 118, 121, 123, 124, 127–129

ELBO Evidence Lower Bound. 18, 93

FFNN Feed Forward Neural Network. 37, 50–52

GAE Generalized Advantage Estimation. 70

GAIfO Generative Adversarial Imitation from Observation. 16

GAIL Generative Adversarial Imitation Learning. 4, 15, 16, 20

GAN Generative Adversarial Network. 15

GCL Guided Cost Learning. 19

GNN Graph Neural Network. iii, 33, 50, 52, 58, 59, 65, 118, 128, 143

GO-Navigation Goal-Only Navigation. iii, 1, 6, 30, 45, 46, 50, 51, 53–55, 72–74, 77, 78, 81, 82, 84, 96–98, 102–104, 108–110, 113–115, 117–119, 123, 124, 127

GS Grid Search. 40–42

GT ground-truth. 54

HE High Entropy. 73, 75, 77, 81, 113–115, 117

IL Imitation Learning. 2, 4–6, 8, 14–16, 20, 45, 47, 125, 126

ILEED Imitation Learning by Estimating Expertise of Demonstrators. 15

IRL Inverse Reinforcement Learning. iii, 1, 2, 4–6, 8, 9, 12, 14, 16, 17, 21–25, 28, 30, 35, 36, 42, 43, 47, 48, 50, 53, 55, 67, 69, 93, 115, 124–127, 129, 130

IRLEED Inverse Reinforcement Learning by Estimating Expertise of Demonstrators. 20, 129

LE Low Entropy. 73, 75, 77, 81, 82, 113–115

LfD Learning from Demonstration. 14, 36

LfO Learning from Observation. 14, 16

LSTM Long Short Term Memory. 129

MAP Maximum-A-Posteriori. 18

MaxEnt-IRL Maximum Entropy Inverse Reinforcement Learning. 4, 16, 19, 20

MaxMargin-IRL Maximum Margin Inverse Reinforcement Learning. 4, 16, 17

MC Monte Carlo. 11

MDP Markov Decision Process. iii, 2, 5, 6, 8, 9, 11, 14, 18, 19, 28, 45

ML Machine Learning. ii, 5

MMP Maximum Margin Planning. 17

MMSI Maritime Mobile Service Identity. 21, 22, 54, 138

MPC Model Predictive Control. 1, 2

MSE Mean Squared Error. 13, 37, 48, 70

MWAL Multiplicative Weights for Apprenticeship Learning. 17

NED North-East-Down. 35

OV Own Vessel. 2, 4, 5, 28, 30–34, 36–38, 47, 55, 57–59, 63–65, 72–75, 78, 80, 81, 85–87, 89–91, 97, 99–102, 104, 105, 108–110, 114–116, 119, 120, 128

PPO Proximal Policy Optimization. 3, 4, 6, 13, 69–73, 77, 83, 125, 129

RAM Random Access Memory. 5

ReLU Rectifier Linear Unit. 51, 52

RL Reinforcement Learning. ii, 1, 3, 6, 8–16, 28, 30, 47, 50, 69, 77, 125, 126, 129

RRT Rapidly Exploring Random Tree. 3

SAC Soft Actor Critic. 3, 13, 69, 129

SGD Stochastic Gradient Descent. 11, 13

SOG Speed over Ground. 21, 22, 28, 29, 79, 80, 100, 128, 138

SQIL Soft Q Imitation Learning. 15

Std standard deviation. 35, 41, 54, 57, 63, 73, 75, 85, 97, 104, 113, 118

TD Temporal Difference. 11

TD3 Twin Delayed Deep Deterministic Policy Gradient. 3, 13

TRPO Trust Region Policy Optimisation. 4, 12, 13, 69

TV Target Vessel. 4, 26, 28, 30–32, 35, 58, 59, 63–65, 86, 87, 90, 91, 104, 105, 108–111, 119, 120, 128

UTC Coordinated Universal Time. 21, 138

VILD Variational Imitation Learning with Diverse-Quality Demonstrations. 20, 129

List of Figures

List of Tables

1 Introduction

1.1 Motivation & Background

In recent years, more and more transportation systems have seen major advances towards fully autonomous operation. Examples of this development are autonomously operating metro systems like the Copenhagen Metro [1] or Waymo's autonomously driving taxis in San Francisco, Phoenix and Los Angeles [2].
Many advances have also been made in the maritime sector. Especially, Autonomous Underwater Vehicles (AUVs) are merely operated autonomously, as this creates substantial benefits w.r.t. time of operation, safety of personnel and cost [3]. Autonomous Surface Vehicles (ASVs) are also subject to recent research and commercial activities, as autonomously operated surface vessels offer a wide range of benefits. These range from lower operational cost due to reduced needs of trained and skilled personnel over increased efficiency in navigation and route planning to higher safety due to automated collision avoidance systems [4] [5]. Cheaper and more optimised ship designs might also be a benefit of the adoption of ASVs [5].

In order to achieve the required level of autonomy, ASVs have to comply with the Convention on the International Regulations for Preventing Collisions at Sea (COLREG) [6]. This ruleset encodes how vessels shall be operated at sea to avoid dangerous encounters and collisions. However, encoding these rules is not trivial, as some of them rely on a notion of human intuition and judgement. Therefore, the research community has proposed various approaches on achieving COLREG-compliant path following and collision avoidance. These approaches range from search algorithms like A* over control strategies like Model Predictive Control (MPC) to Reinforcement Learning (RL) approaches.

Especially, RL approaches are a promising research direction, as the agent manoeuvering the vessel learns the optimal behaviour by interacting with a simulation of the target environment. However, this approach has one downside: A human expert has to define a reward function that indicates desired behaviour to the agent, which poses a great challenge due to the not always strict notion of the COLREGs. Inaccurate reward functions may therefore lead to agents showing non-human-like behaviour.

Recent advances in the field of Inverse Reinforcement Learning (IRL) provide a tool to circumvent this issue by learning a reward function from expert demonstrations. This may lead to more human-like behaviour from artificial agents. Additionally, the learned reward function allows to interpret the intentions in ASV operation.

1.2 Problem Statement & Project Objectives

This thesis aims at investigating the use of IRL for decision making in Autonomous Ship Navigation (ASN) within confined and congested waterways.

One novelty of this thesis is the use of Automatic Identification System (AIS)-data, which contains real-life vessel trajectories. Since this data is provided in a non-IRL compliant format, identifying a suitable representation is a significant part of this thesis.

Another focus of the thesis is to design and evaluate IRL-based approaches for ASN that can achieve either one of two objectives. The first is to develop systems that navigate to a goal location within a maritime scenario (Goal-Only Navigation (GO-Navigation)).

The other is to extend this setup to the combination of waypoint following and COLREG-compliant collision avoidance into a single decision making system (Environment-Aware Navigation (EA-Navigation)).

Accordingly, the objectives of the thesis are:

1. **Formulation of the ASN problem as an IRL problem:** The tasks of waypoint following and COLREG-compliant collision avoidance shall be formulated as a Markov Decision Process (MDP) that allows for the application of IRL methods and builds on top of the available AIS-data.
2. **Development of IRL-based solutions for ASN:** IRL algorithms, that are suitable for the development of waypoint following and collision avoidance systems, shall be selected and implemented according to the constraints of the ASN problem.
3. **Evaluation of developed IRL-based solutions for ASN:** The developed IRL approaches shall be evaluated with focus on the learned behaviours and the obtained reward functions in order to assess their advantages and disadvantages w.r.t. the ASN problem.

1.3 Literature Review

In this section, relevant approaches to solving the ASN problem are reviewed, since the research community has already investigated various technical solutions to solve this problem. Those encompass either only navigation to a goal-position, path-following or collision avoidance or a combination of those in a single system. A plethora of approaches ranging from control-based over genetic algorithms up to Imitation Learning (IL)-based approaches have been proposed. Among those, the IL-based approaches are the most interesting for the scope of this thesis, because these methods offer the possibility to learn navigational behaviour from human expert demonstrators. Please note, that this literature review aims to provide an overview over different research directions but does not cover every approach suggested by the worldwide research community.

Control-based approaches

The first category of approaches is from the domain of control. MPC is used by many authors for decision making in ASN, as it predicts the dynamics of the scenario into the future and derives control actions by minimising a suitable cost function. The design of the cost-function and the constraints are the main differentiating factors between the approaches. Some of them succeed in establishing COLREG-compliant collision avoidance systems. Please refer to [7, 8, 9] for more details.

Search algorithm-based approaches

Search algorithms have also been used for path planning in congested waterways. The A* algorithm has successfully been applied by [10] and [11]. The authors of [10] use the A* algorithm to successfully avoid underwater obstacles detected from sonar images. In [11] the authors expand on A* to develop a collision avoiding system that complies with the COLREGs.

Velocity obstacle-based approaches

Another approach used for collision avoidance is the velocity obstacle method. The velocity obstacle method constructs cone-shaped obstacles in the velocity space of the Own Vessel (OV). By avoiding this velocity range, an agent can successfully avoid collisions. The authors of [12] show that this approach can be used to achieve COLREG-compliant collision avoidance. In [13] the authors also use velocity obstacles but expand it to take interactions with other vessels into account.

Dynamic window-based approaches

The dynamic window method has also been used to achieve collision avoidance in maritime scenarios. By scoring predicted trajectories this method allows to choose one future collision-free trajectory. The authors of [14] successfully managed to transfer the dynamic windows approach to the maritime collision avoidance domain. In [15], the authors develop a hybrid approach using dynamic windows to avoid collisions and using a trajectory planner like Rapidly Exploring Random Tree (RRT) to follow a trajectory.

Genetic algorithm-based approaches

Genetic and evolutionary algorithms are also part of the solution space for decision making in ASN. In [16], a collision avoidance system using an evolutionary algorithm has been developed. The authors of [17] use evolutionary neural networks for the same purpose. Ant Colony Optimisation has been used by [18] for COLREG-compliant collision avoidance.

Potential field-based approaches

Constructing a potential field, which plans a collision-free trajectory, has also been subject to research. The authors of [19] design a system that allows for route finding and collision avoidance in ASN using potential fields. In [20], the authors also use potential fields for collision avoidance but apply this method to an autonomous sailboat.

Deep Learning-based approaches

Supervised Deep Learning (DL) approaches were also proposed for collision avoidance in maritime navigation. The authors of [21] use Convolutional Neural Network (CNN) architectures and simulated visual inputs to learn collision avoiding models. Other DL based approaches are summarised in [22] and range into the field of RL as elaborated in more detail in the next section.

Reinforcement Learning-based approaches

A big field of research for decision making in ASN are RL methods with a plethora of approaches being developed. Those differentiate by using different reward function definitions, targeting different navigational problems and using different RL algorithms.

Some approaches utilise RL to just solve the path-following problem. The common denominator in this setting is the use of the Deep Deterministic Policy Gradient (DDPG) algorithm and the cross-track error as part of the reward function. However, both the exact reward definition and the state spaces differ slightly between the approaches shown in [23, 24, 25, 26, 27, 28].

Other approaches solve both, the path-following and collision avoidance problem, with one single policy. A Deep Q-Network (DQN) is used by [29] and [30] for this task. The first uses a state description that retains historical information, while the later uses images to represent its surroundings. DDPG is used by [31] to achieve collision avoidance of static obstacles. [32] also uses DDPG but combines it with AIS-data. Most other approaches utilise the Proximal Policy Optimization (PPO) algorithm. [33, 34, 35] base their approaches on the same sector-based state representation. [33] thereby introduces the use of collision-risk indices to define a reward function and achieves COLREG compliance with its policy. PPO is also used by [36] and [37].

A comparison of different RL-algorithms for decision making in ASN has also already been conducted. The authors of [38] show that the PPO algorithm outperforms DDPG, Twin Delayed Deep Deterministic Policy Gradient (TD3) and Soft Actor Critic (SAC).

Imitation Learning-based approaches

While the previously outlined approaches show promising results, most of the approaches do not really leverage data from human experts for maritime decision making. Therefore, the research community has also started to explore IL and IRL methods for the purpose of decision making in ASN:

- In [39], the Generative Adversarial Imitation Learning (GAIL) [40] algorithm is utilised to achieve avoidance of static obstacles and navigation to a goal-position. This approach learns a stochastic policy using the Trust Region Policy Optimisation (TRPO) [41] algorithm and a non-linear discriminator function based on a CNN. The state and action spaces are continuous. The state includes 2D radar images and the distance and bearing to the goal-position, while the action is defined as the change to the current heading with the velocity being fixed. The data used was manually recorded using radar sensors and encompasses 35 trajectories.
- Maximum Margin Inverse Reinforcement Learning (MaxMargin-IRL) [42] is used in [43] to achieve COLREG compliant collision avoidance. For this purpose, a stochastic policy is learned via the cross-entropy algorithm. The reward function is only linear while the state and action spaces are discrete. The approach uses different state definitions for policy and reward, but both rely on a discrete description of the Target Vessels (TVs) around the OV. The action is the discretised rudder angle, while the velocity is again fixed. To obtain the data, the authors use a 3-Degree of Freedom (DoF) simulation environment which allows for 1000 scenarios with expert-behaviour.
- The authors of [44] use Maximum Entropy Inverse Reinforcement Learning (MaxEnt-IRL) [45] to learn collision avoiding behaviour in multi-vessel scenarios. This is done using a deterministic policy and a linear reward function. The state & action spaces are discrete, as the TV positions are discretely encoded in sections. The action is one of three future waypoints. The data has been obtained by experts manoeuvering vessels in a dedicated simulation environment. One novelty of this approach is a detailed analysis of the obtained reward function.
- [46] proposes the use of GAIL [40] to achieve collision avoidance in single-TV scenarios. This is done using a stochastic policy learned via the PPO [47] algorithm and a non-linear discriminator function. The state space is continuous and encompasses proprioceptive features about the OV and the relative position and movement of the TV. In a second variant the authors also evaluate a visual state description of the environment, which uses a CNN as part of the policy and reward networks. The action is discrete and defined as the course change angle. The data is again obtained from experts acting in a simulation environment and consists of 15 demonstrations.
- GAIL [40] is also used by [48]. However, the authors use it to achieve path following during the presence of environmental disturbances. A stochastic policy is learned via TRPO [41], while the discriminator is again non-linear. The state and action spaces are continuous. Position, orientation and information about the wind and current are part of the state, while the action is defined as the desired velocity and heading of the OV. The data has been obtained using a 3-DoF simulator that allowed to act in a real-world setting with recorded environmental disturbances.
- The authors of [49] also achieve path following with their approach. However, they utilise a modification of the Adversarial Inverse Reinforcement Learning (AIRL) [50] method. Doing so, they use PPO [47] to learn a stochastic policy and they are able

to recover a non-linear reward function. The state and action spaces are continuous. While the state encompasses the OVs position, heading, velocities and cross-track-errors, the actions are defined as the surge force and yaw moment of the vessel. The data is obtained using a 3-DoF simulation and a small-scale vessel.

This literature review reveals that there are plenty of research opportunities in the field of IRL for decision making in ASN. None of the IRL-based approaches try to learn a policy that enables ASN combining waypoint-following and collision avoidance in a single policy actuating both the vessels surge and yaw motion. Additionally, no known IRL-based approach has leveraged AIS data to learn reward functions and policies. Thorough analysis of the learned reward functions is also a field that offers potential for further exploration.

1.4 Methods & Tools

This work leverages a multitude of tools and methods to develop IRL-based solutions for ASN using AIS-data.

The tools used for this work encompass:

- **Data:** The thesis is based on AIS data that has been provided by the Autonomous Marine Systems research group of Technical University of Denmark (DTU). It consists of 1001 maritime vessel scenarios recorded in confined and congested Danish coastal waters. Each scenario is complemented by the respective seachart information. No additional data beyond this dataset shall be incorporated into the work.
- **Computational Resources:** To train, simulate and obtain results a workstation with a 48-core AMD Ryzen processor, 64GB Random Access Memory (RAM) and a Nvidia GeForce RTX 3090 graphics card, owned and managed by the Autonomous Marine Systems research group was available. Of these resources 36GB of RAM and 24 cores have been solely dedicated to this thesis. Additional, but less powerful, computational resources were available towards the end of the project.
- **Software tools:** Implementation of the thesis work was conducted in Python [51], as it is the commonly used programming language for Machine Learning (ML) projects. Key libraries that have been used include PyTorch [52], Gymnasium [53], PyTorch Geometric [54], Numpy [55], Shapely [56] and Matplotlib [57].
- **Experiments:** Weights and Biases [58] was used to log the experiments conducted in this thesis. Additionally, the built-in optimisation methods of Weights and Biases were used to optimise the hyperparameters in all experiments.

From a methodological point of view, the following methods have been utilised:

- **Kinematic vessel simulation:** In addition to the AIS-data, a simulation of a vessels movement is needed to train & evaluate policies. For this purpose, a 2-DoF kinematic simulation has been adapted for the needs of this thesis.
- **MDP:** As foundation for every behaviour learning problem the concept of MDPs has been utilised in this thesis. This concept has been explored in combination with methods from the fields of ML, graph theory and state estimation to derive a representation of the AIS-data which is suitable for learning by IRL methods.
- **IL methods:** IL, as overarching set of methods, has been utilised in this thesis to establish a performance baseline for IRL methods. The Behaviour Cloning (BC)

method has been used, but it was not the objective to obtain peak performance using this method.

- **IRL methods:** IRL-based policies and their application to and evaluation for the ASN problem are the main objective of this thesis. The method selection is motivated by a recoverable reward function, continuous state spaces and a model-free definition. The AIRL and Approximate Variational Reward Imitation Learning (AVRIL) methods have been chosen. A modification or extension of the methods is not in scope of this thesis.
- **RL methods:** AIRL requires the use of a RL method to optimise a policy w.r.t. the learned reward. Therefore, the state-of-the-art PPO algorithm has been chosen. The exact technical details of this method are not in the scope of this thesis.

1.5 Contributions

This thesis makes the following contribution to the field of IRL for decision making in ASN:

- Review of literature in the fields of RL, IL and IRL.
- Analysis of the available AIS-data w.r.t. to its suitability for learning policies and reward functions in the IRL setting.
- Development and evaluation of different observation definitions based on AIS-data as part of the MDP.
- Development and evaluation of different methods for action inference from AIS-data.
- Development and evaluation of IL and IRL-based methods that allow for reaching a set goal-position using AIS-data (GO-Navigation).
- Development and evaluation of IL and IRL-based methods that allow for waypoint following and collision avoidance in confined and congested waterways using AIS-data (EA-Navigation).
- Analysis of learned reward functions w.r.t. the expertise of human operators and which environmental factors motivate navigational behaviour.
- Evaluation of all used IL and IRL approaches w.r.t. their benefits and disadvantages and possible future improvements.
- Development of a software package that allows learning from and acting in AIS scenarios.

1.6 Thesis overview

The present work translates AIS-data into a suitable representation for IRL-methods and evaluates the application of IRL-based methods to the problem of decision making in ASN.

Therefore, Chapter 2 introduces the most important technical concepts covering MDPs, RL, IL and IRL. In Chapter 3, the AIS-data used for the project is introduced theoretically and analysed w.r.t. its suitability in a IRL context. Chapter 4 then uses the results of the analysis to derive multiple variants of a MDP. Different observation and action definitions are evaluated, and recommendations w.r.t. the best variant are derived. Chapter 5 introduces the two navigational tasks that shall be completed using IRL-methods and motivates the algorithm selection. The BC method is then outlined in Chapter 6 as possible solution for the decision making problem. BC is used as a baseline and to develop & evaluate good performing policy and reward network architectures. Following up on the insights

gained, Chapter 7 applies the AIRL algorithm to the decision making problem. The implementation and training considerations are presented, and the trained policies and reward functions are analysed. Chapter 8 introduces AVRIL and discusses the considerations w.r.t. implementation & training as well as the obtained results. Chapter 9 compares the three approaches and outlines benefits as well as shortcomings and improvements for both navigational tasks. Finally, Chapter 10 summarises the achievements of this thesis and provides suggestions for future research.

The code corresponding to this thesis can be found in this repository.

2 Decision Making from Expert Data

The core of this thesis is to learn decision making policies via IRL for the ASN problem. Therefore, it is crucial to introduce the necessary technical concepts needed for IRL, starting with MDPs and the foundations of RL. This leads to the fundamental concepts of IL and its sub-field IRL.

This chapter aims at providing an intuitive overview over the subject, that shall allow the reader to understand the technical considerations of the later sections. It does not aim at providing a complete in-depth explanation of each single subject.

2.1 Markov Decision Process

A MDP is the foundation for any learning algorithm using sequential data and leverages the definition of crucial terms for the IRL problem formulation.

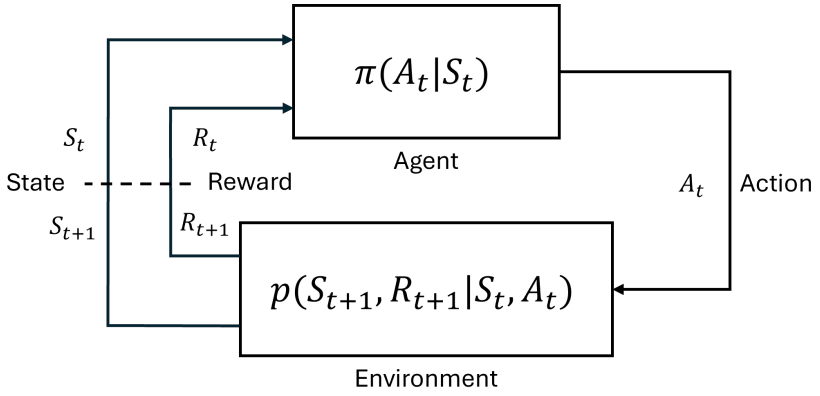


Figure 2.1: Markov Decision Process

The basic MDP model consists of an agent and an environment. The agent is the decision-making unit taking actions based on its current perception of the environment. The environment reacts to the action and generates a new state and a reward. The goal of the agent is to maximise the long-term reward with its actions. An example for a MDP would be a human playing a computer game with the objective to achieve a high score. A basic MDP is depicted in Figure 2.1 and its parts are formally defined by:

- **Agent**: The agent interacts with the environment by issuing actions A_t to the environment based on the current state S_t to increase the obtained rewards R_t. [59]
- **Environment**: The environment is a representation of the problem of interest. It accepts an action A_t as its input. The action A_t changes the environments internal state, which results in it issuing the next state S_{t+1} and the next reward R_{t+1} to the agent. [59]
- **Action** A_t: An action A_t is issued by the agent and shall create a change in the environment. A particular instance of an action is defined by $A_t \in \mathcal{A}$ where $\mathcal{A}$ is the action space that contains all possible actions. [59]
- **State** S_t: The state S_t is a representation of the current state of the environment. In fully-observable settings it exactly matches the environments internal state. However, in partially-observable settings there is a divergence between both and the

state S_t is called observation O_t. Both terms are used interchangeably in this report. A particular instance of a state is defined by $S_t \in \mathcal{S}$ where $\mathcal{S}$ is the state space that contains all possible states. [59]

- **Reward** R_t: The reward R_t is a scalar signal issued by the environment to the agent. The agent's goal is to maximise the future rewards with its actions. A discount factor $\gamma \in [0, 1)$ is used to weigh the rewards at different timesteps. This gives rise to the definition of the return G_t as future discounted cumulative reward $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$. [59]
- **Policy** $\pi(A_t|S_t)$: The policy π is the decision making part of the agent. The policy is meant to predict actions A_t based on the current state S_t to maximise the return G_t. The terms policy and agent are used interchangeably in this thesis. [59]
- **Environment Dynamics** $p(S_{t+1}, R_{t+1}|S_t, A_t)$: The environment dynamics model the transition probability $p(S_{t+1}|S_t, A_t)$ from one state to another based on an action issued by the agent. Additionally, it also models the reward issued by the environment to the agent.[59]

This constitutes the basic MDP. One aspect, that makes this model so powerful is the *markovian property*. It defines that the next state issued by the environment is only dependent on the current state and independent of the history of past states in case the above components are well-defined. [59]

When an agent is interacting with an environment it creates a so-called *trajectory*, which is an ordered list of all state-action pairs (S_t, A_t) or state-state pairs (S_t, S_{t+1}) the agent and environment have encountered. Trajectories can be generated during the learning process by the policy as is typically done in RL. But they can also be generated by a different policy that has a higher level of expertise w.r.t. acting in the environment. [60]

This gives rise to the following categorisation of policies:

- **Learner** π_L: The learner policy π_L is the policy learned in a RL or IRL setting by maximising the returns G_t. [60]
- **Expert** π_E: The expert policy π_E is relevant in the IRL-setting, where learning a policy from observed behaviours is the goal. The expert policy is a policy that has high expertise w.r.t. its task and provides expert-trajectories. [60]

To learn behaviours, the expert policy π_E can be utilised in different ways. The two main variants are based on the definitions of demonstrations and observations:

- **Demonstration**: A demonstration is a state-action pair (S_t, A_t) sampled from an expert policy π_E acting inside an environment. It assumes that one has knowledge of the expert policies actions, which is not always the case. [60]
- **Observation**: An observation is a state-state pair (S_t, S_{t+1}) sampled from an expert policy π_E. This definition assumes, that the actions of the expert policy cannot be observed and one would have to infer the action from the expert data. [60]

2.2 Reinforcement Learning

One approach to use a MDP to learn decision-making policies is RL. In RL a full MDP consisting of a state space $\mathcal{S}$, action space $\mathcal{A}$, environment dynamics $p(S_{t+1}, R_{t+1}|S_t, A_t)$ and a predefined reward function $R(S_t, A_t)$ is available. The goal in RL is to learn a policy π_L by acting with the same policy inside the environment and discovering behaviours that maximise the expected return G_t. [59]

2.2.1 Reinforcement Learning Basics

In general, modern RL methods can be split into three distinct categories. First, there are value-based methods that try to learn a so-called value function and infer a policy from this value function. Second, policy-based methods try to learn a policy directly without needing a value-function. Last, actor-critic-methods combine both of the previous categorises by learning both a value function and a policy. [61]

Value-based Reinforcement Learning

Value-based RL is the origin of all RL methods and is ideal to understand the fundamental concepts of RL.

As already outlined the main objective of RL is to learn a policy π_L that maximises the future return G_t. This objective leads to the definition of the state-value function V and the state-action-value function Q which express the expected future return conditioned on a state or state-action pair. Intuitively, this expresses the desirability of being in state S_t or taking action A_t in state S_t. [59]

Both value functions are defined by:

$$V_\pi(S_t) = \mathbb{E}_\pi[G_t \mid S_t] \tag{2.1}$$

$$Q_\pi(S_t, A_t) = \mathbb{E}_\pi[G_t \mid S_t, A_t] \tag{2.2}$$

The two value functions in equations 2.1 and 2.2 are a fundamental part of policy evaluation. In policy evaluation one is trying to estimate the expected return of a policy π. [59]

The main goal of RL however is to learn and improve a policy π_L. This can be done via policy optimisation, where the goal is to find a policy that maximises future returns to derive the optimal value functions. Equations 2.3 and 2.4 show this. [59]

$$V^*(S_t) = \max_\pi V_\pi(S_t) \tag{2.3}$$

$$Q^*(S_t, A_t) = \max_\pi Q_\pi(S_t, A_t) \tag{2.4}$$

Both steps, policy evaluation and policy improvement are chained together in an integrated procedure which learns optimal policies. This procedure guarantees convergence to optimal value function estimates and an optimal policy in discrete state and action spaces. Policy iteration alternates between the evaluation and improvement step to jointly improve both, the value function estimates and the policy towards their optimal versions. In this setting an optimal policy π_L can be derived by taking the action that maximises the current estimate of the state-action-value function. [59]

The following equation shows the definition of an otpimal policy:

$$\pi_L(A_t \mid S_t) = \begin{cases} 1 & \text{if } A_t = \arg\max_{A_t \in \mathcal{A}} Q_\pi(S_t, A_t) \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

Deriving an optimal policy therefore requires learning the state-value function V or the state-action-value function Q. Three different approaches exist for this problem:

- **Dynamical Programming**: Dynamical Programming (DP) utilises the recursive bellman equations to solve the MDP and estimate the state-value function V and state-action-value function Q. However, to do this DP requires the environment dynamics to be known. DP is only feasible in small and discrete state & action spaces $\mathcal{S}$ & $\mathcal{A}$ as well as for tabular representations of the value functions. [59]
- **Monte Carlo Estimation**: If the environment dynamics are unknown one would have to estimate the state-value function and state-action-value function from samples. Monte Carlo (MC) estimation is a technique that samples full trajectories of a policy π acting in the environment. The obtained rewards in a trajectory can be used to update the value estimates of all states or state-action pairs encountered. This provides an unbiased estimate of the value functions, which unfortunately suffers from high variance. [59]
- **Temporal Difference Learning**: Instead of sampling full trajectories to estimate the value functions, one can also just sample n-steps of a trajectory and use those as target for the value function update. This method is called Temporal Difference (TD)-learning and provides higher bias but lower variance in its value function estimates. Additionally, this method is more sample-efficient. [59]

A policy defined as in equation 2.5 in combination with either MC estimation or TD learning suffers from one problem that would hinder the learning success. Such a policy would exploit the current estimate of the state-action-value function and not reach its optimum. Instead, the policy has to explore new actions to assess if they might provide higher future returns. To do so one can define a ϵ-greedy policy that takes a random action with a probability of ϵ. This approach is suitable for discrete state and action spaces. [59]

Another approach to fostering exploration is to use an external exploratory behaviour policy to generate samples via MC estimation or TD. The policy π_L being learned is different in this case. This approach is called off-policy learning and might require aligning the distributions of both policies via importance sampling. [59]

So far, all methods are theoretically motivated by using a tabular expression of the state-value function V and the state-action-value function Q. However, such a representation does not scale well in large and continuous state and action spaces. Function approximation improves this aspect as the value functions are differentiable functions learned via Stochastic Gradient Descent (SGD). Unfortunately, using function approximation has implications on the convergence of RL algorithms. Especially, if off-policy learning, TD-learning and function approximation are applied, the RL algorithm might diverge. [59]

Policy-based Reinforcement Learning

Value-based RL methods learn a policy by first learning a representation of the state-value function V or the state-action-value function Q. Policy-based RL methods skip the intermediate step of learning a value function and directly learn a parametrised policy $\pi_{L,\theta}$. This allows to use high-dimensional and continuous state and action space. Additionally, stochastic policies can be learned, that are modelled as distributions and inherently allow for exploration by the policy. [59]

Policy-based RL methods lend themselves the definition of the policy gradient, which makes the reward expectation differentiable w.r.t. the parameters θ of the policy $\pi_{L,\theta}$. [59]

The policy gradient is defined by:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_{L,\theta}} \left[G_t \nabla_\theta \log \pi_{L,\theta} \left(A_t \mid S_t \right) \right] \tag{2.6}$$

This gives rise to the definition of a simple gradient-ascent algorithm that can be used to learn an optimal policy using sampled trajectories from the environment. [59]

Actor-Critic-based Reinforcement Learning

Actor-critic RL methods combine both, value-based and policy-based RL methods, by learning both a policy π_L and a state-value function V. The state-value function is called *critic* as it provides an indication whether the taken action leads to higher rewards than the current estimate of the state-value function suggests. It is learned via the already introduces methods and leads to the definition of the advantage function. [59]

The advantage function is defined by:

$$\delta_t(S_t, A_t) = Q_\pi(S_t, A_t) - V_\pi(S_t) = R_{t+1} + \gamma V_\pi(S_{t+1}) - V_\pi(S_t) \tag{2.7}$$

The advantage function can be used in the same gradient ascent framework as introduced for policy-based RL methods: [59]

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_{L,\theta}} \left[\delta_t(S_t, A_t) \nabla_\theta \log \pi_{L,\theta}(A_t \mid S_t) \right] \tag{2.8}$$

2.2.2 Reinforcement Learning Algorithms

Based on the core-principles of RL, multiple advanced RL algorithms have been proposed in the literature that try to address some of the still existing problems with RL. The most important algorithms shall be introduced here, as some of them have been used in existing literature of IRL for marine applications. This overview shall also enable the decision for a RL algorithm in the context of IRL.

Trust-Region Policy Optimisation

A major disadvantage of basic policy-gradient or actor-critic RL methods is, that these methods suffer from instabilities, when the update steps of the gradient ascent get too large. TRPO improves on that aspect by constraining the largest possible update step by a KL-Divergence term (eq. 2.11) and defining a surrogate objective function (eq. 2.9) using an importance sampling term (eq. 2.10). [41] [62]

The following defines the surrogate objective function that is optimises w.r.t. the parameters θ of the policy. δ_t is the advantage function and r_θ is the importance sampling term:

$$\max_\theta \mathbb{E}_{\pi_{L,\theta_{old}}} [\delta_t(S_t, A_t)\ r_\theta(A_t \mid S_t)] \tag{2.9}$$

The next equation describes the importance sampling term, where $\pi_{L,\theta_{old}}$ is the policy estimate used to sample trajectories prior to the policy update:

$$r_\theta(A_t \mid S_t) = \frac{\pi_{L,\theta}(A_t \mid S_t)}{\pi_{L,\theta_{old}}(A_t \mid S_t)} \tag{2.10}$$

The update steps are constrained by the following KL-divergence between current and old policy estimates, where ι is a constant defining the maximum divergence:

$$\mathbb{E}_{\pi_{L,\theta_{old}}} \left[D_{KL} \left(\pi_{L,\theta}(\cdot \mid S_t) \mid\mid \pi_{L,\theta_{old}}(\cdot \mid S_t) \right) \right] \leq \iota \tag{2.11}$$

Doing so, TRPO is an on-policy actor-critic algorithm suitable for continuous environments that trains a stochastic policy. [41] [62]

Proximal Policy Optimisation

PPO builds on top of TRPO by simplifying the constraint on the update step. Instead of using the KL-Divergence, PPO simply limits the size of the update step by clipping the gradient by a fraction ϵ. [47] [63]

PPO is maximising the following objective:

$$\max_{\theta} \mathbb{E}_{\pi_{L,\theta_{old}}} [\min (r_\theta(A_t \mid S_t)\ \delta_t(S_t, A_t),\ \text{clip}(r_\theta(A_t \mid S_t), 1-\epsilon, 1+\epsilon)\ \delta_t(S_t, A_t))] \quad (2.12)$$

PPO is an on-policy actor-critic algorithm suitable for continuous environments and trains a stochastic policy. [47] [63]

Deep Deterministic Policy Gradient

DDPG takes a different approach by combining the value-based Q-learning algorithm with an actor-critic-algorithm. One step involves learning the critic state-action-value function Q_ϕ via a Mean Squared Error (MSE)-loss and SGD, where ϕ are the parameters of the state-action-value function which are being learned. [64] [65]

The minimisation objective is the following:

$$\min_{\phi} \mathbb{E}_{\pi_{L,\theta}} \left[(Q_\phi(S_t, A_t) - (R_{t+1} + \gamma\ Q_{\phi_t}(S_{t+1}, \pi_{L,\theta_t}(A_{t+1}, S_{t+1}))))^2 \right] \quad (2.13)$$

As the policy is deterministic and a parametrised function, it is possible to compute the derivate of the estimated state-action-value function Q_ϕ w.r.t. the policy parameters θ using gradient ascent, as shown by the expression below. [64] [65]

$$\max_{\theta} \mathbb{E}_{\pi_{L,\theta}} [Q_\phi(S_t, \pi_{L,\theta}(A_t \mid S_t))] \quad (2.14)$$

All this is done in an off-policy fashion and is suitable for continuous environments. The algorithm characterises as an actor-critic algorithm that trains a deterministic policy. [64] [65]

Twin Delayed Deep Deterministic Policy Gradient

TD3 improves on DDPGs shortcomings w.r.t. overestimation by learning two state-action-value functions Q_ϕ. Additionally, TD3 introduces delayed policy updates, where the policy is updates less often than the state-action-value functions Q_ϕ. Noise is also added to the target policy to foster exploration. [66] [67]

Overall, TD3 is an off-policy actor-critic algorithm that can use continuous environments to learn a deterministic policy. [66] [67]

Soft Actor Critic

The SAC algorithm introduces entropy regularisation to the field of RL, which allows for better exploration properties. Similar to TD3, two state-action-value functions $Q_{\phi,j}$ are learned. [68] [69]

The state-action value function is learned via minimising the following regularised objective w.r.t. the state-action-value functions parameters ϕ. α is a configurable hyperparameter:

$$\min_{\phi} \mathbb{E}_{\pi_{L,\theta}}\Big[\big(Q_{\phi,i}(S_t, A_t) - \big(R_{t+1} + \gamma \min_{j=1,2}\big(Q_{\phi_t,j}(S_{t+1}, \pi_{L,\theta}(A_{t+1} \mid S_{t+1})) - \alpha \log \pi_{L,\theta}(A_{t+1} \mid S_{t+1})\big)\big)\big)^2\Big] \tag{2.15}$$

To learn the policy using the estimated state-action-value functions, one has to use the reparametrisation trick, to derive a differentiable policy $\tilde{\pi}_{L,\theta}$. [68] [69]

Learning the policy is done by maximising the following equation w.r.t. the policies parameters ϕ:

$$\max_{\theta} \mathbb{E}_{\pi_{L,\theta}}\left[\min_{j=1,2} Q_{\phi,j}(S_t, \tilde{\pi}_{L,\theta}(A_t \mid S_t)) - \alpha \log \tilde{\pi}_{L,\theta}(A_t \mid S_t)\right] \tag{2.16}$$

This is an off-policy actor-critic algorithm that learns a stochastic policy and can be used in continuous environments. [68] [69]

2.3 Imitation Learning

An alternative to RL is IL, which also utilises a MDP to learn a policy π_L. While RL requires the definition of a reward function, which is difficult in high-dimensional state and action spaces or in complex tasks where codifying the desired objectives into a reward function is not possible, IL avoids using a reward function by using trajectory samples from an expert policy π_E. Instead of maximising the expected return G_t, it is the objective of IL to learn a policy π_L that imitates the expert policy π_E. [70] [60]

Learning policies π_L from expert trajectories can be achieved using different methods. The three main categories are BC, Adversarial Imitation Learning (AIL) and IRL. BC follows a supervised approach, where a direct mapping between states and actions is learned. In AIL a discriminator is used to distinguish between expert and learner trajectories, which leads to a reward-like signal that can be used for policy learning. IRL usually uses two steps to learn an imitating policy. Initially an interpretable reward function is learned from expert trajectories, which can then be used by a RL algorithm to optimise and learn a policy. [70]

On top of this main separation of approaches the algorithms are also characterised by a couple of other properties.

Approaches can be separated into Learning from Demonstration (LfD) and Learning from Observation (LfO). In LfD the expert trajectories contain demonstrations meaning that the actions of the experts are available. LfO describes the case where the expert trajectories do not contain the actions of the experts and are just composed of observations. [60]

An algorithm may either be online or offline. An online method requires an environment which it can use to sample new trajectories using its current policy estimate. An offline method works purely with already recorded trajectories, making it computationally more efficient. [70] [60]

Algorithms may also be split into model-free and model-based approaches. Model-based approaches require a model of the environment dynamics to learn a policy. This limits their use to only those cases, where such a model can be derived. Model-free methods do not need such a model and are therefore more flexible. [70] [60]

Last, methods can either learn purely from expert trajectories or also from non-expert trajectories. In the case of expert trajectories, it is assumed that the experts act optimal w.r.t. the task. In reality, that might not always be the case. Therefore, some methods can use non-expert trajectories to learn optimal policies. [70] [60]

Figure 2.2 provides an overview over a selection of different methods that will be elaborated further below.

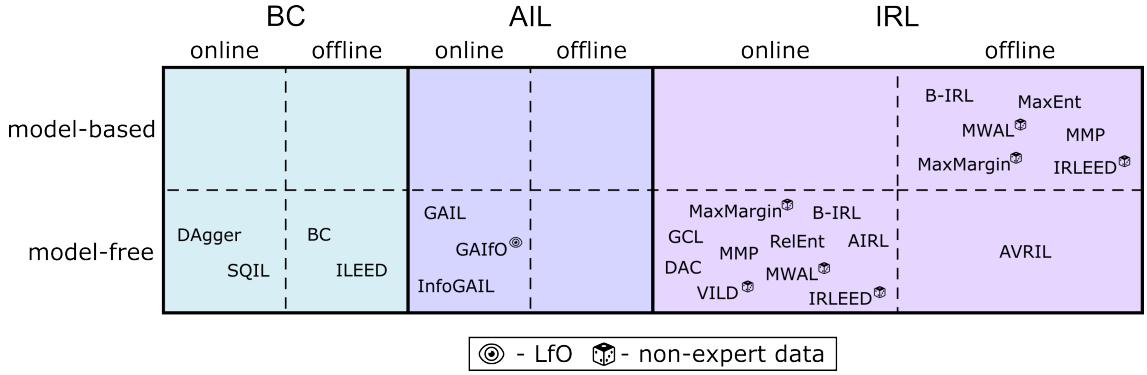


Figure 2.2: Categorisation of IL methods

2.3.1 Behaviour Cloning

BC is based on a simple supervised learning approach. The aim is to learn a direct mapping between states and actions from expert demonstrations. This can be done completely offline and is therefore computationally very efficient. However, there is one big downside to BC called the *compounding error* effect. This effect describes that in a sequential decision making setting the inaccuracies of a policy will lead the encountered states to become out of distribution w.r.t. the training data. Once such a state is encountered the actions taken by the policy will lead to an even larger error which magnifies the problem. [71, 72, 70]

The most common approach building on top of BC is Dataset Aggregation (DAgger), which is an iterative algorithm. It constructs its policy estimate from a linear combination of the current and the previous policy estimate and uses it to sample new demonstrations from the environment. Those demonstrations then get added to the dataset and are used to learn a new policy estimate. By doing so DAgger improves on BC in a model-free manner but requires access to an environment for the generation of new demonstrations. [73]

Some other BC algorithms exist as well. Imitation Learning by Estimating Expertise of Demonstrators (ILEED) expands BC by also learning to estimate the expertise of different experts contained in the expert trajectories [74]. Soft Q Imitation Learning (SQIL) treats BC as a RL problem by incentivising the policy to return to states from expert demonstrations upon acting out-of-distribution. Therefore, a reward of 1 is issued for matching a state-action pair from demonstrations and a reward of 0 otherwise [75].

2.3.2 Adversarial Imitation Learning

AIL is based on an adversarial and generative framework originating from Generative Adversarial Networks (GANs) [76]. The first IL algorithm based on this core principle was GAIL [40], which learns a discriminator $D(S_t, A_t)$ and a policy π_L. The discriminator $D(S_t, A_t)$ is trained via a simple Binary Cross Entropy (BCE) objective to distinguish between expert demonstrations and demonstrations sampled from the latest policy estimate. The discriminator output is then used as part of a reward function to learn a policy using a stochastic RL algorithm. The reward is thereby defined as:

$$R_{\text{GAIL}}(S_t, A_t) = -\log D\left(S_t, A_t\right) \tag{2.17}$$

The two steps of discriminator training and policy optimisation are done interchangeably in an iterative learning paradigm. Still, this method does not qualify as IRL method as the reward function R_{GAIL} is not interpretable. The reason for this is, that the discriminator would predict 0.5 if the policy would act as optimal as the expert does. In this case the reward is just uniform and therefore not interpretable. [40] [77]

The working principle of GAIL was also transferred to the LfO setting. Generative Adversarial Imitation from Observation (GAIfO) was proposed by [78] and modifies the discriminator to be a function of the current and next state.

Another algorithm using this core principle is InfoGAIL proposed by [79]. This algorithm not only allows to learn a policy imitating experts, but it also learns latent features that enable a differentiation between different behaviours.

2.3.3 Inverse Reinforcement Learning

IRL is the third field that proposes solutions to the IL problem. The core principle of all methods within the IRL domain is that they learn an interpretable reward function that can be used to either optimise policies or to gain understanding about the underlying desires and motivations of experts acting in their respective domain.

Within IRL, different types of approaches have been developed. MaxMargin-IRL tries to learn a reward function, which maximises the value-margin between rewards obtained by expert trajectories and policy trajectories. Bayesian Inverse Reinforcement Learning (B-IRL) uses a different approach and tries to learn a posterior reward distribution using Bayesian inference. MaxEnt-IRL on the other hand lends itself the principle of maximum entropy to learn a reward that is based on the minimal evidence from the expert trajectories. Last, Adversarial-based Inverse Reinforcement Learning (A-IRL) builds on top of the already shown AIL approaches but transfers the discriminator to an interpretable reward function. [80]

The main problem with the IRL-problem is that multiple reward functions can motivate the same policy. This is called the *reward ambiguity problem* and is solved differently by the different IRL algorithms. [80]

Maximum Margin IRL

MaxMargin-IRL is the origin of all IRL algorithms. It tries to find a reward function so that the expert behaviour is the result of an optimal policy acting w.r.t. this learned reward function. It does so by maximising the margin between the expected values of the optimal policy and all other policies. Doing this iteratively will lead to a reward function that mirrors the expert behaviours. [80]

[81] was the starting point for all IRL research as it proposes three basic MaxMargin-IRL algorithms. The first algorithm assumes that the expert policy and a model of the environment is known and that the state space is finite. The following constraint is the result of this algorithm and defines all possible reward functions that are a solution to the RL problem:

$$\left(\mathbf{P}_{\pi(S_t)} - \mathbf{P}_{A_t}\right)\left(\mathbf{I} - \gamma \mathbf{P}_{\pi(S_t)}\right)^{-1}\mathbf{R} \geq 0 \tag{2.18}$$

where **P** are transition matrices and **R** is a vector of rewards. However, this approach is limited, as $\mathbf{R} = 0$ is a solution and its scope is quite constrained.

The second algorithm improves certain aspects, by assuming an infinite state space and using a linear function approximation for the reward function. The constraint is solved via linear programming and defined by:

$$\mathbb{E}_{S_t \sim P_{S,\pi_E(S)}} [V^{\pi}(S_t)] \geq \mathbb{E}_{S_t \sim P_{S,A}} [V^{\pi}(S_t)] \tag{2.19}$$

where the reward function shall make the expected value for the expert policy larger than the expected value of any other policy.

The third algorithm does not rely on the knowledge of the expert policy π_E. Instead, it uses sampled trajectories to find a reward function so that the expert policy maximises:

$$\mathbb{E}_{S_0 \sim \mathcal{D}_{\mathcal{E}}} [V^{\pi}(S_0)] = \sum_{i=0}^{N} w_i \, \mathbb{E}_{S_0 \sim \mathcal{D}_{\mathcal{E}}} \left[\sum_{t=0}^{T} \gamma^t \phi(S_t) \right] \tag{2.20}$$

where $\phi(S_t)$ are the basis functions from which the linear reward function is constructed in combination with the learnable weights w_i. $\mu(\pi_E) = \mathbb{E}_{S_0 \sim \mathcal{D}_{\mathcal{E}}} \left[\sum_{t=0}^{T} \gamma^t \phi(S_t) \right]$ is also known as the feature expectation.

In practical terms the maximisation of the expression above is done by maximising the margin between the estimated values of the optimal policy w.r.t. the current reward function estimate and all previous policy estimates. Two assumptions are needed for this to work. It has to be possible to estimate the optimal policy, and new trajectories need to be sampled, making it an online algorithm. [81] [80]

The authors of [42] propose a MaxMargin-IRL algorithm which first estimates a reward function based on the expert's behaviour and then finds a policy that is optimal w.r.t. this reward function. Using a linear reward function estimate, its goal is to minimise the difference in feature expectations of the learned policy and the expert policy $\|\mu(\pi) - \mu(\pi_E)\|$ in an iterative and converging algorithm.

Another MaxMargin-IRL method called Maximum Margin Planning (MMP) is introduced by [82], where the authors propose to find a linear reward function so that the policy learned from this reward function is similar to the expert demonstrations. This is done as a quadratic optimisation problem which uses a slack variable and a loss function to maximise the margin between expert and learner trajectories.

Some other approaches focusing on different aspects or short-comings of the shown approaches have also been developed. [83] propose an algorithm called Multiplicative Weights for Apprenticeship Learning (MWAL) based on [42] that can handle non-expert demonstrations by treating the IRL problem as two-player zero sum game. The authors of [84] use gradient optimisation techniques to learn a reward function based on the MaxMargin-IRL problem formulation.

MaxMargin-IRL is the origin of IRL methods, however they show considerable downsides in their applicability to complex problems. All shown algorithms rely on linear reward functions and feature expectations, which requires careful feature engineering. Additionally, all algorithms require complex optimisation, which makes them not scale well to large continuous partially-observable state and action spaces. Also, the reward ambiguity problem is not explicitly addressed by these methods. [80]

Bayesian IRL

B-IRL methods use the Bayesian framework to learn a posterior distribution $P(R \mid \mathcal{D}_E)$ of the reward over trajectories of expert demonstrations $\mathcal{D}_E$. The goal is to learn a reward function R which maximises the probability that it is underlying the expert demonstrations $\mathcal{D}_E$. The posterior probability can be computed using the Bayes theorem: [85] [80]

$$P(R \mid \mathcal{D}_E) = \frac{P(\mathcal{D}_E \mid R)\, P(R)}{P(\mathcal{D}_E)} \tag{2.21}$$

The likelihood $P(\mathcal{D}_E \mid R)$ is modelled as exponential distribution as this choice attributes higher probabilities to demonstrations with high rewards. [85] [80]

The likelihood is defined by:

$$P(\mathcal{D}_E \mid R) = \frac{1}{Z} e^{\alpha E_R(\mathcal{D}_E)}, \quad \text{where } E_R(\mathcal{D}_E) = \sum_{t=1}^{T} Q_R^*(S_t, A_t) \tag{2.22}$$

The main issue with this likelihood is the computation of the normalising constant Z, which is done differently in various algorithms. [80]

The prior $P(R)$ may be used to induce prior knowledge about the reward distribution into the B-IRL problem. Therefore, the distribution type may be chosen according to the problem setup but will most likely be an uniform, Gaussian or beta distribution. [85]

To learn policies using the posterior distribution, point estimates of the reward distribution are needed. Multiple approaches to solve this issue problem have been proposed in the literature. [80]

The first approach can be found in [85], where the authors propose the so-called policy walk algorithm which is based on a markov chain monte carlo algorithm to compute the mean of the posterior distribution needed to learn the reward function and the policy.

Another approach proposed by [86] uses Maximum-A-Posteriori (MAP) estimation to estimate the most probable reward according to the posterior distribution. They achieve this by using a gradient method which solves the following maximisation objective:

$$R_{MAP} = \text{argmax}_R \left[\log P(\mathcal{D}_E \mid R) + \log P(R)\right] \tag{2.23}$$

This requires the likelihood and prior to be a differentiable function in order to use gradient descent for the estimation of R_{MAP}.

Most B-IRL approaches do not scale well, as their rewards are modelled in tabular form, being closely tied to the dimensionality of the state-space. Additionally, they require to solve the MDP to learn a policy, which requires acting in the environment. [80]

The authors of [87] mitigate those issues by proposing an offline B-IRL algorithm that works with continuous state-spaces to learn a reward distribution. AVRIL is based on a variational framework which learns a policy and a reward distribution by maximising a Evidence Lower Bound (ELBO). For more details on this algorithm please refer to Chapter 8 as a detailed explanation of this method is provided there.

Maximum Entropy IRL

MaxEnt-IRL, introduced by [45], primarily addresses the reward ambiguity problem, as it maximises the likelihood of expert trajectories given a reward function while also ensuring that the entropy of the trajectory distribution is maximised. This ensures that among all reward functions that might explain expert behaviour, only the reward function which imposes the least unnecessary assumptions is chosen. [80]

To achieve this, [45] use a linear reward function and try to match expected feature counts with observed feature counts from the experts. Therefore, the probability of observing a single trajectory is modelled as:

$$P(\mathcal{T} \mid w) = \frac{1}{Z(w)} e^{\sum_{s \in \mathcal{T}} w^T f(s)} \tag{2.24}$$

where w are the weights of the reward function, $\sum_{s \in \mathcal{T}} f(s) = f(\mathcal{T})$ is the trajectory feature count and $Z(w)$ is the normalising function summing over all possible trajectories. The reward weights can be learned using expert demonstrations by maximising the log-likelihood via gradient descent:

$$\begin{aligned} w^* &= \text{argmax}_w \sum_{\mathcal{T} \in \mathcal{D}_E} \log P(\mathcal{T} \mid w) \\ &= \text{argmax}_w \left(w^\top \frac{1}{|\mathcal{D}_E|} \sum_{\mathcal{T} \in \mathcal{D}_E} f(\mathcal{T}) - \log Z(w) \right), \quad \text{where} \quad Z(w) = \sum_{\mathcal{T}} e^{w^\top f(\mathcal{T})} \end{aligned} \tag{2.25}$$

The optimisation consists of solving the forward MDP to compute the feature counts of the normalising function $Z(w)$ using value iteration given the current reward estimate. This assumes knowledge of an optimal policy. Using this the log-likelihood can be maximised to update the reward estimate. [45]

So far, MaxEnt-IRL assumes known environment dynamics, a linear reward function and that a near-optimal policy can be computed. However, this setup becomes computationally intractable for larger state and action-spaces due to the large number of feature counts that would have to be computed. Other MaxEnt-IRL algorithms improve exactly those aspects. [80]

[88] proposes a method that expands MaxEnt-IRL to the model-free setting, which allows to compute the feature counts using samples. It achieves this by minimising the relative entropy between the trajectory distribution under a baseline policy and the trajectory distribution under a policy that matches the experts feature counts. Additionally, it uses importance sampling for gradient estimation which loosens the requirement for learning an optimal policy.

In [89] this is even further expanded to also use non-linear function approximators as reward functions. The Guided Cost Learning (GCL) algorithm unifies policy and reward learning, as samples for policy improvement are also used to update the reward function. The reward function is updated in the inner loop of policy learning, which makes it more efficient.

Some newer approaches based on MaxEnt-IRL also try to solve the problem of learning from non-expert demonstrations. In [90] the authors propose their method Variational Imitation Learning with Diverse-Quality Demonstrations (VILD) which models the expertise of different demonstrators with a probabilistic graphical model which allows to determine a demonstrator's expertise and to learn a reward function. Another method attempting the same is called Inverse Reinforcement Learning by Estimating Expertise of Demonstrators (IRLEED) and is proposed by [91]. This method combines MaxEnt-IRL with a model for demonstrator suboptimality that allows for a quantification of demonstrator precision and accuracy.

Adversarial-based IRL

As shown by [77], the AIL case from section 2.3.2 is closely related to MaxEnt-IRL. In fact, GAIL, the adversarial IL method proposed by [40], can be modified so that it learns an interpretable reward function.

The algorithm doing this is called AIRL and has been introduced by [50]. All that is needed to receive an interpretable reward function is to use a different definition of the discriminator and reward function. The discriminator is composed of a learnable function f_ϕ and the current policy estimate:

$$D_\phi(S_t, A_t) = \frac{\exp(f_\phi(S_t, A_t))}{\exp(f_\phi(S_t, A_t)) + \pi_L(A_t \mid S_t)} \tag{2.26}$$

To not have uniform rewards if the learner imitates the expert well, the reward is defined via:

$$R_{\text{GAIL}}(S_t, A_t) = \log D_\phi\left(S_t, A_t\right) - \log(1 - D_\phi\left(S_t, A_t\right)) \tag{2.27}$$

For more details on the AIRL method please refer to Chapter 7.

A variant of both GAIL and AIRL that provides benefits in terms of sample efficiency is Discriminator Actor-Critic (DAC). This method uses off-policy learning of the discriminator and the policy while also learning an unbiased reward function [92].

3 AIS-Data

The development of IRL based policies for waypoint-following and collision avoidance in maritime scenarios builds on the available data and simulation environment. As stated in Chapter 1.5 the novelty is the use of AIS data and requires thorough analysis.

3.1 Background on AIS-data

The AIS is a system that is used to automatically identify vessels by transmitting navigational information like position, Speed over Ground (SOG) and Course over Ground (COG) to improve navigational safety and vessel-to-vessel communication. Therefore, most vessels have to be equipped with AIS-transceivers and communicate AIS related information periodically [93]. The information is received by shore or satellite-based receivers to track ship movement or other AIS-capable vessels to enhance navigational safety. [94] [95]

The information contained in the AIS protocol spans different domains and consists of:

- **Static information:** This is information that is related to a specific vessel, like the Maritime Mobile Service Identity (MMSI), the vesseltype or the vessels length & width. This information is rarely changed as it is specific to one vessel. [96] [95]
- **Dynamic information:** This information is visualising the vessel's current navigational status characterised by the vessels position, the time in Coordinated Universal Time (UTC), the COG, the SOG and different navigational stati. [96] [95]
- **Voyage related information:** This information relates to the current voyage. This includes information like the vessels current payload-induced draught or the start and destination ports. [96] [95]
- **Short safety messaging:** These are short navigational safety related messages transmitted via AIS. They are not standardised and mostly used to warn other vessels about navigational risks. [96] [95]

The above, is only a very brief overview of the different information that AIS entails. For more information on the exact information transmitted by AIS please refer to [97]. The provided AIS-dataset uses a preprocessed format of the AIS-data that does not contain the full set of AIS data instances.

3.2 Analysis of AIS-Scenarios

The available dataset consists of so-called AIS-scenarios, which capture a sequence of vessel movements in a smaller geographical region in Danish waters. Before using those for any IRL algorithm a thorough analysis of the data is required. The analysis will be made w.r.t. the EA-Navigation task.

3.2.1 Features of AIS-Scenarios

The AIS-scenarios contain preprocessed information from real AIS recordings. The preprocessing makes it easier to process the scenarios for the purpose of learning and analysing navigational behaviours. The scenarios contain three parts:

- **Metadata:** The metadata of an AIS-scenario contains general information about the scenario. This is a unique identifier, the start and end time of the scenario as well

as the number of vessels included in the scenario. Additionally, the location of the scenario is provided by latitude and longitude ranges. Lastly, the sampling time of the vessel trajectories is included.

- **Vessel data & trajectories:** Multiple vessels are usually included in one scenario. Each vessel is characterised by metadata that entails the MMSI, the vessel-type, the width, length and draught of the vessel as well as its navigational status. Additionally, the trajectory of the vessel is defined by the latitude, longitude, SOG and COG the vessel has at every timestep of the scenario.
- **Seachart:** The third component of the data is the seachart, which is naturally not a part of AIS-data but has been included into the dataset. This seachart provides location information about land, buoys, searoutes and different depths of the ocean inside the latitude and longitude boundaries of the scenario. Land and depths are represented by polygons, while buoys are points and searoutes are lines.

All these features are sufficient for learning navigational behaviours from data, as the features encode all information that a human operator of a vessel would need to act in a maritime environment. Only weather information might be a useful addition but is out of scope at this point.

A detailed overview over all contained features may be found in appendix A.

3.2.2 Analysis of AIS-Scenarios

The dataset comprises a total of 1001 AIS scenarios recorded in Danish coastal waters. All of these scenarios have been extracted as they contain situations of closer encounters between vessels. However, it is crucial to initially analyse the data and assess whether it is suitable in the context of IRL.

First, the geographical locations of all the scenarios are of interest. Figure 3.1 shows that all scenarios are indeed from Danish coastal waters. The majority of the scenarios is from the *Storebælt* region, while the other two main waterways *Øresund* and *Lillebælt* are underrepresented.

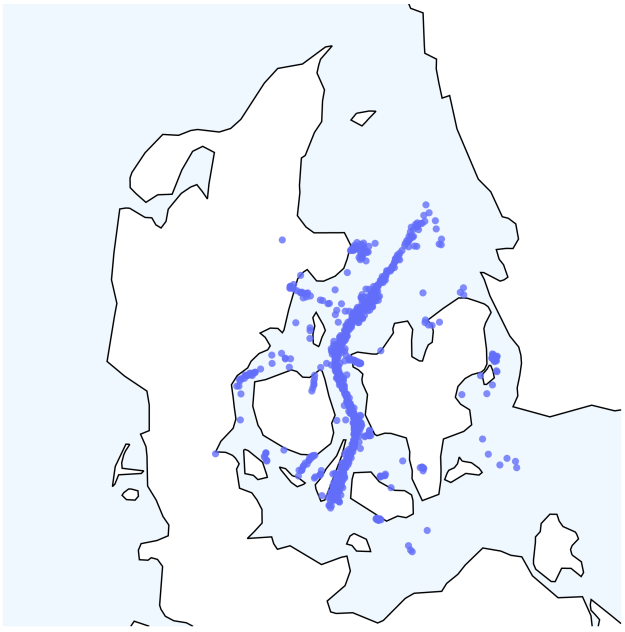


Figure 3.1: Geographical locations of all scenarios in Danish coastal waters

Second, the sampling-rates and duration of all scenarios are of interest. Analysis reveals that most of the scenarios cover a duration of 30 min. However, very few scenarios have significantly shorter durations as can be seen in Figure 3.2. The sampling time is constant with 10 s in all scenarios.
Closer examination of all scenarios additionally reveals that not all vessel trajectories span the whole duration of the scenario, as their AIS-transmitter has been enabled or disabled in the middle of the scenario. These scenarios need special attention to ensure the integrity of the dataset.

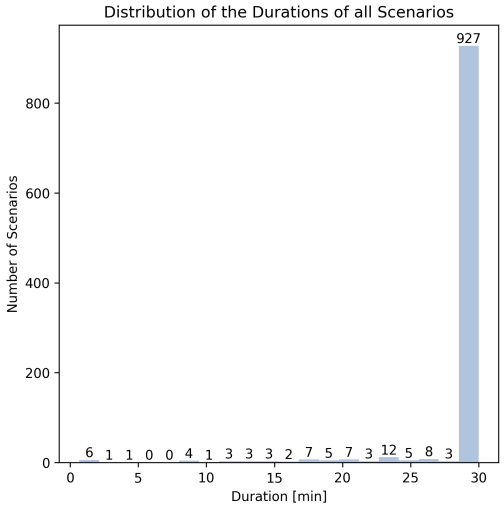


Figure 3.2: Distribution of the durations of all scenarios

Next part of the analysis is the number of vessels included in each scenario. Figure 3.3 reveals most scenarios contain 2-5 vessels. The mean of vessels per scenario is 3.84 while the median is 4. A maximum of 13 vessels per scenario can be observed once, while the minimum number is 1.

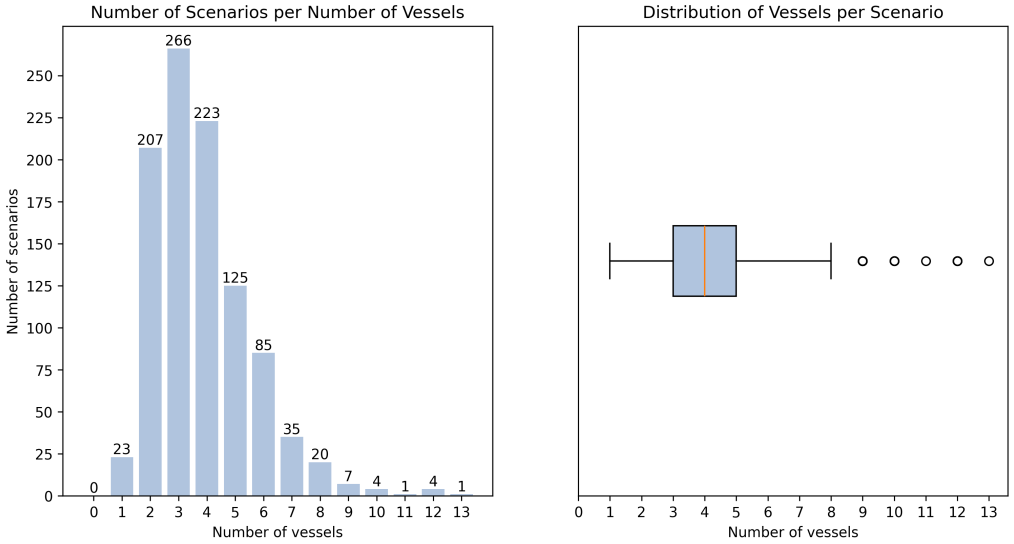


Figure 3.3: Distribution of number of vessels per scenario

Another important aspect of the analysis is the distribution of vessel types. This reveals that most scenarios contain vessels of the categories *Cargo*, *Passenger* and *Tanker*. Their number per scenario is also rather large in comparison to all other vessel types, as shown in Figure 3.4. Among the other vessel types the *Pilot* vessels are most notably. Their share is relatively high, however they often act by making planned contact with another vessel, as seen in Figure 3.5. This needs to be considered when choosing the AIS scenarios to use for training via IRL algorithms.

Additionally, it is of interest if the scenarios contain a sufficient number of vessel-to-vessel encounter situations. For this purpose, an encounter is defined as a situation where two vessels are within 2.5 km of each other. Figure 3.6 shows that a significant number of

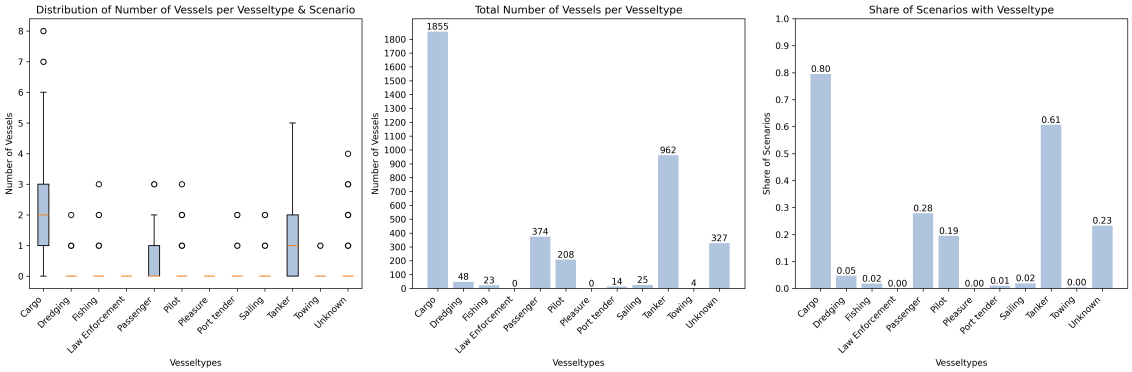


Figure 3.4: Statistics w.r.t. the vessel type in all AIS-scenarios

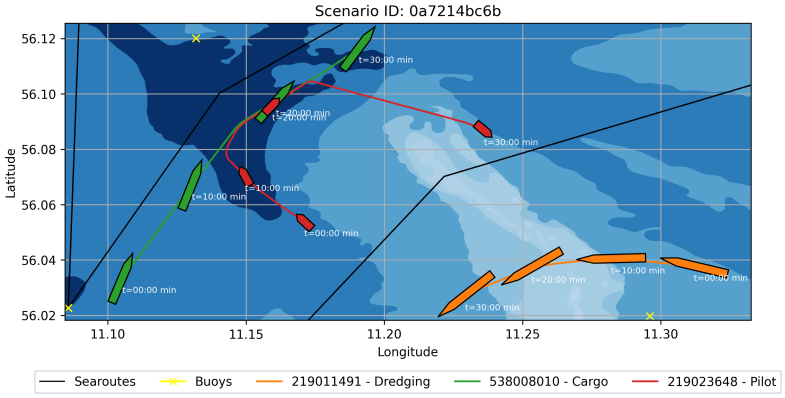


Figure 3.5: Scenario 0a7214bc6b - A *Pilot* vessel performs a planned encounter with very close proximity to the *Cargo* vessel

scenarios contains encounters according to this definition. A differentiation into *head-on*, *cross* and *overtake* is also possible via the relative heading angle between two vessels. This reveals that all three encounter types are almost equally represented in the data.

The navigation status of every vessel should also be analysed. This is necessary to assess if a significant number of vessels is showing active waypoint following and collision avoiding behaviour. Figure 3.7 shows that most vessels are operating using the *UNDER_WAY_USING_ENGINE* status. This status is used to indicate that the vessel is in normal operation and actively steered by the vessels captain. All other stati are significantly less common.

Last, some scenarios reveal vessels acting inside ports. A total of 22 % of all scenarios includes at least one vessel acting inside a port (see Figure 3.8). This is the case if a vessel is within a distance of 1 km to a port location as provided by [98]. This is of interest as the behaviours inside ports are more dynamic and feature closer encounters with shore and other vessels compared to open waters.

3.3 Preprocessing of AIS-Scenarios

Based on the gained insights it is now crucial to extract the most-valuable scenarios for the IRL setting. This includes selecting the scenarios that should be used for training, validation and testing, as well as the valid vessel trajectories that can be used as expert demonstrations.

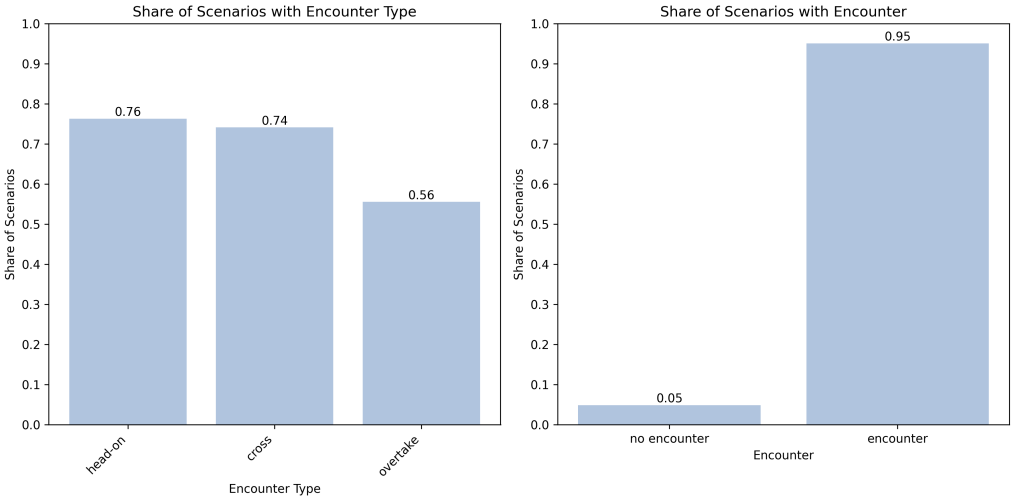


Figure 3.6: Share of scenarios with encounters.

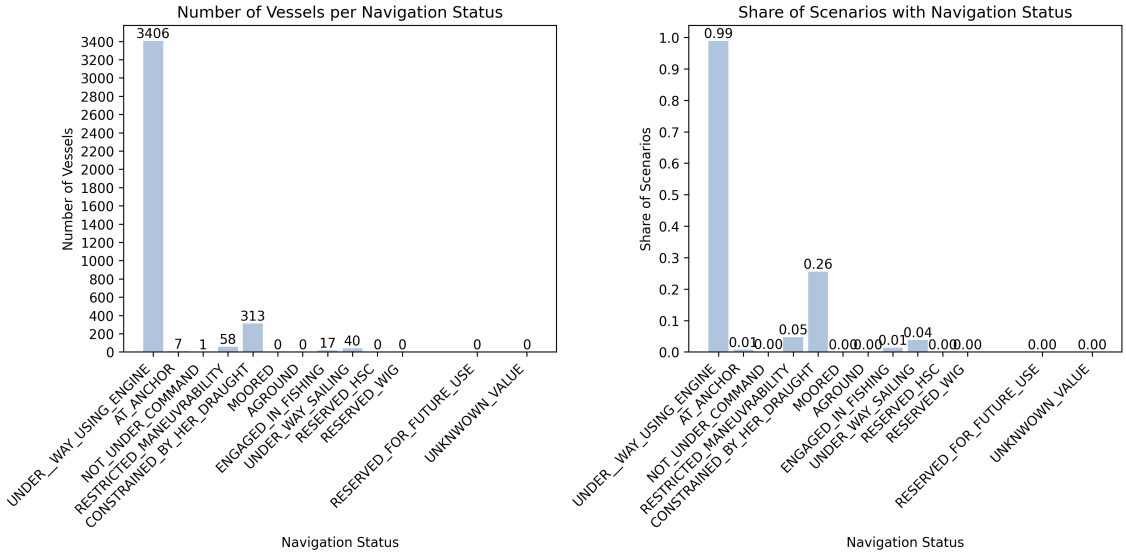


Figure 3.7: Navigation stati of vessels across all scenarios

3.3.1 Selection of valid vessel trajectories

First, the criteria for valid vessel trajectories shall be defined. A valid vessel is defined as a vessel whose trajectory can be used as expert behaviour to learn a policy and reward function in the IRL setting.

The previous analysis gives rise to the following criteria:

- A valid vessel shall act in a scenario that has a duration of 30 min and its own trajectory shall also span the whole 30 min. This ensures the consistency of the dataset.
- A valid vessel shall be of type *Cargo*, *Passenger* or *Tanker*, as those are the most represented in the AIS-scenarios.
- A valid vessel shall have the navigation status *UNDER_WAY_USING_ENGINE*, as this indicates that the vessel is controlled by an expert captain.
- A valid vessel shall not act inside a port, as the behaviour demonstrated inside a port shall not be learned as part of this report. The focus shall be on open-water behaviour.

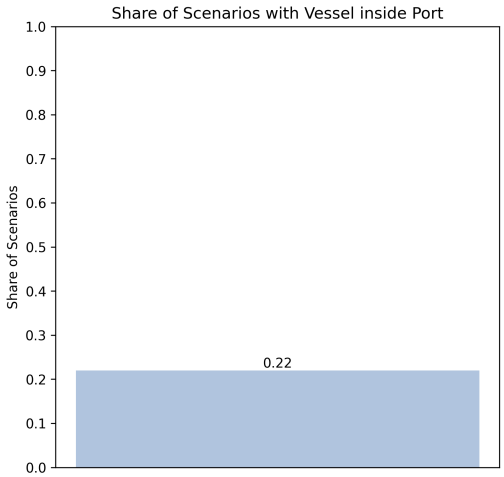


Figure 3.8: Share of scenarios that include at least one vessel acting inside or near a port

- A valid vessel shall experience at least one encounter situation during its trajectory. As it is the goal to learn collision avoiding behaviour, this requirement increases the representation of meaningful expert behaviour in the dataset.

3.3.2 Selection of valid scenarios

The selection of valid vessel trajectories also gives rise to a selection of scenarios. The selection is structured as follows:

- A scenario that does not have a duration of 30 min shall be removed from the selection.
- A scenario that does not contain at least two vessels shall be filtered out from the selection. This step is necessary, because collision avoiding behaviour shall be learned from the scenario.
- A scenario that does not contain at least one active vessel shall be filtered out from the selection, since it does not provide any trajectory that can be learned from.

Applying these selection filters reduces the number of scenarios from 1001 to 801. This number of scenarios is still sufficient and should allow to learn meaningful behaviours from the data.

Another consideration is the large number of *Pilot* vessels among all scenarios. These are of interest, as *Pilot* vessels perform planned encounters with other vessels, which is not ideal in a setting where collision avoiding behaviour shall be learned. However, as their share across all scenarios is relatively big, removing those scenarios would limit the available data too much. Therefore, the learned policies and reward functions shall be learned in a way that allows for a differentiation between different types of TVs.

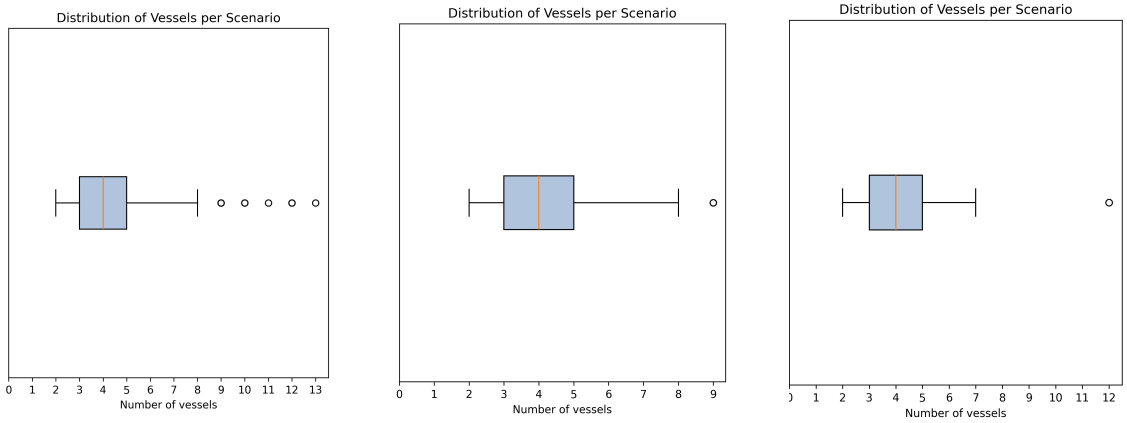
3.3.3 Split of scenarios

Based on the selection of scenarios it is necessary to split these scenarios into a training, validation and test set. A split of 80 % training scenarios, 15 % validation scenarios and 5 % test scenarios has been chosen. This leads to 640 training scenarios, 120 validation scenarios and 41 test scenarios.

For the validation and test scenarios it has also been made sure that easily interpretable scenarios are included. Those scenarios shall function as a qualitative verification method

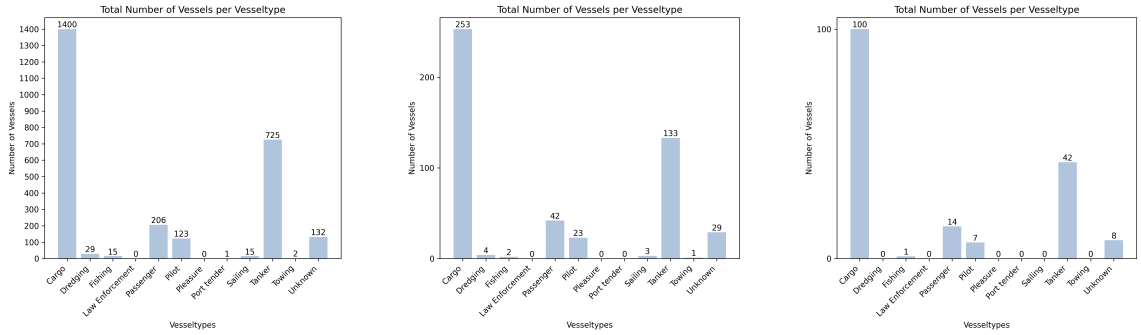
in the results parts of this thesis and allow for an intuitive judgement on the waypoint-following and collision avoiding capabilities of the learned policies and reward functions.

The applied splits of the data have no significant effect on the key features. As visualised in Figures 3.9, 3.10 and 3.11 no significant changes to the distributions of number of vessels per scenario, number of vessels per vessel type and the share of scenarios with a specific encounter type have been introduced by the splits.



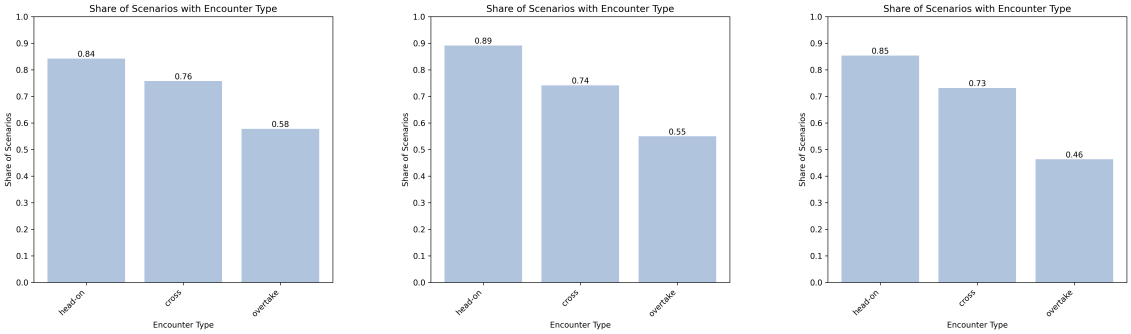
(a) Training dataset (b) Validation dataset (c) Test dataset

Figure 3.9: Distributions of number of vessels per scenarios for all three datasubsets



(a) Training dataset (b) Validation dataset (c) Test dataset

Figure 3.10: Distributions of number of vessels per vesseltype for all three datasubsets



(a) Training dataset (b) Validation dataset (c) Test dataset

Figure 3.11: Distributions of encounter types for all three datasubsets

4 Definition of an AIS-based MDP

As already outlined in Chapter 2 the IRL framework builds on top of a MDP. Therefore, a MDP based on the available AIS-scenarios has to be defined. As the policy and rewards of the MDP depend on the specific IRL-method, only three components of the MDP need to be defined at this stage. Those are the environment, the observation and the actions.

It assumed that the states are only partially-observable requiring the definition of observations based on the available AIS-data. Additionally, it is assumed that the environment dynamics are unknown as the movements of the TVs do not follow a known model. The task of the environment is to model the OVs movement.

4.1 Environment

The IRL framework, as outlined in Chapter 2, often requires an environment which an RL agent can use to learn a behaviour policy in using the simultaneously learned reward function. This environment has to be built on top of the available data of the AIS-scenarios to emulate the scenario the OV acts in.

To achieve this the environment has to comply with the following two requirements:

- The environment shall use a simulator of the OVs movement to allow an agent to act inside a scenario. This shall be possible for training as well as evaluation purposes.
- The environment shall be able to replay the motion of all TVs inside a AIS-scenario. Additionally, measurements to seachart instances shall be possible.

This leads to the development of a kinematic simulator, that allows for the simulation of the OVs movement inside a AIS-scenario and allows to replay the remaining parts of the AIS-scenario.

4.1.1 Kinematic Simulator

A simple 2-DoF kinematic simulator is used and allows to simulate the surge and yaw motion of a vessel. Surge motion corresponds to SOG and yaw motion to COG in the AIS data. A third DoF cannot be modelled using the available AIS scenarios, as no sway information is available.

According to [99] the kinematic model in the north east frame is defined by:

$$\dot{\eta} = \begin{bmatrix} cos(\psi) & 0 \\ sin(\psi) & 0 \\ 0 & 1 \end{bmatrix} \cdot \nu \tag{4.1}$$

where $\eta = \begin{bmatrix} N & E & \psi \end{bmatrix}^T$ is the position & course of the vessel and $\nu = \begin{bmatrix} u & r \end{bmatrix}^T$ is the surge velocity and yaw rate.

A vessel can be manoeuvred inside the simulation environment using two approaches.

The first approach is based on manoeuvring the vessel by directly applying $\dot{u}$ and r to the kinematic model. This approach is open-loop as it is not feeding back information about the current velocity and course.

Another approach issues a desired surge u_{des} and a desired course ψ_{des} in a closed-loop fashion to two separate PID-controllers as described by Equations 4.2 and 4.3. The errors are defined by $e_u = u_{des} - u$ and $e_\psi = \psi_{des} - \psi$.

$$\dot{u}(t) = K_{p,u} \cdot e_u(t) + K_{i,u} \cdot \int e_u(\tau)d\tau + K_{d,u} \cdot \dot{e_u}(t) \tag{4.2}$$

$$r(t) = \dot{\psi}(t) = K_{p,\psi} \cdot e_\psi(t) + K_{i,\psi} \cdot \int e_\psi(\tau)d\tau + K_{d,\psi} \cdot \dot{e_\psi}(t) \tag{4.3}$$

In order to act using the kinematic model of Equation 4.1 the surge velocity needs to be computed from the acceleration $\dot{u}$ obtained via Equation 4.2:

$$u(t) = \int \dot{u}(\tau)d\tau \tag{4.4}$$

To align the kinematic model with the actual AIS-data the acceleration $\dot{u}$ and yaw rate r are limited in their absolute magnitude. The limits have been extracted from the AIS-scenarios by computing the change in SOG and COG between two AIS-states. The results are displayed in Figure 4.1.

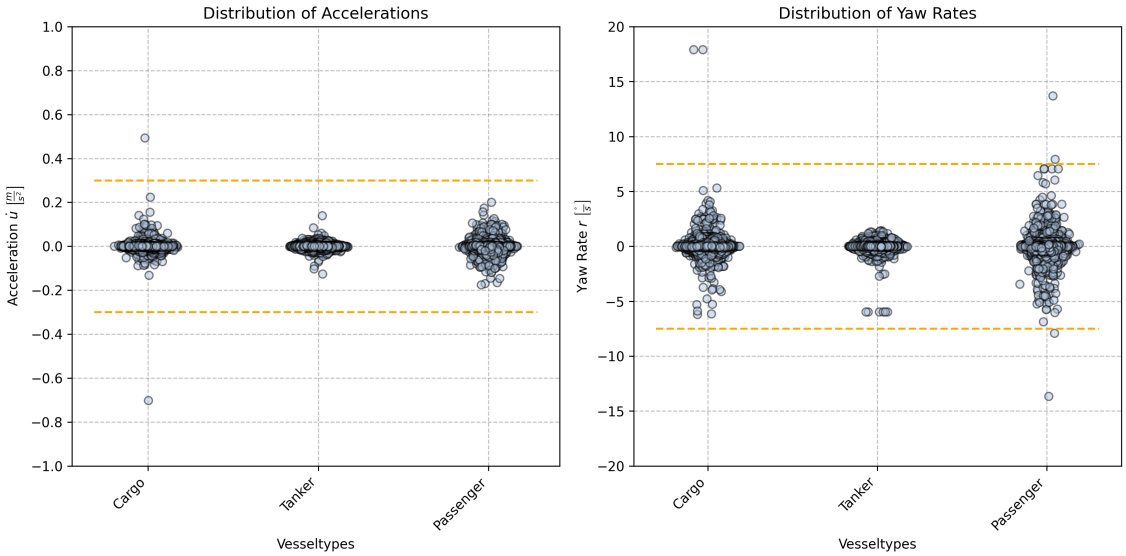


Figure 4.1: Distribution of acceleration $\dot{u}$ and yaw rate r among all AIS-scenarios for the vessel types *Cargo*, *Passenger* and *Tanker*

The majority of absolute accelerations $\dot{u}$ is smaller than $0.3\,\mathrm{m\,s^{-2}}$ while the majority of absolute yaw rates r is smaller than $7.5\,°\,\mathrm{s^{-1}}$. Therefore, the acceleration and yaw rate are limited according to the following clipping objectives:

$$\dot{u} = \text{clip}\left(\dot{u}, -0.3\frac{m}{s^2}, 0.3\frac{m}{s^2}\right) \tag{4.5}$$

$$r = \text{clip}\left(r, -7.5\frac{°}{s}, 7.5\frac{°}{s}\right) \tag{4.6}$$

The sampling time of this kinematic simulator has been chosen to be 1 s to allow for relatively fast motion control. However, the policy only makes decisions every 10 s to be aligned with the AIS-dataformat. This leads to the actions only being updated every 10 s. During the 10 s the action issued by the policy remains the same.

4.1.2 AIS-scenario replay

To represent the full information of the AIS-scenarios the trajectories of the TVs inside a scenario need to be replayed. Additional metadata, like the seachart, also needs to be represented. This happens by simply replaying the trajectories of the TVs with the 10 s sampling-rate of the AIS-data. The metadata is present statically.

The TV trajectories & metadata as well as the additional seachart and scenario-metadata form a complete state of the environment. This is the foundation for the observation, which is an abstract representation of this AIS-based state.

4.2 Observation

The observation is issued to the policy every 10 s and represents the OVs perception of its surroundings. This information is used by the policy to make decisions w.r.t. the future movements of the OV. In order to fulfil the GO-Navigation and EA-Navigation tasks the observation needs to fulfil some specific requirements:

- The observation shall be absolute-position agnostic. This means that all features included in the observation shall be relative to the OV. Learning universal policies and reward functions is only possible by utilising this approach.
- The observation shall be modular in the sense that it shall allow for both navigation tasks to be solved using it. For GO-Navigation this may be achieved by only using a subset of the full observation definition.
- The observation shall always include the distance and bearing to the goal-position. As it is the goal to achieve waypoint-following in both navigational tasks, the observation shall include the relative position to the navigational goal in the respective scenario. The goal-position shall always be the last AIS-state of an expert trajectory, since no other waypoint information is available.

Based on these requirements two different approaches to the observation have been developed. One approach uses a visual representation for the observation, which is more human-understandable. The other approach describes the OVs surroundings by multiple graph structures. Both approaches will be assessed w.r.t. suitability for IRL-algorithms in terms of computational and memory size limitations.

4.2.1 Visual-based observation

The first approach to an observation description leverages visual features. The motivation for this approach is, that such features would be usable by a human operator to navigate a vessel in a maritime scenario. Additionally, the authors of [30] have used a similar approach for their RL system. The observation consists of the following components:

- I_{BEV} **- Birds-Eye-View (BEV) Image:** As shown in Figure 4.2 a BEV image can be calculated from the AIS-states. It captures an area of 2000 m by 2000 m around the OV, while being centred at the current position of the OV and rotated according to the OVs course. Different features are included in the BEV image. Firstly, each TV is included at its relative distance, bearing and heading to the OV. The physical dimension (length & width) of each vessel is visualised. The vessel type is encoded via the colour of the vessel, while the navigation status is indicated via a colour

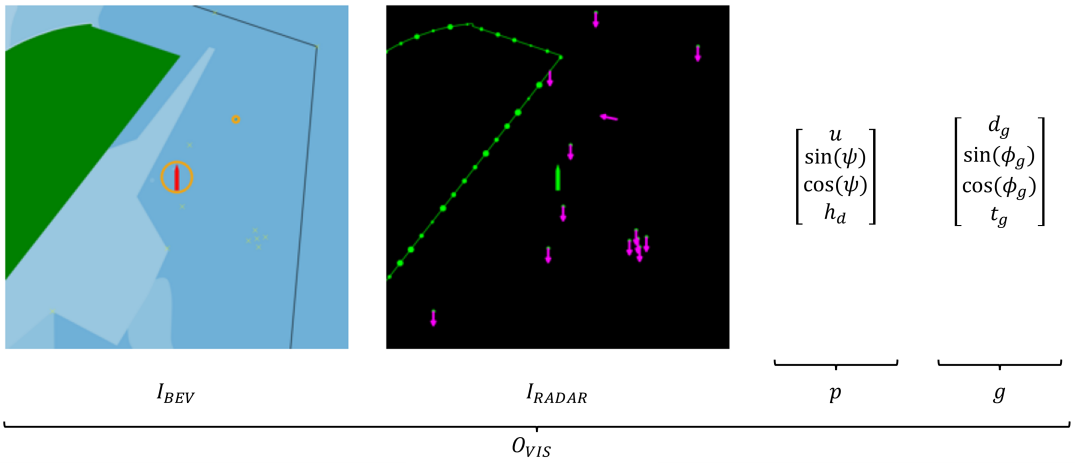


Figure 4.2: Visual-based observation

encoded circle around a vessel. The seachart is used to provide further information w.r.t. the environment, as land, water-depths, searoutes and buoys are included to provide further navigational information.
With these features the BEV image provides information about the relative position and orientation of all navigation relevant entities in the OVs near surroundings.

- I_{RADAR} **- Radar Image:** Another visual observation that can be extracted from the AIS-data is a radar-like image (see Figure 4.2). This image maps the same 2000 m by 2000 m area around the OV. It also depicts the shorelines, TVs and buoys, as uniformly coloured areas inside the image. However, it also provides information that is not present in the BEV image. This is the relative velocity of other objects, as would be obtained by the doppler-velocity estimation of a radar. The relative velocities and relative headings are visualised by magenta coloured arrows attached to the objects. The length of the arrow encodes the velocity, while its orientation the relative heading.
With this the radar image supplements the BEV image by providing additional information w.r.t. the relative movement of objects in the OVs surroundings.

- p **- Proprioceptive Features:** In addition to features of the OVs surroundings, features about its own motion state shall also be included. This gives rise to the definition of the proprioceptive features:

$$p = \begin{bmatrix} u & \sin(\psi) & \cos(\psi) & h_d \end{bmatrix} \tag{4.7}$$

where u is the OVs surge velocity, ψ is the OVs course and h_d is the draught expressed by the AIS-data. The course ψ is decomposed into the sine and cosine components.
By providing these features as part of the observation, the policy may act according to the OVs current dynamic situation. Features like the vessel length and width are not encoded as part of the proprioceptive features, as they are implicitly included in I_{BEV} and I_{RADAR}.

- g **- Goal Location:** To be able to do waypoint-following, the policy requires information about the goal-position. It is defined by:

$$g = \begin{bmatrix} d_g & \sin(\phi_g) & \cos(\phi_g) & t_g \end{bmatrix} \tag{4.8}$$

where d_g is the distance and ϕ_g is the bearing to the goal-position which is also decomposed into the sine and cosine components. t_g is the time until the goal-position

shall be reached.
Conditioning a policy on these features should be sufficient to learn waypoint-following behaviour in both navigational tasks.

All four parts of the visual observation definition are calculated at every timestep and can be summarised via the following expression:

$$O_{VIS} = \begin{bmatrix} I_{BEV} & I_{RADAR} & p & g \end{bmatrix} \tag{4.9}$$

4.2.2 Graph-based observation

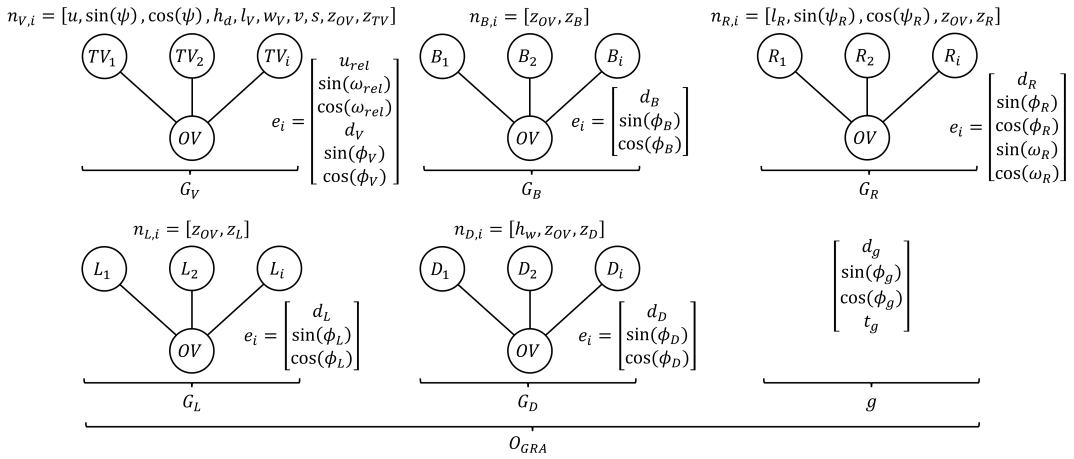


Figure 4.3: Graph-based observation

The second observation definition leverages an approach, which uses direct measurements to all objects in the OVs surroundings and arranges those as graphs. This representation is less interpretable for a human, but it offers more precise and descriptive features. The observation definition is visible in Figure 4.3 and consists of the following components:

- G_V **- Vessel Graph:** The vessel graph depicts the relation of the OV to the TVs. Therefore, each vessel inside a scenario (OV & TV) is represented by a node inside the graph. Each node is then characterised by the following node feature vector:

$$n_{V,i} = \begin{bmatrix} u & \sin(\psi) & \cos(\psi) & h_d & l & w & v & s & z_{OV} & z_{TV} \end{bmatrix} \tag{4.10}$$

  where u is the surge velocity and ψ is the course of the respective vessel. The course ψ is decomposed into the sine and cosine component. Geometrical information about the vessel is provided via the draught h_d, the length l and the width w. v describes the vessel type, while s expresses the navigation status of the vessel. Both are just encoded as scalar values rather than a one-hot encoded representation. Additionally, z_{OV} and z_{TV} are binary variables indicating whether the node is corresponding to the OV or a TV. As the OV is also included as a node, no dedicated proprioceptive features are necessary.
  Only the node of the OV is connected to nodes of TVs by edges, which represent relative measurements between both vessels. Each edge is undirected and defined by the following feature vector:

$$e_{V,ij} = \begin{bmatrix} u_{rel} & \sin(\omega_{rel}) & \cos(\omega_{rel}) & d_V & \sin(\phi_V) & \cos(\phi_V) \end{bmatrix} \tag{4.11}$$

  where u_{rel} and ω_{rel} are the relative velocity and heading between both vessels. Additionally, the distance d_V and the bearing ϕ_V from the OV to the TV are included.

Both angular features ω_{rel} and ϕ_V are decomposed into their sine and cosine components. In addition to the edges between two nodes, each node is also equipped with a self-loop edge connecting the node with itself, indicating the relative measurements to itself. This is needed for potentially using Graph Neural Networks (GNNs) as policies.
This graph encodes all relevant information that a decision-making policy would need to avoid collisions with other vessels.

- G_B **- Buoy Graph:** The buoy graph depicts the relation of the OV with all buoys inside a scenario. Therefore, all buoys and the OV are represented by nodes in this graph. Since the buoys do not offer relevant physical features the node feature vector will just be two-dimensional and contain the binary variables z_{OV} and z_B to indicate whether the node corresponds to the OV or a buoy.
The OV node is connected to all buoy nodes via undirected edges. The feature vectors characterising the edges are defined by:

$$e_{B,ij} = \begin{bmatrix} d_B & \sin(\phi_B) & \cos(\phi_B) \end{bmatrix} \tag{4.12}$$

where d_B is the distance from the OV to the buoy and ϕ_B is the bearing to the buoy as seen from the OV, which is decomposed into a sine and a cosine component. Again, each node contains a self-loop edge.
This encodes all relevant information to navigate around buoys. Specifying the relative velocity and heading w.r.t. buoys is not necessary as this is implicitly included in the OVs own surge velocity and course.

- G_R **- Searoute Graph:** For navigational purposes the observation should also contain information about nearby searoutes. The searoute graph covers this aspect. As searoutes are represented by multiple line segments it is computationally not feasible to calculate the minimum distance to each searoute. Therefore, the searoutes are represented by their length, orientation and the distance to the start point of the searoute. One searoute segment corresponds to one node in the graph, which is represented by:

$$n_{R,i} = \begin{bmatrix} l_R & \sin(\psi_R) & \cos(\psi_R) & z_{OV} & z_D \end{bmatrix} \tag{4.13}$$

where l_R indicates the length of the searoute segment. ψ_R expresses the global orientation of the searoute decomposed into sine and cosine. Lastly, the two binary flags z_{OV} and z_R indicating correspondence to either the OV or a searoute are included. Undirected edges connect the OV-node to all searoute-nodes to indicate relative measurements to the searoute. The edge feature vector is defined by:

$$e_{R,ij} = \begin{bmatrix} d_R & \sin(\phi_R) & \cos(\phi_B) & \sin(\omega_R) & \cos(\omega_R) \end{bmatrix} \tag{4.14}$$

where d_R is the distance to the start point of the searoute segment and ϕ_R is the bearing to this searoute point from the OVs point of view. The bearing ϕ_R is decomposed into a sine and a cosine component. Additionally, the relative orientation ω_R between the OVs course and the searoutes global orientation is included in decomposed manner. Again, self-loop edges are included for every node.
This component of the graph-based observation provides the necessary relative information about the searoutes in the scenario in a computationally efficient manner.

- G_L **- Shore Graph:** Information about shore shall also be represented as a graph. However, computing minimal distances or distances at certain bearing angles suffers from the same computational problems as elaborated for the searoute graph. Therefore, m_L points are uniformly sampled from each shore boundary. These

points are together with the OV the nodes of the graph. Again, the node feature vector has 2-dimensions which indicate the correspondence to either the OV via z_{OV} or the shore via z_L.
Undirected edges connect the OV-node with all shore nodes and their feature vectors are defined by:

$$e_{L,ij} = \begin{bmatrix} d_L & \sin(\phi_L) & \cos(\phi_L) \end{bmatrix} \tag{4.15}$$

where d_L is the distance to the shore point and ϕ_L is the bearing to the shore point from the OVs perspective, which is decomposed into sine and cosine. A self-loop edge is included for every node.
This approach reduces the accuracy of the shore due to the sampling. However, this is needed to achieve a computationally efficient representation that provides enough information about the shore.

- G_D **- Water Depth Graph:** The seachart provides comprehensive information about the water depth in a geographical area. In order to assess this information as part of the graph-based observation definition, it has to be translated into a graph. To accomplish this, a number of points proportional to the size of a depth polygon are sampled for each of those polygons contained in the seachart. The number of points per squarekilometer is variable according to the parameter m_D. This ensures all regions of the seafloor are properly represented. The OV and each of the seabed points is represented by a node in the graph characterised by the following node feature vectors:

$$n_{D,i} = \begin{bmatrix} h_w & z_{OV} & z_D \end{bmatrix} \tag{4.16}$$

where h_w is the water depth at the specific point in the seachart. The other two features are again binary features indicating whether the current node corresponds to the OV or a point on the seabed.
The undirected edges of the graph connect the OV node with all other nodes. One edge is characterised by the following feature vector:

$$e_{D,ij} = \begin{bmatrix} d_D & \sin(\phi_D) & \cos(\phi_D) \end{bmatrix} \tag{4.17}$$

where d_D is the distance and ϕ_D is the bearing to the point of interest from the OV perspective. The bearing ϕ_D is again decomposed into sine and cosine features. This graph also contains self-loop edges for every node.
This graph provides a sparse representation of the seabed inside a graph structure. This makes it computationally much more efficient, but still expressive enough to identify shallow waters.

- g **- Goal Location:** This observation approach also requires the information about the goal-position. It follows the same definition as already provided in Equation 4.8.

All these components provide a full graph-based characterisation of the OVs surroundings that can be used to learn policies and reward-functions. At every timestep the graph-based observation is defined by:

$$O_{GRA} = \begin{bmatrix} G_V & G_B & G_R & G_L & G_D & g \end{bmatrix} \tag{4.18}$$

4.2.3 Evaluation

A clear evaluation of the suitability of one or the other observation definition can only be done once policies and reward functions have been trained. However, within the scope of this thesis an assessment w.r.t. computational complexity and memory size shall be

done prior to training to assess which definition is beneficial for usage in combination with IRL-algorithms.

Computational Complexity

As the IRL problem requires an agent to act inside an environment it is crucial for a fast and seamless training process to generate the observations in a reasonable time. To assess both observation definitions in this regard the observations of all valid vessels in 10 valid scenarios are generated and their computational complexity is assessed.

In order to do so I_{BEV} and I_{RADAR} from the visual-based observation O_{VIS} have a resolution of 256×256 pixels. The number of sampled points m_L and m_D from the graph-based observation O_{GRA} are set to 5 points per shore and 0.1 points per squarekilometer respectively. These parameters have been determined by a qualitative assessment of the resulting observations.

The following mean and standard deviation (Std) of the computation times for each observation has been obtained:

	O_{VIS}		O_{GRA}	
	Mean	Std	Mean	Std
Computation Time (s)	0.40	0.10	0.10	0.06

Table 4.1: Computation times for O_{VIS} and O_{GRA}

This reveals that the graph-based observation O_{GRA} can be computed faster than the visual-based observation O_{VIS}. In practice it is also not feasible to calculate the visual-based observation O_{VIS} in large scenarios, as additionally to the computation of one observation instance the whole seachart would have to be converted into the North-East-Down (NED)-frame at least once per scenario. This is a computationally very demanding operation. This favours the use of the graph-based observation O_{GRA}.

Memory Size

The memory size of one observation instance is also important as it is required to sample and store expert observations as part of an IRL algorithm. Reduced memory size may speed up the loading and saving of observations from and to memory. Therefore, the total number of features is analysed.

For the visual-based observation O_{VIS} both images are again defined with a resolution of 256×256 pixels. The number of sampled points m_L and m_D from the graph-based observation O_{GRA} are set to set to 5 points per shore and 0.1 points per squarekilometer.

The following numbers of features per observation definition have been obtained for all valid vessels in 10 valid scenarios:

	O_{VIS}		O_{GRA}	
	Mean	Std	Mean	Std
Number of Features	393224.00	0.00	16705.39	10928.07

Table 4.2: Number of features for O_{VIS} and O_{GRA}

For the visual-based observation O_{VIS} the number of features is always constant, as the RGB-images I_{BEV} and I_{RADAR} maintain the same resolution. The number of features of the graph-based observation O_{GRA} varies with the number of TVs, buoys and sampled

seachart points. In comparison the graph-based observation O_{GRA} is favourable since it uses considerably less features.

Conclusion

Overall, the graph-based observation O_{GRA} is favourable in comparison to the visual-based observation O_{VIS}. This is due to its lower computational complexity as well as its smaller memory footprint. Additionally, the information density is considerably higher in the graph-based observation O_{GRA} as the visual approach contains a lot of information-less pixels. Table 4.3 summarises these aspects. The evaluation symbols are defined as follows: + + is very good performance, + is good performance, o is moderate performance, - is poor performance - - is very poor performance. This also applies to all future uses of such evaluation tables.

	O_{VIS}	O_{GRA}
Computational Complexity	-	+
Memory Size	-	+
Information Density	-	+

Table 4.3: Evaluation of O_{VIS} and O_{GRA}

Considering this, the graph-based observation shall be used for the remainder of this thesis.

4.3 Action

The actions are computed by the policy and applied to the environment which utilises these to compute the next AIS-state. On one hand the actions are used by the policy to act inside the environment. On the other hand, they are also crucial in the LfD setting. As the AIS-scenarios do not include the real actions the expert vessels have taken, they have to be inferred from the AIS-trajectories to do LfD. This leads to the following requirements for the definition of the actions:

- The action definition shall allow to use the kinematic simulator to maneuver the OV as was defined in section 4.1.1.
- The action definition shall allow for inferring expert actions from AIS-trajectories. This is needed to be able to apply LfD-algorithms to the problem setting.

In total three different approaches have been developed. The first uses a learned inverse model that extracts actions from state transitions. The second approach redefines actions to be future waypoints of the trajectory. The last approach utilises a Kalman-filter to infer the actions using a kinematic model. All three approaches will be evaluated w.r.t. their accuracy and suitability for the IRL-problem.

4.3.1 Velocity & Course using an Inverse Model

One approach to define actions is to issue desired control variables to a lower-level controller, which takes care of the motion control. Exactly this approach shall be followed for the first action definition. This approach defines the actions as the desired surge velocity u_{des} and desired course ψ_{des}:

$$A_{vc} = \begin{bmatrix} u_{des} & \psi_{des} \end{bmatrix}^T \tag{4.19}$$

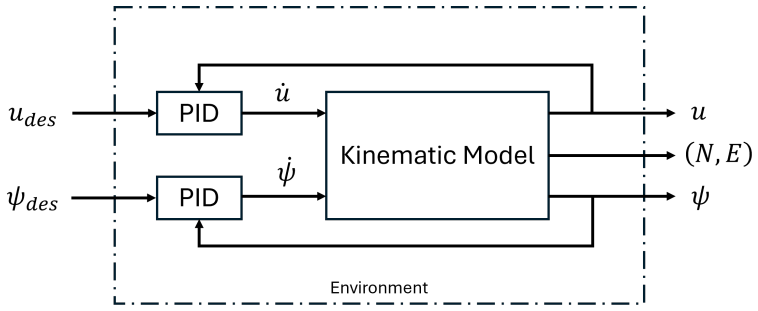


Figure 4.4: A_{vc} - Action definition using desired speed u_{des} and desired course ψ_{des}

Lower-level PID-controllers, as already outlined in section 4.1.1, interact with the kinematic model using A_{vc} to maneuver the OV. Figure 4.4 visualises this action definition.

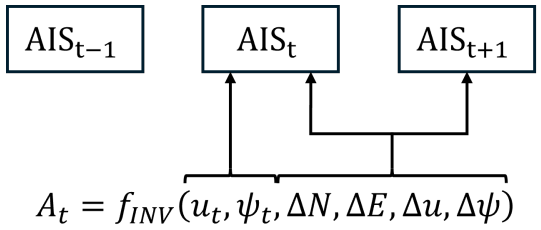


Figure 4.5: Inference of desired speed u_{des} and desired course ψ_{des} from AIS-states

Unfortunately, both the desired surge velocity u_{des} and desired course ψ_{des} cannot be inferred directly from the AIS-based expert trajectories. An inverse model needs to be defined which takes the relative change between the current and the next AIS-state as input and predicts the action A_{vc} that was necessary to create this relative AIS-state change. The inverse model is defined by:

$$\hat{A}_{vc} = f_{INV}\left(\Delta N, \Delta E, \Delta u, \Delta \psi, u, \psi\right) \tag{4.20}$$

where $\Delta N, \Delta E, \Delta u, \Delta \psi$ are the changes in AIS-state from the current to the next state and u and ψ are the current surge velocity and course as visualised in Figure 4.5.

The inverse model f_{INV} is modelled as a Feed Forward Neural Network (FFNN) and trained using the MSE objective. The training data can be generated by first uniformly sampling actions A_{vc} and initial surge velocities u and courses ψ. In a second step the actions are then applied to the PID-controllers and the kinematic simulator to record the relative state changes. This procedure generates the necessary training data.

A full characterisation and evaluation of all the parameters like the PID-coefficients or the network architecture of f_{INV} will be conducted in Chapter 4.3.4.

4.3.2 Waypoints using full trajectories

Another approach to defining the actions is formulating the problem as waypoint following task. An action consists of j waypoints the OV shall reach in the future. Each waypoint is defined as the following vector:

$$WP_j = \begin{bmatrix} t_j & d_j & \phi_j \end{bmatrix} \tag{4.21}$$

where t_j indicates the time until this waypoint shall be reached, d_j expresses the distance and ϕ_j expresses the bearing to the j-th waypoint.

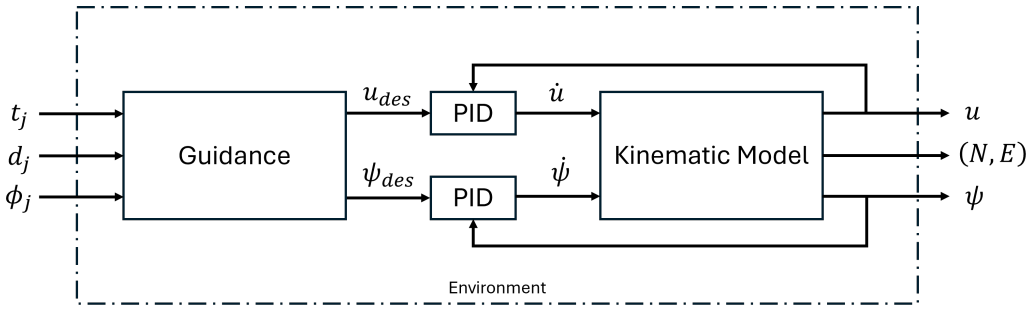


Figure 4.6: A_{wp} - Action definition using future waypoints

Using this waypoint definition the action is defined by:

$$A_{wp} = \begin{bmatrix} WP_0 & \dots & WP_j \end{bmatrix}^T \tag{4.22}$$

Based on these waypoints the already developed PID-controllers can be used for waypoint following. Using the distance d_j and the time t_j to a waypoint the desired surge velocity u_{des} can be computed. The bearing ϕ_j allows for the computation of the desired course ψ_{des}. Both estimates are updated with the sampling rate used by the motion control. This also requires updating the waypoints by reducing t_j and recalculating d_j and ϕ_j. The waypoint of interest is always the one closest in time. Switching to the next waypoint occurs whenever the OV is within d_{accept} of a waypoint or the updated estimate of t_j reaches 0. The whole setup is visualised in Figure 4.6.

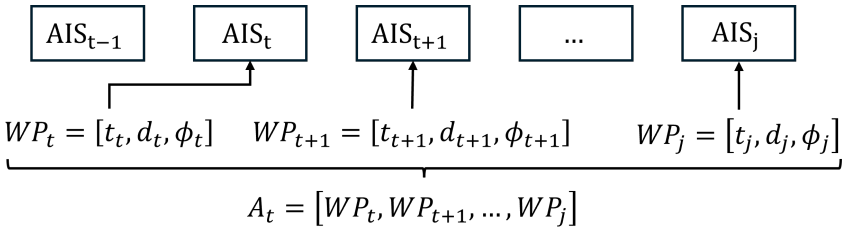


Figure 4.7: Inference of future waypoints from AIS-states

Inferring the waypoint action from AIS-based expert trajectories is straightforward, as j future AIS-states can be sampled and used to calculate the respective time t_j, distance d_j and bearing ϕ_j. The sampling rate for these waypoints should match the sampling rate of the AIS-scenario. When a AIS-trajectory ends the remaining waypoints are calculated via linear interpolation. Figure 4.7 visualises this inference method.

A full parametrisation and evaluation of this approach will be outlined in section 4.3.4.

4.3.3 Acceleration & Yaw Rate using a Kalman-Filter

The last approach abandons the use of the intermediary PID-controllers and defines the actions directly as the surge acceleration $\dot{u}$ and the yaw rate r as visualised in Figure 4.8. The action A_{kf} is defined by:

$$A_{kf} = \begin{bmatrix} \dot{u} & r \end{bmatrix}^T \tag{4.23}$$

A Kalman-filter [100] can be used to infer the actions A_{kf} from AIS-based expert trajectories as shown in Figure 4.9. Aligned with the kinematic simulator the motion prediction part of the Kalman-filter is defined by:

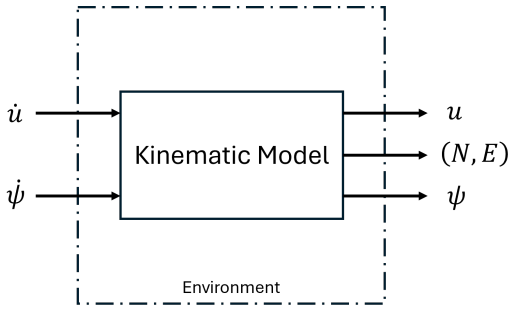


Figure 4.8: A_{kf} - Action definition using direct assignment of acceleration $\dot{u}$ and yaw rate $\dot{\psi}$

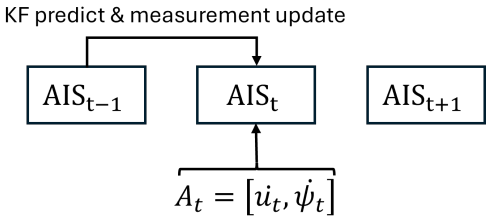


Figure 4.9: Inference of acceleration $\dot{u}$ and yaw rate $\dot{\psi}$ using a Kalman-Filter

$$X = \begin{bmatrix} N & E & u & \psi & \dot{u} & r \end{bmatrix}^T \tag{4.24}$$

$$F = \begin{bmatrix} 1 & 0 & cos(\psi) \cdot t & 0 & \frac{1}{2} \cdot cos(\psi) \cdot t^2 & 0 \\ 0 & 1 & sin(\psi) \cdot t & 0 & \frac{1}{2} \cdot sin(\psi) \cdot t^2 & 0 \\ 0 & 0 & 1 & 0 & t & 0 \\ 0 & 0 & 0 & 1 & 0 & t \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.25}$$

$$P_0 = \text{diag} \begin{bmatrix} \sigma_N^2, \sigma_E^2, \sigma_u^2, \sigma_\psi^2, \sigma_{\dot{u}}^2, \sigma_r^2 \end{bmatrix} \tag{4.26}$$

where X is the state vector that consists of the position in the north-east frame (N & E), the surge velocity u, the course ψ, the surge acceleration $\dot{u}$ and the yaw rate r. F is the state transition matrix based on kinematic equations, where t is the time from the last measurement update. P_0 describes the initial uncertainty covariance matrix, which is a diagonal matrix featuring all measurement uncertainties.
The measurement update part of the Kalman-filter is defined by:

$$Z = \begin{bmatrix} N & E & u & \psi \end{bmatrix}^T \tag{4.27}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{4.28}$$

$$R = \text{diag} \begin{bmatrix} \sigma_N^2, \sigma_E^2, \sigma_u^2, \sigma_\psi^2 \end{bmatrix} \tag{4.29}$$

where Z is the measurement vector, that just consist of the data available in a AIS-state. H is the measurement matrix mapping Z to X, while R describes the uncertainties about the measurements using the variances of the measurement variables.

The actions can be inferred as results of one prediction and measurement step of the Kalman-Filter, which provides estimates of the surge acceleration $\dot{u}$ and yaw-rate r, that can in return be applied to the kinematic model.

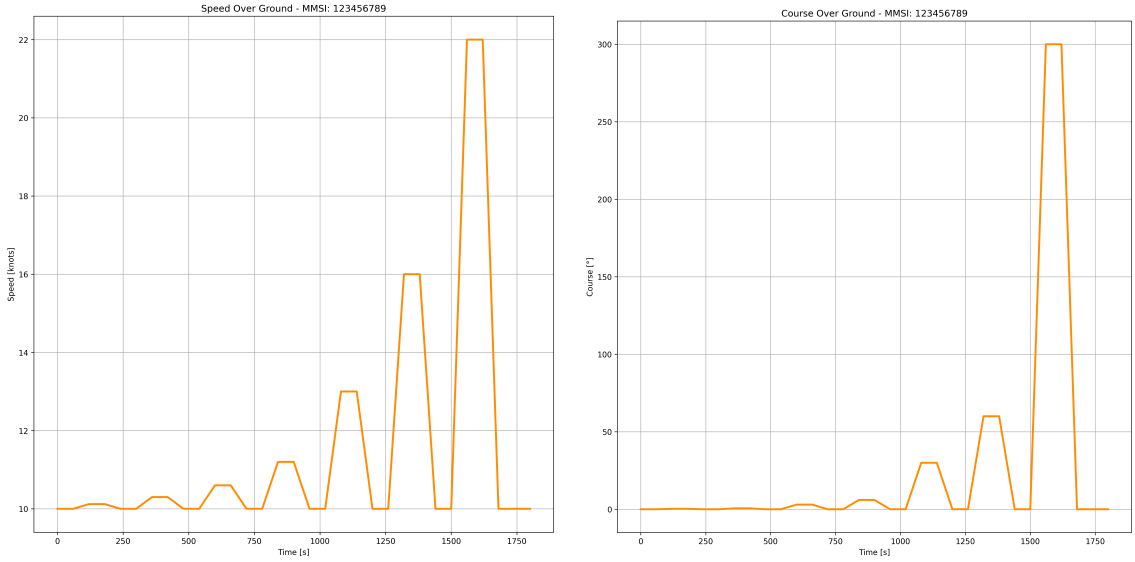
A full parametrisation and evaluation of this approach will be conducted in section 4.3.4.

4.3.4 Evaluation

All three approaches have to be evaluated. Therefore, the parameters of all action definition shall be optimised and their performance shall be assessed w.r.t. their accuracy in inferring actions from AIS-states.

Setup

The evaluation can be done by replaying the inferred action sequence in a scenario and evaluating the average position, surge velocity and course error obtained in the scenario. For this purpose 96 artificial AIS-scenarios from two categories have been defined. One type of scenarios consists of iteratively growing accelerations and decelerations, while the second type of scenarios consists of iteratively growing yaw rates turning to the starboard and port side. This is done for a multitude of initial surge velocities and courses. Figure 4.10 visualises the surge velocity and course of a scenario from each type. This evaluation approach provides an independent evaluation basis for the hyperparameter-optimisation and action type selection.



(a) Acceleration Scenario

(b) Turning Scenario

Figure 4.10: Comparison of artificial acceleration and turning scenarios

As part of the evaluation the hyperparameters of all action definitions shall be tuned. Therefore, a Grid Search (GS) for specific parameters shall be done.

For the velocity-course action A_{vc} defined in 4.3.1 the coefficients of the surge velocity PID-controller ($K_{p,u}$, $K_{i,u}$ & $K_{d,u}$), the course PID-controller ($K_{p,\psi}$, $K_{i,\psi}$ & $K_{d,\psi}$) and the network architecture for f_{INV} are determined via the GS. The number of samples for training f_{INV} is set to 25 000 with a 80 % training, 10 % validation and 10 % test split. This sampling is done for every PID-controller configuration and the best architecture is

chosen using the R^2-measure on the validation set. The learning rate is set to 1×10^{-3} and the model is trained for 25 epochs.

For the waypoint action A_{wp} defined in section 4.3.2 the coefficients of both PID-controllers shall be determined via the GS. Other parameters that are not subject to the GS are the number of waypoints which is set to 3 future AIS-states and the acceptance radius d_{accept} which is set to 0.0 m.

The acceleration-yaw rate action A_{kf} requires less hyperparameters. Only the measurement uncertainties of the Kalman-Filter need to be defined. The uncertainties w.r.t. position in the north σ_N and east dimension σ_E, surge velocity σ_u and course σ_ψ shall be determined via a GS. The uncertainties of the acceleration $\sigma_{\dot{u}}$ and yaw rate σ_r are set to $0.3\,\text{m}\,\text{s}^{-2}$ and $7.5\,°\,\text{s}^{-1}$ based on the analysis provided in Figure 4.1.

Results

Conducting the GS for all three action definitions and obtaining the best action hyperparameters by the minimal mean average position error over all evaluation scenarios leads to the following results:

	A_{vc}		A_{wp}		A_{kf}	
	Mean	Std	Mean	Std	Mean	Std
Avg. Pos. Error (m)	602.58	541.31	530.07	269.26	278.07	1199.92
Avg. Vel. Error ($\frac{m}{s}$)	0.249	0.133	0.047	0.024	0.070	0.065
Avg. Cou. Error (°)	7.34	13.97	20.42	21.29	8.89	18.75

Table 4.4: Evaluation of A_{vc}, A_{wp} and A_{kf}

The best velocity-course action A_{vc} uses a configuration of the PID-controllers, where $K_{p,u} = 0.1$. $K_{i,u} = 0.0$ and $K_{d,u} = 1.0$. The course PID-controller is configured via $K_{p,\psi} = 1.0$, $K_{i,\psi} = 0.0$ and $K_{d,\psi} = 0.1$. The best performing network architecture uses 4 layers with 512, 256, 128 and 64 neurons in descending order. It has been trained to a validation loss of 0.08 and a validation R^2-score of 0.74 over 25 epochs using a mean-squared-error objective and the Adaptive Moment Estimation (Adam) optimiser [101] with a learning rate of 1×10^{-3}.

In case of the waypoint action A_{wp} the best results have been obtained using a parametrisation of the velocity PID-controller of $K_{p,u} = 1.0$, $K_{i,u} = 0.1$ and $K_{d,u} = 0.1$. The course PID-controller is only configured as a P-controller with $K_{p,\psi} = 0.01$.

The best parameters of the acceleration-yaw rate action A_{kf} are $\sigma_N = \sigma_E = 50m$, $\sigma_u = 0.01\frac{m}{s}$ and $\sigma_\psi = 0.01°$, which intuitively sounds realistic.

Comparing the accuracy of all three action definitions one can see that the acceleration-yaw rate action A_{kf} using a Kalman-Filter to infer expert actions outperforms the other two action definitions. In terms of the position accuracy the acceleration-yaw rate action A_{kf} has the lowest mean position error by a factor of approximately two. The standard deviation is quite large and the reasons for that will be elaborated in the last section of this chapter. However, both, the velocity-course and waypoint inferred actions have a lower variance of the error. Looking at the velocity error, both the waypoint A_{wp} and acceleration-yaw rate action A_{kf} definitions feature an almost negligible error. The error of the velocity-course action A_{vc} is considerably higher. For the course error the velocity-course action A_{vc} has the lowest error closely followed by the acceleration-yaw rate action A_{kf}. Only the waypoint action shows a rather large error in its resulting course.

Verifying the results obtained from the artificially generated trajectories with a real AIS-scenario confirms the results. Figure 4.11 shows the deviation in position, surge velocity and course for the same scenario using all three action definitions. The acceleration-yaw rate action A_{kf} shows almost no deviation and outperforms both of the other action definitions. Especially the waypoint action A_{wp} shows large deviations w.r.t. the course of the vessel and the replayed surge velocity is quite noisy. The velocity-course action A_{vc} shows good performance w.r.t. the surge velocity under- and overestimating the velocity just slightly. Similar observations can be made w.r.t. the course which is slightly overestimated.

Conclusion
Considering all aspects, the acceleration-yaw rate action A_{kf} not only has the highest average accuracy w.r.t. the position and course, but it is also less complex with fewer hyperparameters and does not require extensive tuning or training. This conclusion can be observed in Table 4.5.

	A_{vc}	A_{wp}	A_{kf}
Accuracy	o	o	+
Parameters	-	o	+
Tuning	-	o	+

Table 4.5: Comparison of A_{vc}, A_{wp} and A_{kf}

Therefore, the acceleration-yaw rate action A_{kf} definition is the right choice for IRL problems and will be used for the remainder of this thesis.

Errors in the Implementation of the Acceleration-Yaw rate Action
Upon a final review of the implementation of the acceleration-yaw rate action A_{kf} three errors inside the code emerged. As this happened six days before submission of the project report, no correction of the succeeding work could be done. Therefore, the following discussion shall motivate why the obtained results are still valid.

The first mistake in the code was that the measurement uncertainties have been fixed to $\sigma_N = \sigma_E = 0.1m$, $\sigma_u = 0.1\frac{m}{s}$ and $\sigma_\psi = 0.1°$. Thus they were not part of the GS. The second error affected the state transition matrix F, where the course angle ψ was not correctly translated into radians when computing the sine and cosine. Luckily, in the measurement update the small uncertainties lead to a recovery of the error made in the prediction step. Figure 4.12 validates this claim as it compares the resulting course of replaying the action sequences before and after the removal of this error.

The third error is the most severe among the three identified issues. The course angle error has not been rescaled correctly in the case of turning motion over the 0° course angle. This did not result in small negative course errors, but in large positive course errors ranging up to 360°. This then resulted in the prediction of high yaw rates, which are the reason for the large standard deviation of the position error shown in Table 4.4. These large yaw rates only occur in cases where the course changes across 0°. The share of those transitions within all valid transitions is 0.26 % and therefore very small. 14.51 % of the valid trajectories contain at least one such state transition. As all state transitions after such a transition may also be affected by larger deviations of the yaw rate, the share of all compromised states among all valid states is estimated with 8.57 %. Those numbers are significant, however, the learned policies should be able to generalise and average out the unplausible and large yaw rates. Therefore, the obtained results of

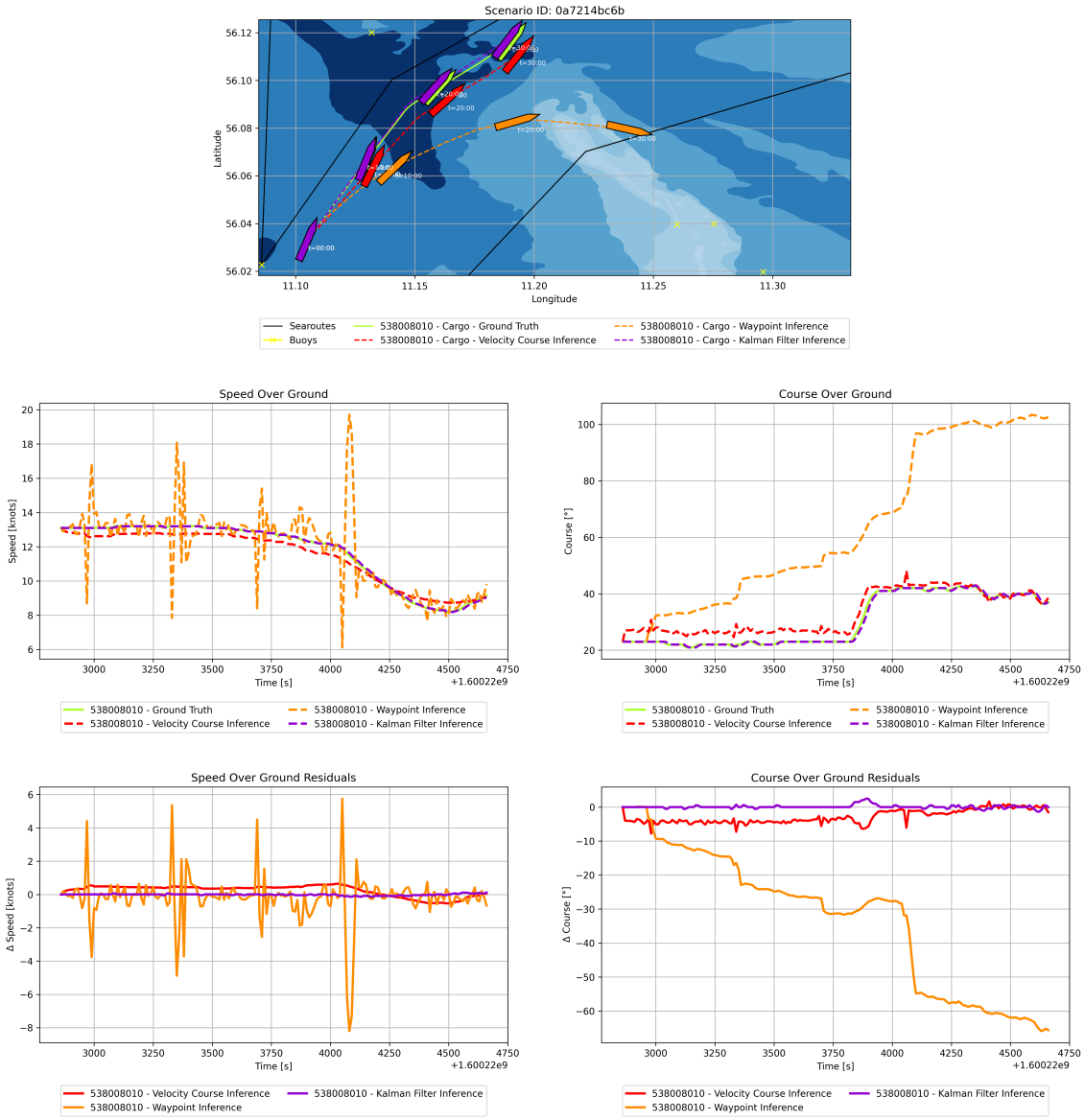
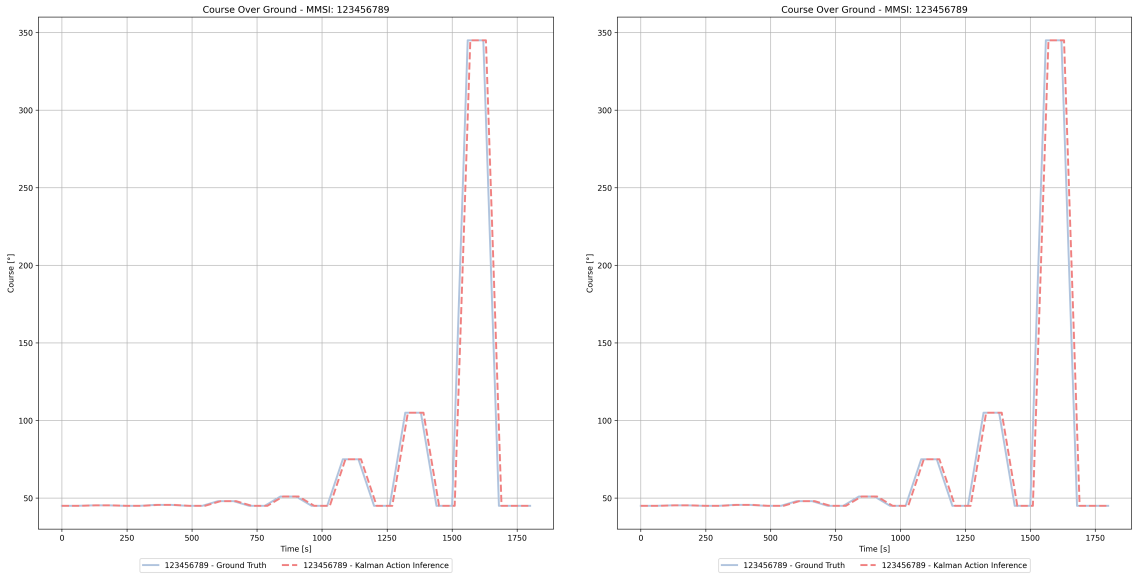


Figure 4.11: Validation of A_{vc}, A_{wp} and A_{kf} via a real AIS-scenario

the IRL-algorithms in the following chapters should still be valid, but caution is needed in the analysis of those results.

The evaluation of the acceleration-yaw rate action has been repeated with the implementation errors removed. The evaluation shows that the following parameters yield the best performance: $\sigma_N = \sigma_E = 10.00m$, $\sigma_u = 0.01\frac{m}{s}$, $\sigma_\psi = 0.01°$. The average position, velocity and course errors are also considerably lower with a mean of 10.58 m, 0.07 m s^{-1}, 2.21° and a standard deviation of 4.66 m, 0.06 m s^{-1}, 2.21° respectively. This proves, that the acceleration-yaw rate action using a Kalman-filter for action inference remains the best option.



(a) COG before error removal

(b) COG after error removal

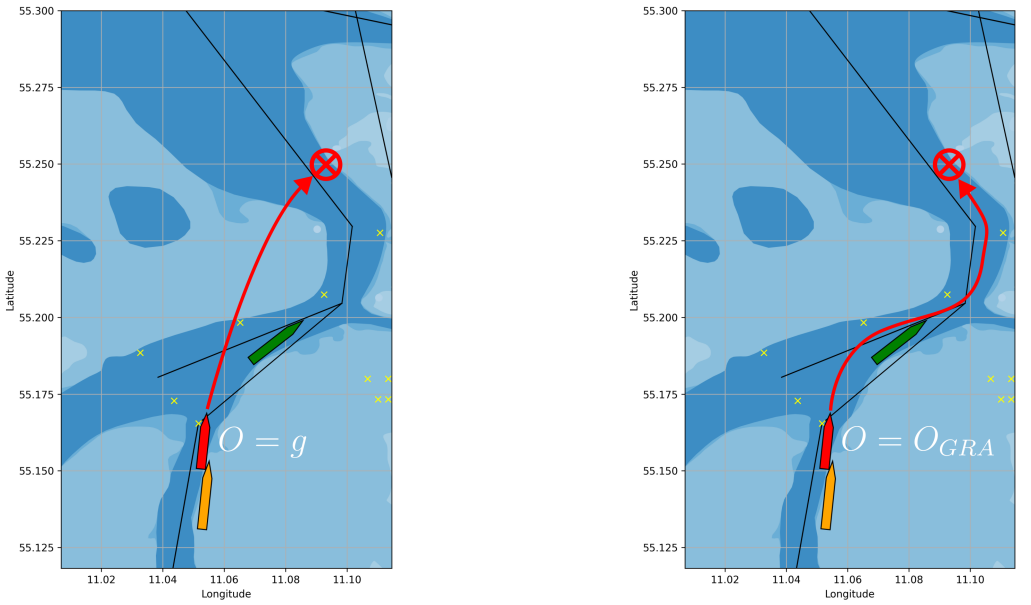
Figure 4.12: Comparison of COG of action replay before and after error removal

5 Navigation from Expert Data

With the analysis of the AIS-data in Chapter 3 and the definition of a suitable MDP in Chapter 4 the foundations for learning decision making policies have been laid.

However, first it is crucial to understand the navigation problems the decision making policies shall solve. As already introduced in Chapter 1 two distinct navigation tasks shall be addressed: GO-Navigation and EA-Navigation. For this suitable IL algorithms need to be identified.

5.1 Navigation tasks



(a) Goal-Only Navigation

(b) Environment-Aware Navigation

Figure 5.1: Navigation tasks

The two navigation tasks of interest are the *GO-Navigation* and the *EA-Navigation*. Figure 5.1 visualises the navigational goal and the corresponding observation of both.

5.1.1 Goal-Only Navigation

The objective of a policy in GO-Navigation is to reach a goal location without reacting to environmental features. Therefore, the observation is just the goal-position vector g.

In this setting, it is not possible to learn decision-making policies that are capable of collision avoidance and waypoint following. However, the complexity of this task is lower, making it computationally more feasible to learn good performing policies. For this reason, this task will be used as an initial step to validate the feasibility of IL algorithms.

The performance for a GO-Navigation policy can be evaluated based on the position, velocity and course errors relative to the expert at the end of a trajectory. Especially, the position error is of interest, as a lower error indicates, that the vessel is closer to the desired goal-position. Logically, all policies are optimised w.r.t. to this metric.

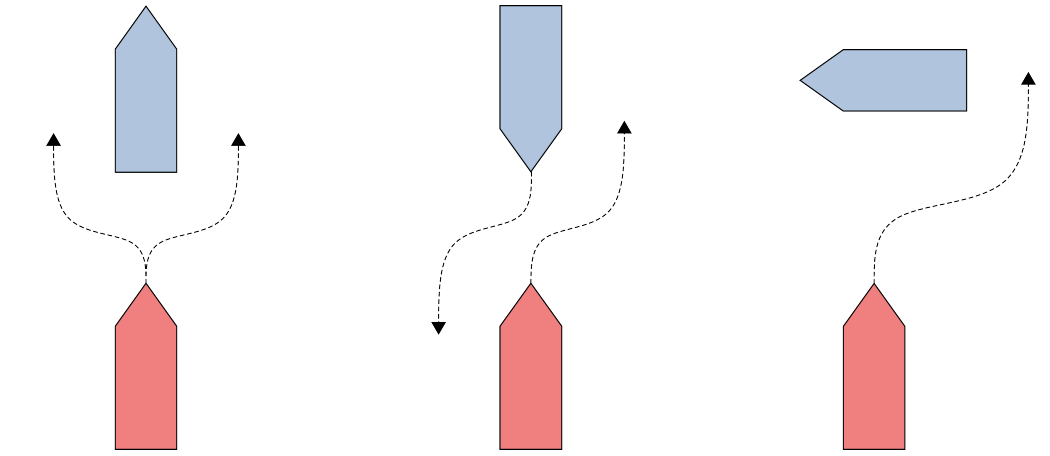
The incorrectly inferred actions, as outlined in Chapter 4.3.4, should not have an effect in this task, as the policy is not informed about the course of the vessel.

5.1.2 Environment-Aware Navigation

In EA-Navigation it is the objective to reach a goal-position while also reacting to environmental features. This includes aspects regarding navigational safety, like avoiding collisions with vessels or buoys and avoiding to run aground. But it also includes aspects regarding the overall navigational performance, like following known searoutes to a given goal-position. In this setting the graph-observation O_{GRA} is the basis for every decision making policy.

This task is more complex, as it includes all aspects of full maritime autonomy with COLREG-compliant collision avoidance and waypoint-following. Therefore, it shall be addressed as a second step once decent results have been obtained from the simpler GO-Navigation task.

The relevant metrics for this task are split into two categories. First, there are the *navigational performance metrics*. These include the average position, velocity and course errors and indicate how good a policy is able to imitate the behaviour of the expert. Optimisation w.r.t. the mean average position error is therefore the ideal choice for this task. Second, there are also the *navigational safety metrics*. These include the share of trajectories in which the policy causes a collision with another vessel, a buoy or land or runs aground.



(a) Rule 13 - Overtaking (b) Rule 14 - Head-On (c) Rule 15 - Crossing

Figure 5.2: COLREG rules 13-15

Additionally, a qualitative assessment w.r.t. the COLREGs is paramount for EA-Navigation. The primary focus is on the following three rules of the COLREGs, which are also displayed in Figure 5.2:

- **Rule 13 - Overtaking:** The overtaking vessel shall keep out of the way of the overtaken vessel. This shall be accomplished by either moving to the starboard or port side. [102]
- **Rule 14 - Head-on situation:** In a head-on situation with collision-risk both vessels shall alter their course to the starboard side in order to pass each other on the port side. The necessary actions shall be clearly recognisable and taken in decent advance as required by rule 8. [102]
- **Rule 15 - Crossing situation:** In a crossing situation with collision-risk the vessel which has the other on its starboard side shall give-way and avoid crossing in front of the other vessel. The give-way vessel shall take substantial and early action in

compliance to rule 16, while the stand-on vessel shall maintain a predictable course as required by rule 17. [102]

The incorrectly inferred actions, as introduced in Chapter 4.3.4, can have an effect on the learned policies in this task, as the observation contains information about the OVs course. Assessment of the results will show the effect.

5.2 Algorithm selection

To address both of these tasks, different algorithms from the domain of IL shall be chosen. The following requirements have been defined for the algorithm selection:

- An IL algorithm that not necessarily learns a reward function shall be chosen as baseline to compare the performance of IRL policies.
- The chosen algorithms shall support continuous state spaces and ideally also continuous action spaces in accordance with the definitions made in Chapter 4.
- The chosen IRL algorithms shall learn an explicit and interpretable reward function to allow for a derivation of the navigational goals of the experts.

Based on these requirements the following three algorithms were chosen and will be evaluated in the Chapters 6, 7 and 8.

5.2.1 Behaviour Cloning

The first algorithm is BC [71] [72]. As will be elaborated in more detail in Chapter 6 it is based on a conceptually simple supervised learning approach that uses expert data to only train a policy.

This method was chosen to offer a good baseline for the more sophisticated IRL algorithms. In theory IRL algorithms should outperform BC as it suffers from the *compounding error* problem (see Chapter 2). Additionally, it can be used to determine suitable neural network architectures for the learned policies, which then may be used for other methods. The intuition is that if a policy network architecture is able to learn meaningful behaviour via BC it is safe to assume it can also do this in an IRL setup.

5.2.2 Adversarial Inverse Reinforcement Learning

AIRL [50] has been chosen as the first IRL algorithm. It can be categorised as adversarial approach and its choice is motivated by two aspects. First, this algorithm recovers an interpretable and potentially non-linear reward function that can be used to train RL agents and to interpret the navigational goals of the expert demonstrators. Second, the algorithm has already shown good results in literature. Not only the authors of the original paper [50] show good results on general benchmarks, but also in the maritime domain this algorithm has already been applied successfully [49]. This makes it an ideal choice for the first IRL algorithm.

5.2.3 Approximate Variational Reward Imitation Learning

The last algorithm implemented and evaluated in this thesis is AVRIL [87]. This method is categorised as a Bayesian approach, which leads to its first positive aspect: AVRIL recovers non-linear rewards that are modelled as Gaussian normal distributions allowing to reason about the uncertainty in the reward. This is useful in cases, where the reward function shall be interpreted to extract the experts' motives. Additionally, the algorithm is completely offline, which provides great benefits in terms of training time. One downside is, that the action space needs to be discretised. Overall, this algorithm shows potential and will therefore be evaluated and compared against the other two algorithms.

6 Behaviour Cloning

The first method to be implemented and evaluated is BC. It's purpose is to serve as the performance baseline for IRL methods, since BC-policies should suffer from the compounding error problem. Therefore, IRL methods should be able to outperform BC.

6.1 Method

As already briefly introduced in Chapter 2, BC [72, 71] uses supervised learning on recorded expert demonstrations to learn a policy π_L. Figure 6.1 provides an overview of the method.

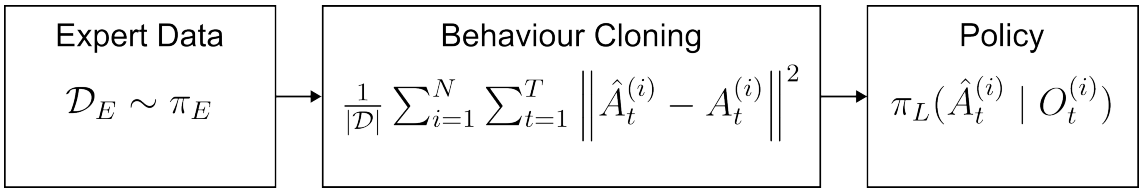


Figure 6.1: Behaviour Cloning

BC starts with a dataset $\mathcal{D}_E$ consisting of demonstrations sampled from trajectories of length T conducted by the expert policy π_E. The dataset consisting of N trajectories is split into three parts based on the scenarios, where the *train* split is used for training and the *val* split is used for evaluation and hyperparameter optimisation. [72, 71]

The expert dataset is formally defined by:

$$\mathcal{D}_E = \left\{ O_t^{(i)}, A_t^{(i)}) \right\}_{i=1,t=1}^{N,T}, \quad \text{where } (O_t^{(i)}, A_t^{(i)}) \sim \pi_E \tag{6.1}$$

The BC method assumes an agent with a learner policy π_L. This agent learns a mapping between an observation $O_t^{(i)}$ and the action $A_t^{(i)}$ to take using the expert data. The policy π_L is modelled as non-linear function, in this case a task-specific neural network, due to their high representational capacity. [72, 71]

BC is defined as a supervised regression problem. This means learning the policy π_L is framed as the minimisation of the MSE objective w.r.t. the weights θ of the learner policy π_L as shown in Equation 6.2. This is done via backpropagation and stochastic gradient descent using the Adam optimiser [101]. [72, 71]

$$\mathcal{L}_{\text{MSE}}(\theta) = \frac{1}{|\mathcal{D}_E|} \sum_{i=1}^{N} \sum_{t=1}^{T} \left\| \hat{A}_t^{(i)} - A_t^{(i)} \right\|^2, \quad \text{where } \hat{A}_t^{(i)} \sim \pi_L(\cdot \mid O_t^{(i)}, \theta) \tag{6.2}$$

6.2 Implementation

To successfully apply BC to the given ASN problem a few considerations regarding the implementation are required. Those include the scaling and normalisation of the data, the regularisation within the learning process and, most importantly, the design of the policy network.

6.2.1 Scaling

Since the learned policies are non-linear function approximators based on neural networks, scaling of the observation features and the action into the range $[-1, 1]$ is necessary to account for different feature value magnitudes.

Linear scaling is used for the observation features. This applies to all features, except for the angles which are naturally in a valid range due to the decomposition into sine and cosine components. Appendix B provides more details on the exact scaling parameters.

The actions are also scaled. This is done by scaling them linearly and clipping them into the range of $[-1, 1]$. To spread out small action values more within the action space, the actions are further transformed by taking the square root. This is done to increase the resolution of small action values and to enhance gradient informativeness. For more details on the scaling process, please refer to appendix C and D.

6.2.2 Normalisation

To enhance learning performance in BC, all observations O should be normalised.

The goal features g of the graph-based observation O_{GRA} are represented as a vector. To normalise these features, the mean and standard deviation are calculated over the *train* split of the expert dataset $\mathcal{D}_E$ and subsequently applied to the observations in all dataset splits.

The other components of the graph-based observation O_{GRA} are graphs. In this case the mean and standard deviation of all non-categorical and non-binary features are computed w.r.t. all nodes and edges of all graphs inside the *train* split of the expert dataset $\mathcal{D}_E$. Again, they are then applied to all data splits.

6.2.3 Regularisation

Another important aspect of the BC implementation is sufficient regularisation to learn generalised behaviour traits. Mainly two measures shall be used. First, L_2 regularisation shall be used to keep the magnitude of the model weights reasonably small to avoid overfitting. The regularisation parameter λ_{L_2} as shown in Equation 6.3 is subject to hyperparameter tuning.

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{MSE}}(\theta) + \lambda_{L_2}\|\theta\|_2^2 \tag{6.3}$$

Second, dropout, which is the probability that a parameter of the model is set to 0 during training, shall be a part of every policy network architecture to allow for learning more robust policies. The dropout rate is also subject to hyperparameter optimisation.

6.2.4 Dataset imbalance

Since the dataset contains mostly straight trajectories, observations with large action magnitudes are underrepresented in the data as shown in Figure 6.2. This can possibly result in local minima of the optimisation landscape that correspond to linear motions without turning or accelerating.

As this might hinder the learning of policies that are capable of waypoint-following or collision avoidance, one can oversample observations with high magnitudes within the *train* split of the data. For this purpose, a weight has been applied to each observation in the *train* split of the expert dataset $\mathcal{D}_E$. Observations where either the acceleration $\dot{u}$ or the yaw rate r have an absolute magnitude larger than 0.15 are assigned a weight of 10 while all other observations are assigned the weight 1. Each observation is then sampled

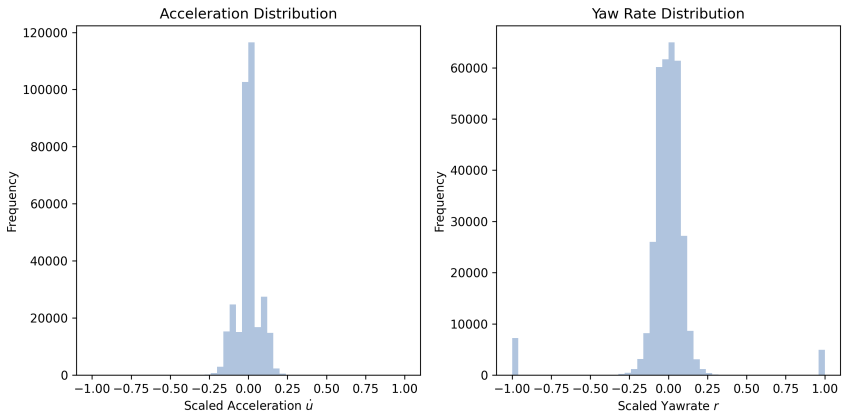


Figure 6.2: Distribution of action values in the training dataset without oversampling

randomly with a probability proportional to its weight. This procedure ensures that the bias towards small action values is smaller and that the gradient updates are more informative.

Obviously, this step also magnifies the weight of the of the incorrectly inferred large yaw rates as elaborated in Chapter 4.3.4. However, Figure 6.2 shows that they do not represent the majority of values to be oversampled. As a result, the trained policies should still be capable of generalising beyond these incorrectly inferred yaw rates. Additionally, all experiments are structured as Bayesian hyperparameter optimisations, that should be able to identify if oversampling hurts the performance.

6.2.5 Policy neural networks

As it is the main decision making part of the agent the design of the policy neural networks is crucial. The network is specific to the way the observation and the action is defined, which gives rise to a few requirements to the definition of its architecture:

- The policy network shall be able to handle the graph-based observation as input, by offering the possibility to use GNN architectures as well as classical FFNN architectures.
- The policy network architecture shall be modular. This implies that the number of layers, the activation function and other regularisation methods shall be configurable as parameters. Additionally, the architecture shall allow that both navigational tasks can be solved with it (GO-Navigation and EA-Navigation).
- The policy network shall also offer the opportunity to configure either a deterministic or stochastic policy. This property is needed, since different IRL and RL train different kinds of policies. Additionally, this allows for the expansion of the architecture to value and reward functions in later stages.

Based on these requirements the fully modular architecture depicted in Figure 6.3 has been developed.

It is composed of multiple feature extractors and a common regressor. One feature extractor has one part of the observation O_{GRA}, either one graph or the goal features, as input and learns a different vectorised representation of its input. These vectorised representations are then concatenated and serve as the input to the regressor. The regressor may either be stochastic or deterministic and computes the actions to be taken by the agent.

Benefits of this architecture are, that the feature extractors and the regressor can be varied

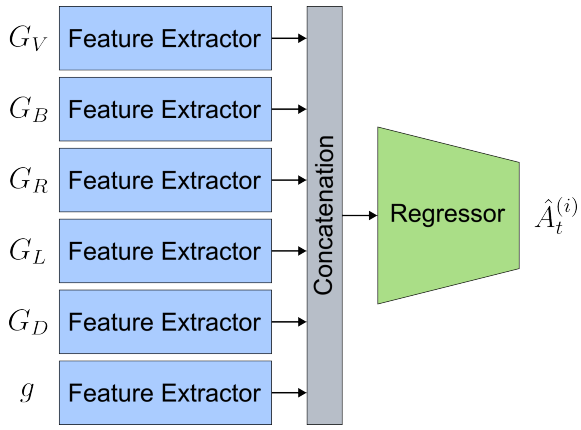


Figure 6.3: Policy network architecture

in their respective architecture. Additionally, it allows to add other feature extractors to the architecture, so that value or reward functions can be easily built based on it.

In total, two different feature extractors, with different levels of complexity, have been developed. One can handle vectorised input, while the other works directly on the graph data structures. Only one regressor type has been developed, which is capable of both, deterministic and stochastic predictions. This choice makes the regressors flexible to later demands from AIRL which requires stochastic outputs for policies and deterministic outputs for value and reward functions.

For EA-Navigation the whole architecture depicted in Figure 6.3 is used. In GO-Navigation it is simplified to just use the feature extractor for g, which ultimately simplifies the architecture to a single FFNN.

Vector feature extractors

The conceptually and computationally simpler of the two feature extractors is the vector feature extractor, which utilises an abstraction of the graph inputs.

First, the mean of all nodes and all edges inside a graph is computed. Obviously, this reduces the available information, but it also allows for a fixed size input representation. After that the node and edge features are concatenated to a single vector representation. This vector is then fed into multiple FFNN layers. Each layer consists out of a linear layer, the Rectifier Linear Unit (ReLU) activation function as well as a dropout component for regularisation purposes.

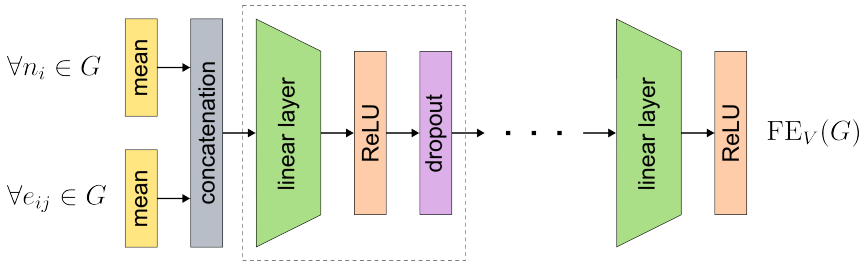


Figure 6.4: Vector feature extractor architecture

Figure 6.4 visualises this architecture. The number and size of the hidden layers is configurable, while the level of dropout can also be adjusted. All these parameters are subject to hyperparameter optimisation as described in section 6.3.

The goal features g use the same feature extractor architecture. However, as g is already

a vector, the mean and concatenation part is not included in its feature extractor.

Graph feature extractors

The second type of feature extractor utilises the full graph for its computations, as it is based on GNNs [103].

The feature extractor is composed of multiple graph-processing components. Each graph-processing component takes the whole graph as input and computes a new node representation for every node. This is done via a graph-layer that essentially computes this new node representation using the features of the neighbouring nodes and the connecting edges. The new node representation is then normalised using batch normalisation, fed through a ReLU activation function and a dropout layer for regularisation. Multiple of these components can be stacked on top of each other. In the end a pooling layer over all computed node representations is applied to derive a fixed size vector representation. This is followed by another linear layer and the ReLU activation function to derive the final extracted features.

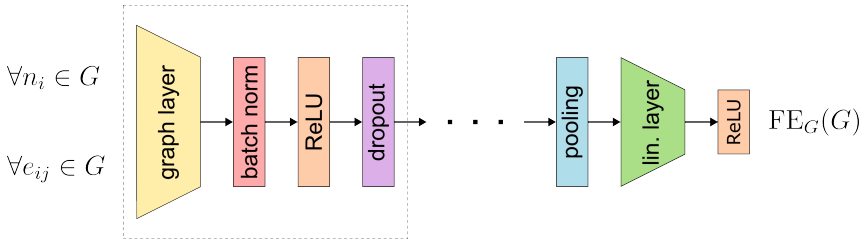


Figure 6.5: Graph feature extractor architecture

Figure 6.5 visualises this feature extractor architecture. This architecture also offers great flexibility regarding its parameters. First, the type of graph-layer can be configured. Four different types have been chosen as they are able to process edge features together with node features, which is a necessity for the chosen observation definition. The four types are NNConv [104], GINEConv [105], GATConv [106] and a custom designed layer called ConcatConv. Appendix E provides details about the design of those layers. Additionally, the number of graph-layers and the size of each graph-layer can be configured. The dropout rate is also a hyperparameter. Last, the pooling method can be one of three types. Those are *mean*, *max* and *sum*. The best hyperparameters will be discovered via hyperparameter optimisation as described in section 6.3.

As this feature extractor is only applicable to graph-structures the goal-position input g would still use the already introduced vector-based feature extractor.

Regressors

The final part of the policy architecture is the regressor, which computes the actions based on the extracted features.

As the feature extractors provide the features in vectorised form, the regressor can be modelled as simple FFNN. It is composed of multiple linear layers with ReLU activation function and dropout for regularisation. The number and size of the layers is again variable.

As the regressor supports both, stochastic and deterministic outputs, its final layer requires more attention. For the deterministic regressor, the output has the dimensionality of the action space with each output predicting one part of the action. The stochastic regressor, however, has its output modelled as Gaussian normal distribution, where both the mean and variance are predicted by the model. A prediction can be done by sampling

from the distribution and backpropagation is possible via the reparametrisation trick. The outputs are additionally clamped to be in the valid action range to improve stability of the learning process.

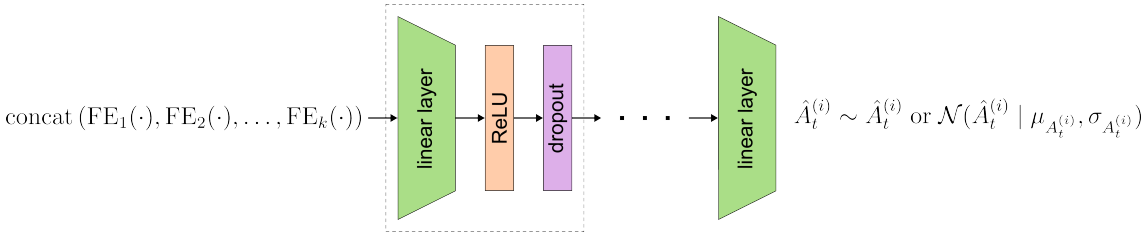


Figure 6.6: Regressor architecture

Figure 6.6 visualises the regressor architecture. The architecture of the regressor is also subject to parameter tuning. The number and size of the linear layers, as well as the dropout rate and the output type are optimisation objectives.

The choice to include both, deterministic and stochastic, regressors has been made, because some IRL-algorithms specifically require the usage of one of the two output types.

6.3 Experiments

As introduced in Chapter 5, two navigational tasks shall be solved by all algorithms. First, BC will be evaluated on the simpler GO-Navigation task. In a second step BC is evaluated on the EA-Navigation task. However, the evaluation of policies for EA-Navigation is done in two steps. First a vectorised representation shall be evaluated and used as baseline for the performance of graph neural networks as policy.

6.3.1 Goal-Only Navigation

The first experiment applies BC to the GO-Navigation task where the trained policy takes the goal features g as input and computes the actions needed to go to the goal-position without learning any collision avoiding behaviour. This simplifies the policy architecture to effectively just the regressor of the policy network architecture.

The training of the policy π_L is set up as a hyperparameter optimisation. The best set of hyperparameters is determined by a Bayesian hyperparameter optimisation, which optimises w.r.t. the average end position error in all validation scenarios. The lower this error, the better the performance of the goal-position reaching policy. A total of 50 different configurations are trained in this setting. Each configuration is trained for 25 epochs and uses the Adam optimiser [101]. The parameters that are subject to the optimisation are shown in Table 6.1.

The best model can be chosen according to the minimum average end position error on the validation scenarios. In this case the best model managed to achieve an error of 870.81 m using the hyperparameters highlighted green in Table 6.1. Judging based on the 10 best performing models of the Bayesian hyperparameter search, it can be concluded that smaller L_2 regularisation weights perform better. For all the other hyperparameters no clear tendency can be observed.

The navigational performance of the best performing model can be expanded beyond the mean end position error of 870.81 m on the validation scenarios, which already indicates that the trained policy succeeds in manoeuvring the vessel towards the goal location. However, the standard deviation on this error is rather large with 1026.62 m indicating

Symbol	Parameter	Values			
—	Batch size	2048	4096	8192	
—	Hidden layers	$(64, 32)$	$(64, 64, 32)$	$(128, 64)$	$(128, 128, 64)$
		$(256, 128)$	$(256, 256, 128)$	$(512, 256)$	$(512, 512, 256)$
—	Learning rate	$5 \cdot 10^{-3}$	10^{-3}	$5 \cdot 10^{-4}$	
—	Dropout rate	0.0	0.1	0.2	0.3
—	Orthogonal init.	`True`	`False`		
—	Stochastic output	`True`	`False`		
—	Oversampling	`True`	`False`		
λ_{L_2}	L_2 regularisation	0.0	10^{-5}	10^{-4}	10^{-3}

Table 6.1: Hyperparameters for goal-only BC

that the GO-Navigation is not successful in every validation scenario. This can also be seen when analysing the mean end course error, which is far from zero with 38.54°. This indicates that the policy often ends a trajectory with a significantly different course compared to the ground truth trajectory. The velocity error at the end of a trajectory, however, is reasonably small with a mean of 0.34 m s^{-1}. This indicates that the velocity behaviour has been cloned decently well. Table 6.2 shows a more detailed view on the navigational performance metrics computed on the validation scenarios.

	Mean	Std	Min	Max
End position error (m)	870.81	1026.62	27.13	7465.21
End course error (°)	38.54	25.44	0.38	178.38
End velocity error ($\frac{m}{s}$)	0.34	0.41	0.002	3.29

Table 6.2: Navigational performance of goal-only BC in validation AIS scenarios

The shown metrics indicate decent waypoint-following performance. However, it is crucial to also qualitatively assess the performance inside the validation AIS-scenarios. Figure 6.7 shows one of the validation scenarios. The learned BC-agent is manoeuvring the vessel with the MMSI 314 462 000 and its trajectory can be compared with the AIS ground-truth (GT) trajectory. The agent is coloured pink, while the GT is green. This is the case for all future visualisations of this kind. It can be seen that the agent manages to navigate to the goal-position with minimal deviation from the expert's path. This shows that the desired behaviour of waypoint-following has been successfully learned.

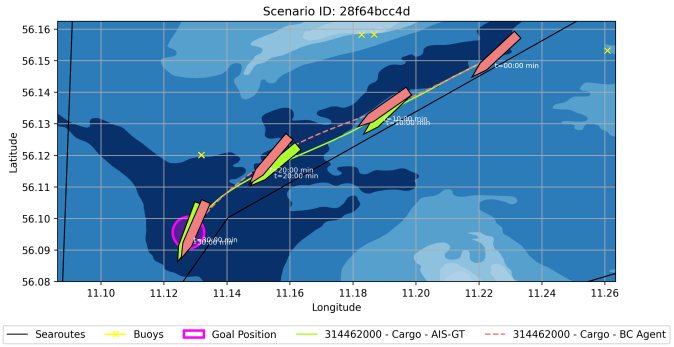


Figure 6.7: Scenario 28f64bcc4d - Goal-only BC policy

However, the policy does not imitate the experts in all situations this accurate. In several other scenarios the agent also manages to go towards the waypoint, but it does so by

deviating much more from the expert's path. At some point it then turns towards the waypoint and reaches it from a different angle. This causes the large end course errors. Figure 6.8 shows such a scenario.

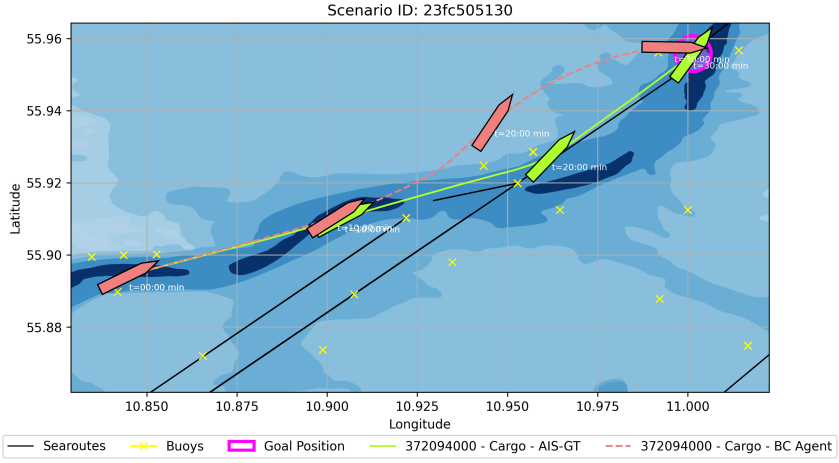


Figure 6.8: Scenario 23fc505130 - Goal-only BC policy

In some occasions this behaviour leads to the agent taking considerably longer routes, which results in the OV not getting close to the waypoint. Fortunately, in most cases it still moves in the direction of the waypoint, indicating that the correct intentions have been learned. Figure 6.9 displays such a situation.

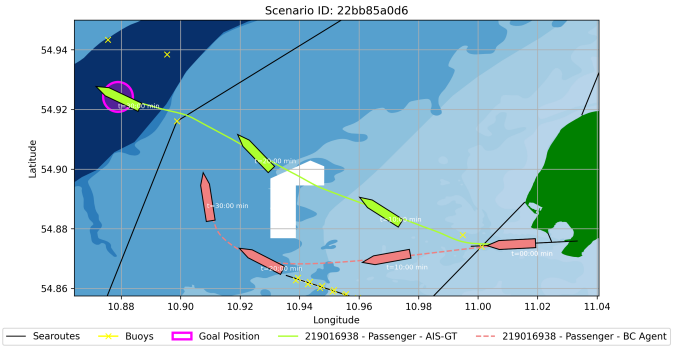


Figure 6.9: Scenario 22b85a0d6 - Goal-only BC policy

Unfortunately, the learned policy does not master every scenario. In rare occasions it completely misses to navigate towards the goal-position. This may be attributed to the *compounding error* effect of BC. As soon as the agent enters a region of the observation space that is not part of the training data, such as locations behind the goal-position, it fails to successfully navigate towards it. Figure 6.10 is an example of such behaviour.

Overall, BC performs well in the GO-Navigation task. In most cases the learned policy manages to move the OV close to the goal-position. However, the policy exhibits deficits in accomplishing the task, when deviating too much from the shortest path. Accordingly, there is still opportunity to improve using the IRL algorithms. The incorrectly inferred actions from Chapter 4.3.4 did not influence the results.

6.3.2 Environment-Aware Navigation with Vector-Based Policy

In the second experiment the whole graph-based observation O_{GRA} is used to learn policies that are meant to solve the full EA-Navigation problem. This experiment uses the vector-based feature extractor definition, which just provides a *summarised* description of the OVs surroundings. This shall be a baseline for the graph-based policies of Chapter 6.3.3. If a policy of the graph-based experiment is unable to outperform the vector-based policy, it can safely be discarded.

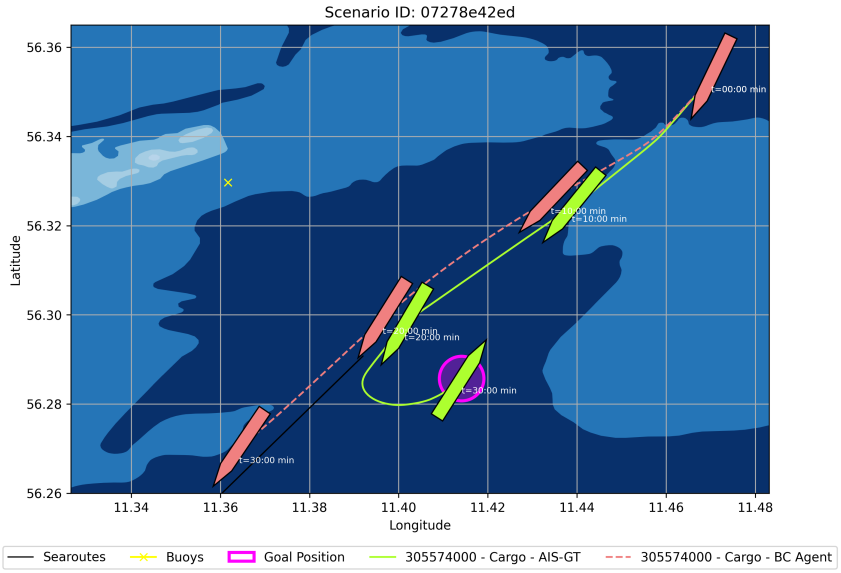


Figure 6.10: Scenario 07278e42ed - Goal-only BC policy

Again the experiment is set up as Bayesian hyperparameter optimisation, but this time the optimisation happens w.r.t. the mean average position error in all validation scenarios. To achieve a low error, the policy needs to imitate the expert policy including collision avoidance and adherence to good seamanship practices. The number of different configurations trained has been reduced to 20, to account for the increased computational complexity of the EA-Navigation task. Each configuration is trained for 25 epochs with the Adam optimiser [101]. Table 6.3 shows the parameters that are optimised in the hyperparameter search.

Symbol	Parameter	Values			
—	Batch size	1024	2048	4096	
—	Reg. Hidden layers	$(256, 128)$	$(256, 128, 64)$	$(512, 256)$	$(512, 256, 128)$
—	Feat. Ext. Hidden layers	$(256, 128)$	$(256, 128, 64)$	$(512, 256)$	$(512, 256, 128)$
—	Feat. Ext. Output Dim.	64	128		
—	Learning rate	$5 \cdot 10^{-3}$	10^{-3}	$5 \cdot 10^{-4}$	
—	Dropout rate	0.0	0.1	0.2	0.3
—	Orthogonal initialisation	`True`	`False`		
—	Oversample	`True`	`False`		
—	Stochastic output	`True`	`False`		
λ_{L_2}	L_2 regularisation	0.0	10^{-5}	10^{-4}	10^{-3}

Table 6.3: Hyperparameters for environment-aware BC with vector-based policy

As mentioned, the best policy is chosen according to the smallest mean average position error, which has been recorded with 530.28 m for a policy trained using the parameters highlighted in Table 6.3. Analysing the 5 best performing configurations regarding common parameters one can observe that the L_2 regularisation λ_{L_2} should be set to 1×10^{-4} or 1×10^{-5} and the learning rate to 5×10^{-4} in order to improve performance. Regarding the policy architecture only one thing can be summarised: the feature extractor output dimension should be large with a value of 128.

The policy trained using these parameters can now be assessed w.r.t. its performance in the validation scenarios. First, the navigational performance metrics shown in Table 6.4 are of interest. The average position error, with a mean of 530.28 m and a standard deviation of 529.51 m, shows that the policy has learned to imitate the experts. The same applies to the average course and velocity errors with mean values of 7.56° and 0.23 m s^{-1}. These metrics indicate that the trained policy manages to imitate the expert trajectories with some error. Additionally, the waypoint following performance of this policy can be assessed using the end position, course and velocity errors. One can observe that the performance w.r.t. waypoint following is equal to policies trained just for this purpose (see Table 6.2). The position error is slightly higher, while the end course error is considerably lower. This may indicate that the policy acts more like the experts do. However, this has to be confirmed by qualitative assessment.

	Mean	**Std**	**Min**	**Max**
Avg. position error (m)	530.28	529.51	26.96	3272.12
Avg. course error (°)	7.56	6.80	0.61	49.66
Avg. velocity error ($\frac{m}{s}$)	0.23	0.30	0.02	2.75
End position error (m)	938.71	1113.89	25.99	6920.73
End course error (°)	13.14	15.51	0.11	177.13
End velocity error ($\frac{m}{s}$)	0.34	0.42	0.002	3.41

Table 6.4: Navigational performance of vector-based BC in validation AIS scenarios

Next to the navigational performance metrics, the navigational safety metrics, shown in Table 6.5, can also be used for the evaluation. These express the share of trajectories in the validation scenarios in which the OV encounters a collision or a grounding. Additionally, this is put into perspective by computing the difference to the experts' collision rates. Due to the inaccuracies of the AIS-data and seachart they also encounter collisions and groundings. This is the case for the buoy collisions, where the learned policy outperforms the experts by 25 %. Collisions with shore are on a similar level, however for both the collisions with other vessels and the groundings the experts are far superior. Especially, the awareness of the learned policy for shallow waters is worse, as it runs aground in over a fifth of all trajectories. This shows potential for further improvement.

	Value (%)	**Expert Value (%)**	**Difference**
Traj. with groundings	22.69	9.80	+132%
Traj. with vessel collisions	4.48	3.08	+45%
Traj. with shore collisions	8.40	8.12	+3%
Traj. with buoy collisions	1.68	2.24	−25%

Table 6.5: Navigational safety of vector-based BC in validation AIS scenarios

Apart from the quantitative assessment the performance of the vector-based policy also needs to be evaluated qualitatively. Therefore, several representative validation scenarios are used. Those scenarios include situations where the OV encounters an overtake, a head-on and a crossing situation to cover all COLREG-relevant situations. Additionally, the awareness of searoutes, shallow waters and shore is evaluated in distinct scenarios.

First, the general goal-position following behaviour shall be assessed. Looking at scenar-

ios as shown in figures 6.11 or 6.13a the policy has generally managed to learn moving close to the goal-position. However, it not always manages to go exactly to this position due to other navigational constraints or previous decisions.

Exactly this is the case in Figure 6.11, where the policy encounters an overtake situation. It manages to successfully overtake the TV but decides to do this on the starboard instead of the port side of the TV. This behaviour is COLREG-compliant but leads to a larger deviation from the goal-position. One might argue that the shown behaviour is preferable compared to the expert as the distance in the head-on situation with the second TV is maximised.

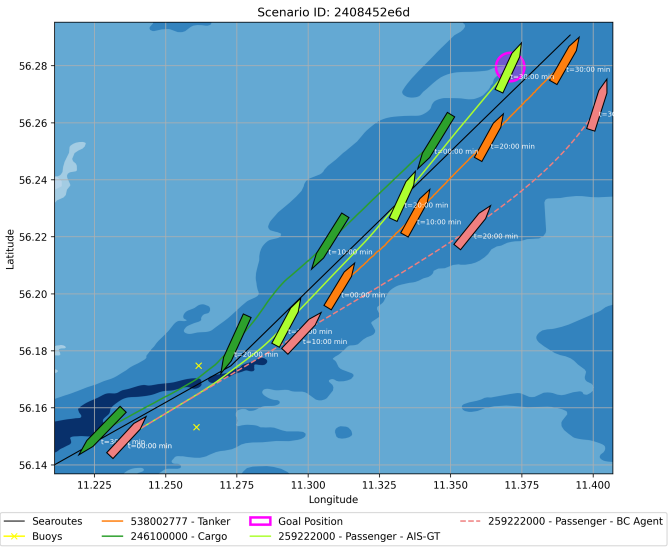
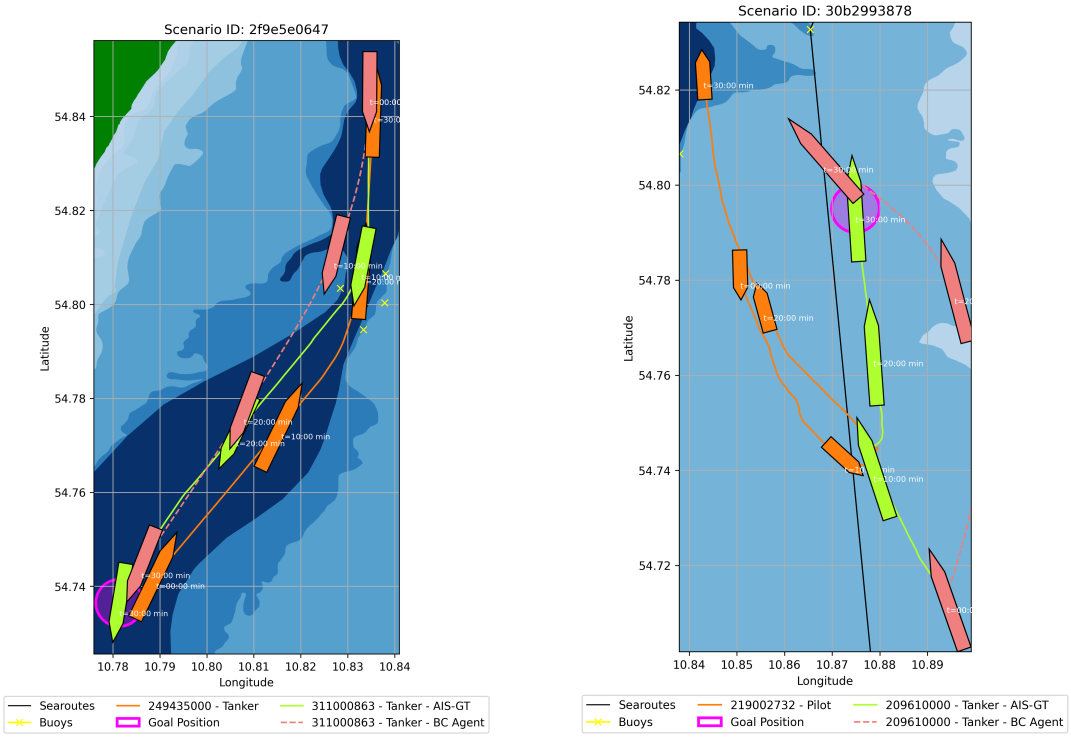


Figure 6.11: Scenario 2408452e6d - Vector-based BC policy in overtake scenario

Head-on collision situations can be analysed in more depth using Figure 6.12. As one can see in Figure 6.12a the OV successfully avoid a head on-collision by staying out of the path of the TV. This is also done in the correct way by keeping the TV on its port side. However, by doing so it risks navigating through shallower water and disobeying buoys. The second example shown in Figure 6.12b reveals two interesting things. First, the OV manages to avoid the head-on collision with the pilot vessel by making a big avoiding action towards its starboard side. This is clearly good behaviour according to the COLREGs. Second, as already mentioned in Chapter 3, the *pilot* vessels perform *intended collisions* with other vessels. The learned policy seems not to be able to distinguish between pilot and other vesseltypes as it avoids getting close to the pilot vessel. The *pilot* vessel information is likely lost during vectorisation of the graphs, which motivates the use of GNNs.

The third situation, that is of interest for COLREG compliance, is a crossing situation, where the OV is the give-way vessel. Figures 6.13a and 6.13b show two examples of such situations with different resulting behaviours. In Figure 6.13a the OV would have to follow the searoute to avoid shallow waters while also taking care of the crossing TV. However, it fails at the first challenge. It does not follow the deeper waters and the searoute, as it takes a shortcut over shallow water. With doing this it clearly shows limited awareness of shallower waters and complex searoutes. By taking the shortcut, it crosses the TVs trajectory behind the TV, which is COLREG-compliant. Figure 6.13b reveals better reactions to a crossing scenario. Here the OV actively steers towards the starboard side to maximise the distance to the crossing TV. By doing so it effectively avoids a collision in a COLREG-compliant manner while still moving towards the goal-position.

All of the so far analysed scenarios feature just a few TVs and relatively simple navigational tasks w.r.t. the searoutes, buoys and depth of the sea. Therefore, the performance shall also be analysed for more complex scenarios. Figure 6.14 shows two of these. As one can see the agent manages to avoid collisions with other vessels. However, when doing so it is not perfectly following the searoutes nor avoiding shallower waters. These



(a) Scenario 2f9e5e0647 - Vector-based BC policy in head-on scenario

(b) Scenario 2f9e5e0647 - Vector-based BC policy in pilot-encounter scenario

Figure 6.12: Vector-based BC policy in head-on & pilot-encounter sceanrios

worse behaviour traits can be mainly observed in scenarios, where the number of TVs rises and the environment in terms of the depths and other navigational supports gets more complicated. In some occasions, which are not shown here, the OV even shows no sign of any collision avoiding or environment aware behaviour.

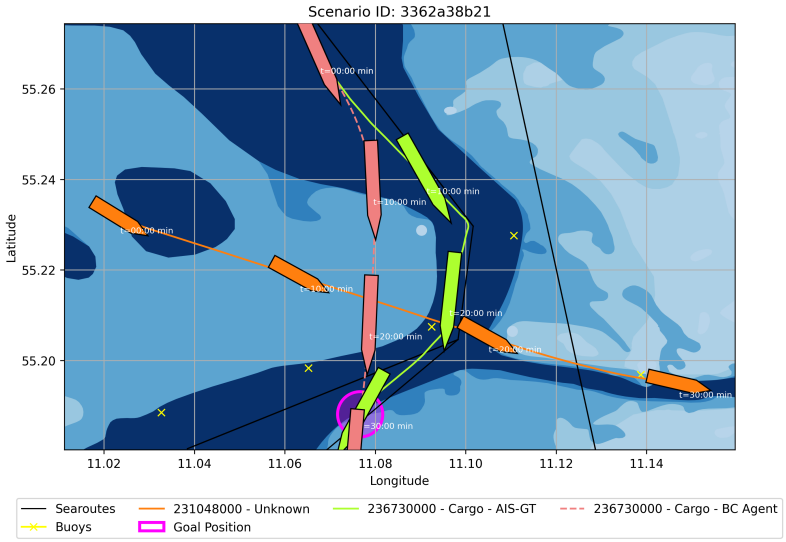
Summarising the performance of the vector-based BC policy, one can say that it in fact managed to learn some COLREG-compliant traits and it shows, as expected, limited awareness of the confined waters around it. However, there is still a lot of room for improvement in acting according to the COLREGs and being aware of shallower waters. Applying GNNs is s first step in this direction. The faulty inferred actions, as described in Chapter 4.3.4 did not have a negative impact on the policies.

6.3.3 Environment-Aware Navigation with Graph-Based Policy

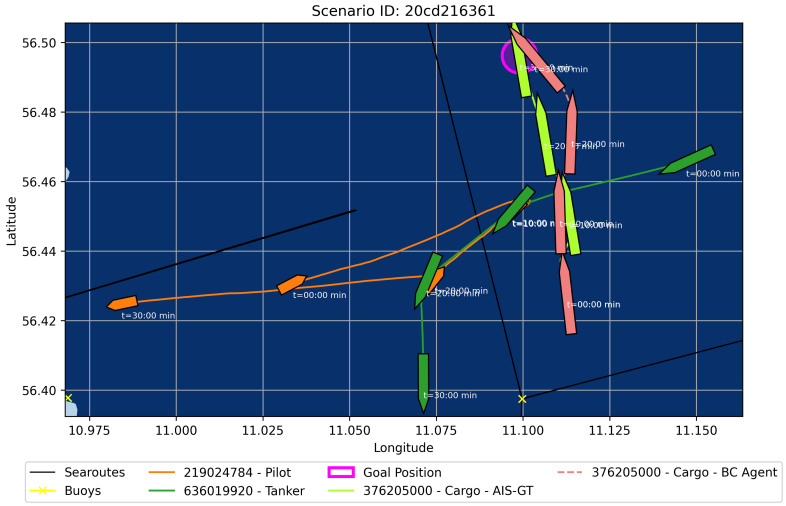
Since the observation O_{GRA} is defined as a graph-structure the use of GNNs as part of the policy network is the logical next step. Therefore, the already introduced graph-based feature extractors are used and evaluated in this experiment.

The setup of the experiment is similar to the one for the vector-based policy. Bayesian hyperparameter optimisation w.r.t. the mean average position error is applied. Each configuration is trained for 25 epochs using the Adam optimiser [101]. Table 6.6 shows the parameters that are subject to optimisation. Each graph layer is optimised in a separate optimisation with a total of 20 configurations, as each layer type might require completely different parameter sets.

Conducting all four hyperparameter optimisations leads to the results shown in Table 6.7. As one can see the NNConv layer performed best with an mean average position error of 387.61 m, which is significantly better than the baseline vector-based policy trained



(a) Scenario 3362a38b21 - Vector-based BC policy in crossing scenario

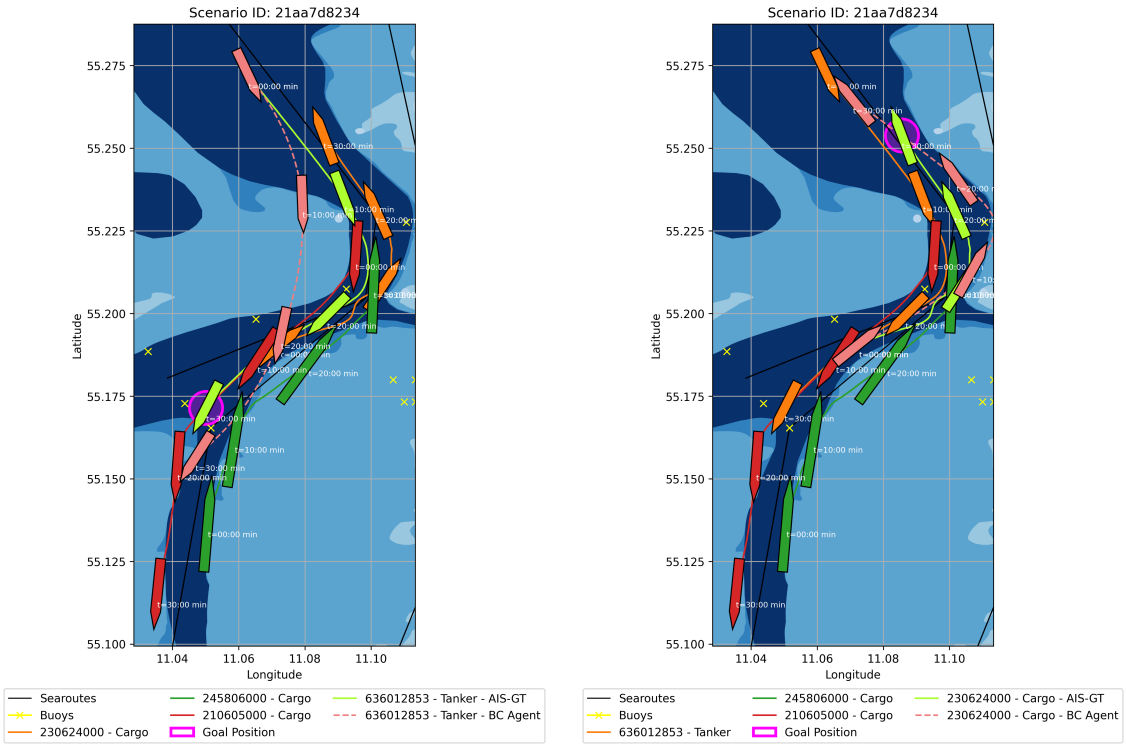


(b) Scenario 20cd216361 - Vector-based BC policy in crossing scenario

Figure 6.13: Vector-based BC policy in crossing scenarios

in section 6.3.2. Also the models using the GINEConv and ConcatConv graph layers outperform the vector-based policy. Only the GATConv layer does not manage to do so. Analysing the navigational safety metrics, one can observe that the NNConv policy outperforms the baseline vector-based policy significantly for groundings and collisions with other vessels. The ConcatConv model does so for collisions with other vessels and collisions with buoys. The GINEConv model does so only for collisions with vessels while the GATConv model is not able to outperform the baseline vector-based policy by a big margin. As it is the best performing policy in a majority of metrics the NNConv-based policy shall be used.

The parameters used by the NNConv-based policy are highlighted green in Table 6.6. From the parameters of other 5 well performing models one can see that smaller learning rates of 1×10^{-3} are preferable. In terms of regularisation smaller dropout rates in the range of 0 to 0.1 and smaller levels of L_2 regularisation with values of 1×10^{-4} or 1×10^{-3} perform better. Additionally, policies trained with smaller batch sizes and using



(a) Scenario 21aa7d8234 - Vector-based BC policy in env.-restricted scenario

(b) Scenario 21aa7d8234 - Vector-based BC policy in env.-restricted scenario

Figure 6.14: Vector-based BC policy in environment restricted scenarios

a stochastic regressor perform better. Apart from these insights, the other parameters do not show significant biases.

The NNConv-based policy trained using these parameters, can now be assessed in more detail. First, the navigational performance metrics are shown in Table 6.9. The average position error is 387.61 m with a standard deviation of 341.75 m. This is better than the vector-based policy. The average course and velocity errors are 7.12° and 0.24 m s^{-1}, which is comparable to the vector-based policy. Overall, this indicates that the policy can successfully imitate the experts to some extent. It is also worth analysing the waypoint-following performance via the end position, course and velocity errors. With an average end position error of 527.95 m and an average end course error of 22.51° the policy shows good waypoint-following behaviour, which is better than for BC policies trained specifically for this purpose (see Table 6.2). This indicates that the additional information provided by the graph-based observation O_{GRA} allows the agent to imitate the experts better and make improved navigational decisions.

Table 6.9 shows the navigational safety metrics, which express the share of trajectories with either a collision or a grounding. They reveal that the NNConv-based policy performs worse than the experts in terms of avoiding groundings by 86 %. However, this has improved compared to the vector-based policy. In terms of collision with other vessels, this policy outperforms the experts, showing that collision avoiding behaviour has been learned. The policy also achieves better than expert performance when avoiding collisions with buoys, but the vector-based policy acts even better in this discipline. Collisions with shore are on a similar level with the experts. All this leads to the conclusion that the NNConv-based policy has awareness of other vessels, buoys and shore as it manages to

Symbol	Parameter	Values		
—	Graph layer	`ConcatConv`	`NNConv`	`GINEConv`
		`GATConv`		
—	Graph layer hidden layers*	$()$	(16)	(32)
		$(16, 16)$	$(32, 32)$	
—	Attention Heads†	1	2	3
—	Graph Feat. Ext. Hidden layers	$()$	(16)	(32)
		$(16, 16)$	$(32, 32)$	
—	Goal Feat. Ext. Hidden layers	$(32, 16)$	$(32, 32, 16)$	$(64, 32)$
		$(128, 64)$	$(64, 64, 32)$	$(128, 128, 64)$
—	Feat. Ext. Output Dim.	8	16	32
—	Regressor Hidden layers	$(32, 16)$	$(32, 32, 16)$	$(64, 32)$
		$(128, 64)$	$(64, 64, 32)$	$(128, 128, 64)$
—	Batch size	128	256	512
η_{BC}	Learning rate	10^{-2}	10^{-3}	
—	Dropout rate	0.0	0.1	0.2
		0.3		
—	Orthogonal initialisation	`True`	`False`	
—	Oversample	`True`	`False`	
—	Stochastic output	`True`	`False`	
λ_{L_2}	L_2 regularisation	0.0	10^{-5}	10^{-4}
		10^{-3}		
—	Pooling	`mean`	`add`	`max`

* Applies only to GINEConv, NNConv and ConcatConv.
† Applies only to GATConv.

Table 6.6: Hyperparameters for environment-aware BC with graph-based policy

	Vector	GINEConv	NNConv	GATConv	ConcatConv
Mean Avg. position error (m)	530.28	474.25	387.61	615.25	494.76
Mean Avg. course error (°)	7.56	7.35	7.12	8.77	8.13
Mean Avg. velocity error (m/s)	0.23	0.23	0.24	0.23	0.23
Traj. with groundings (%)	22.68	21.85	18.21	22.69	21.00
Traj. with vessel collisions (%)	4.48	2.52	2.80	3.92	2.52
Traj. with buoy collisions (%)	1.68	1.96	1.96	1.68	1.12
Traj. with shore collisions (%)	8.40	8.68	8.40	8.40	8.68

Table 6.7: Evaluation of different feature extractor architectures

	Mean	Std	Min	Max
Avg. position error (m)	387.61	341.75	26.55	2183.02
Avg. course error (°)	7.12	6.42	0.62	43.93
Avg. velocity error ($\frac{m}{s}$)	0.24	0.28	0.02	2.62
End position error (m)	527.95	624.53	8.18	4216.49
End course error (°)	22.51	26.07	0.06	177.41
End velocity error ($\frac{m}{s}$)	0.38	0.39	0.004	3.11

Table 6.8: Navigational performance of NNConv-based BC in validation AIS scenarios

perform at least on expert level. The awareness of shallow waters is improved compared to the vector-based policy, but still very limited.

	Value (%)	Expert Value (%)	Difference
Traj. with groundings	18.21	9.80	+86%
Traj. with vessel collisions	2.80	3.08	−9%
Traj. with shore collisions	8.40	8.12	+3%
Traj. with buoy collisions	1.96	2.24	−12%

Table 6.9: Navigational safety of NNConv-based BC in validation AIS scenarios

A qualitative assessment in the validation scenarios can provide more insight into the COLREG-compliance of the policy. The analysis is focused on the behaviours in overtake, head-on and crossing situations but also covers the awareness of seachart properties.

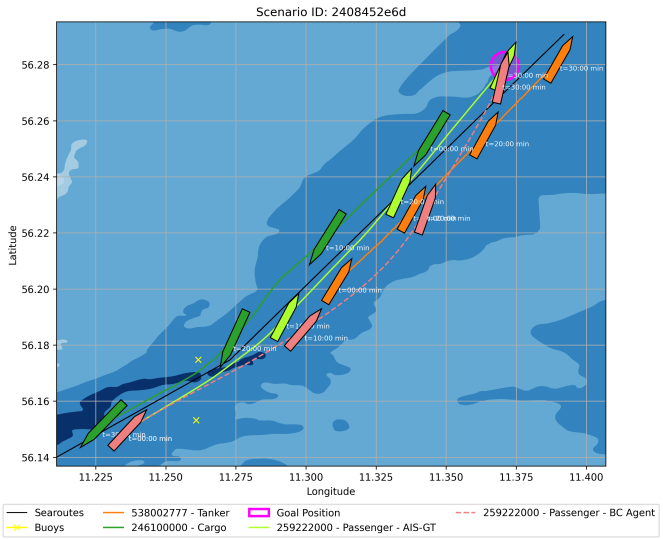


Figure 6.15: Scenario 2408452e6d - NNConv-based BC policy in head-on scenario

First, the waypoint-following behaviour shall be assessed qualitatively. In general the NNConv-based policy shows good waypoint-following capabilities, as it usually steers the OV towards the goal-position as shown in Figure 6.15. However, in some occasions it fails at this task and takes decisions that lead to massive deviations from the goal-position or the OV not even moving in the direction of the goal-position. Figure 6.19 shows such an example.

The policy is capable of initiating overtaking maneuvers. Figure 6.15 shows such a scenario. The policy thereby prefers to overtake on the starboard side of the TV and initiates the overtake on this side. Thereby also maximising the distance to the approaching head-on TV. However, reaching the waypoint is more important to the policy, which is why it crosses behind the to be overtaken TV to reach the waypoint. This behaviour minimises collision risk and can be regarded as COLREG-compliant.

Crossing situations are also important when analysing the COLREG-compliance of the learned policy. Figure 6.16 shows a representative crossing situation where the OV is the give-way vessel. The OV follows the searoute and avoids shallow waters and passes behind the TV. This behaviour complies with the COLREGs and shows good awareness of other navigational features of the scenario.

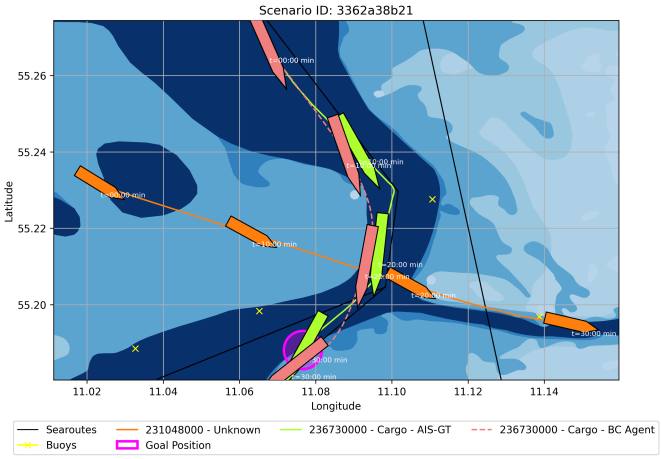
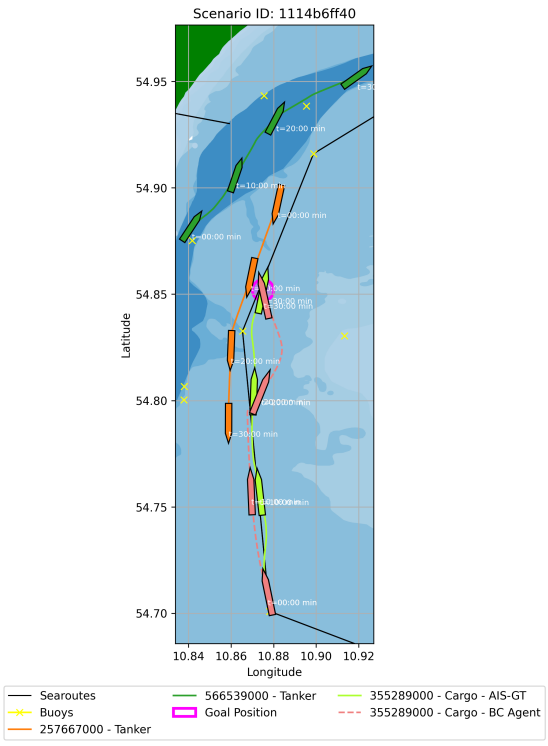
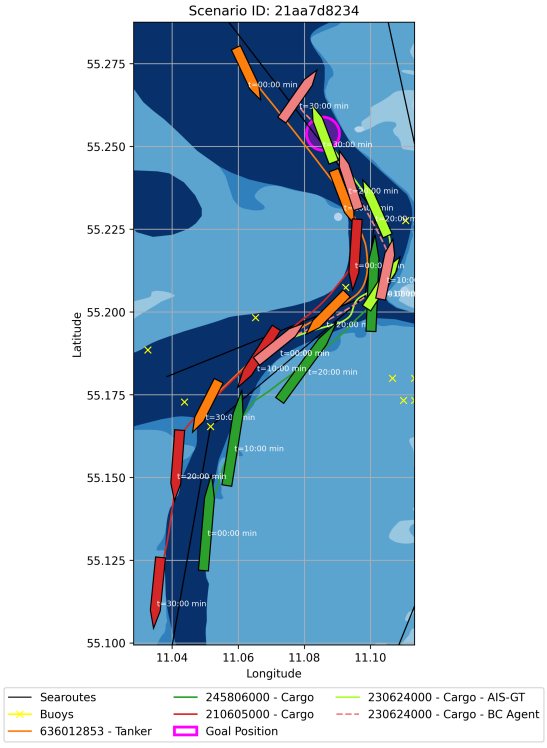


Figure 6.16: Scenario 3362a38b21 - NNConv-based BC policy in crossing scenario

The learned policy performs also well in head-on scenarios. Figure 6.17a shows an example where the OV steers towards the starboard side as the Closest Point of Approach (CPA) with the TV gets closer. After the TV has passed, the policy steers the OV back towards the original path to still reach the waypoint. This example shows that the policy has learned safe and COLREG-compliant behaviour.



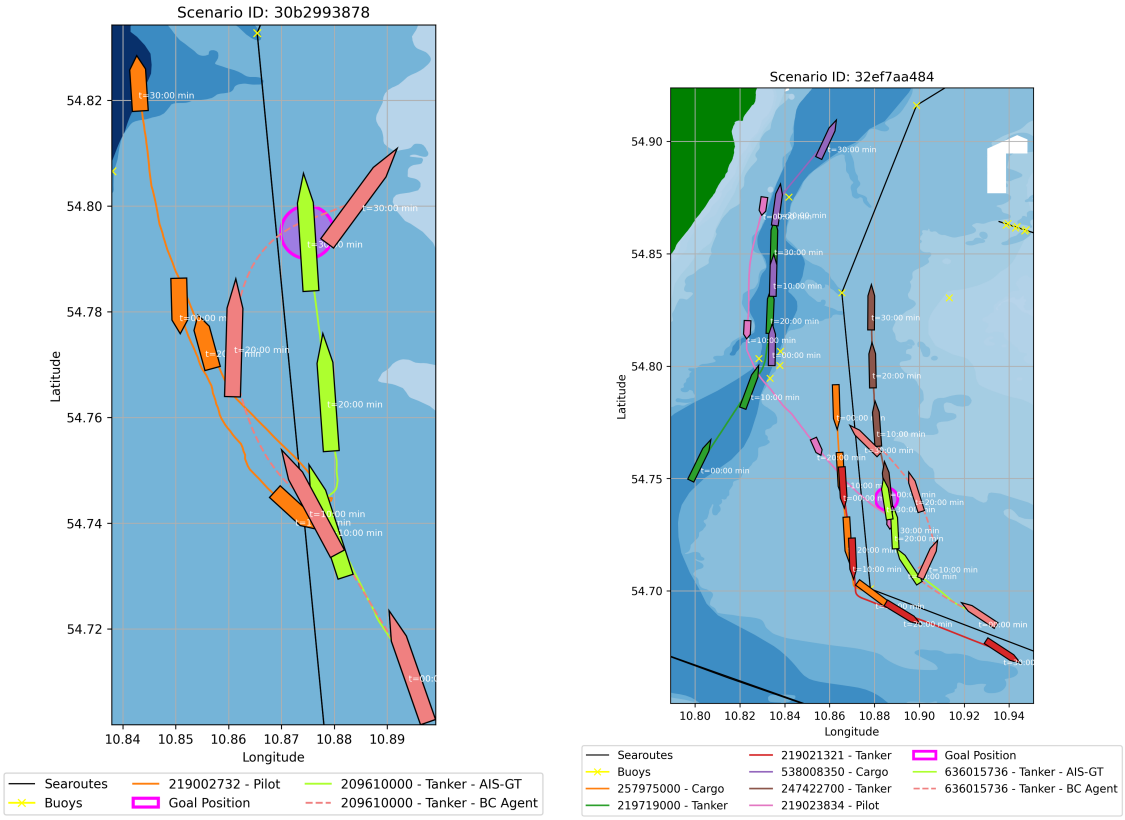
(a) Scenario 1114b6ff40 - NNConv-based BC policy in head-on scenario



(b) Scenario 21aa7d8234 - NNConv-based BC policy in env.-restricted scenario

Figure 6.17: NNConv-based BC policy in head-on and environment-restricted scenarios

In terms of interactions with *pilot* vessels, this policy shows mixed performance. Sometimes it avoids getting close to *pilot* vessels and sometimes it accepts collisions with them. The vector-based policy generally performed better in this aspect as it avoided collisions. The difficulties probably stem from the fact that the policy does not know if a *pilot* vessel



(a) Scenario 30b2993878 - NNConv-based BC policy in pilot-encounter scenario

(b) Scenario 32ef7aa484 - NNConv-based BC policy in env.-restricted scenario

Figure 6.18: NNConv-based BC policy in pilot-encounter & env.-restricted scenarios

wants to approach the OV or another TV. This can be observed in Figure 6.18.

The policies awareness of searoutes, buoys and shallow waters can also be assessed. Figure 6.16 already shows that the learned policy can follow searoutes and avoid shallow waters. Also in scenarios with more TVs the policy manages to successfully avoid shallow waters and follow the marked paths as shown in Figure 6.17b. These observations indicate that the NNConv-based policy has learned how to interact with searoutes, buoys and shallow waters to at least a limited extent.

Unfortunately, the learned policy does not always act according to the COLREGs and does not always obey to shallow waters or searoutes. Figure 6.19 visualises such an example, where the policy gets irritated by the complex searoute and buoy configuration and acts not in a desired way. Such behaviour occurs rarely in the validation scenarios, but in all cases, it seems like the policy is reacting to some environmental factors, mainly buoys, and is not just going straight. It can also not be attributed to the incorrectly inferred actions of Chapter 4.3.4, as it also occurs in situations where the OV has a course vastly different from 0°.

Overall, the NNConv-based policy shows superior performance compared to the vector-based policy of section 6.3.2. In most scenarios it acts COLREG-compliant and shows awareness of searoutes and buoys. However, improvements are still needed as the policy sometimes takes wrong and unplausible decisions that increase the risk of collision and grounding. Possible improvements include expanding the size of the graph-layers to increase the representational power or using different GNN architectures.

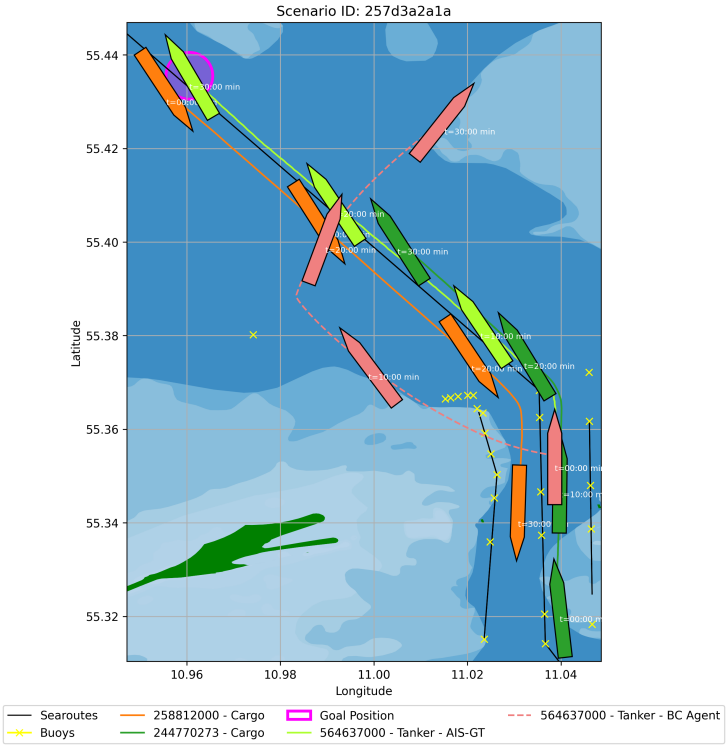


Figure 6.19: Scenario 257d3a2a1a - NNConv-based policy in miscellaneous scenario

For further experiments using AIRL and AVRIL the NNConv-based architecture shall be used for EA-Navigation, as it shows superior performance compared with the vector-based policy.

7 Adversarial Inverse Reinforcement Learning

AIRL is one IRL method to be implemented and evaluated, as explained in Chapter 5. It learns an explicit & interpretable reward function and is compatible with continuous state and action spaces.

7.1 Method

AIRL is an adversarial IRL method, that alternates between learning a reward function R_{AIRL} via a binary discriminator and optimising a policy π_L w.r.t. the learned reward. The schematics of this algorithm are visualised in Figure 7.1. [50]

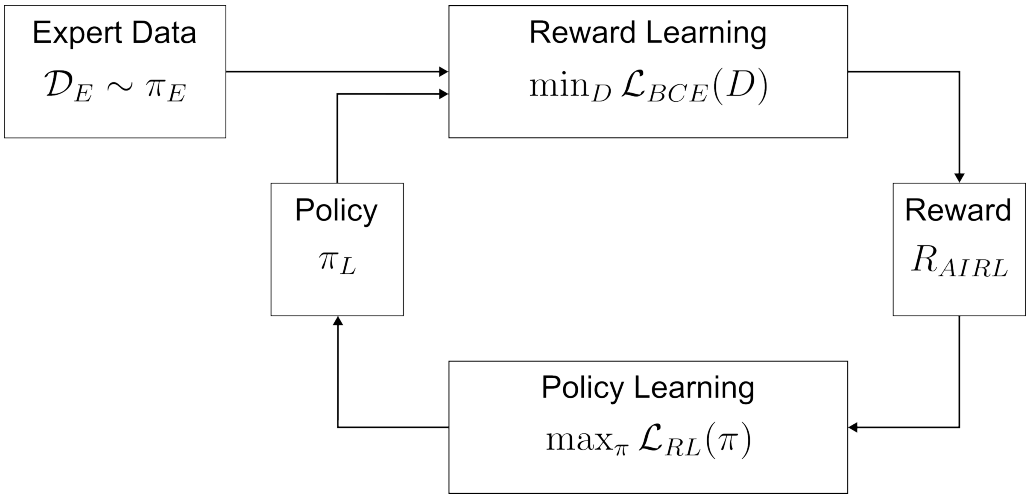


Figure 7.1: Adversarial Inverse Reinforcement Learning

AIRL starts with a dataset of demonstrations $\mathcal{D}_E$ generated by an expert policy π_E. The dataset $\mathcal{D}_E$ is split into three parts, where the *train* split is used for training and the *val* split is used for validation and hyperparameter tuning. [50]

The whole dataset is formally defined by:

$$\mathcal{D}_E = \left\{ \left(O_t^{(i)}, A_t^{(i)} \right) \right\}_{i=1,t=1}^{N_E,T}, \quad \text{where } \left(O_t^{(i)}, A_t^{(i)} \right) \sim \pi_E \tag{7.1}$$

where N_E expresses the number of expert trajectories inside the dataset and T is the fixed length of one trajectory.

An agent with a learner policy π_L must be defined for AIRL. The policy π_L is modelled as non-linear function mapping an observation $O_t^{(i)}$ to an action $A_t^{(i)}$. [50]

As shown in Figure 7.1, the algorithm follows an alternating fashion, where the first step is learning the reward function R_{AIRL}. This is done by sampling expert demonstrations from $\mathcal{D}_E$ and demonstrations from the current learner policy π_L as defined by Equation 7.2, where N_L is the number of learner trajectories. [50]

$$\mathcal{D}_L = \left\{ \left(O_t^{(i)}, A_t^{(i)} \right) \right\}_{i=1,t=1}^{N_L,T}, \quad \text{where } \left(O_t^{(i)}, A_t^{(i)} \right) \sim \pi_L \tag{7.2}$$

Using those two sets of demonstrations, a binary classifier D_ϕ is trained to distinguish between the expert demonstrations and the learner demonstrations. Equation 7.3 shows the definition of the discriminator. The learnable part is the function f_ϕ, which is modelled as non-linear function, a neural network in this case. The learner policy π_L, also being a part of the discriminator definition, is not subject to optimisation in the reward learning step. [50]

$$D_\phi\left(O_t^{(i)}, A_t^{(i)}\right) = \frac{\exp\left(f_\phi\left(O_t^{(i)}, A_t^{(i)}\right)\right)}{\exp\left(f_\phi\left(O_t^{(i)}, A_t^{(i)}\right)\right) + \pi_L\left(A_t^{(i)} \middle| O_t^{(i)}\right)} \tag{7.3}$$

The discriminator D_ϕ is trained by minimising the BCE-loss as shown in Equation 7.4. $\mathcal{D}'_E$ and N'_E are batches of expert demonstrations sampled from the *train* split of the dataset. The intuitive idea is that the discriminator is supposed to differentiate between expert and learner demonstrations to issue high rewards to demonstrations that are similar to ones the expert would have taken. [50]

$$\begin{aligned}\mathcal{L}_{\text{BCE}}(\phi) = &-\frac{1}{|\mathcal{D}'_E|}\sum_{i=1}^{N'_E}\sum_{t=1}^{T}\log D_\phi\left(O_t^{(i)}, A_t^{(i)}\right) \\ &-\frac{1}{|\mathcal{D}_L|}\sum_{j=1}^{N_L}\sum_{t=1}^{T}\log\left(1 - D_\phi\left(O_t^{(j)}, A_t^{(j)}\right)\right)\end{aligned} \tag{7.4}$$

This intuition is embedded in the reward function R_{AIRL}, which can be computed after having trained the discriminator D_ϕ. R_{AIRL} is dependent on the function f_ϕ and the log-probabilities of the current learner policy π_L The reward function is defined by Equation 7.5. [50]

$$R_{\text{AIRL}}\left(O_t^{(i)}, A_t^{(i)}\right) = f_\phi\left(O_t^{(i)}, A_t^{(i)}\right) - \log \pi_L\left(A_t^{(i)} \middle| O_t^{(i)}\right) \tag{7.5}$$

The second part of the AIRL algorithm is the policy learning step, where the learned reward function R_{AIRL} is used to update the policy π_L. This is done by conducting policy optimisation w.r.t. R_{AIRL} using some policy optimisation method. The only requirement for the policy optimisation method is, that it has to train a stochastic policy. Equation 7.6 shows the optimisation objective of the policy learning step. Intuitively, the policy learning step shall train a policy that is able to fool the discriminator by getting close to the experts' behaviour. [50]

$$\pi_L^* = \arg\max_{\pi_L} \mathbb{E}_{\tau\sim\pi_L}\left[\sum_{t=0}^{T}\gamma^t R_{\text{AIRL}}(O_t^{(i)}, A_t^{(i)})\right] \tag{7.6}$$

Both steps, reward learning and policy learning, are repeated interchangeably for a total of N epochs. The result of this adversarial learning process is a reward function R_{AIRL} that is dependent on states & actions and a policy π_L that acts near-optimal w.r.t. R_{AIRL}. The whole process is summarised in Algorithm 1. [50]

Algorithm 1 Adversarial Inverse Reinforcement Learning [50]

Input: Expert data $\mathcal{D}_E$
Output: Learned policy π_L & reward function R_{AIRL}
Initialise policy π_L and discriminator D
for $j = 1, \ldots, N$ **do**
    Sample $\mathcal{D}_L$ using π_L
    Train D by minimising $\mathcal{L}_{\text{BCE}}$ using $\mathcal{D}_L$ and $\mathcal{D}'_E$
    Update the reward function R_{AIRL}
    Update the policy π_L by optimising w.r.t. R_{AIRL}
end for

7.2 Implementation

AIRL is a reasonably complex algorithm. Therefore, a couple of aspects need to be addressed in its implementation.

7.2.1 Reinforcement Learning

The policy optimisation part of the AIRL algorithm requires the implementation of an RL algorithm that optimises the policy π_L w.r.t. the current reward function estimate R_{AIRL}.

Three of the RL algorithms introduced in Chapter 2.2 are suitable for usage in AIRL since they learn a stochastic policy and are compatible with continuous environments. Those are TRPO [41], PPO [47] and SAC [68].

PPO is the ideal candidate for the given problem for two reasons. First, it promises robust learning behaviour in the adversarial setting of AIRL due to the clipping of the policy updates. Second, PPO has already been used in other applications of IRL and RL for decision making in ASN and is said to be the ideal candidate in this context by [38].

A brief introduction to PPO has already been provided in Chapter 2.2, however, to motivate specific design choices a more thorough explanation of the algorithm is needed. PPO is an on-policy actor-critic method which learns a policy by iteratively sampling trajectories using the current policy estimate and updating the policy estimate based on the sampled trajectories. Algorithm 2 shows this procedure. [47]

Algorithm 2 Proximal Policy Optimisation [47]

Input: Reward function R, environment $\mathcal{E}$, initial policy π_{L,θ_0}, initial value function V_{Θ_0}
Output: Learned policy $\pi_{L,\theta}$
for $j = 1, \ldots, N_{\text{EPOCH}}$ **do**
    Sample J trajectories using $\pi_{L,\theta}$ in $\mathcal{E}$
    Estimate advantages δ_t and returns G_t from trajectories
    for $k = 1, \ldots, K_{\text{EPOCH}}$ **do**
        Minimise $\mathcal{L}_{\text{PPO}}(\theta, \Theta)$ w.r.t. θ and Θ using minibatches
    end for
end for

The policy optimisation happens w.r.t. the PPO loss objective $\mathcal{L}_{\text{PPO}}$ shown in Equation 7.7. It is composed of three separate loss functions and is subject to minimisation. The factors c_{VF} and c_{ENTROPY} are hyperparameters that modulate the weight of the value-function and entropy loss components. The optimisation is done using the Adam optimiser, as suggested by [107]. [47]

$$\mathcal{L}_{\text{PPO}}(\theta, \Theta) = -\mathcal{L}_{\text{CLIP}}(\theta) + c_{\text{VF}} \cdot \mathcal{L}_{\text{VF}}(\Theta) - c_{\text{ENT}} \cdot \mathcal{L}_{\text{ENT}}(\theta) \tag{7.7}$$

The first and most important loss objective is the clip loss $\mathcal{L}_{\text{CLIP}}$ which optimises the policy π_L to maximise the obtained rewards (eq. 7.8). For this purpose, it limits the size of the policy updates via a clipping mechanism if the probability ratio between updated and old policy as shown in Equation 7.9 grows too large. [47]

$$\mathcal{L}_{\text{CLIP}}(\theta) = \frac{1}{J \cdot T} \sum_{i=1}^{J} \sum_{t=1}^{T} \text{min} \Bigg(r_\theta \left(A_t^{(i)} \middle| O_t^{(i)} \right) \delta_t \left(O_t^{(i)}, A_t^{(i)} \right), \\ \text{clip} \left(r_\theta \left(A_t^{(i)} \middle| O_t^{(i)} \right), 1-\epsilon, 1+\epsilon \right) \delta_t \left(O_t^{(i)}, A_t^{(i)} \right) \Bigg) \tag{7.8}$$

$$r_\theta \left(A_t^{(i)} \middle| O_t^{(i)} \right) = \frac{\pi_{L,\theta} \left(A_t^{(i)} \middle| O_t^{(i)} \right)}{\pi_{L,\theta_{old}} \left(A_t^{(i)} \middle| O_t^{(i)} \right)} \tag{7.9}$$

Since the algorithm is an actor-critic method, it is required to also learn a value function to estimate the advantages. This is done via a simple MSE objective. [47]

The following equation, where V_Θ is the value function, shows this:

$$\mathcal{L}_{\text{VF}}(\Theta) = \frac{1}{J \cdot T} \sum_{i=1}^{J} \sum_{t=1}^{T} \left\| V_\Theta \left(O_t^{(i)} \right) - G_t \right\|^2 \tag{7.10}$$

Last, an entropy loss can be added to $\mathcal{L}_{\text{PPO}}$ to foster exploration of the learned policy. This regularises the learned policy and prevents overfitting and instability in the AIRL setting. [47]

$H[\pi_{L,\theta}]$ expresses the entropy of the policy as part of the following objective:

$$\mathcal{L}_{\text{ENT}}(\theta) = \frac{1}{J \cdot T} \sum_{i=1}^{J} \sum_{t=1}^{T} H[\pi_{L,\theta}] \left(O_t^{(i)} \right) \tag{7.11}$$

The implementation of the algorithm is done according to [47] and [107] and uses a few additional considerations to improve the stability of the algorithm. Those additional considerations include:

- The actor and critic do not share the same model. Instead, each of them is modelled as separate neural network, since this is supposed to increase performance of the algorithm. [107]
- PPO requires the estimation of advantages. This is done using Generalized Advantage Estimation (GAE) [108] as proposed by [47]. The estimated advantages are also normalised after each sampling step.
- The actor is stochastic and may therefore generate actions that are not valid in the environment. Therefore, the actions are clipped to be inside a valid range. This improves stability of the learning process.

- Another important aspect, which improves the stability of the algorithm, is the global clipping of all gradients. This limits the magnitude of the gradients, thus limiting the magnitude of the updates and improving stability. [107]
- The initialisation of the actor and critic networks is also supposed to have an effect on the algorithms performance. Therefore, all models can use orthogonal initialisation as suggested by [107].
- Another consideration is the option for a separate learning rate for the critic, to improve the value function estimation. This is optional, and if set to `None` the learning rate of the policy is used.

7.2.2 Normalisation

Normalisation is also an important aspect for AIRL. The observations are normalised w.r.t. the training dataset, following the same procedure described for BC in Chapter 6. Furthermore, as already discussed, the advantages are normalised within PPO.

7.2.3 Regularisation

In the adversarial setting of AIRL, regularisation plays a crucial role, since an overfitting discriminator could make the whole learning process collapse. In order to avoid this issue a multitude of regularisation methods has been implemented.

All actor, critic and reward networks include dropout for regularisation. The dropout magnitude can be adapted individually and is subject to hyperparameter optimisation.

Additionally, the Adam optimisers of the discriminator and the actor and critic networks inside PPO are subject to L_2 regularisation to keep the magnitude of the network weights small and promote generalisation.

As already mentioned, the entropy loss $\mathcal{L}_{\text{ENTROPY}}$ regularises the policy π_L and prevents it from exploiting known rewards too early.

Last, the discriminator updates are modified by adding demonstrations from previous iterations to the learner dataset $\mathcal{D}_L$. This replay buffer, suggested by [50], makes the discriminator not overfit to samples from the latest learned policy and allows to learn a more generalised reward function.

7.2.4 Actor & Critic networks

PPO requires the definition of an actor and a critic policy neural network. Since both are, similar to BC, just dependent on the observation, the architectures shown in Chapter 6 can be used. However, the actor and critic require different regressors.

The actor has to use a stochastic regressor, which predicts the mean and log-variance of a Gaussian distribution for each action. The critics output, on the other hand, is modelled as deterministic scalar output value, as it has to predict the value of the current state.

7.2.5 Reward networks

The rewards in AIRL are represented as non-linear functions through the function f_ϕ. This function is dependent on the observation $O_t^{(i)}$ and action $A_t^{(i)}$. The already known neural network architectures shall be used, however they require some small adjustments.

In AIRL the action $A_t^{(i)}$ needs to be handled as additional input to the network. Therefore, it is concatenated to the goal-position features g and handled by its feature extractor. The regressor of this network is deterministic and a single output, since it just has to predict a scalar reward.

7.3 Experiments

A series of experiments is also conducted for AIRL to evaluate its suitability for the GO-Navigation and EA-Navigation tasks. First, AIRL is used in the GO-Navigation task. In a second step, the performance of AIRL in the EA-Navigation task shall be evaluated.

7.3.1 Goal-Only Navigation with AIRL

In a first experiment, AIRL shall be used and evaluated for the GO-Navigation task. In GO-Navigation the input to the policy and reward function is only the goal-position vector g. Based on this, the goal is to learn a policy that guides the OV to the goal-position and a reward function which motivates this behaviour.

The whole experiment for AIRL is set up as a Bayesian hyperparameter optimisation, which optimises w.r.t. the average end position error in all validation scenarios. The lower this error, the better the performance of the policy. A total of 20 configurations is trained in this setting due to computational limitations. Each configuration is trained for 100 epochs using the Adam optimiser [101] for both the discriminator and the policy inside PPO. Additionally, some other hyperparameters have been fixed prior to the hyperparameter search. Those are the batch sizes for discriminator learning and inside PPO, which are set to 2048. Based on the results of the BC experiments the policy network architecture has been set to (256, 128) and uses orthogonal initialisation of the the weights. The replay buffer for discriminator training memorises demonstrations from the last 20 epochs. Some parameters for the PPO algorithm have also been fixed beforehand based on commonly used default values. Those are $\epsilon = 10^{-5}$, $\gamma = 0.99$, $\lambda_{\text{GAE}} = 0.95$ and $K_{\text{EPOCH}} = 10$. The other parameters are subject to optimisation and are listed in Table 7.1.

Symbol	Parameter	Values			
η_{DISC}	Discriminator learning rate	10^{-3}	10^{-4}		
—	Disc. epochs	2	5		
—	Disc. replay buffer coeff.	0	0.25	0.5	
$\lambda_{L_2,DISC}$	Disc. L_2 regularisation	0	10^{-3}	10^{-4}	
—	Disc. oversampling	`true`	`false`		
—	Disc. dropout	0	0.1	0.2	0.3
—	Disc. hidden layers	$(256, 128)$	$(128, 64)$		
N_{EPOCH}	Policy epochs	2	5		
—	Policy dropout	0	0.1	0.2	0.3
ϵ	PPO epsilon	0.1	0.2		
c_{ENT}	PPO entropy coefficient	0	10^{-2}	10^{-3}	
c_{VF}	PPO value coefficient	0.3	0.5	0.7	
J	PPO sampled trajectories	20	40		
η_{POLICY}	PPO learning rate	10^{-3}	10^{-4}		
η_{VALUE}	PPO value learning rate	`None`	10^{-3}	10^{-4}	
$\lambda_{L_2,PPO}$	PPO L_2 regularisation	0	10^{-3}	10^{-4}	

Table 7.1: Hyperparameters for goal-only AIRL

Two sweeps have been conducted due to observations about the policies and the reward functions related to the entropy of a trained policy. Therefore, first a High Entropy (HE) and a Low Entropy (LE) policy are evaluated. In a second step the corresponding reward functions are analysed too.

Policy - High Entropy

The best performing parameter configuration is determined by the minimum mean end position error in all validation scenarios. In the hyperparameter sweep an error of 90.71 m has been achieved using the parameters that are highlighted green in Table 7.1. Analysing the parameters of the five best performing policies trained as part of the hyperparameter optimisation one can draw some more conclusions about suitable parameters. Those reveal that the learning rates inside PPO should be set to 1×10^{-3} for the policy and to 1×10^{-4} for the value function. Additionally, the ϵ parameter of PPO should be set to 0.2. In terms of regularisation the policy should feature a dropout rate of 0.3 and the PPO L_2 regularistaion should be rather small with values of either 0 or 1×10^{-4}. Last, setting N_{EPOCHS} to 5 is another common parameter among well performing policies. No other clear parameter choices can be observed.

As already stated, the best performing model of the hyperparameter search achieves a mean end position error of 90.71 m in the validation scenarios. The standard deviation is also considerably low, indicating that the policy trained using the AIRL method has successfully learned to go to the goal-position. However, there are also some negative aspects. The course error is rather large with a mean of 92.46° and a large standard deviation. This indicates that the learned policy not necessarily ends the trajectory similar to the expert. The velocity error is also large with a mean of 4.95 m s^{-1}, hinting at the policy having learned a different behaviour that achieves GO-Navigation. Table 7.2 shows the detailed metrics.

	Mean	**Std**	**Min**	**Max**
End position error (m)	90.71	58.41	9.31	355.64
End course error (°)	92.46	51.27	0.22	179.95
End velocity error ($\frac{m}{s}$)	4.95	3.38	0.01	16.28

Table 7.2: Navigational performance of goal-only HE-AIRL in validation AIS scenarios

A qualitative assessment of the policy can provide a clearer picture of its performance. Figure 7.2 shows one validation scenario, where the agent successfully manages to reach the goal-position, while also maintaining a path that is similar to the expert and looks natural. Thereby, the OV has a higher surge velocity resulting in it reaching the goal-position sooner. As a result, it decelerates and starts rotating at the goal-location. It succeeds in GO-Navigation, but dissimilar to the experts.

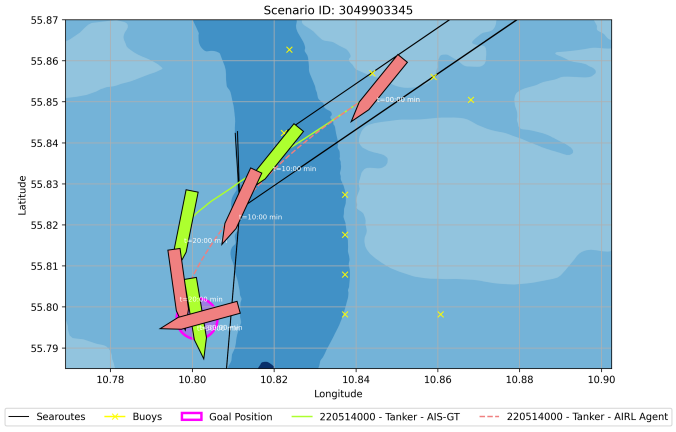


Figure 7.2: Scenario 3049903345 - Goal-only HE-AIRL policy

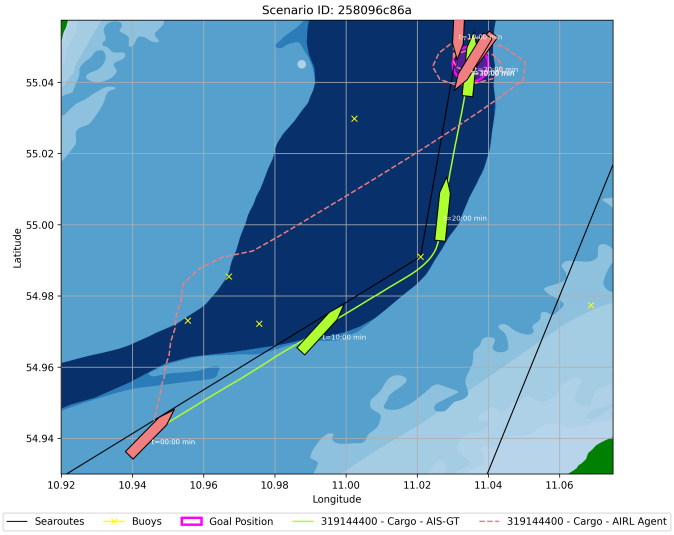
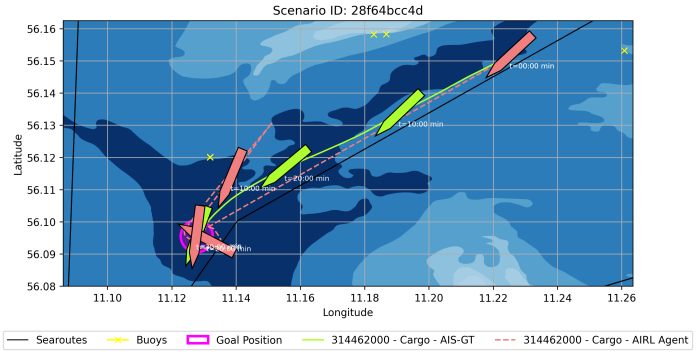


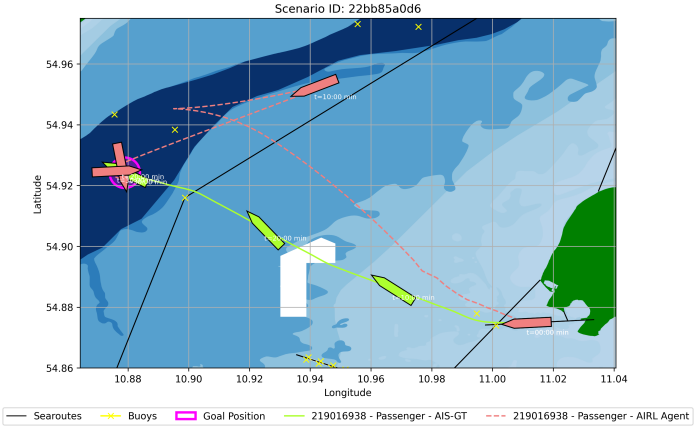
Figure 7.3: Scenario 258096c86a - Goal-only HE-AIRL policy

In most other scenarios this behaviour is even more severe. The OV accelerates to high velocities at the beginning of the trajectory and then decelerates close to the goal-location. While doing so it shows large and fluctuating deviations from the expert trajectory. Upon reaching the goal-location it spins around the goal-location achieving GO-Navigation in a different way to the experts. Figure 7.3 visualises this behaviour in a representative scenario.

In some other scenarios the spinning is combined with a short backward motion, which increases the distance to the goal-position momentarily. Figure 7.4 shows two of these cases, where the backward motion either starts at the goal-location directly or occurs on the way to the goal-location. Still the policy manages to be at the goal-location when the trajectory ends, which is the success criterium.



(a) Scenario 28f64bcc4d - Goal-only HE-AIRL policy



(b) Scenario 22bb85a0d6 - Goal-only HE-AIRL policy

Figure 7.4: Miscellaneous Scenarios - Goal-only HE-AIRL policy

Overall, the policy learned by AIRL successfully achieves moving the OV to the goal-position. It does so in an unnatural way with high accelerations, decelerations and yaw rates. This highly dynamic behaviour is only possible in simulation and would not be applicable in reality. The entropy of the policy was 5.3 at the end of training, which indicates that the policy still acts highly exploratory and dynamic. However, AIRL can, depending on the hyperparameter selection, also learn policies with low entropy that show comparable performance.

Policy - Low Entropy

A low entropy policy has been learned as best performing policy by a second hyperparameter sweep optimising w.r.t. the mean end position error on the validation scenarios. The optimisation was based on the same parameters shown in Table 7.1.

This time the optimisation algorithm found another local optimum in the parameter space. The parameter selection is highlighted blue in Table 7.1. The common parameter choices of the five best performing policies of this sweep are the following: The learning rate of the discriminator was set to 10^{-3}. No other clear trends are observable. In comparison to the HE-policy this policy has been trained using less regularisation as both the dropout rates and the L_2 regularisation of the policy were set to 0. This should allow the policy to optimise its weights better towards the seen data.

The identified policy has an entropy of -0.16, which is considerably lower than the HE-policy trained by the other hyperparameter sweep. The evaluation metrics computed in all validation scenarios do not show large differences to the HE-policy (see Table 7.3). While the end course and velocity errors are in a similar range, the position error is larger. This indicates that the LE-policy is not as successful in waypoint-following as the HE-policy.

	Mean	**Std**	**Min**	**Max**
End position error (m)	243.81	489.80	20.24	4983.28
End course error (°)	80.31	68.12	0.07	179.89
End velocity error ($\frac{m}{s}$)	4.39	3.49	0.001	10.76

Table 7.3: Navigational performance of goal-only LE-AIRL in validation AIS scenarios

Qualitative assessment of the policy can reveal more details about the behavioural traits of this policy. Figure 7.5 shows one representative scenario. The policy steers the OV towards the goal-position by choosing a shorter route than the expert. At the end of the trajectory the OV also reaches the goal-location with good accuracy. Additionally, the imitation performance of this policy is much better. It acts less dynamic by maintaining one consistent trajectory without sudden turning manoeuvrers. This more realistic behaviour can be attributed to the lower entropy, which allows the policy to act with higher certainty in the environment.

When the policy reaches the goal-location sooner than the expert does, it remains at the goal-location and starts rotating at this location. This explains the large course and velocity errors but is not as severe as observed for the HE-policy. Figure 7.6 visualises this behaviour in a representative scenario.

The end position error is larger compared to the HE-policy. This can be attributed to the LE-policies fewer dynamic movements. As a result, it might occasionally happen that the policy does not manage to reach the goal-location in time. In this case a larger position error occurs. Figure 7.7 shows this.

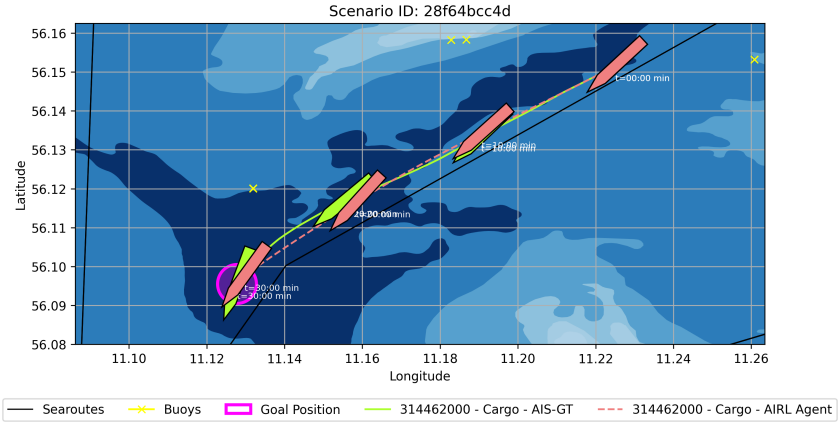


Figure 7.5: Scenario 28f64bcc4d - Goal-only LE-AIRL policy

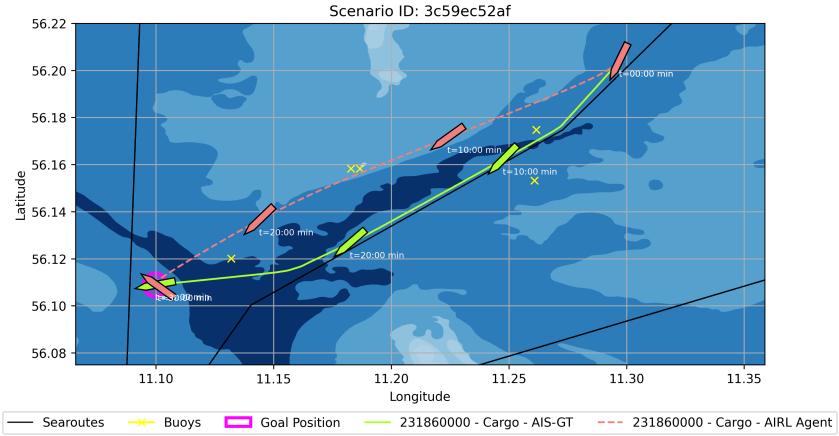


Figure 7.6: Scenario 3c59ec52af - Goal-only LE-AIRL policy

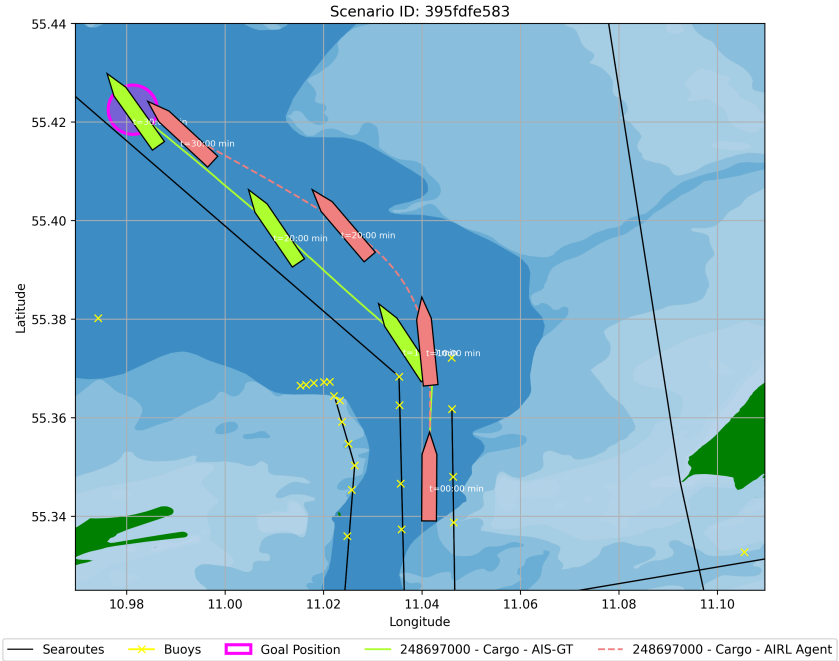


Figure 7.7: Scenario 395fdfe583 - Goal-only LE-AIRL policy

In conclusion, the LE-policy shows good performance in the GO-Navigation task. It especially acts more natural and less dynamic which indicates better imitation performance. Whether a HE or LE-policy is trained by AIRL depends on the selected parameters.

Reward - High Entropy

Next, the reward function associated with both the HE and the LE-policy are analysed and compared. This provides further possibilities for interpretation of the policy and the navigational motives of the experts in the GO-Navigation task.

The reward function linked to the HE-policy is analysed first. Initially, the cumulated rewards of the expert and policy trajectories in all validation scenarios are compared. From Figure 7.8 it is visible that the HE-policy does not act as optimal w.r.t. the reward function R_{AIRL} as the experts do. The experts on average obtain a reward of 607.65 while the learner policy only obtains an average cumulated reward of -433.52. This strongly suggests that the learned policy can be further improved by optimising it for more epochs w.r.t. the latest reward function estimate using a RL algorithm like PPO.

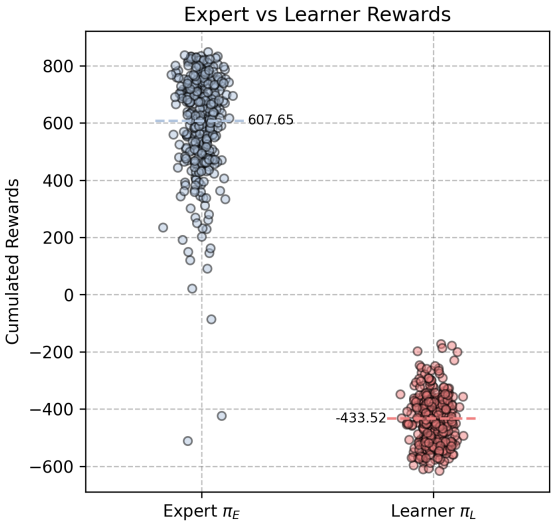


Figure 7.8: Cumulated expert vs. learner rewards under goal-only HE-AIRL reward function in validation AIS scenarios

A more detailed analysis of the reward function can provide further insights. First, the reward function is characterised w.r.t. all its input features. Therefore, Figure 7.9 visualises the gradient-based saliency of all input features of the reward function.

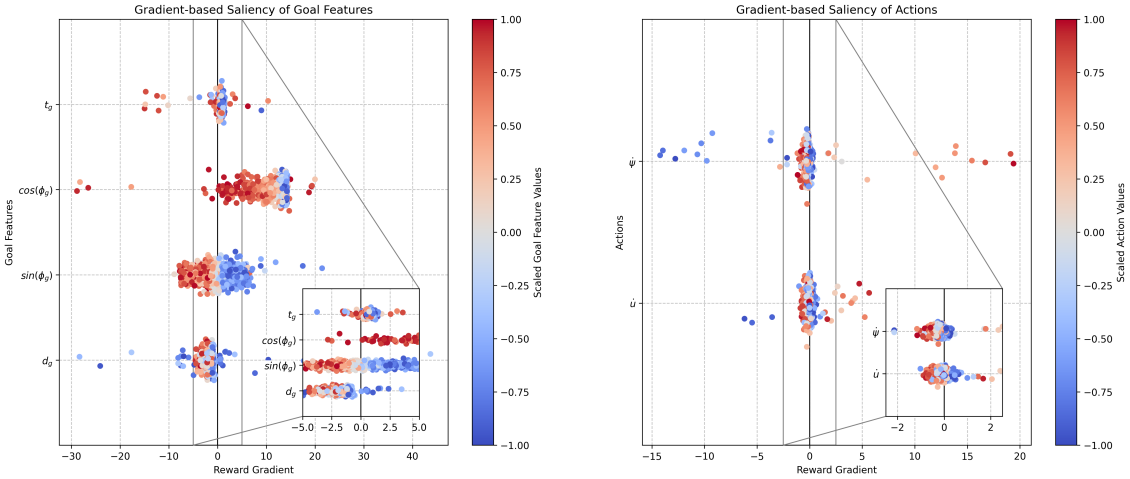


Figure 7.9: Gradient-based saliency of all input features of the HE-AIRL reward function

Figure 7.9 can be interpreted as follows: The x-axis shows the gradient of the reward w.r.t. each input feature. The y-axis indicates the input features to the reward, where the left plot covers all goal-position features and the right plot the two actions. A total of 1000 input features are sampled randomly from the valid feature ranges and the gradient of the reward w.r.t. those input features is computed. The value of a feature is indicated via

the colour coding, where red indicates positive and blue negative feature values. To draw conclusions from this visualisation, one first has to understand what positive and negative feature values correspond to in the navigation task. Then one can assess whether increasing or decreasing specific feature values leads to either an increase or decrease in the reward. [109]

This analysis method leads to the following insights about the GO-Navigation reward function learned by AIRL:

- **Time to goal-position t_g:** The scaled feature values of t_g are smallest when the time to the goal-position is smallest, while large scaled feature values correspond to the vessel still being far from the goal-position in the time domain. Looking at the gradients one can now see, that increasing the scaled feature values from very low feature values in fact increases the reward. In practice this means that the larger the time to the goal-position the higher the reward. However, this trend gets less intensive with higher feature values. If the time to the goal-position grows the gradient is smaller and can even become negative. All this indicates that the reward function issues higher rewards to states that are far from the goal-position in the time compared to states that are close in time. However, the feature is the least important among the goal-position features, as the gradient magnitudes are the smallest.
- **Bearing to goal-position ϕ_g:** The bearing to the goal-position is decomposed into the sine and cosine component. The cosine component indicates whether the goal-position is in front or in the back of the OV. High values indicate the goal-position is ahead and negative values indicate the goal-position is behind the OV. Analysis shows that if the cosine feature value is low, increasing it, leads to higher reward. This matches with the simple intuition, that the reward is higher when the OV is steering towards the goal-position. This trend continues, but the larger the scaled feature value grows, the smaller the gradient gets stopping close to a gradient of 0. This shows that the reward w.r.t. cosine component of the bearing becomes optimal at 0° bearing.
  The sine component of the bearing has positive feature values if the goal-position is on the starboard side of the OV and negative values if the goal-position is on the port side. Accordingly, turning so that the goal-position is not on either the starboard or port side increases the reward. For negative feature values an increase, which corresponds to turning towards the goal-position, leads to higher rewards. For positive values a decrease leads to higher rewards, which also corresponds with turning towards the goal-position. This matches the intuition one would have about an optimal reward function.
- **Distance to goal-position d_g:** The scaled feature values of d_g are positive for large distances and negative for small distances. Figure 7.9 shows that decreasing the distance for large feature values (large distances) increases the reward. For small feature values a decrease generally also increases the reward. However, there are a few outliers for small feature values, where the reward is decreased. All this means that the reward generally grows the closer the goal-location is to the OV. The outliers with positive gradient indicate that there is an intermediate distance at which the reward is maximised. Further analysis needs to confirm this. The overall importance of this feature is smaller compared to the bearing features as the magnitude of the gradients is smaller.
- **Yaw rate r:** Positive scaled yaw rate values indicate turning to the starboard side, while negative values express a turn to the port side. Analysing the gradients of the

reward w.r.t. this feature one can see that increasing the yaw rate for small scaled feature values and decreasing the yaw rate for large scaled feature values generally increases the reward. This corresponds to minimising the absolute yaw rate in order to maximise the reward. This is in line with the provided data, where most vessels have trajectories with very small yaw rates. Some outliers that reverse this relationship are present in the data. Those probably correspond to turning manoeuvrers that increase the reward. The importance of this feature for the reward is very small as the absolute value of most gradients is smaller than 2.

- **Surge acceleration $\dot{u}$:** The surge acceleration is scaled in the same manner as done for the yaw rate. Positive feature values correspond to positive accelerations while negative feature values correspond to decelerations. The conclusion for this feature is, that reducing the absolute acceleration magnitude increases the reward. This is in line with the AIS-scenarios, where most vessels move with a uniform velocity. Some outliers, that reverse this relationship are also present for this feature. These can also be attributed to situations where a larger acceleration or deceleration action is necessary to reach the goal-position in time. Last, the absolute gradient magnitude of this feature is comparable to the yaw rate feature. Therefore, it is also less important to the overall reward.

Figure 7.9 provides an overview over the whole reward landscape. The observations made, can be made more concrete by analysing explicit scenarios. Therefore, each scenario can be represented using a heatmap that visualises the reward obtained by a vessel at a specific location with the specified SOG and COG if it was taking the specified action. The vessel orientation is illustrated in the top left corner of each figure, while the actions are specified in the title of the plot. High reward regions are shown red and low reward regions are shown green. This is called a scenario reward heatmap.

Figure 7.10 uses this visualisation method to analyse the contribution of the goal-position features in more detail, as it shows the scenario reward heatmap for three unscaled values of t_g and a vessel with a SOG of 10 kn & COG of 225° while neither accelerating nor turning.

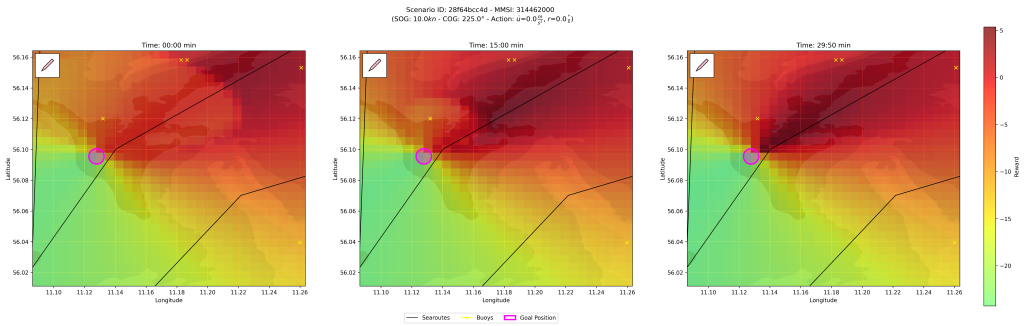


Figure 7.10: Scenario 28f64bcc4d - Scenario reward heatmap of goal-only HE-AIRL for different t_g

This reveals that the high reward regions are along the path towards the goal-location. It shows that the reward grows the closer the goal-location gets, which is in line with the results from the gradient analysis. However, there is a correlation with the t_g feature. If the time to goal-position is still large, the maximum reward is issued not directly at the goal-location. This corresponds to the outliers observed for the d_g feature in the gradient analysis. An effect of the t_g feature on the absolute reward magnitudes cannot be observed, as its importance is also the lowest. Overall, this reward function encourages the policy to learn moving slowly to the goal-location, which is desired to imitate the experts.

The bearing feature ϕ_g can also be further analysed by using the visualisations from Figure 7.11. Here the heatmap is plotted for three different orientations 15 min into the scenario with a SOG of 10 kn and no acceleration or turning.

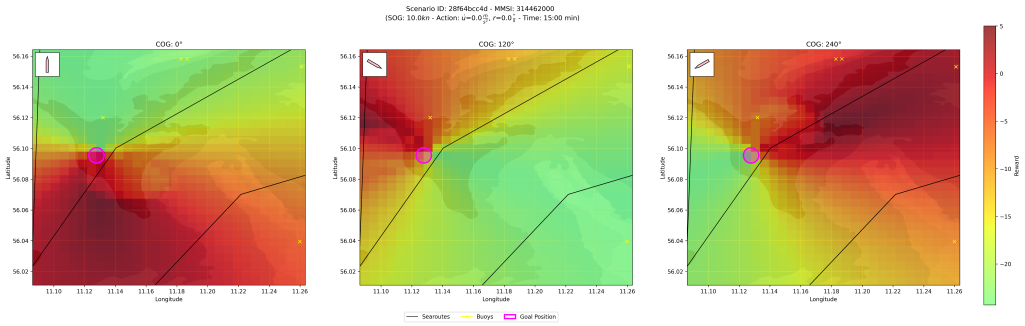


Figure 7.11: Scenario 28f64bcc4d - Scenario reward heatmap of goal-only HE-AIRL for different ϕ_g

Using this visualisation one can clearly see that the reward is higher if the OV is approaching the goal-position and considerably lower whenever the OV is moving away from the goal-position. The reward is also lower when the goal-position is on the starboard or port side of the vessel, which results in the cone like shape of the high reward regions. All this is in line with the conclusions from the gradient analysis. Going into more detail, one can also spot that approaching the goal-position from the OVs starboard side leads to higher rewards. This explains why one observation of the gradient analysis was that turning towards the goal-position might in fact decrease the reward.

On top of these insights, it is worth to also confirm the observations regarding the effect of the actions on the reward. This can be done by assessing the reward w.r.t. the surge acceleration and yaw rate at a specific state in a AIS-scenario. Figure 7.12 shows the action reward heatmap for a representative situation where the OV has a surge velocity of 10 kn, a course of 0° and is 15 min into the scenario.

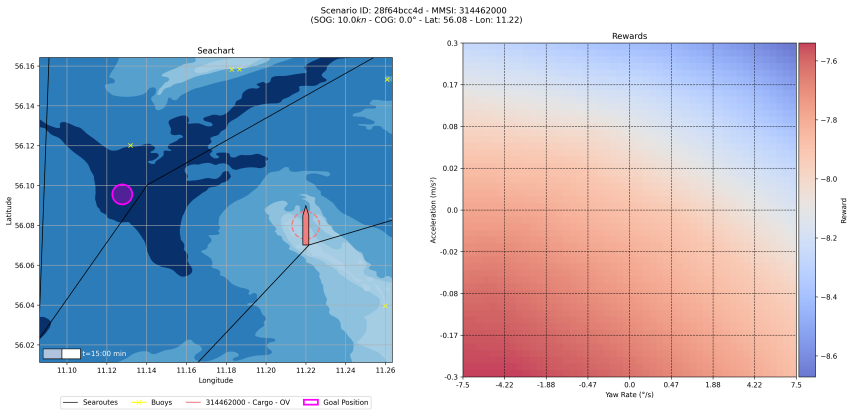


Figure 7.12: Scenario 28f64bcc4d - Action reward heatmap of goal-only HE-AIRL

The figure reveals, that, in this particular situation, decelerating and turning to the port side increases the reward. In combination with Figure 7.11 this is the logical action, as it brings the OV closer to the goal-location. Additionally, it also confirms the observation from the gradient-analysis that turning towards the goal-location increases the reward and that the importance of the action is much lower as can be seen from the scale of the heatmap.

The in-depth analysis of the reward function reveals that AIRL can learn reward functions that correctly encode the task-specific navigational goals of the experts. Despite the policy

imitating not very well and being highly exploratory, the reward function correctly encodes the navigational goals of the GO-Navigation task. Further optimisation of the HE-policy w.r.t. the latest reward function estimate can most likely lead to better imitation performance without losing goal reaching performance but was out of scope for this project. In regard to the analysis tools, the gradient analysis provided insights that are aligned with those gained from the heatmaps. Therefore, both tools are a viable approach for reward function analysis in the maritime domain.

Reward - Low Entropy

The reward function associated to the HE-policy represented the navigational goals very well despite the exploratory policy. Now, it is interesting to analyse, whether the reward function learned jointly with a LE-entropy policy shows any considerable differences.

First, the obtained cumulated rewards of both the expert and the LE-policy w.r.t. the learned reward function are compared. Figure 7.13 visualises this. It shows that the learner policy manages to obtain similar rewards compared to the experts. The experts on average achieve a cumulated reward of 330.89, while the learner policy achieves an average cumulated reward of 352.20. The LE-policy imitates the experts very well according to this observation, especially compared to the large deviation observed in this plot for the HE-policy and reward function.

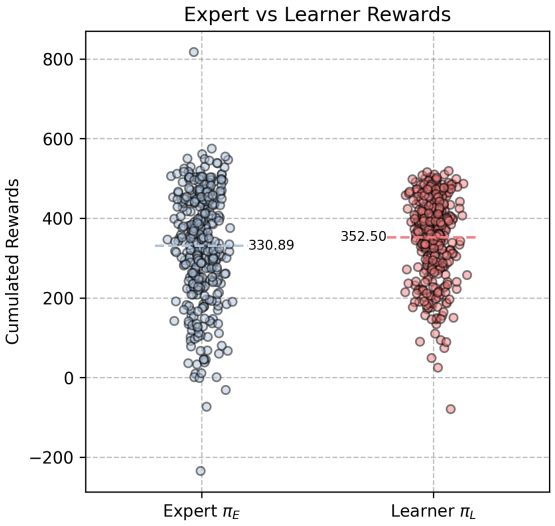


Figure 7.13: Cumulated expert vs. learner rewards under goal-only LE-AIRL reward function in validation AIS scenarios

A more thorough analysis can reveal the differences between the reward functions. For brevity a detailed gradient-analysis is skipped for this reward function. Instead, the easier to interpret and analyse heatmaps are used.

Figure 7.14 visualises the reward function in a scenario, where the OV has a course of 225°, a velocity of 10 kn and is not accelerating or turning. The displayed reward function is different to the one learned by HE-AIRL (see Figure 7.11). The highest reward is issued behind the goal-position, which does not represent the navigational goals of GO-Navigation. Nevertheless, the LE-policy has still maximised the reward. It exploits a local maximum of the reward which is highlighted in Figure 7.14. This local optimum has a similar shape and location compared to the overall structure of the HE-reward function.

A look at the reward relative to the actions also shows differences (see Figure 7.15). In a scenario where the OV is approaching the goal-position with a course of 225° and a velocity of 10 kn the reward is highest if the policy would turn to the port side and decelerate. Keeping the course, which intuitively would be the right action, is only a local maximum of the reward function. Also, the scale of the rewards w.r.t. the actions is much larger, which indicates that the actions have much higher importance to the learned reward function.

One question remains from these observations: Why does the reward function associated with a LE-policy represent the navigational goals in the GO-Navigation task worse? One likely reason for this can be found in the definition of the AIRL reward function and discrim-

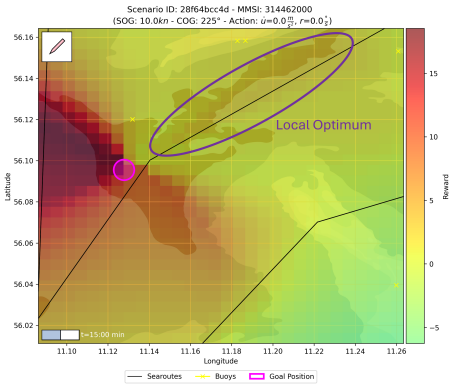


Figure 7.14: Scenario 28f64bcc4d - Scenario reward heatmap of goal-only LE-AIRL

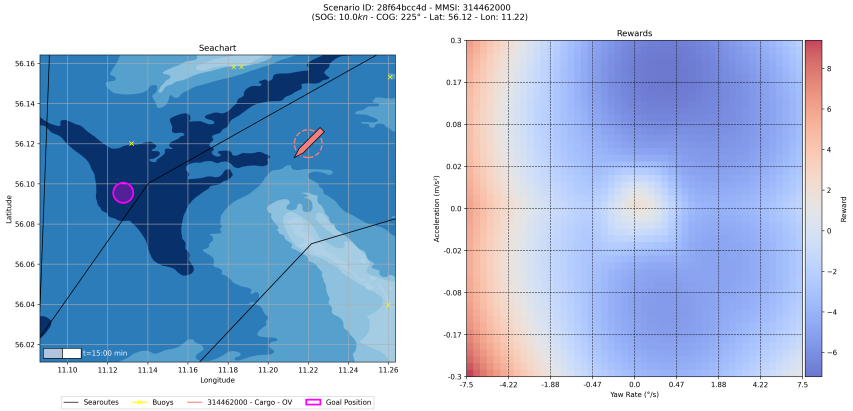


Figure 7.15: Scenario 28f64bcc4d - Action reward heatmap of goal-only LE-AIRL

inator and is associated with the policy having low entropy. According to the LE-policy the actions taken in the high reward regions of the reward function would have low probability as those are not similar to the experts. This lets the log-probability of the policy become large for those states and actions. Additionally, a well trained discriminator would classify those states and actions as expert demonstrations, which would require f_ϕ to become large according to Equation 7.4. In combination with the large log-probability this results in the reward becoming large according to Equation 7.5.

Overall, the reward function associated with a LE-policy models the true navigational goals of the GO-Navigation task only as local maxima of the reward function. The reward function is less interpretable due to the lower entropy of the learned policy.

Conclusion
In conclusion, AIRL shows that well performing and imitating policies and reward functions can be learned. However, there seems to be a trade-off between well imitating policies and interpretable reward functions which is related to the entropy of the policy. Further exploration beyond this report is needed to gain further insights on the decisive parameters for well-imitating and performing policies or well-interpretable reward functions. The incorectly inferred actions of Chapter 4.3.4 did not show an effect in this experiment.

7.3.2 Environment-Aware Navigation with Graph-Based Policy

AIRL shall also be applied to the EA-Navigation task. The goal of a policy in this task is to move towards the goal-position while also taking environmental limitations like other vessels or shallow waters into account. This information is encoded in the graph-based observation O_{GRA}.

The experiment is set up has Bayesian hyperparameter optimisation which optimises the hyperparameters w.r.t. the mean average position error in all validation scenarios. Each configuration is only trained for 50 epochs as computation times would otherwise be too large for the scope of the project. Additionally, the number of sampled trajectories per PPO step J was set to only 10 and the number of epochs per policy and discriminator update were set to 2 for the same reason. Therefore, the results may not show the full potential of the AIRL method for the EA-Navigation-task. Some of the other hyperparameters were also fixed. This includes the majority of the policy, critic and reward neural network architectures. Those are aligned with the best performing architecture of Chapter 6.3.3. This entails the use of NNConv as graph layer with only a linear layer inside of it. The pooling operation was set to *mean*. The graph feature extractors have an architecture of (16, 16), the goal-feature a network architecture of (64, 32) and the output dimension of one feature extractor is set to 16. The regressors of the policy and the critic are aligned in their architecture as they both use an architecture of (128, 128, 64). However, the regressor of the policy is stochastic and the regressor of the critic is deterministic. The batch size in PPO has been set to 512 and in the discriminator learning part to 256 due to memory constraints. Apart from that the following parameters are also fixed throughout the hyperparameter optimisation: $\epsilon = 10^{-5}$, $\gamma = 0.99$, $\lambda_{\text{GAE}} = 0.95$, $K_{EPOCH} = 10$, orthogonal initialisation has been applied, and the replay buffer stores trajectories from the past 20 epochs. The other parameters are subject to optimisation and are listed in Table 7.4.

Symbol	Parameter	Values			
η_{DISC}	Disc. learning rate	10^{-3}	10^{-4}		
—	Disc. replay buffer coeff.	0	0.25	0.5	
$\lambda_{L_2,DISC}$	Disc. L_2 regularisation	0	10^{-3}	10^{-4}	
—	Disc. oversampling	`true`	`false`		
—	Disc. dropout	0	0.1	0.2	0.3
—	Disc. reg. hidden layers	$(128, 128, 64)$		$(64, 64, 32)$	
—	Policy dropout	0	0.1	0.2	0.3
ϵ	PPO epsilon	0.1	0.2		
c_{ENT}	PPO entropy coefficient	0	10^{-2}	10^{-3}	
c_{VF}	PPO value coefficient	0.3	0.5	0.7	
η_{POLICY}	PPO learning rate	10^{-3}	10^{-4}		
η_{VALUE}	PPO value learning rate	`None`	10^{-3}	10^{-4}	
$\lambda_{L_2,PPO}$	PPO L_2 regularisation	0	10^{-3}	10^{-4}	

Table 7.4: Hyperparameters for environment-aware AIRL

Two Bayesian hyperparameter searches have been conducted using the above set of parameters. One used untrained feature extractors, while the second approach used the pretrained feature extractors from the BC experiment of Chapter 6.3.3 to allow for quicker learning due to the computation time limitations. The search without pretrained feature extractors tried a total of 16 configurations, while the search using pretrained feature extractors only used a total of 8 different configurations.

Policy

First, the best policy for the EA-Navigation task has to be identified from the two parameter searches. The most important performance metrics of the best performing policies from both hyperparameter searches are therefore shown in Table 7.5. The policy using pretrained feature extractors performs better in terms of the mean average position error, course error and share of trajectories with groundings and collisions with shore. The policy with untrained feature extractors only performs better in avoiding collisions with vessels and shows a lower mean average velocity error. Preliminary visual inspection of this policy reveals that the policy often moves straight in the validation scenarios without taking any actions. As this is not the case and the majority of the metrics is in favour for the pretrained policy, it is chosen for further analysis.

	Pretrained Feature Extractors	**Untrained Feature Extractors**
Avg. position error (m)	634.88	728.20
Avg. course error (°)	8.75	11.49
Avg. velocity error ($\frac{m}{s}$)	5.67	0.51
Traj. with groundings (%)	19.89	22.13
Traj. with vessel collisions (%)	4.20	2.52
Traj. with buoy collisions (%)	1.96	1.96
Traj. with shore collisions (%)	7.84	8.40

Table 7.5: Evaluation of pretrained vs. untrained environment-aware AIRL

The parameters used for training this policy are highlighted green in Table 7.4. As the number of configurations used by the Bayesian hyperparameter search was small, no insights about other well performing configurations are provided. In fact the chosen hyperparameter selection is likely to be not ideal and further optimisation might lead to better results.

The detailed navigational performance metrics of the chosen environment-aware AIRL-policy are shown in Table 7.6. The mean average position error has a value of 634.88 m, which indicates that the policy is acting well in most validation scenarios. However, the large maximum error of 7402.38 m shows that there must be some scenarios, where the policy is acting vastly different compared to the experts. The mean average velocity error confirms this, as it is large with 5.67 $\mathrm{m\,s^{-1}}$. The maximum error is again very large with 249.98 $\mathrm{m\,s^{-1}}$. As the difference between mean and maximum error is large, there are likely only a few scenarios, where the policy acts different to the experts. The mean average course error is in a range comparable to the previous BC policies. The maximum value is again large supporting the claim that the policy shows bad navigational performance only in a few scenarios.
The mean end position error is also large compared to the GO-Navigation-policies trained by AIRL (see Table 7.3.1). The same applies to the mean end velocity error. Only the end course error is smaller compared to the GO-Navigation policies. However, all three metrics also have very large maximum values, which indicate that only a minority of scenarios is deviating massively from the expert's path. Qualitative analysis will have to assess this.

The navigational safety metrics are analysed next. They reveal that the policy is not aware of shallow waters as it runs aground in 19.80 % of all validation trajectories, which is an in-

	Mean	Std	Min	Max
Avg. position error (m)	634.88	727.56	117.37	7402.38
Avg. course error (°)	8.75	12.49	0.52	86.26
Avg. velocity error ($\frac{m}{s}$)	5.67	32.23	0.06	249.98
End position error (m)	1276.75	1625.03	99.33	19479.26
End course error (°)	19.76	32.03	0.02	178.39
End velocity error ($\frac{m}{s}$)	14.96	72.90	0.001	518.63

Table 7.6: Navigational performance of env.-aware AIRL in validation AIS scenarios

crease of 102 % compared to the experts. Also in terms of collisions with other vessels the policy performs worse than the experts by 36 %. Positive is the performance for collisions with shore and buoys. The policy performs equal to the experts when avoiding collisions with shore and outperforms the experts in avoiding collisions with buoys by 12 %. This shows that the policy has gained awareness of shore and buoys, but the awareness of other vessels and especially shallow waters is very limited.

	Value (%)	Expert Value (%)	Difference
Traj. with groundings	19.80	9.80	+102%
Traj. with vessel collisions	4.20	3.08	+36%
Traj. with shore collisions	7.84	8.12	−3%
Traj. with buoy collisions	1.96	2.24	−12%

Table 7.7: Navigational safety of environment-aware AIRL in validation AIS scenarios

A qualitative analysis of representative validation scenarios will provide more insights into the learned behaviours. The COLREGs and the awareness of navigational features like shore or other vessels are of interest in this analysis.

First, the reason for the large maximum errors can be identified. These errors occur as the policy applies high accelerations and yaw rates in a few scenarios. As a result, the OV starts circular motions with high speeds, which is clearly not possible in reality. Figure 7.16 shows such a scenario. This behaviour most likely originates from the fact that the policy still acts exploratory with an entropy of 3.5 at the end of training. These behaviours are likely to be less severe, if the policy would be trained further.

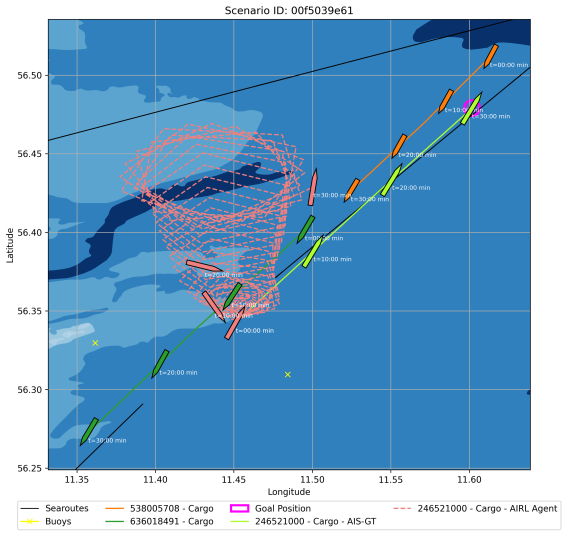


Figure 7.16: Scenario 00f5039e61 - Environment-Aware AIRL policy shows spinning

In the majority of the validation scenarios the learned policy does not act this dynamic. In fact, it has learned decent waypoint-following behaviour as shown in Figures 7.19a and 7.19b. In all, but the previously men-

tioned dynamic scenarios, the policy steers the OV towards the goal-position. However, due to earlier decisions along the trajectory the OV does not always get very close to the goal-position or exhibits a different course at the goal-location.

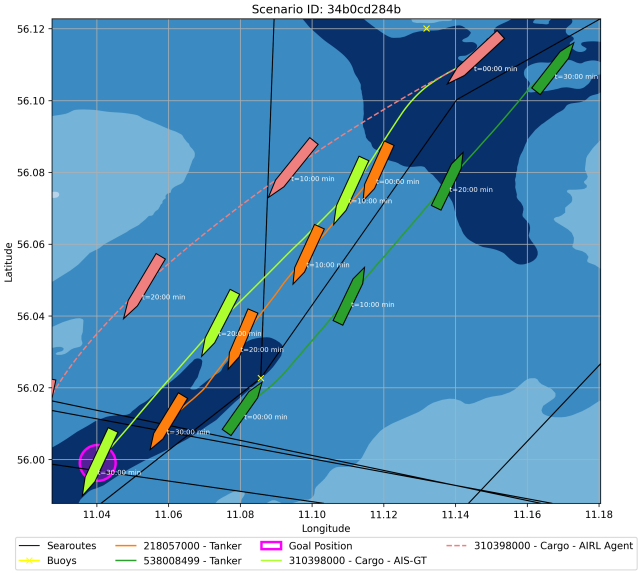


Figure 7.17: Scenario 34b0cd284b - Environment-aware AIRL policy in overtake scenario

In dedicated overtaking situations the policy generally shows good performance as it manages to overtake the TV with a safe distance. Figure 7.17 shows such an example where the OV overtakes on the starboard side of the TV while also maintaining a safe distance to both TVs in the scenario. As the distance is large, it does not exactly reach the goal-location.

Head-On scenarios are not mastered successfully by the policy. Figure 7.18 shows a situation where the OV does not turn towards the starboard side to avoid collision. Instead, it keeps its course and collides with the upcoming TV. This behaviour is dangerous and not COLREG-compliant.

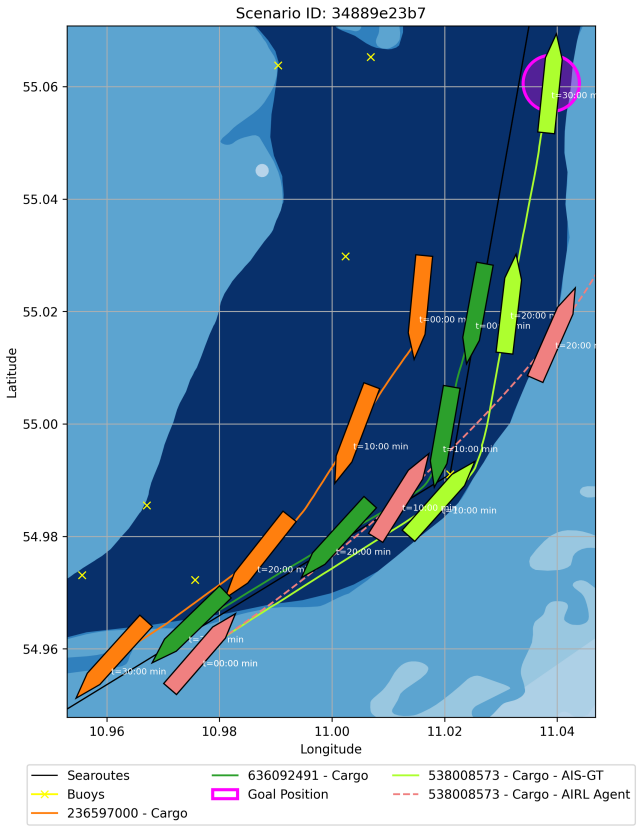
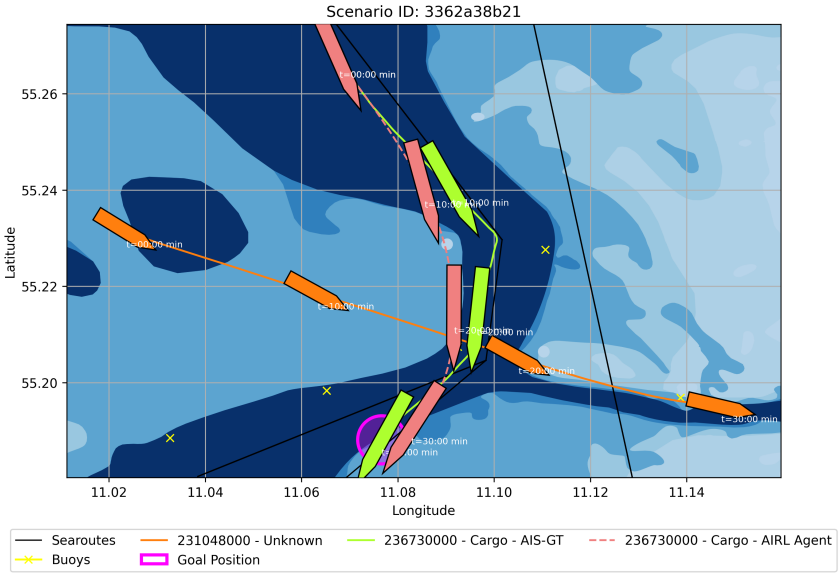
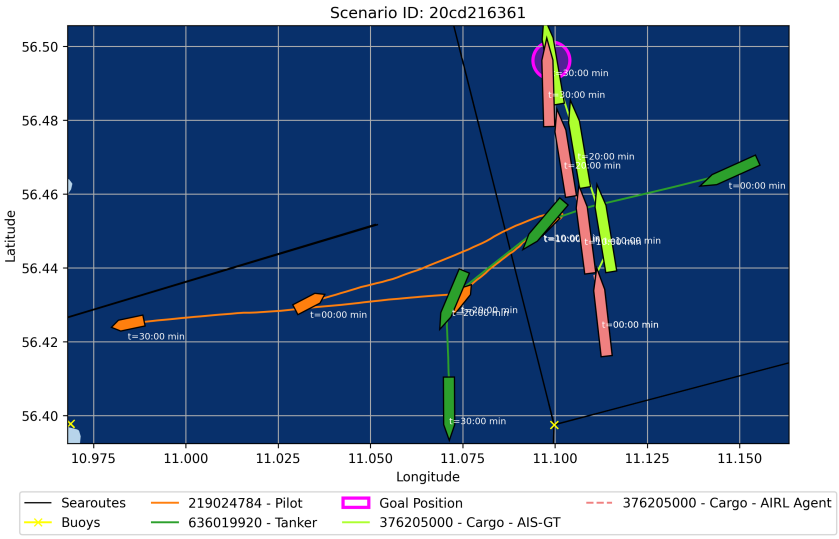


Figure 7.18: Scenario 34889e23b7 - Environment-aware AIRL policy in head-on scenario

In crossing scenarios, the policy shows a mixed performance. While it manages to comply with the COLREGs in some situations it fails to do so in others. Figure 7.19a shows a situation where the policy almost follows the deeper waters and correctly crosses the trajectory of the TV behind it. Figure 7.19b, on the other hand, does not show ideal behaviour. The OV does not collide, but it also does not show any reaction to maximise the Distance to Closest Point of Approach (DCPA) by turning to the starboard side.



(a) Scenario 3362a38b21 - Environment-aware AIRL-policy in crossing scenario

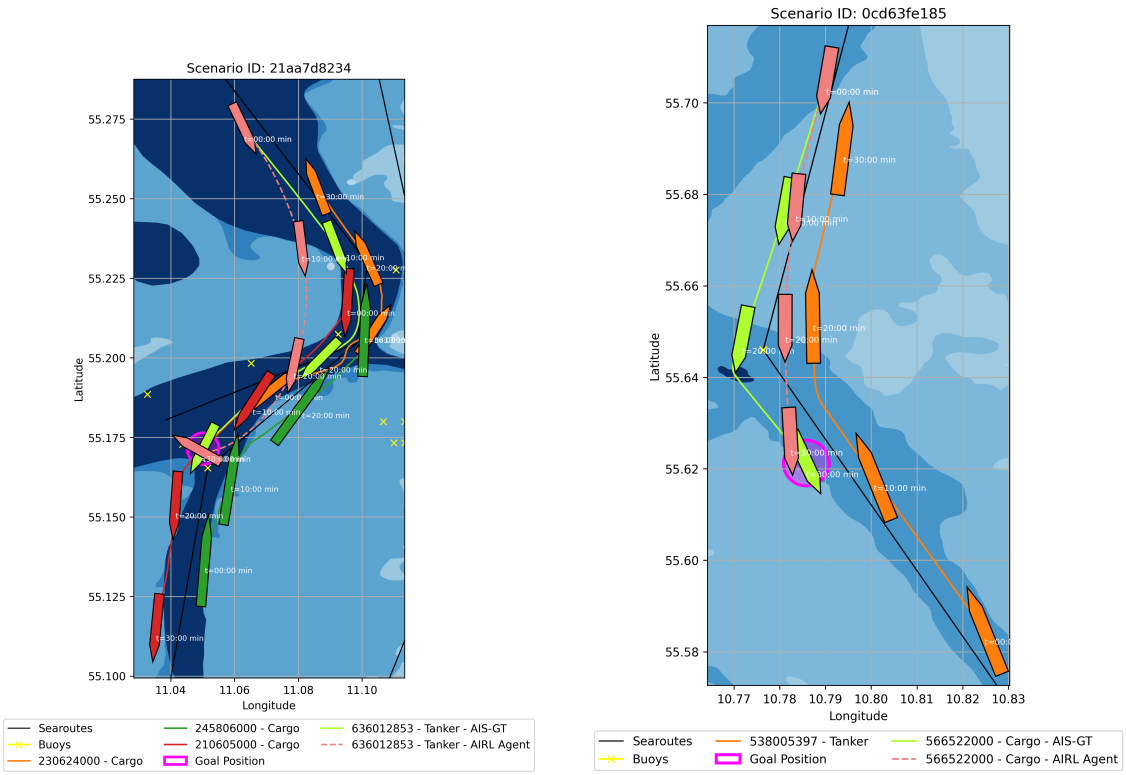


(b) Scenario 20cd216361 - Environment-aware AIRL-policy in crossing scenario

Figure 7.19: Environment-aware AIRL policy in crossing scenarios

The policy does not seem to have good awareness of other features of the maritime environment. Figure 7.20a shows that it is not aware of the shallower waters, the buoys and the searoutes to effectively take a shortcut to the goal-position. Another example is shown in Figure 7.20b where the policy is unable to steer the vessel around the buoy and along the searoutes. Instead, it takes a more direct route.

When a *pilot* vessel approaches the OV the policy does not show any avoiding action.



(a) Scenario 21aa7d8234 - Environment-aware AIRL-policy in env.-restricted scenario

(b) Scenario 0cd63fe185 - Environment-aware AIRL-policy in env.-restricted scenario

Figure 7.20: Environment-aware AIRL-policy in environment-restricted scenarios

It keeps its course and proceeds with its movements as if the *pilot* vessel would not be inside the scenario. This further stresses that the learned policy has limited awareness of its surroundings. Figure 7.19b shows an example of this behaviour.

Overall, the policy learned by AIRL in this experiment does not show good performance. It is not aware of important features like searoutes or shallow waters. It also does not show signs of COLREG-compliant behaviour in all scenarios. On top of that, it occasionally exhibits unrealistic spinning motions. However, AIRL-policies may perform better if more training time was available and hyperparameters would be optimised more thoroughly.

Reward

In addition to the policy, AIRL also learned a reward function that should ideally encode the navigational goals of the EA-Navigation task. This reward function shall be analysed in this section.

First, the performance of the policy w.r.t. the learned reward function is analysed. Figure 7.21 shows the comparison between the obtained cumulated rewards of the trained AIRL-policy and the expert trajectories. It shows that the experts on average obtain higher cumulated rewards with 481.53 compared to 325.04. Still, when assessing the whole distribution, the AIRL-policy acts very similar to the experts according to the reward function. The outliers of the obtained rewards of the AIRL-policy are probably related to the scenarios, where the policy starts the spinning movements.

Besides this, the reward function can also be analysed in depth. The analysis focuses on the COLREGs and the awareness of environmental features like buoys, searoutes or shallow waters. No explicit attention will be given to reward function properties explaining

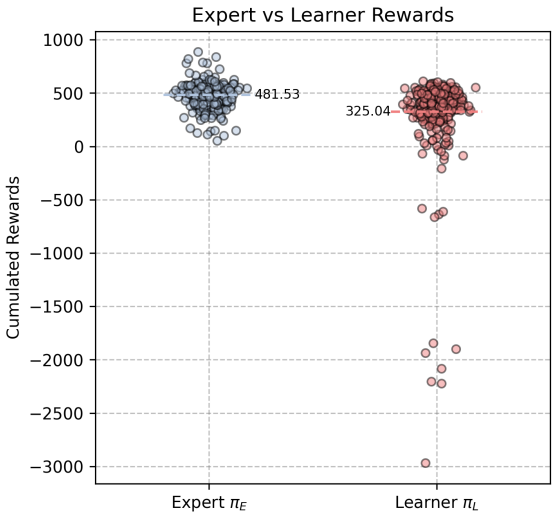


Figure 7.21: Cumulated expert vs. learner rewards under environment-aware AIRL reward function in validation AIS scenarios

the waypoint-following behaviour, as this has already been covered by the dedicated experiment in Chapter 7.3.1. For the analysis, the already introduced heatmaps are used as the main analysis tools. A gradient-analysis is not conducted, because the visualisation of the gradients of all graph inputs features would be unfeasible and this analysis method would not explicitly target the encoding of the COLREGs in the reward function.

First, the awareness of features like buoys, searoutes, shallow waters, land and other vessels is assessed. Therefore, Figure 7.22 visualises the heatmap in a scenario where all these features are present. The OV is travelling with a speed of 10 kn and a course of 0° while taking no actions. The learned reward function is mainly focused on proximity to the goal-position, as this is rewarded the most. In terms of other maritime features, these are only relevant to the reward function with sufficient distance to the goal-location. On the bottom right corner of the scenario, the deeper water area along the searoutes and the buoys is a lower reward region compared to the surrounding shallow waters. This relation is not intuitive and should be reversed to comply with same navigational behaviours.

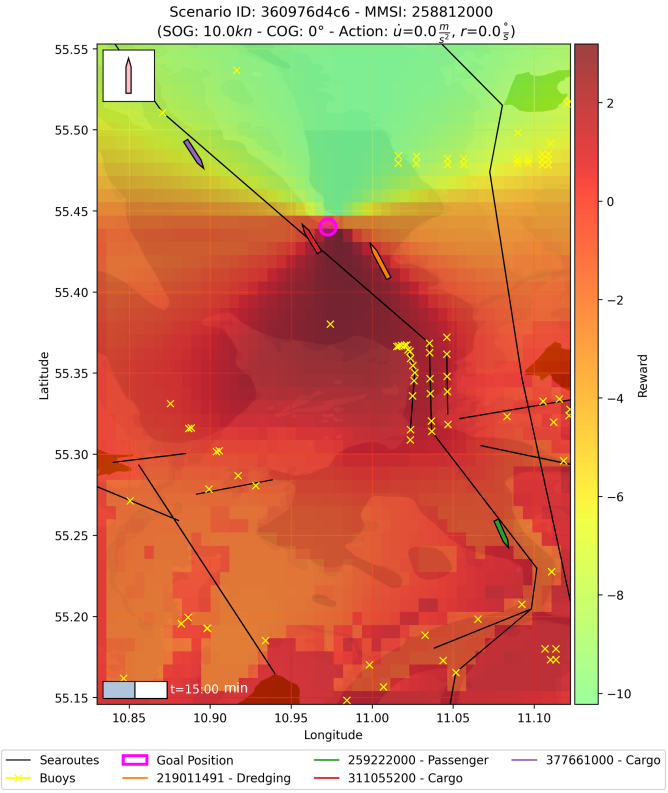


Figure 7.22: Scenario 3609764d4c6 - Scenario reward heatmap of env.-aware AIRL

This scenario can also be used to

assess the reward landscape relative to all possible actions. Figure 7.23 shows this for the same scenario and the OV moving with the same parameters. The figure shows that the reward function rewards large yaw rates towards the port side the most, which would in this situation lead to unsafe and unintuitive behaviour.

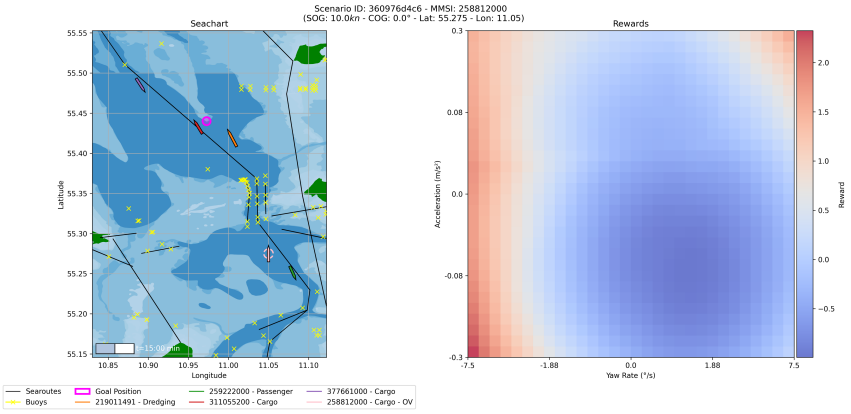
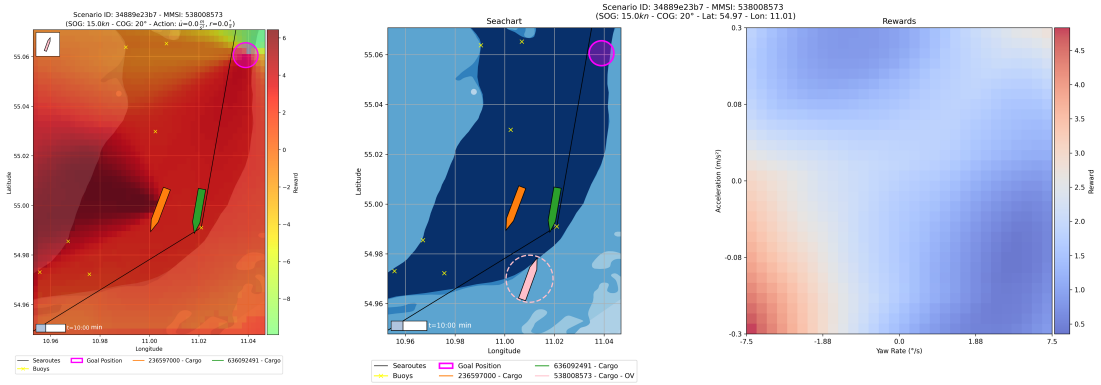


Figure 7.23: Scenario 3052e0a031 - Action reward heatmap of env.-aware AIRL

Next is the analysis of the reward function in COLREG relevant situation. Figure 7.24 shows a head-on scenario where the OV has a course of 20° and a velocity of 15 kn while not accelerating or turning. The reward function is again focused on reaching the goal-position. However, the highest rewards are not issued on the direct path to the goal-position, which would result in a head-on collision. Instead, the highest reward can be obtained by the OV if it would pass the two TVs on its starboard side. This way the reward function promotes collision avoiding but also not COLREG-compliant behaviour. With respect to the actions the reward function also encourages turns to the port side and decelerations, which potentially could avoid a collision.



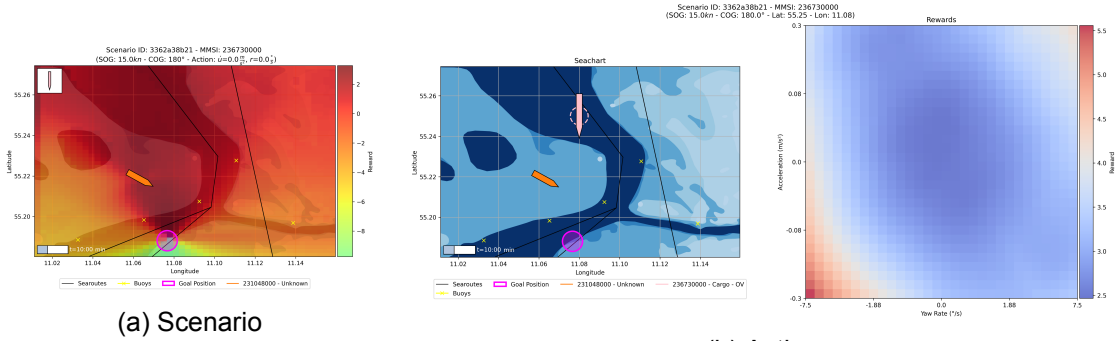
(a) Scenario

(b) Action

Figure 7.24: Scenario 34889e23b7 - Reward heatmaps of environment-aware AIRL in head-on scenario

Next, in Figure 7.25, a crossing situation is visualised. The OV is moving from north to south with a velocity of 15 kn applying no actions. The reward function does again mainly issue high rewards based on the proximity to the goal-location. Shallow waters, searoutes or buoys are not considered by the reward function. However, the TV has a small influence on the reward function. Upon closer inspection once can see in Figure 7.25a that there is a small local minimum of the reward directly in front of the TV. Additionally, the area behind the TV, as seen from the OV, is also associated with smaller rewards. The reward function can identify TVs even though their influence is small. The heatmap relative to

the actions shows that collision avoiding actions generate higher rewards. Turning to the port side and decelerating as well as accelerating and turning to the starbaord side are the two maxima of the reward function in this situation, although only the later would be COLREG-compliant.

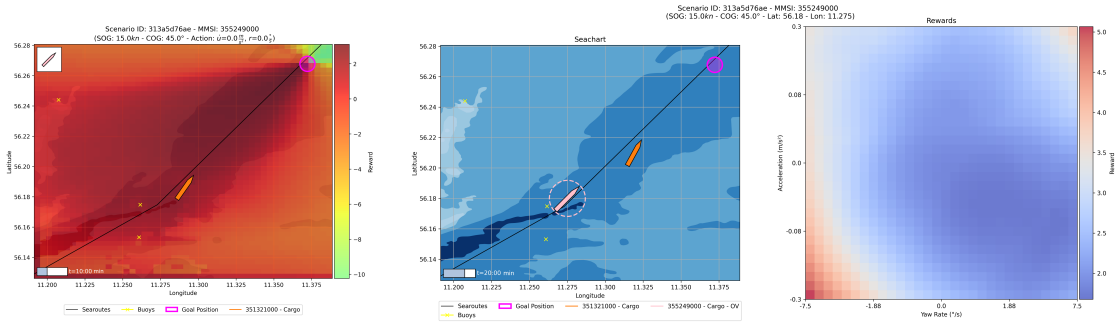


(a) Scenario

(b) Action

Figure 7.25: Scenario 3362a38b21 - Reward heatmaps of environment-aware AIRL in crossing scenario

Last, an overtake scenario is shown in Figure 7.26. In this setting the OV is moving with a velocity of 15 kn and a course of 45° towards the goal-location, while also taking no actions. In this setting the reward function is also mainly influenced by the proximity to the goal-location. Interesting is the fact, that higher rewards are issued on the port side of the TV. However, the TV itself is still in a high reward region, with only minor artifacts showing a reduction in reward. The reward function visualised w.r.t. the actions at a later stage of the scenario shows that either decelerating and turning to the starboard side or accelerating and turning to the port side would be optimal. Both would be safe and COLREG-compliant options.



(a) Scenario

(b) Action

Figure 7.26: Scenario 3362a38b21 - Reward heatmaps of environment-aware AIRL in overtake scenario

The identified optimal actions, according to the reward function, in all three COLREG-scenarios are similar. A reason for this could indeed be, that the reward function in general incentives large actions, which would lead to unstable behaviour if a policy would be optimal w.r.t. the reward function. Resolving the computational restrictions of this experiment could potentially lead to a resolution of this observation.

Overall, the learned reward function is very focused on the proximity to the goal-location. Other environmental factors influence the reward function only a little. In combination with high rewards for large actions, this suggests that further training beyond the computational

limitations of this experiments is needed to learn a reward function that correctly and reliably encodes the COLREGs and environmental features.

Conclusion

In conclusion, AIRL was not able to learn a COLREG-compliant and environment-aware policy for the EA-Navigation task and a reward-function that fully explains the navigational goals of the experts. While the policy is unaware of its environment and does not reliably succeed in COLREG-relevant situations, the reward function is mainly focused on motives for waypoint-following and attributes only minor reward patterns to environmental features. Thereby, the policy and reward function are aligned with each other. Nevertheless, AIRL provides potential for further improvements as this experiment was restricted in computation times. The incorrectly inferred actions of Chapter 4.3.4 did not exhibit a significant effect on the shown policy and reward function.

8 Approximate Variational Reward Imitation Learning

AVRIL is the second IRL algorithm to be evaluated in the context of ASN. It learns a policy and a reward distribution using discrete action spaces.

8.1 Method

AVRIL is a Bayesian IRL approach, as it not only learns a stochastic policy but also a reward distribution. The overall concept of this algorithm is visualised in Figure 8.1. [87]

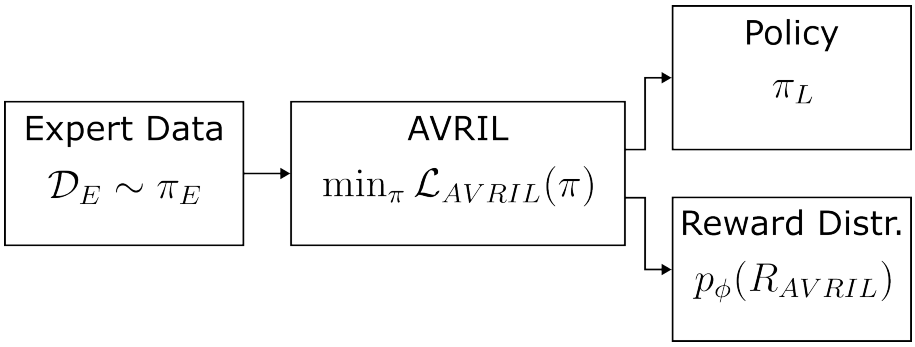


Figure 8.1: Approximate Variational Reward Imitation Learning

Similar to the other two approaches, AVRIL starts with a dataset of expert demonstrations $\mathcal{D}_E$. However, in this setting the dataset consists of tuples that include the demonstration at the current and at the next time step. This dataset is also split into a *training*, *validation* and *test* split, from which the training set is used to learn a policy and the validation set is used for evaluation and hyperparameter optimisation. [87]

The entire expert dataset is formally defined as:

$$\mathcal{D}_E = \left\{(O_t^{(i)}, A_t^{(i)}, O_{t+1}^{(i)}, A_{t+1}^{(i)}\right\}_{i=1,t=1}^{N_E,T}, \quad \text{where } \left(O_t^{(i)}, A_t^{(i)}, O_{t+1}^{(i)}, A_{t+1}^{(i)}\right) \sim \pi_E \tag{8.1}$$

The method requires two neural networks. The first is the variational reward distribution p_ϕ, which learns a distribution of rewards R_{AVRIL} based on a given observation $O_t^{(i)}$ and action $A_t^{(i)}$. The reward distribution is expressed as Gaussian normal distribution with learnable mean and variance. The second is the q-function Q_θ which predicts the state-action value associated with taking an action $A_t^{(i)}$ in a state characterised by observation $O_t^{(i)}$. The policy π_L can be derived from the learned q-value function using softmax over q-values. To be computationally feasible, this requires the action space to be discrete.

Both function approximators are optimised jointly by minimising the objective shown in Equation 8.2. This objective is a ELBO which is optimised via theAdam optimiser [101].

$$\mathcal{L}_{\text{AVRIL}}(\theta,\phi) = \frac{1}{|\mathcal{D}_E|}\sum_{i=1}^{N}\sum_{t=1}^{T}\Bigg(-\log \frac{\exp\left(\beta_{\text{AVRIL}} Q_\theta\left(O_t^{(i)}, A_t^{(i)}\right)\right)}{\sum_{B_t\in\mathcal{A}}\exp\left(\beta_{\text{AVRIL}} Q_\theta\left(O_t^{(i)}, B_t^{(i)}\right)\right)} + \alpha_{\text{AVRIL}} D_{KL}\left(p_\phi\left(R_{\text{AVRIL}}\left(O_t^{(i)}, A_t^{(i)}\right)\right) \middle\| \mathcal{N}\left(R_{\text{AVRIL}}\left(O_t^{(i)}, A_t^{(i)}\right)\right)\right) - \lambda_{\text{AVRIL}} \log p_\phi\left(Q_\theta\left(O_t^{(i)}, A_t^{(i)}\right) - \gamma Q_\theta\left(O_{t+1}^{(i)}, A_{t+1}^{(i)}\right)\right)\Bigg) \tag{8.2}$$

The first term of $\mathcal{L}_{\text{AVRIL}}$ computes the log-probability of the softmax policy, where β_{AVRIL} is a configurable temperature parameter that modulates the randomness of the softmax distribution. Intuitively, this term expresses that the likelihood of the demonstration shall be maximised.
The second term is the KL-divergence between the learned variational reward distribution and a prior. The prior is modelled as Gaussian normal distribution with zero mean and a standard deviation of one. This term is subject to minimisation and encourages the reward distribution to stay close to the prior distribution. The KL-divergence is also multiplied with a weight factor α_{AVRIL} that models its importance within the loss function. If this term's weight in the overall loss function is too large, the reward distribution will collapse towards the prior and will contain no meaningful reward information.
The third term describes the reward learning objective. It is learned by maximising the log probability of the estimated reward obtained by moving one step in the environment. This is estimated by $Q_\theta(O_t^{(i)}, A_t^{(i)}) - \gamma Q_\theta(O_{t+1}^{(i)}, A_{t+1}^{(i)})$ which includes the discount factor γ. By maximising this term, the q-function and reward are getting aligned. The strength of this term can be configured via the weight λ_{AVRIL}.

This approach to policy and reward learning is fully offline as it does not require to sample new demonstrations with an intermediate policy. Algorithm 3 summarises the procedure.

Algorithm 3 Approximate Variational Reward Imitation Learning [87]

1: **Input:** Expert data $\mathcal{D}_E$
2: **Output:** Learned policy π_L & Distribution of rewards $p_\phi(R_{\text{AVRIL}})$
3: Initialise Q-function Q_θ & variational reward distribution Q_θ
4: **for** $j = 1, \ldots, N_{\text{AVRIL}}$ **do**
5: Train Q_θ and p_ϕ by minimising $\mathcal{L}_{\text{AVRIL}}$ using $\mathcal{D}_E$
6: Update the policy π_L by computing arg max of Q_θ
7: **end for**

8.2 Implementation

The implementation of AVRIL is less complex in comparison to AIRL. However, a few points still need to be addressed.

8.2.1 Scaling & Discretisation

Since neural networks are used as function approximators for the reward distribution p_ϕ and the q-function Q_θ the observations and actions need to be scaled. This follows the definitions introduced in Chapter 6.

Additionally, the actions need to be discretised. This is done by extracting n equidistant action values across the whole scaled action range. This is done for both the acceleration $\dot{u}$ and the yaw rate r resulting in a total of n^2 discrete actions. Each continuous action

is then mapped to the closest discretised action. The parameter n is subject to hyperparameter tuning but should be an odd number to include a neutral action (acceleration and yaw rate are 0) in the discretised action space.

8.2.2 Normalisation

The observations also need to be normalised in AVRIL. This follows the definitions introduced in Chapter 6.

8.2.3 Regularisation

Regularisation is also an important part of AVRIL in order to train generalised policies and reward functions.

The main regularisation of this algorithm is embedded into the loss function. The KL-divergence ensures that the reward distribution is not deviating too far from the prior distribution. However, if this term is too big, the reward distribution will just collapse to a standard normal distribution with zero mean and a standard deviation of one, which results in a not informative reward. The parameter α_{AVRIL} can modulate the strength of the KL-divergence term and lead to meaningful policies and reward distributions.

Additionally, both models, the reward distribution and the q-function, should be subject to dropout layers, to prevent them from overfitting. This ensures that generalised policies and reward distributions are learned.

To achieve the same, L_2 regularisation of variable strength is applied as part of the Adam optimiser.

8.2.4 Dataset imbalance

Since AVRIL also relies on the expert dataset $\mathcal{D}_E$, it has the same action magnitude imbalances as discussed in Chapter 6. Therefore the same oversampling procedure can be applied here. The oversampling is done w.r.t. the action magnitude of the action at time t and not at time $t+1$. The influence of the errors in the yaw rate inference has to be considered when applying oversampling, but in case of a negative effect oversampling will not be selected by the hyperparameter search.

8.2.5 Q-function and variational reward distribution network architectures

The network architecture of the q-function Q_θ and the variational reward distribution p_ϕ need definition and shall reuse as many components as possible from the previously defined architectures of Chapter 6.

Contrary to the definition of both functions, the networks do not take the observation and the action as its input. Instead, they base their predictions just on the observation and predict the q-values as well as the reward distribution parameters for all actions at once. By doing this the feature extractors from Chapter 6 can be reused and just one forward pass is needed during computation of the softmax function, which ultimately increases computational efficiency.

Consequently, the same overall architecture and the same feature extractors as shown in Chapter 6 can be used. But the regressors need adaption. The regressor of the variational reward distribution p_ϕ predicts n^2 means and n^2 log-standard deviations at once, where each output pair corresponds to one discrete action. The q-function Q_θ predicts n^2 continuous outputs, where every single one corresponds to the q-value of taking this action while observing the current observation.

8.3 Experiments

The performance of AVRIL shall be evaluated on both navigational tasks. The experiments again start with the conceptually simpler task of GO-Navigation and expand to the more complex EA-Navigation task using a graph-based policy.

8.3.1 Goal-Only Navigation

In GO-Navigation, both the variational reward distribution p_ϕ and the q-function Q_θ take only the goal features g as input. The goal of this task is to successfully navigate to a desired goal-position.

The training is set up as Bayesian hyperparameter optimisation that optimises w.r.t. the mean end position error on all validation scenarios. The lower this error, the better the performance of the learned policy. A total of 50 different configurations is trained and evaluated. Each configuration uses the Adam optimiser [101] for optimisation and relies on a network architecture of (256, 128) for both the variational reward distribution and the q-function. This architecture is chosen as it performs best in the BC experiments. The discount factor γ is set to 0.99. The other parameters, as shown in Table 8.1, are subject to the optimisation.

Symbol	Parameter	Values				
—	Batch size	2048	4096	8192		
η_{AVRIL}	Learning rate	$5 \cdot 10^{-3}$	10^{-3}	$5 \cdot 10^{-4}$		
N_{AVRIL}	Number of epochs	10	20	50		
—	Encoder dropout rate	0.0	0.1	0.2	0.3	
—	Decoder dropout rate	0.0	0.1	0.2	0.3	
—	Oversampling	`True`	`False`			
$\lambda_{L_2,\text{AVRIL}}$	L_2 regularisation	0.0	10^{-5}	10^{-4}	10^{-3}	
β_{AVRIL}	Temperature	0.1	1.0	10		
λ_{AVRIL}	Reward coefficient	0.1	1.0	10		
α_{AVRIL}	KL-Divergence coefficient	0.001	0.01	0.1	1	10
n	Discrete action values	9	15	21		

Table 8.1: Hyperparameters for goal-only AVRIL

First, the best performing policy and after that the corresponding reward distribution for the GO-Navigation task will be assessed.

Policy

The best performing policy has been chosen based on the mean end position error in all validation scenarios, which in this case was 380.95 m. The policy and reward distribution that achieved this used the parameters highlighted green in Table 8.1. Analysing the parameters of the 10 best performing parameter combinations one can observe some trends. First, the coefficient λ_{AVRIL} for the reward learning objective should have a value of 0.1, while the magnitude of the coefficient α_{AVRIL} does not have a significant influence. The temperature parameter was either chosen to be 0.1 or 1.0 across all ten parameter combinations. Additionally, there is a preference for low numbers of action bins. This is probably due to a trade-off between model size and output-dimension. Last, oversampling and a L_2 regularisation with a value of 0.001 benefit the performance of the trained policies. The increased share of erroneous inferred yaw rates due to the oversampling seems not to have an effect on the performance.

The performance evaluation of the best model reveals a low mean end position error of 380.95 m with a standard deviation of the position error of 387.80 m. The average end course error is small with just 32.45°, however the minimum and maximum values indicate that different orientations occur in the validation scenarios. This indicates that the goal-position might be reached with trajectories differing from the experts. The mean end velocity error is 0.54 m s^{-1}, which shows that the goal-position is reached with a similar velocity profile compared to the experts. Table 8.2 shows the performance metrics in detail.

	Mean	**Std**	**Min**	**Max**
End position error (m)	380.95	387.80	7.35	2773.39
End course error (°)	32.45	26.65	0.06	179.38
End velocity error ($\frac{m}{s}$)	0.54	0.95	0.001	6.41

Table 8.2: Navigational performance of goal-only AVRIL in validation AIS scenarios

A qualitative assessment of the policy can provide a more in-depth understanding of the policies performance. Figure 8.2 displays one validation scenario, where the agent successfully manages to reach the goal-position and follows a similar path in comparison to the expert. This behaviour is representative for a lot of scenarios and shows that good waypoint-following behaviour has been learned using AVRIL.

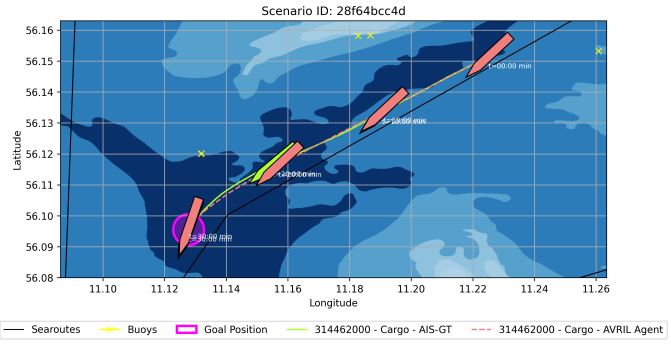


Figure 8.2: Scenario 28f64bcc4d - Goal-only AVRIL policy

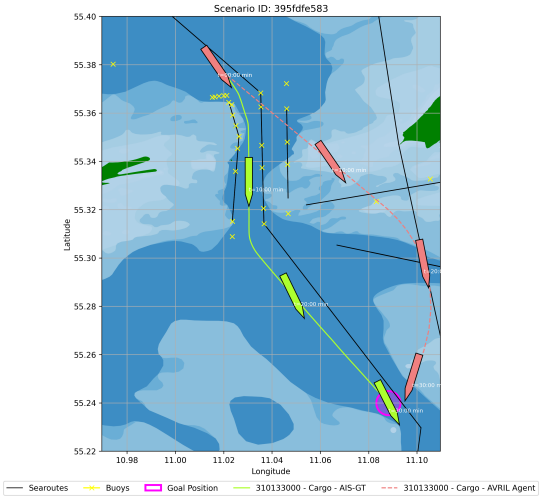
However, not every learner trajectory mimics the expert trajectory this accurately. In many cases the paths are more winding and indirect, resulting in longer distances travelled. Especially in scenarios, where the goal-position is not straight ahead of the OV it is evident, that the policy often takes more winding and longer routes. In most cases the agent still manages to reach the goal-position, but in some occasions the length of the trajectory is so long that the goal is not reached in time. Still, the shown behaviours look natural, as if they were done by an expert. Figure 8.3a visualises such behaviour.

In some occasions the agent also reaches the goal-position too early. In this case the agent starts to spin and move back towards the goal-position, which is in the scope of this experiment a decent reaction. This behaviour also explains the rather large variance in end course errors. Figure 8.3b shows one example of this behaviour.

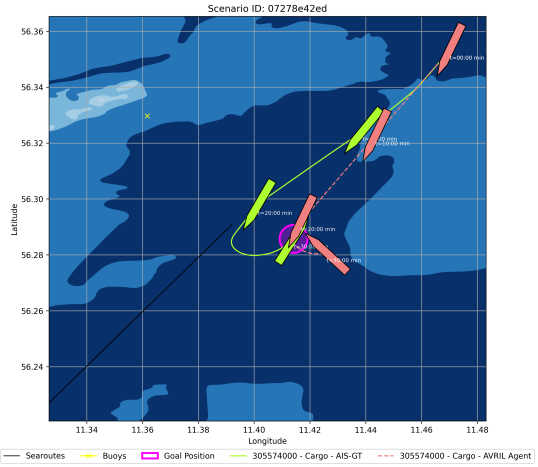
Overall, the policy trained by AVRIL shows good waypoint-following performance in the GO-Navigation task. In all validation scenarios it manages to move the OV close to the goal-location. By doing so it also acts naturally without any unreasonably high accelerations or yaw rates. This suggests it imitates the experts well.

Reward

AVRIL also recovers a reward distribution. This reward distribution can provide further insides into the learned behaviour. Initially, the mean and standard deviation are assessed



(a) Scenario 395fdfe583 - Goal-only AVRIL policy



(b) Scenario 07278e42ed - Goal-only AVRIL policy

Figure 8.3: Miscellaneous scenarios - Goal-only AVRIL policy

qualitatively and independent of their absolute values to assess whether the reward distribution encodes the navigational goals of GO-Navigation. After that both are jointly assessed to see whether the full reward distribution is informative.

First, the cumulated rewards obtained from the expert and learner trajectories can be assessed. This assessment is done for the mean of the reward distribution. Figure 8.4 shows this and one can see that the learner in fact acts more optimal w.r.t. the mean of the reward distribution than the expert. While the expert trajectories only average a cumulative reward of 155.23 the learner policies manage to achieve a mean cumulative reward of 236.77. Additionally, the variance of the cumulated rewards obtained by learner policies is smaller, however both policies manage to achieve the same upper bound in cumulative reward. The expert trajectories probably score lower cumulated rewards compared to the learned policy because the reward distribution does not encode the additional behaviour traits which avoid buoys or other vessels inside the full AIS-scenarios. All of this suggests that the learned mean of the reward distribution aligns with the behaviour the policy has learned.

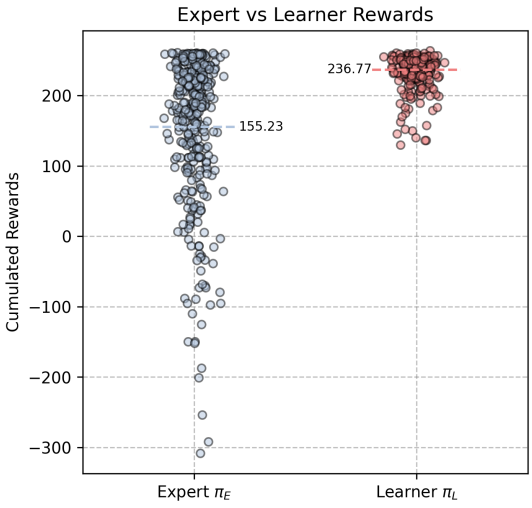


Figure 8.4: Cumulated expert vs. learner rewards under mean of goal-only AVRIL reward distribution in validation AIS scenarios

The learned reward distribution offers more potential for analysis by computing the gradient of the predicted mean and standard deviation of the reward distribution w.r.t. the input features of the reward network. In this case the input is the goal-position feature vector

g as elaborated in Chapter 8.2.5. Figure 8.5 visualises the gradients, where the left side shows the gradients for the mean and the right side for the standard deviation. A detailed explanation of this visualisation method can be found in Chapter 7.3.1.

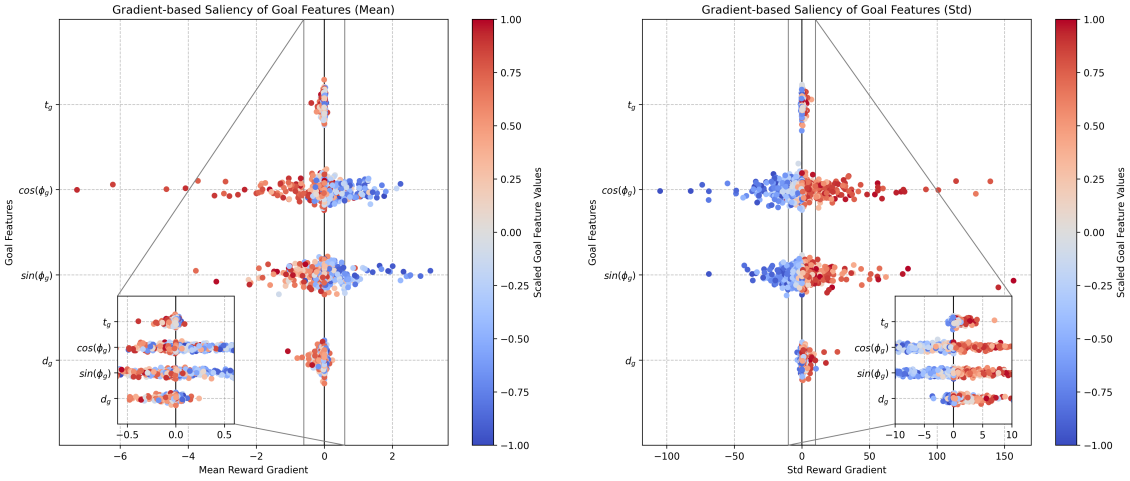


Figure 8.5: Gradient-based saliency of all input features of the AVRIL reward distribution

Using Figure 8.5, one can gain the following insights about the learned reward distribution:

- **Time to goal-position t_g:** The scaled feature value of t_g is large and positive if the time to goal-position is large. A decrease in the feature value is proportional to a decrease in time to goal-position. Analysing the gradients of the mean of the reward distribution w.r.t. this feature, no insight can be gained as the gradients are small and no correlation between gradient magnitude and scaled feature values can be identified.
  This is completely different for the gradient of the standard deviation. Analysing this, one can see that large scaled feature values have a higher reward gradient than lower feature values. This means that decreasing t_g results in a lower standard deviation until a lower bound is reached. There is also another very interesting observation that is valid for all features. The scale of the reward gradient is vastly different between mean and standard deviation. This probably indicates that the standard deviation are considerably larger than the mean reward values.

- **Bearing to goal-position ϕ_g:** The bearing feature provides interesting insights. The cosine of the bearing encodes if the goal-position is in front or behind the OV, with positive feature values symbolising a goal-position in-front of the OV. The sine of the bearing expresses whether the goal-position is on the starboard or port side. Positive values indicate the goal-position being on the starboard side. Using this knowledge one can see that turning towards the goal-position increases the mean of the reward distribution. However, for positive feature values the gradient is negative, meaning that turning away from the goal-position increases the mean of the reward distribution. This indicates that approaching the goal-position with an intermediary bearing angle is optimal. The sine component of the bearing shows that the mean of the reward distribution is maximised if the goal-position is neither on the starboard nor on the port side. This is motivated by the fact that increasing negative feature values and decreasing positive feature values will lead to an increase in the mean of the reward distribution. In comparison to t_g the reward magnitudes are also larger, indicating that the bearing is in fact more important for the computation of the mean of the reward distribution.
  The gradients for the standard deviation of the reward distribution can also be anal-

ysed. Here the observations are reversed. If the OV is facing away from the goal-position, turning towards it will decrease the uncertainty about the reward. Intuitively, this makes sense since almost all AIS-trajectories contain vessels with the goal-position in front of them. This trend is reversed for high feature values, where turning away from the goal-position reduces the standard deviation. This is in line with the observation that non-straight approaches to the goal-position are preferred by the mean of the reward distribution. The sine feature follows the same reversed logic. Having the goal-position on either the starboard or port side increases the uncertainty about the reward. The scale of the bearing features is again larger compared to t_g, indicating that they are more decisive for the reward standard deviation.

- **Distance to goal-position d_g:** The last feature to be analysed is the distance to the goal-position. The scaled feature value is large if the distance is large and small if the distance is small. This leads to the observation that decreasing the distance will increase the mean of the reward distribution. However, for small scaled feature values this trend seems to be reversed as a further decrease also decreases the reward slightly. Additionally, the gradients are rather small in comparison to the bearing features, which hints at a smaller importance of d_g. For the standard deviation of the reward distribution the observation is again reversed. The standard deviation gets smaller if the distance is decreased until this trend reverses for small scaled feature values. The gradient magnitude is again smaller compared to the bearing, indicating at this feature not being as important. Overall, the insights for the distance feature are plausible, as the reward rises and uncertainty shrinks the closer the goal-position gets. This is in line with the intuition that the closer the goal-position the higher the mean of the reward distribution and smaller the uncertainty about being in a valuable state.

These observations can now be made more concrete by analysing actual AIS-scenarios and their reward landscape. Similar to AIRL, reward heatmaps for both the mean and standard deviation of the reward distribution are generated and analysed. An explanation of this analysis method is provided in Chapter 7.3.1.

In Figure 8.6 this method is used to analyse the contribution of the t_g and d_g features as it plots the scenario reward heatmap for three timesteps and a vessel with a SOG of 10 kn & COG of 225° and no acceleration or yaw rate.

Analysis of the figure reveals that the magnitude of the mean of the reward distribution shrinks over time, which is a new insight that could not be extracted from the gradients of Figure 8.5. It can also be seen that the reward is at its peak not directly at the goal-position. The maximum is present for intermediate distances but also depends on the time to goal-position. This confirms the observations made from the gradient analysis, that getting closer to the goal-position to some extend increases the mean of the reward distribution. The standard deviation of the reward deviation is, as expected, completely contrary to the mean of the reward distribution, as the uncertainty is larger behind the goal-position. Such states are not present in the AIS-scenarios. Lower uncertainties can be observed for areas which are represented in the AIS-scenarios. Regarding the observations from the gradient analysis, the statement that t_g has an influence on the reward can be confirmed with these visualisations. This is probably due to the small magnitude of the gradients. The influence of the distance feature d_g on the reward standard deviation is also very small. One can slightly see that the standard deviation decreases the closer the goal-position gets, however the dependency on the bearing to the goal-position overpowers this trend.

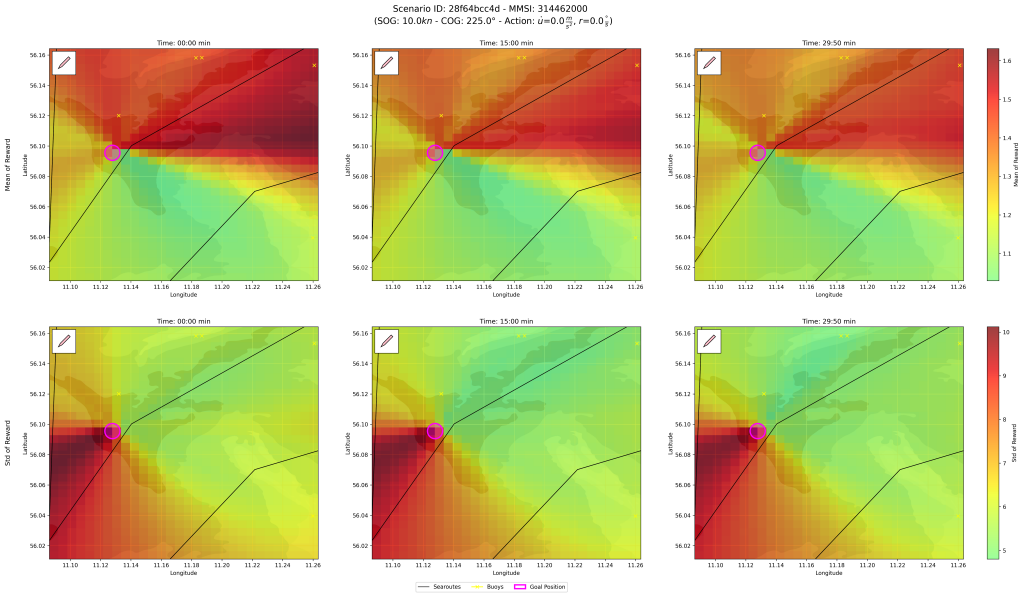


Figure 8.6: Scenario 28f64bcc4d - Scenario reward heatmap of goal-only AVRIL for different t_g

The bearing seems to be the most influential factor for the reward distribution. Therefore, Figure 8.7 plots the reward heatmaps for three different courses of the OV.

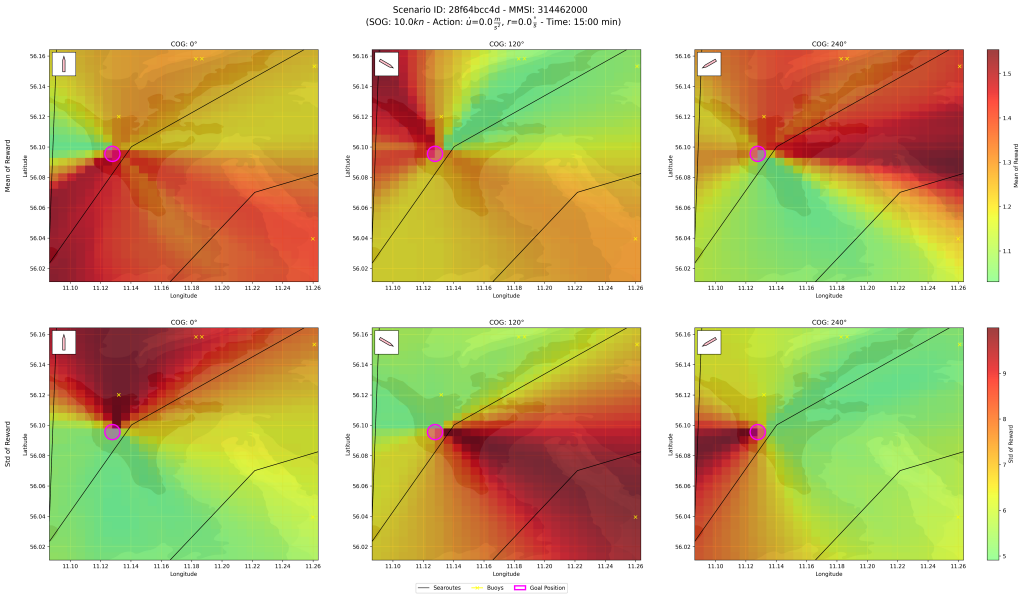


Figure 8.7: Scenario 28f64bcc4d - Scenario reward heatmap of goal-only AVRIL for different ϕ_g

As one can see the mean of the reward distribution is highest when the goal-position is ahead of the OV and lower if it is behind. However, as already identified by the gradient analysis, the reward is not highest if the bearing is 0°. Instead, the reward is maximised if the OV approaches the goal-position with a slight offset on the starboard-side. The reward for a direct approach is slightly lower. Having the goal-position fully to the starboard or port side has considerably lower rewards as consequence, which corresponds to the observations made in the gradient analysis of Figure 8.5. However, the reward magnitude is lower, when the goal-position is on the starboard-side, encouraging the OV to steer towards the goal-position more quickly in such a state. The standard deviation of the

reward distribution is largest right behind the goal-position seen in direction of travel. This complies with the observations from the gradient analysis. If the goal-position is towards the side of the OV the standard deviation of the reward distribution can, as shown by the gradient analysis, be decreased by decreasing the absolute bearing angle.

The actions have not been part of the analysis yet, as they are not an input to the model architecture. However, they also play a key role in the reward distribution. Figure 8.8 shows this. An in-depth explanation of this visualisation method can be found in Chapter 7.3.1.

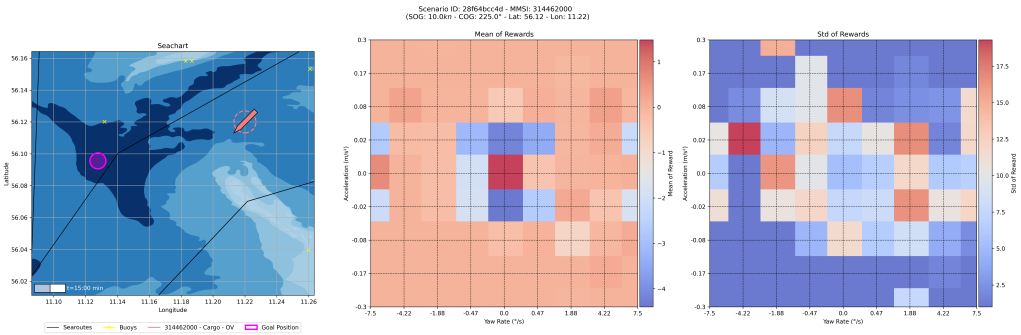


Figure 8.8: Scenario 28f64bcc4d - Action reward heatmap of goal-only AVRIL

The plot in the centre visualises the mean of the reward distribution for all action possibilities in the specific state. Not accelerating and not turning has the highest mean, while small accelerations and decelerations without significant turning have the lowest mean. The remaining reward landscape does not provide much information as the reward is mostly uniform. This shows that not taking excessive actions is incentivised by the mean of the reward distribution. The standard deviation is shown in the right plot. The largest standard deviation is observed for an acceleration of $0.02\,\mathrm{m\,s^{-2}}$ and a yaw rate of $-4.22\,^\circ\,\mathrm{s^{-1}}$. All actions with non-zero mean reward are also assigned a distinct standard deviation, however no clear trend is observable.

Aside from analysing the reward heatmaps, the scales of the mean and standard deviation of the reward distributions are of interest. While the mean is quite small with values between -4 and 2 the standard deviation is very large with values in the range from 2 to 18. This means the uncertainty is much larger than the differences in the mean of the reward distribution. As a result, the recovered reward distribution actually provides no meaningful insights if the reward would be sampled from the distribution. Analysis of other well performing policy-reward pairs confirms this observation. One possible reason for this is that for all those combinations the reward coefficient of the loss function λ_{AVRIL} was set to 0.1. Since the Bayesian hyperparameter optimisation optimises w.r.t. the position error, it achieves this by down-weighing the reward learning term of the AVRIL loss function via the parameter λ_{AVRIL}. A different hyperparameter optimisation objective could lead to more balanced parameter choices and therefore more interpretable reward distributions. This further investigation was out of scope for this report.

Conclusion

The conclusion for AVRIL in GO-Navigation is that the method is suitable for learning good performing policies. The recovered reward distribution qualitatively shows GO-Navigation-compliant insights, but this is information-less due to its high standard deviation. Therefore, it is not suitable to interpret the motives encoded in the expert data. The incorrectly inferred actions of Chapter 4.3.4 did not show an effect in the results of this experiment.

8.3.2 Environment-Aware Navigation with Graph-Based Policy

AVRIL is also applied to the EA-Navigation task to learn a policy that is capable of navigating complex maritime scenarios. In this setting the graph-based feature extractors are used as part of both the variational reward distribution p_ϕ and the q-function Q_θ.

The experiment is set up as Bayesian hyperparameter optimisation. Its optimisation objective is the minimisation of the mean average position error in all validation scenarios. 20 different configurations are trained and evaluated. Each configuration shares some common parameters. Those include the use of the Adam optimiser [101] and a common neural network architecture, which is based on the BC experiments of Chapter 6.3.3. The goal feature extractor uses a network architecture of (64, 32), the graph feature extractors use an architecture of (16, 16), the feature extractor output dimension is 16 and the encoder and decoder networks layers of size (128, 128, 64). The graph feature extractors use the `NNConv` layer which uses just a linear mapping as internal neural network. Additionally, the discount factor γ is set to 0.99 and oversampling is disabled. Every configuration is trained for 10 epochs as this proved sufficient to achieve convergence of the loss. The other parameters shown in Table 8.3 are subject to optimisation.

Symbol	Parameter	Values				
—	Batch size	256	512			
η_{AVRIL}	Learning rate	$5 \cdot 10^{-3}$	10^{-3}	$5 \cdot 10^{-4}$		
—	Encoder dropout rate	0.0	0.1	0.2	0.3	
—	Decoder dropout rate	0.0	0.1	0.2	0.3	
$\lambda_{L_2,\text{AVRIL}}$	L_2 regularisation	0.0	10^{-5}	10^{-4}	10^{-3}	
β_{AVRIL}	Temperature	0.1	1.0	10		
λ_{AVRIL}	Reward coefficient	0.1	1.0	10		
α_{AVRIL}	KL-Divergence coefficient	0.001	0.01	0.1	1	10
n	Discrete action values	9	15	21		

Table 8.3: Hyperparameters for environment-aware AVRIL

The results of this experiment are a policy and a reward distribution. Both are assessed in the following sections.

Policy

The best performing policy was chosen using the smallest mean average position error among all configurations. This policy and the corresponding reward distribution have been trained using the parameters highlighted in Table 8.3. Assessing the five best performing configurations one can determine some common parameter choices. First, the batch size was set to 512 for all of these configurations. Second, the weight parameters β_{AVRIL} and λ_{AVRIL} have smaller values with 0.1 or 1 for β_{AVRIL} and 0.1 for λ_{AVRIL}. The learning rate was also consistently set to 5×10^{-4} across all five scenarios.

The assessment of the best performing policy reveals a mean average position error of 545.45 m and a standard deviation of the average position error of 612.56 m. This is indicative for imitating behaviour of the learned policy. The mean average course and velocity errors support this claim with values of 8.66° and 0.22 m s^{-1} respectively. The mean end position error provides insights into the waypoint following behaviour of the learned policy. With 949.15 m this error is larger than the error obtained in the dedicated GO-Navigation task (see Chapter 8.3.1). Adding information about the environment, did

not lead to more accurate convergence towards the goal-location. However, the mean end course and velocity errors are smaller compared to the GO-Navigation task. This shows that being informed about the environment can align movement patterns with the experts.

	Mean	**Std**	**Min**	**Max**
Avg. position error (m)	545.45	612.56	28.25	3679.90
Avg. course error (°)	8.66	9.20	0.40	66.11
Avg. velocity error ($\frac{m}{s}$)	0.22	0.30	0.02	2.77
End position error (m)	949.15	1242.76	3.55	7540.94
End course error (°)	21.44	22.04	0.05	135.94
End velocity error ($\frac{m}{s}$)	0.32	0.40	0.001	3.48

Table 8.4: Navigational performance of env.-aware AVRIL in validation AIS scenarios

Table 8.5 shows the navigational safety metrics, which express the share of trajectories in which the OV encounters a collision or a grounding. The metrics show that the learned policy has only limited awareness of shallow waters, running aground in 86 % more scenarios than the expert policies. In terms of collisions with other vessels and shore the policy learned by AVRIL performs equal to the experts, which indicates that collisions with vessels and shore can be avoided. The policy outperforms the experts in terms of collisions with buoys as the share of trajectories with collisions is reduced by 50 %. These metrics show that the policy trained by AVRIL has gained some awareness of its surroundings, especially w.r.t. other vessels, shore and buoys.

	Value (%)	**Expert Value (%)**	**Difference**
Traj. with groundings	18.48	9.80	+89%
Traj. with vessel collisions	3.08	3.08	+0%
Traj. with shore collisions	7.84	8.12	−3%
Traj. with buoy collisions	1.12	2.24	−50%

Table 8.5: Navigational safety of environment-aware AVRIL in validation AIS scenarios

However, a qualitative assessment of some validation scenarios is required to fully understand the capabilities of this policy. The COLREGs are the main focus of this analysis, but also the awareness of seachart features shall be assessed.

The learned policy generally moves towards the goal-position in all scenarios. Figures 8.9 and 8.10a show that the goal-position is reached with only a small error. However, not in all situations the policy performs that well. In some occasions the policy initiates a longer path due to earlier decisions. This results in it not reaching the goal-position. Figure 8.10b shows such behaviour.

The first COLREG situation to be evaluated is overtaking. Figure 8.9 shows a representative overtaking scenario. The OV manages to successfully overtake the TV by maximising the distance to it. By doing so it also avoids collision with the upcoming TV.

The policy also shows COLREG-conform behaviour in crossing situations where the OV is the give-way vessel. Figure 8.10 shows two representative scenarios where the OV avoids collision with the TV by crossing behind it and keeping a safe distance. In Figure

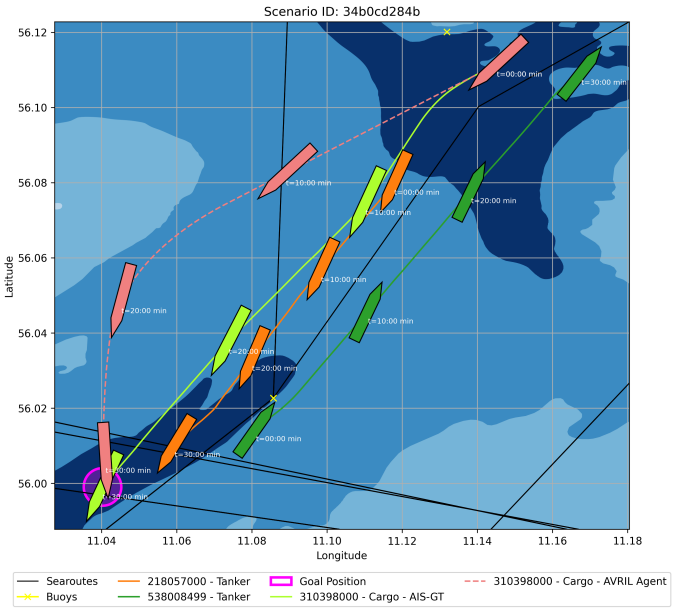


Figure 8.9: Scenario 34b0cd284b - Environment-aware AVRIL policy in overtake scenario

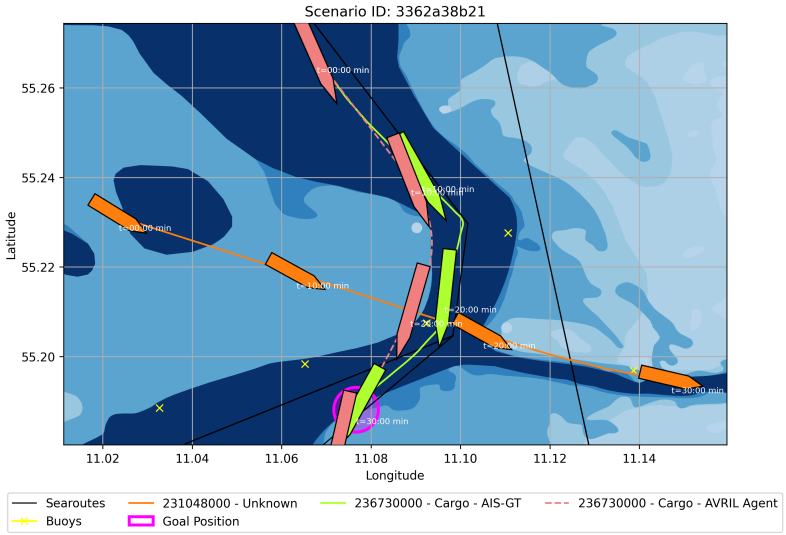
8.10b the evasive action occurs relatively late, but is still sufficient to avoid collision and comply with the requirements of the COLREGs.

The performance in head-on collision scenarios is less consistent. In some cases the policy successfully manages to avoid collision in head-on scenarios and also complies with the COLREGs. Figure 8.11a visualises such an example. In other scenarios the policy shows no awareness of the upcoming TV and collides with the TV. Such behaviour is shown in figure 8.11b and is clearly not COLREG-compliant nor safe.

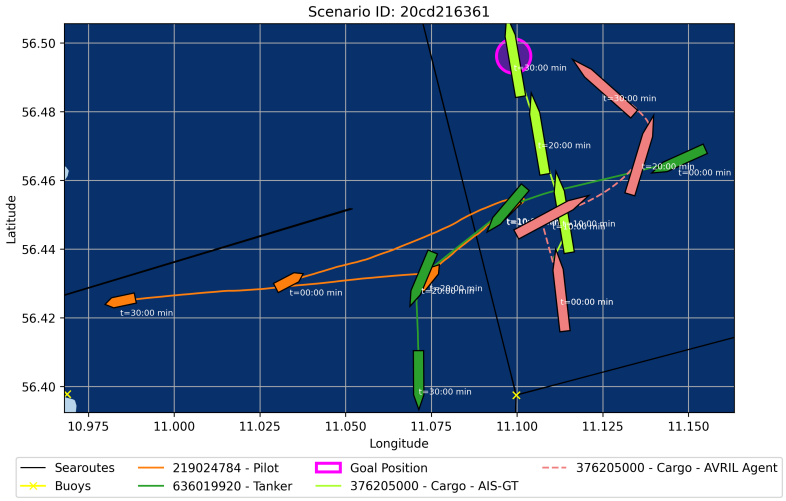
In terms of interaction with *pilot* vessels, the policy has learned to avoid those by taking evasive actions. Figure 8.10b shows such behaviour. even though this situation may also be characterised as crossing, avoiding actions w.r.t. *pilot* vessels have been observed in other scenarios as well. In general, this behavioural trait can be regarded as good and safe, as the policy is unaware of the intention of the *pilot* vessel to perform a *planned collision*.

Last, the performance of the policy w.r.t. environmental features like buoys, searoutes and the seafloor can be assessed. In simple configurations, the policy is aware of searoutes and buoys and can move accordingly. Figure 8.12a shows such an example. However, in scenarios with more buoys and searoutes the policy takes unexplainable and unsafe actions leading it to move through shallow waters and potentially colliding with other obstacles. Figure 8.12b shows this. The awareness of shallow waters is generally low. Figure 8.12c shows how the policy steers the OV straight through shallow waters.

All things considered, the policy learned by AVRIL shows mixed performance in the EA-Navigation task. It generally manages to move towards the goal-position, can comply to some of the COLREGs and is aware of buoys and searoutes. However, it is to reliably obeying the COLREGs and shows limited awareness of shallow waters. The reasons for the insufficiencies are probably related to the collapse of the reward distribution to a standard normal distribution, which led the loss-function into a local minimum. This problem will be elaborated more in the next section.

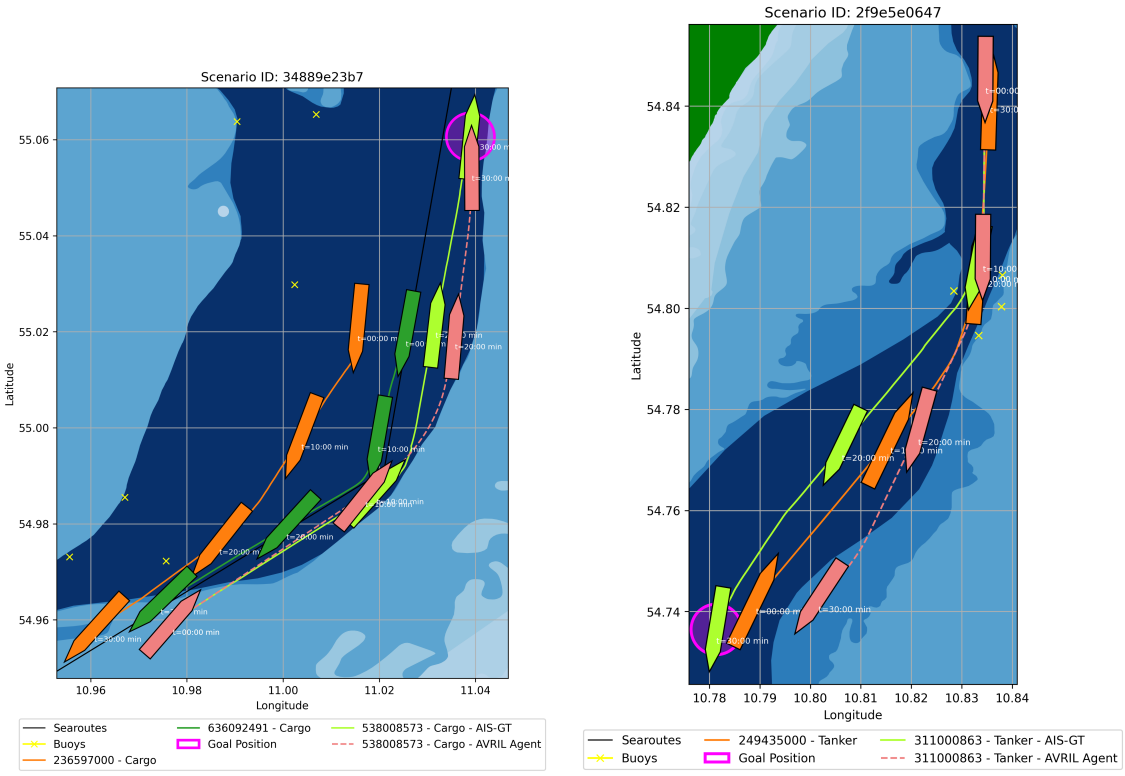


(a) Scenario 3362a38b21 - Environment-aware AVRIL policy in crossing scenario



(b) Scenario 20cd216361 - Environment-aware AVRIL policy in crossing scenarioy

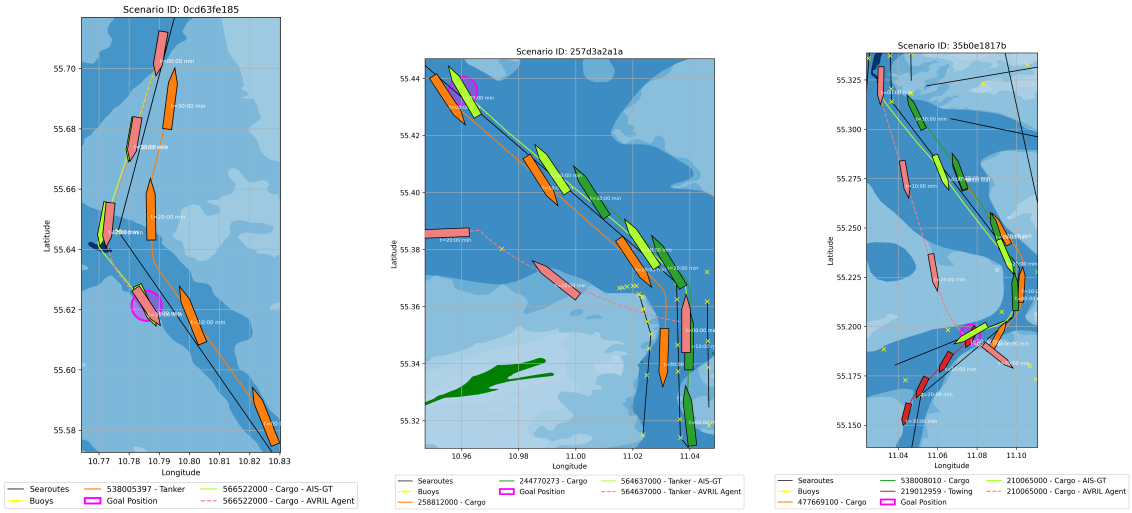
Figure 8.10: Environment-aware AVRIL policy in crossing scenarios



(a) Scenario 34889e23b7 - Environment-aware AVRIL policy in head-on scenario

(b) Scenario 2f9e50647 - Environment-aware AVRIL policy in head-on scenario

Figure 8.11: Environment-aware AVRIL policy in head-on scenarios



(a) Scenario 0cd63fe185 - Env.-aware AVRIL policy in env.-restricted scenario

(b) Scenario 257d3a2a1a - Environment-aware AVRIL policy shows unsafe action

(c) Scenario 35b0e1817e - Environment-aware AVRIL policy in env.-restricted scenario

Figure 8.12: Environment-aware AVRIL policy in miscellaneous scenarios

Reward

Besides the policy, AVRIL also learns a reward distribution that should model the experts navigational motives in the EA-Navigation task. This reward distribution shall now be assessed.

First, the cumulated rewards of the expert and policy trajectories in all validation scenarios are compared with each other. In this setting the rewards are not sampled from the distributions, instead the mean is used. Figure 8.13 visualises this and one can see that the learner acts better than the expert w.r.t. the mean of the learned reward distribution. The learner achieves a cumulated reward of 0.65 while the experts obtain 0.05. However, both values are relatively small, indicating that the mean of the reward distribution is small in all states.

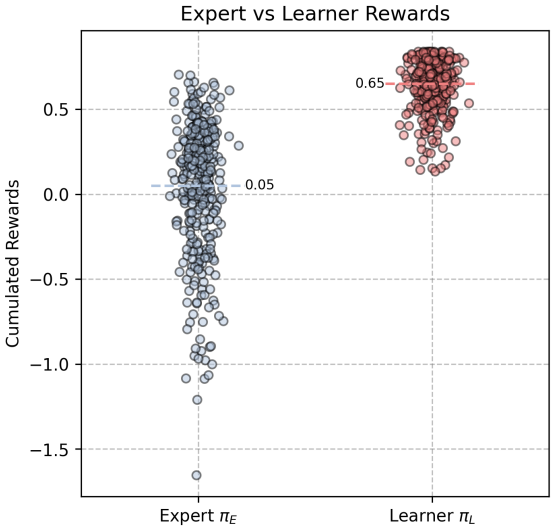


Figure 8.13: Cumulated expert vs. learner rewards under mean of env.-aware AIRL reward distribution in validation AIS scenarios

Further analysis is required to gain insight into the reward distribution. The encoding of the COLREGs and other environmental features like buoys or searoutes is of interest in this analysis. The representation of the waypoint-following behaviour is not of interest. The heatmaps, as introduced in Chapter 7.3.1, are used to assess the learned reward distribution. The gradient analysis is not used for analysis in the EA-Navigation task, as computing the gradients w.r.t. each node and edge of the input graphs is computationally expensive and difficult to visualise.

Figure 8.14 shows the mean and standard deviation for one timestep in a validation scenario, where the OV has a course of 225°, a velocity of 10 kn and is not accelerating or turning. The figure reveals that the reward distribution collapsed towards a standard normal distribution as the mean is close to 0 and the standard deviation is close to 1. This means the KL-divergence term has overregularised the reward distribution. Additionally, the reward is uniform at every location of the scenario. TVs, buoys, searoutes and different water depths do not have any qualitative effect on the either the mean or the standard deviation of the reward distribution. Surprisingly, also the distance to the goal-location is not represented in the reward distribution as it has been in the reward distribution learned by AVRIL in the GO-Navigation task.

The relation between reward distribution and the actions confirm the observation that the reward distribution is completely uninformative. Figure 8.15 shows the reward heatmap of a scenario with the OV having a course of 225° and a speed of 10 kn. The situation is a head-on collision scenario, as the OV is directly moving towards two upcoming TVs. The reward distribution does not indicate that higher yaw rates or decelerating can avoid collision. Instead, a circle around zero acceleration and yaw rate has a slightly lower mean and slightly higher standard deviation of the reward distribution. For larger action values the reward space seems to be unexplored. Overall, the mean of the reward is always close to 0 while the standard deviation is always close to 1 for all actions.

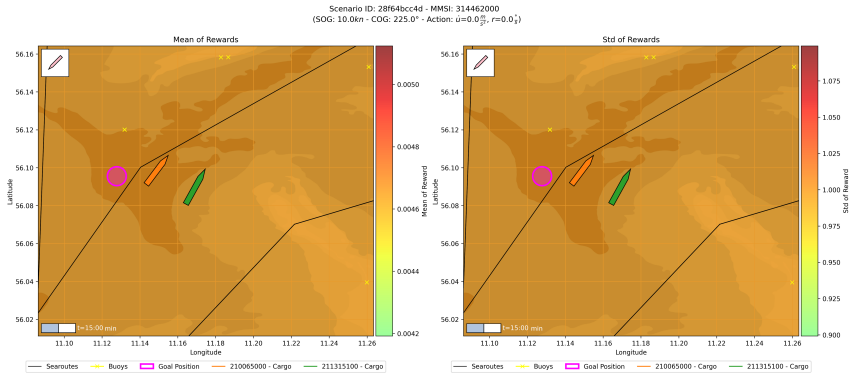


Figure 8.14: Scenario 28f64bcc4d - Scenario reward heatmap of env.-aware AVRIL

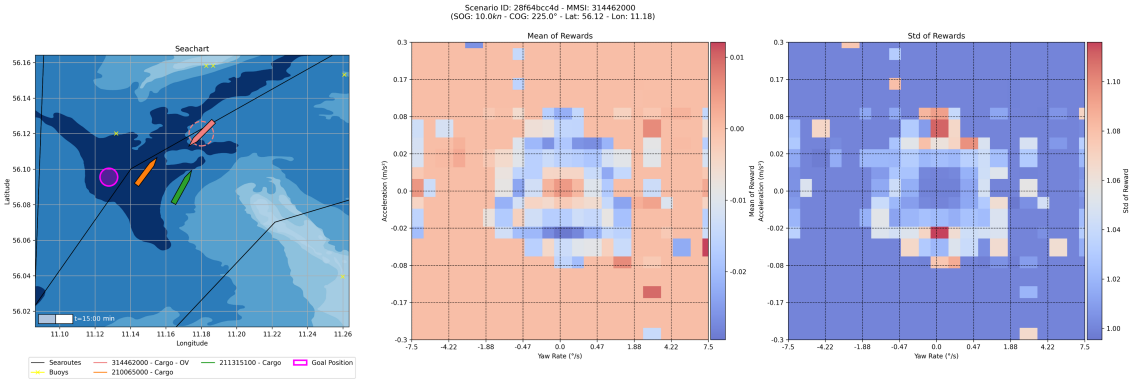


Figure 8.15: Scenario 28f64bcc4d - Action reward heatmap of environment-aware AVRIL

To rule out that the collapse of the reward distribution and the uninformative reward landscape are unique to this specific configuration, the reward distribution of another well performing configuration is analysed. As shown in Figure 8.16 the reward distribution also collapsed towards a standard normal distribution. However, qualitatively the reward distribution of this configuration shows that it has learned a reward representation of a scenario, which shall now be analysed in the light of the COLREGs.
Figure 8.16 shows a scenario, where the OV has a course of 0° and moves from south to north applying no action. The mean of the reward distribution shows patterns that are dependent on some environmental features. At the bottom of the seachart one can see that the deeper waters along the searoutes have low mean of the reward distribution, while the shallower waters have high mean of the reward distribution. This is unintuitive when considering maritime navigation rules. The proximity to the goal-location is not encoded into the mean of the reward distribution. Interestingly, the standard deviation shows similar patterns to the mean. This is different to the GO-Navigation experiment, where a low uncertainty was connected to a low mean. The reward distribution w.r.t. the possible actions shows the same uninformative patterns as shown in Figure 8.15 and will therefore not be analysed in more detail.

Next, a representative head-on scenario is analysed. Figure 8.17 shows the mean and standard deviation of the reward distribution in such a scenario. The OV moves with a velocity of 15 kn and a course of 20° while not applying any actions. The mean of the reward distribution is highest right in front of the TVs and no COLREG-compliant path of high rewards is visible. Instead, the region of high mean of the reward distribution continues towards the goal-location along the searoute. The standard deviation is again directly coupled to the mean, as a high mean implies a high uncertainty.

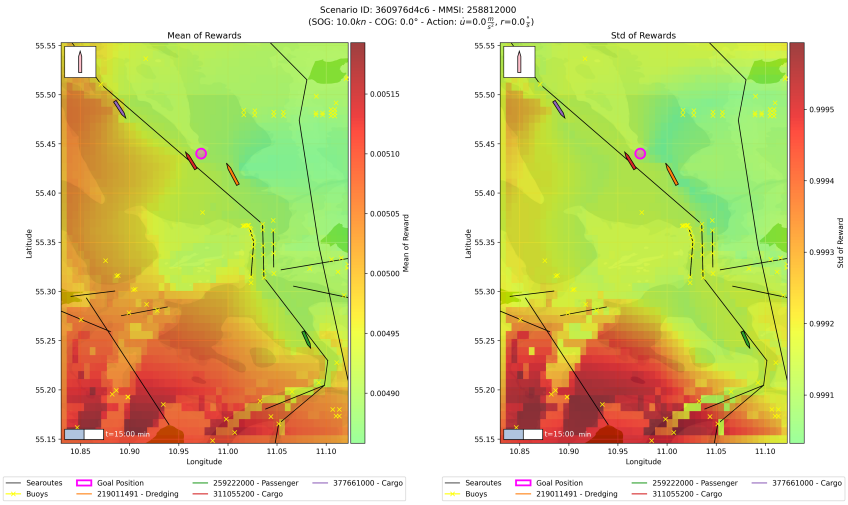


Figure 8.16: Scenario 360976d4c6 - Scenario reward heatmap of env.-aware AVRIL

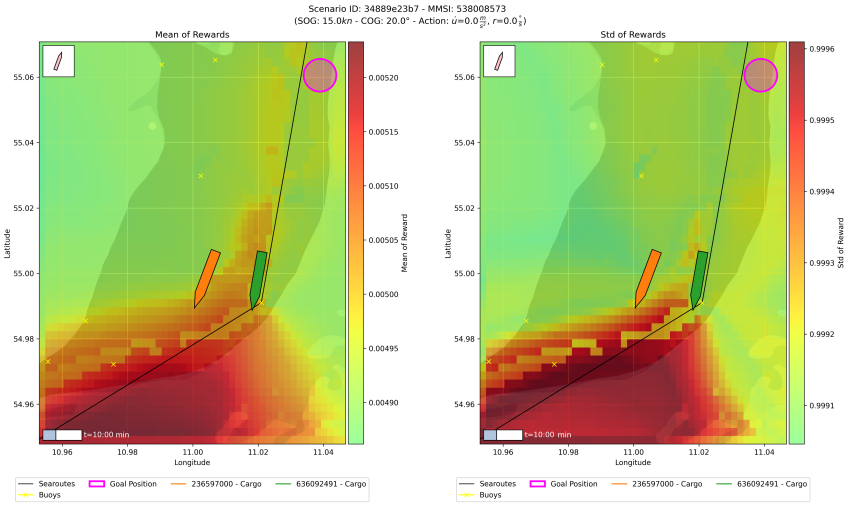


Figure 8.17: Scenario 34889e23b7 - Scenario reward heatmap of env.-aware AVRIL in head-on scenario

A crossing scenario is shown in Figure 8.18. The OV moves with a velocity of 10 kn from north to south without accelerating or turning. The mean of the reward distribution is higher in states along the searoutes and the deeper waters. While the shallower waters on the left of the scenario are regions of low mean of the reward distribution, the shallow regions on the right side of the scenario are regions of high mean of the reward distribution. The TV is also encoded by the reward distribution as the mean reward is lower behind it as seen in direction of travel of the OV. Interestingly, the standard deviation shows the opposite structure to the mean of the rewards, as it has already been for the GO-Navigation task.

Last, also an overtaking scenario shall be analysed. This is shown in Figure 8.19, where the OV has a velocity of 15 kn and a course of 45° in order to overtake the TV. The mean of the reward is highest along the searoute reaching towards the goal-position. On the starboard side of the TV the mean of the reward distribution is lower indicating that an overtake on the port side would be desirable. In this scenario the standard deviation again matches the structure of the mean as the uncertainty is highest along the searoute.

The reward distribution associated to the next best performing policy reveals qualitative patterns in the mean and standard deviation. The reward distribution is aligned to

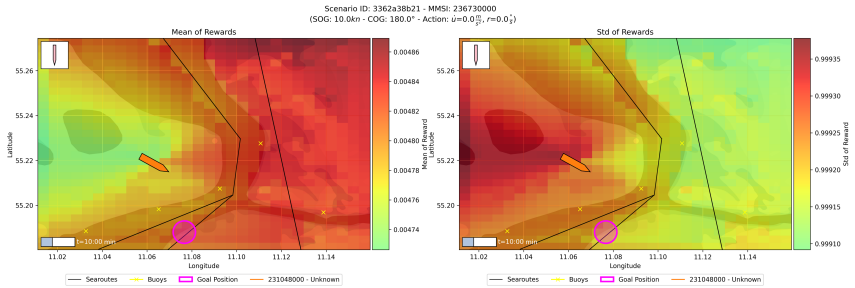


Figure 8.18: Scenario 360976d4c6 - Scenario reward heatmap of env.-aware AVRIL in crossing scenario

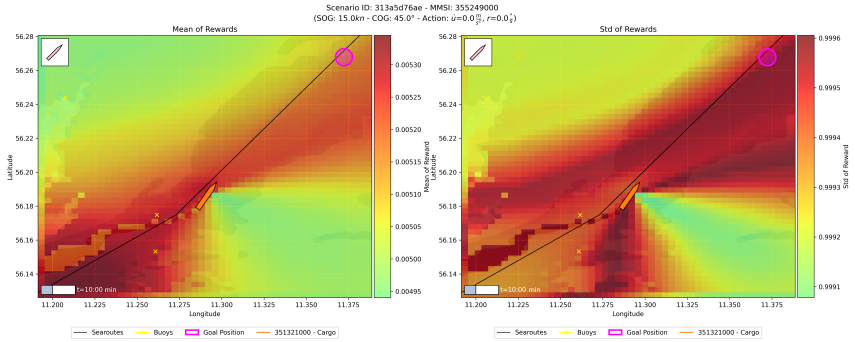


Figure 8.19: Scenario 313a5d76ae - Scenario reward heatmap of env.-aware AVRIL in overtake scenario

searoutes, buoys and TVs and not to the proximity to the goal-location. Nevertheless, the qualitative patterns of the reward distribution do not always correspond to the navigational motives of the EA-Navigation task, as proximity to deeper waters and searoutes is less desirable according to the patterns of the reward distribution. Therefore, AVRIL is able to extract patterns representing environmental influences, but those patterns are inconsistent and do not represent understandable navigation goals of the EA-Navigation task.

Despite this analysis of the patterns of the mean and standard deviation of the reward distribution, the total reward distribution remains uninformative due to the collapse to a standard normal distribution. If one were to sample from the distribution, no informative rewards would be obtained.

The shown results reveal, that AVRIL was not able to learn an interpretable reward distribution in this experiment. The uninformative structure of the first reward distribution shows that the reward network was not able to extract any meaningful information from the training data. In comparison, the reward network of the second best run was able to extract information. The collapse towards the standard normal distribution is a systematic overregularistaion issue. A possible reason is the optimisation objective for the hyperparameters, which is the reduction of the mean average position error. This assigns more importance to the maximisation of the log-probability of the softmax policy (see eq. 8.2). Choosing small λ_{AVRIL} as hyperparameters will help this objective, as the priority of learning an informative reward is scaled down. This leads the reward function to find a local minimum by minimising the KL-divergence term. A lower α_{AVRIL} can help, as this would make the KL-divergence term less influential. Alternatively, scheduling different reward coefficients during training, so that first the reward terms are prioritised and later the policy learning objective, could be another approach to escape the local loss-function minimum. Additionally, a different hyperparameter optimisation objective can possibly lead to more

interpretable reward functions, as the parameters would be tuned more carefully.

Overall, AVRIL did not manage to recover an interpretable reward distribution that explains the behaviours learned by the policy. The parameters λ_{AVRIL} and α_{AVRIL} require careful tuning to learn an interpretable reward distribution in combination with an expressive policy in a more complex task. Further hyperparameter optimisation or scheduling of the loss-function coefficients could therefore unleash better combinations of policies and reward distributions.

Conclusion

In conclusion, the policy learned by AVRIL for the EA-Navigation task exhibits limited environmental awareness and compliance to the COLREGs. In some scenarios it shows safe behaviours that are aware of shallow waters and other navigational features. However, in other scenarios it acts unsafe. The reward distribution learned as part of this experiment is uninterpretable as it collapsed towards a standard normal distribution. Further improvements w.r.t. the hyperparameter optimisation and the choice of hyperparameters is needed to improve this aspect. The incorrectly inferred actions of Chapter 4.3.4 did not show an effect in this experiment.

9 Comparison and Discussion

After evaluating all three methods on both navigation tasks individually, they shall now be compared with each other to assess their strengths and weaknesses. In a first step all policies and reward functions trained for the GO-Navigation task are assessed. After that the the policies and reward functions trained for the EA-Navigation task are compared.

Since all policies and reward functions have been optimised w.r.t. the validation dataset, the comparison shall be done using the so far unused and therefore independent test dataset.

9.1 Goal-Only Navigation

GO-Navigation was subject of three experiments, as BC, AIRL and AVRIL have been used to train policies and, if applicable, reward functions. These policies and reward functions will be compared and advantages and disadvantages w.r.t. the GO-Navigation task are discussed.

AIRL was used to train two policies and reward functions. Both, the HE and LE variant, will be compared in this section.

9.1.1 Policies

The comparison of all three methods starts with an assessment of the learned policies. The first step is thereby the analysis of the performance metrics computed in all test scenarios. Those can be found in Table 9.1.

	BC		**HE-AIRL**		**LE-AIRL**		**AVRIL**	
Error	Mean	Std	Mean	Std	Mean	Std	Mean	Std
End Pos. (m)	881.74	1301.99	92.49	55.61	270.04	616.12	427.61	858.09
End Vel. ($\frac{m}{s}$)	0.31	0.49	5.08	3.41	4.01	3.49	0.55	1.09
End Cou. (°)	34.89	22.01	93.79	53.67	76.16	64.52	28.73	24.90

Table 9.1: Navigational performance of goal-only BC, AIRL and AVRIL in test AIS scenarios

The performance metrics show that the policies trained by AIRL have the lowest mean end position errors. Especially, the HE-AIRL-policy has a very low end position error with a mean of only 92.49 m and a standard deviation of 55.61 m. The policy learned by AVRIL is third with a mean end position error of 427.61 m. BC is least accurate in terms of the end position error with a mean error of 881.74 m. The very good end position accuracy of the AIRL policies does not transfer to the other metrics. Both policies show significantly larger end velocity and end course errors compared to the AVRIL and BC-policies. The AIRL policies have a mean velocity error of over 4 $\mathrm{m\,s^{-1}}$ and a course error that indicates no consistent orientation upon ending a trajectory. This hints at the AIRL-policies achieving the navigational goals different than the experts do. BC on the other hand has a very low mean end velocity error of only 0.31 $\mathrm{m\,s^{-1}}$ and a reasonable mean end course error of 34.89°. This indicates that it imitates the expert better than the AIRL policies. AVRIL achieves similar metrics to BC as the velocity error is slightly higher, but the course error is lower. Therefore, AVRIL should also show better imitation performance compared to the AIRL policies.

Apart from this metric-based assessment a qualitative comparison in real scenarios can confirm these observations and provide a deeper understanding of the learned behaviours.

Figure 9.1 visualises one of the test scenarios where all four policies act to move towards the goal-position. All policies move the OV towards the goal-position, however there are subtle differences. The BC-policy sweeps of the shortest path towards the port side of the OV. This prolongs the path and results in it not fully reaching the goal-position. The LE-AIRL and AVRIL policies manage to reach the goal-position via relatively linear paths that look similar to the expert's path. The LE-AIRL policy thereby takes the more direct route and starts rotating upon reaching the goal-location. The HE-AIRL-policy acts completely different. It accelerates to reach the goal-location very quickly, to then start rotating and circling around the goal-position. This behaviour is clearly not imitating the expert well but is still fulfilling the GO-Navigation task.

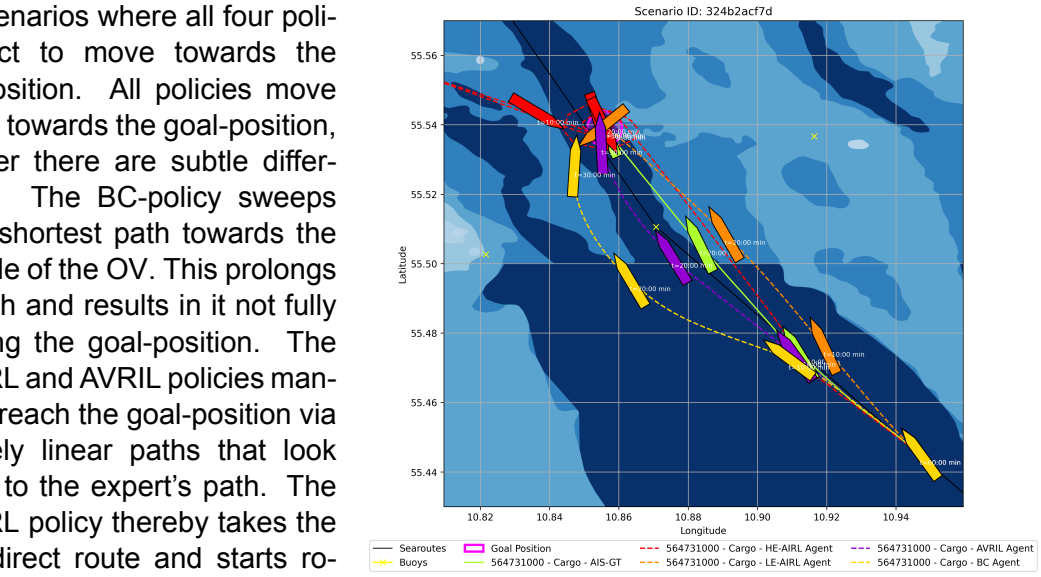


Figure 9.1: Scenario 324b2acf7d - Comparison of goal-only BC, AIRL and AVRIL policies

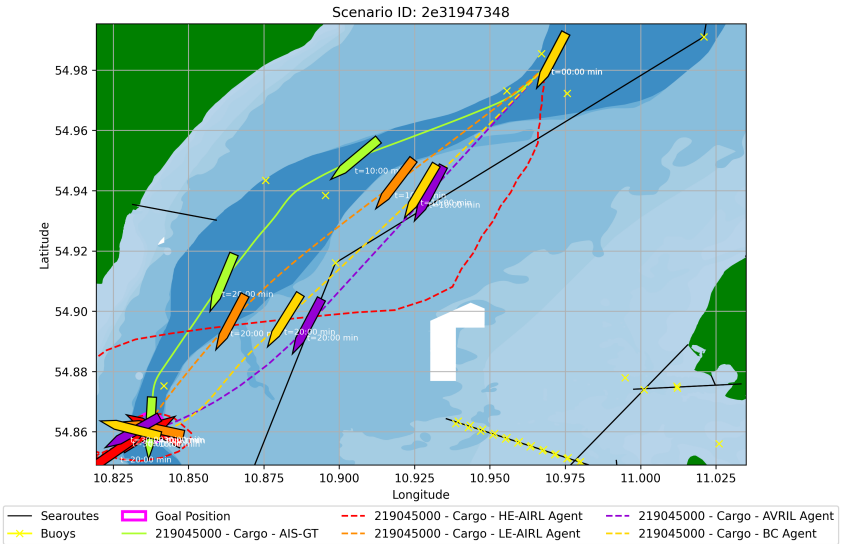


Figure 9.2: Scenario 2e31947348 - Comparison of goal-only BC, AIRL and AVRIL policies

A scenario where the expert does not follow a linear path is shown in Figure 9.2. Again the LE-AIRL, AVRIL and BC-policies act naturally by taking relatively linear paths to the goal-position which are characterised by modest velocities. All three end the trajectory with reasonable course error but while for BC and AVRIL this is due to the angle they reach the goal-location, the LE-AVRIL policy again rotates at the goal-location. Last, the HE-AIRL-policy does not act naturally. It again moves fast towards the goal-location and starts rotating around the goal-location once reaching it. This time the trajectory towards

the goal-location also features considerable deviations from the shortest possible path.

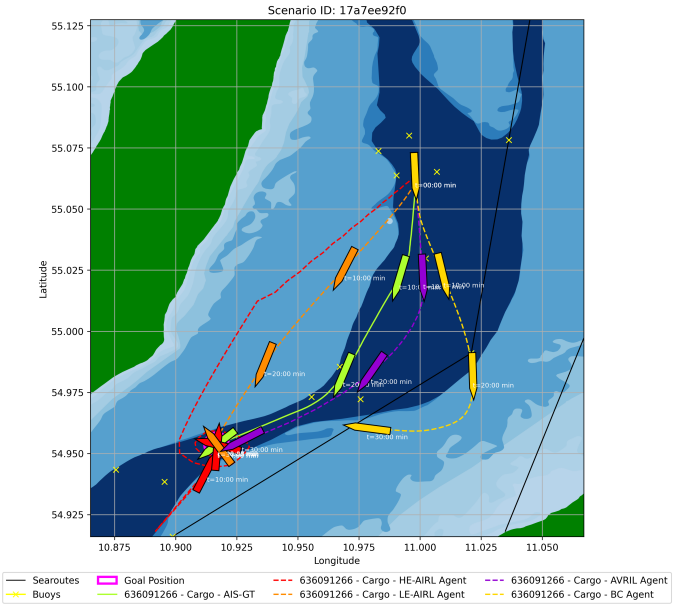


Figure 9.3: Scenario 17a7ee92f0 - Comparison of goal-only BC, AIRL and AVRIL policies

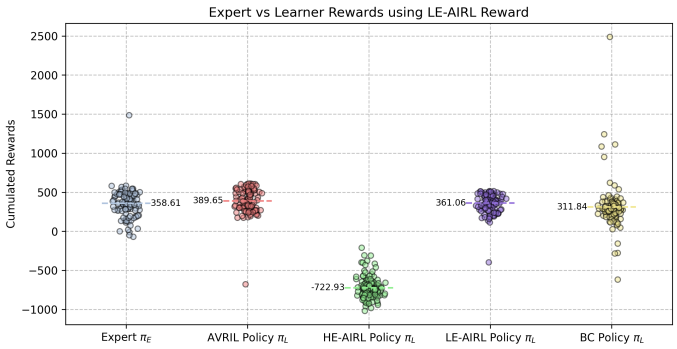
The last scenario shown in Figure 9.3 also requires the policies to turn the OV to reach the goal-position. The policy learned by BC shows the worst result in this scenario, as it does not reach the goal-position. In the initial stages of the trajectory, it accumulates considerable error w.r.t. the expert trajectory. Until the point it turns towards the goal-position, it has drifted so far from the shortest path, that it simply does not manage to reach the goal-position anymore. The policies learned by AVRIL and AIRL manage to reach the goal-position. The HE-AIRL-policy again moves with high velocity and the already explained circling motions around the goal-location. The LE-AIRL-policy takes the most direct path towards the goal-position and again rotates at the goal-location upon reaching it. The AVRIL policy imitates the expert the best, as it moves with decent velocity and uses a similar path.

The conclusion of this analysis is that the policies trained by AIRL and AVRIL manage to reach the goal-position more reliably and with higher accuracy. The AVRIL and LE-AIRL policies show good imitation of the experts in the qualitative assessment of real scenarios. However, AIRL is very sensitive to parameter choices, as a policy with high entropy can achieve good position errors but also acts unnaturally. All learned policies were not affected by the erroneous yaw rates inferred in Chapter 4.3.4. Overall, all IRL methods are able to outperform the BC baseline in the GO-Navigation task.

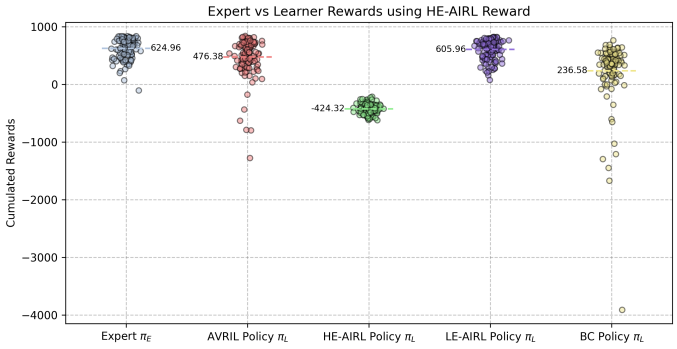
9.1.2 Rewards

Only AIRL and AVRIL learn reward functions, so that the results of only those two algorithms can be compared at this point. This section only compares the reward functions. For a detailed analysis please refer to the respective chapters.

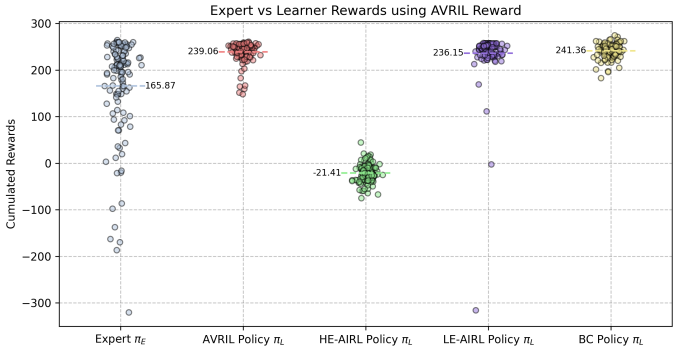
First, the imitation performance of all four policies can be validated by assessing the obtained cumulated rewards of all policies w.r.t. the learned reward functions. Figure 9.4 visualises this for the LE-AIRL, HE-AIRL reward function and the mean of the AVRIL reward distribution. The three policies that show decent imitation performance obtain similar rewards compared to the experts. This confirms that they act similarly to the experts or, in case of the AVRIL-reward, even better than the experts according to the respective reward function. However, the HE-AIRL-policy obtains considerably lower cumulated rewards in the test scenarios for all three learned rewards. This shows that it has learned behaviours that do not imitate the experts well, which was also concluded from the qualitative inspection. Further optimisation w.r.t. one of the AIRL reward functions may lead to more natural behaviours of the HE-AIRL-policy.



(a) LE-AIRL Reward



(b) HE-AIRL Reward



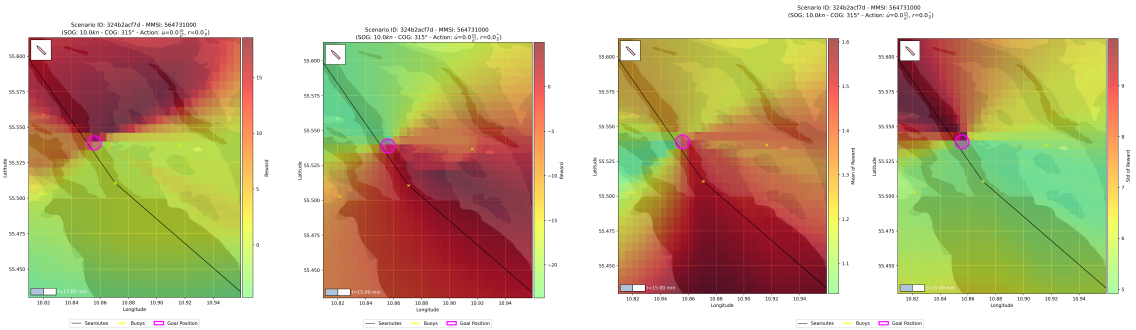
(c) AVRIL Reward

Figure 9.4: Cumulated expert vs. learner rewards under goal-only AIRL and AVRIL reward functions in test AIS scenarios

Next, the similarities and differences between the learned reward functions are analysed. This is based on the analysis of Chapters 7 and 8. The most important aspects are:

- **Navigational Goals:** All three learned rewards encode the navigational goals of the experts at least as local maxima of the reward landscape. Maxima of the reward are present if the OV is moving with a small bearing angle towards the goal-position. Figure 9.5 shows the three reward functions as scenario reward heatmaps for a representative scenario where the OV has a course of 315°, a velocity of 10 kn and is not taking any actions. Within reward regions that comply with the navigational goals, the distance to the goal is also encoded similarly. The highest reward of these maxima is not issued directly at the goal-location. Instead, it is issued along the path to encourage slower movements to the goal-location.

- **Interpretability:** In AIRL the interpretability of the reward function is depending on the entropy of the policy. If the entropy is high, the reward function is able to well represent the navigational goals of the GO-Navigation task. However, if the entropy is low, high rewards are also issued to states and actions that do not correspond to the correct navigation-goals of the experts. A possible reason for this is introduced in Chapter 7. This shows that there is a trade-off between imitation performance of the policy and interpretability of the reward function, which is dependent on the hyperparameter choice. AVRIL learns a reward distribution consisting of the mean and standard deviation of a Gaussian normal distribution. Qualitatively the regions of high reward and low uncertainty confirm the expectations. However, in comparison to the mean the standard deviation is very large. This makes the learned reward distribution uninterpretable as sampling from this distribution would not reliably reward favourable states and actions higher than unfavourable states and actions.



(a) LE-AIRL Reward (b) HE-AIRL Reward (c) AVRIL Reward

Figure 9.5: Scenario 324b2acf7d - Comparison of goal-only AIRL and AVRIL reward functions

These insights show that all three reward functions learned to represent the goal of GO-Navigation at least as local optimums of the reward functions. However, only the reward function corresponding to the HE-AIRL-policy is fully interpretable. Depending on the hyperparameter selection and the training process AIRL may also lead to reward functions that reward states and actions higher which are not desirable for imitation purposes. AVRIL learns a reward distribution, which has very high standard deviation that makes it uninterpretable. This high uncertainty may also be attributed to the optimisation objective of the Bayesian hyperparameter search. Overall, AIRL seems to be the preferable method to infer a reward function from expert data for GO-Navigation based on the insights gained from the conducted experiments. Additionally, it is the only algorithm where the policy truly uses the reward function for its own optimisation, which allows for further optimisation of the policy w.r.t. the last reward function estimate.

9.1.3 Conclusion

The findings of the analysis can be summarised as shown in Table 9.2. While the AIRL policy shows the best performance in GO-Navigation task both in accuracy and imitation depending on the hyperparameter selection, BC and AVRIL policies show less accuracy in terms of the position error, but also act very naturally. A reward function of AIRL can, depending on the hyperparameter choice, correctly encode the navigational goals of GO-Navigation in interpretable way. In this work AVRIL did not offer this interpretability as the optimisation objective possibly lead it to down-prioritise learning an interpretable reward.

This leads to the conclusion that AIRL is the best choice if both a policy and a reward

		BC	AIRL	AVRIL
Policy	Task-specific performance	o	+ +	+
	Imitation of expert	+	+ +	+ +
	Optimality w.r.t. rewards	+	+	+
Reward	Task representation	n/a	+	+
	Interpretability	n/a	+	- -
	Connection to policy	n/a	+	-

Table 9.2: Evaluation of BC, AIRL and AVRIL for goal-only navigation

function shall be learned in the GO-Navigation-task. If only a policy is desired BC and AVRIL are also valid choices, especially as these have computational advantages due to their offline nature.

9.2 Environment-Aware Navigation

All three algorithms - BC, AIRL and AVRIL - have also been applied to the EA-Navigation task to learn policies and reward functions. As GNN-based network architectures showed improved performance to vector-based architectures in the BC experiment of Chapter 6.3, those were used for all three algorithms and are subject to this comparison.

9.2.1 Policies

First, the policies trained by all three algorithms are compared and assessed using the test scenarios. For AIRL it has to be kept in mind, that the learning process was limited by long training times, which offers still considerable room for improvement of its results. Nevertheless, the current best policy is part of this comparison.

The comparison begins with the assessment of the navigational performance metrics computed in all test scenarios. Table 9.3 shows these metrics.

	BC		AIRL		AVRIL	
	Mean	Std	Mean	Std	Mean	Std
Avg. Position Error (m)	454.16	806.54	735.06	942.10	572.34	785.28
Avg. Velocity Error ($\frac{m}{s}$)	0.24	0.40	8.42	39.18	0.21	0.38
Avg. Course Error (°)	8.47	15.14	10.14	15.85	8.19	9.27
End Position Error (m)	656.93	1591.03	1535.31	2211.78	778.38	1038.86
End Velocity Error ($\frac{m}{s}$)	0.36	0.54	21.71	89.57	0.29	0.47
End Course Error (°)	21.53	31.36	21.58	33.31	17.57	19.15

Table 9.3: Navigational performance of environment-aware BC, AIRL and AVRIL in test AIS scenarios

The metrics reveal that the BC-policy performs best in terms of average position error with a mean of 454.16 m compared to 572.34 m for the AVRIL-policy and 735.06 m for the AIRL-policy. BC and AVRIL perform equally well w.r.t. the average velocity and course errors, while AIRL exhibits a large average velocity error, due to occasional fast spinning motions. These metrics indicate that the learned BC-policy and AVRIL-policy perform better in the EA-Navigation task then the learned AIRL-policy does.

The position, velocity and course errors at the end of the trajectories confirm this. The BC and AVRIL-policies show the smallest mean end position and end velocity errors with a large margin compared to AIRL. For the mean end course error, AVRIL acts with the highest accuracy, while BC and AIRL show similar errors. All mean end course errors are also smaller than any mean end course error achieved by a policy trained for GO-Navigation (see Table 9.1). This indicated that all policies have learned to move the OV naturally towards the goal-position in at least the majority of the scenarios. Qualitative assessment has to confirm this.

	BC	**AIRL**	**AVRIL**	**Expert**
Traj. with groundings (%)	21.85	23.53	23.53	11.76
Traj. with vessel collisions (%)	2.52	6.72	2.52	2.52
Traj. with shore collisions (%)	5.04	5.88	7.56	6.72
Traj. with buoy collisions (%)	3.36	3.36	1.68	1.68

Table 9.4: Navigational safety of environment-aware BC, AIRL and AVRIL in test AIS scenarios

Next, is the comparison of all three policies using the navigational safety metrics (see Table 9.4). The share of trajectories with groundings is high among all three policies. It is around two times bigger than for the experts, which shows that all three policies only have limited awareness of shallow waters. In terms of collisions with TVs, the BC and AVRIL policy show similar performance compared to the experts, while the AIRL policy performs worse by colliding with other vessels in almost three times more trajectories. The awareness of shore is on a good level for all three policies. While BC and AIRL collide less with land, the AVRIL-policy collides with land slightly more often compared to the experts. The last features of the maritime environment are buoys. Only the AVRIL-policy performs on a similar level compared to the experts. The BC and AIRL policy collide twice as often with buoys compared to the experts. Overall, these metrics show that the BC and AVRIL policies have better awareness of the environmental features. However, they still do not act on the same level as the experts do.

This leads to the qualitative assessment of all three policies inside representative test scenarios. The focus of the comparison is on the general waypoint-following behaviour, the COLREGs and the awareness of environmental features like buoys or shallow waters.

First, the waypoint-following performance can be compared. Figures 9.6 and 9.7 show that all three policies steer the OV towards the goal-location. Due to earlier decisions along the path, they do not always manage to reach the waypoint with high accuracy. The only exception from this are the cases, where the AIRL-policy starts the highly dynamic spinning movements. One example of this behaviour has been illustrated in Figure 7.16 in Chapter 7.3.2.

The overtaking behaviour of all three policies is shown in Figure 9.6. The scenario shows that not all three policies manage to perform the overtake. The BC-policy shows the best performance as it steers towards the starboard side when conducting the overtake to maximise the distance to the TVs. The AVRIL-policy also succeeds but acts in a less safe manner. It keeps a relatively linear path towards the goal-position, which still avoids collision. The AIRL-policy collides on its way to the goal-location as it drifts towards the port side, which eventually leads to a collision with one of the TVs.

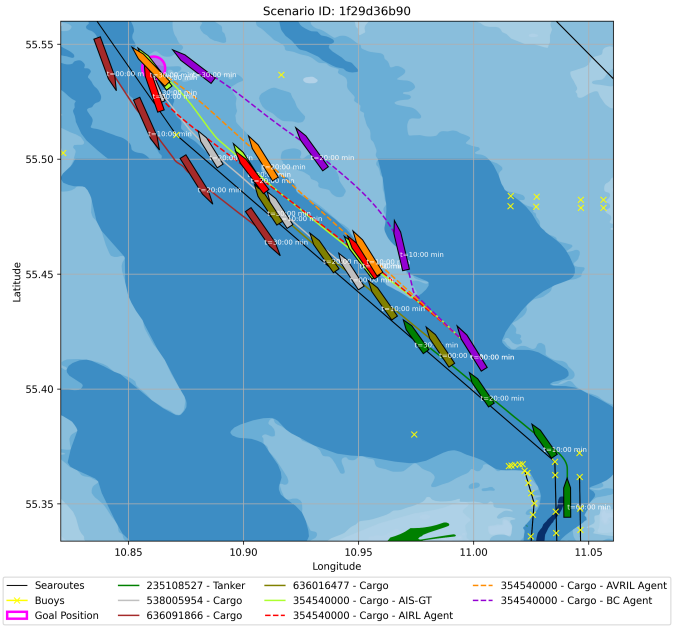


Figure 9.6: Scenario 1f29d36b90 - Comparison of environment-aware BC, AIRL and AVRIL policies in overtake scenario

Figure 9.7 shows a representative head-on scenario. The BC-policy again manages to avoid collision and to act COLREG-compliant by passing the TV on the TVs port side. The AIRL-policy shows similar behaviour in this scenario as it manages the head-on collision scenario successfully. The AVRIL-policy does not avoid collision in this setting. At the start of the trajectory, it deviates towards the port side which results in a collision when the policy tries to steer the OV back towards the port side of the TV.

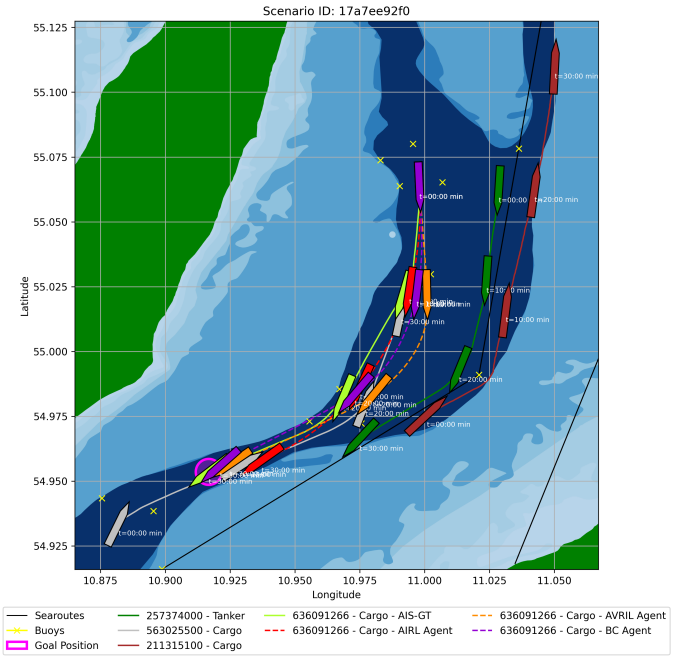


Figure 9.7: Scenario 17a7ee92f0 - Comparison of environment-aware BC, AIRL and AVRIL policies in head-on scenario

The last of the relevant COLREG situations is a crossing scenario. Figure 9.8 shows a crossing scenario. In this case AIRL shows the worst performance as it just follows a linear trajectory towards the goal-position without reacting to the crossing TV. This is clearly not COLREG-compliant and also not safe. The policies learned by BC and AVRIL perform better in this regard. They recognise the crossing TV and initiate a collision-avoiding maneuver. This is characterised by steering towards the port side and crossing in front of the crossing TV. This is also not COLREG-compliant, but because the avoiding action is large and the distance to the TV is maximised this behaviour can still be regarded as safe.

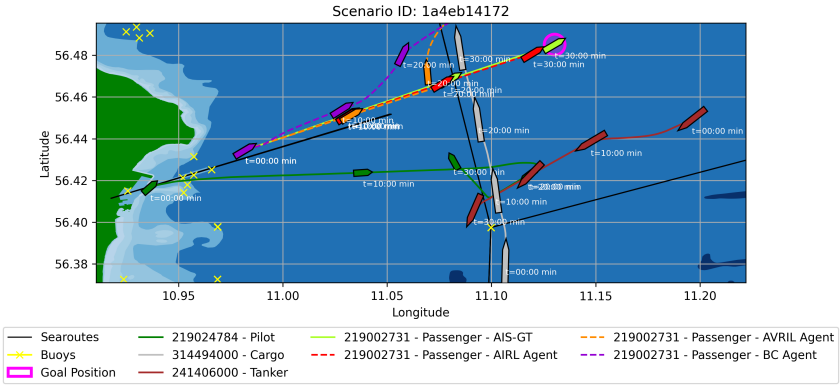


Figure 9.8: Scenario 1a4eb14172 - Comparison of environment-aware BC, AIRL and AVRIL policies in crossing scenario

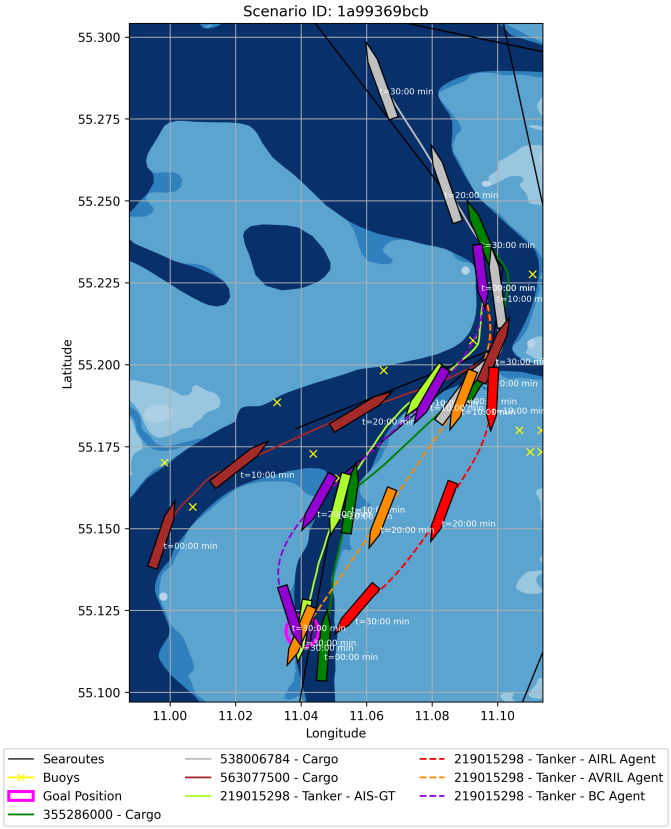


Figure 9.9: Scenario 1a99369bcb - Comparison of environment-aware BC, AIRL and AVRIL policies in environment-restricted scenario

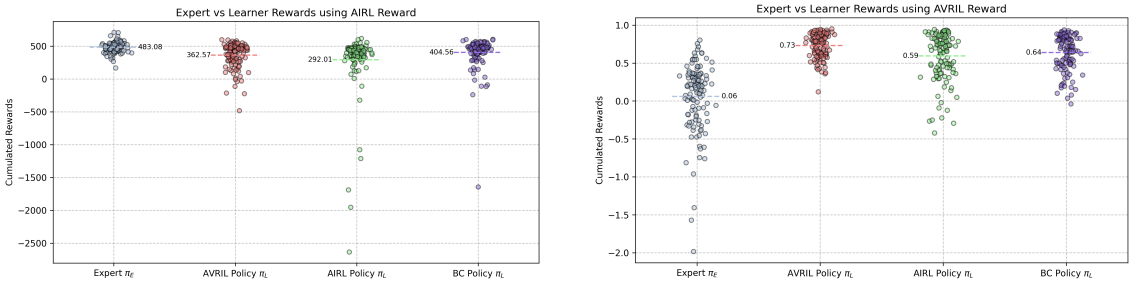
Last, the awareness of all three policies regarding buoys, land, searoutes and water depths shall be compared. For this purpose, Figure 9.9 shows a representative test scenario. It shows that the BC-policy has gained some awareness about its surroundings as it avoids collision with other vessels and moves along the searoutes and the deeper waters. The other two policies do not perform that well. The AVRIL-policy takes a shortcut via shallower waters to move to the goal-position and does not show any reaction to environmental features such as buoys or searoutes. The same applies to the AIRL-policy, which even takes a longer route through shallower waters. Therefore, only the BC-policy can be attributed with environmental-awareness.

Overall, not all policies succeed fully in the EA-Navigation task. BC shows the best performance as it shows reasonable reactions in most challenges. AIRL, on the other hand, often shows insufficient and environment-unaware behaviour. This is probably also attributed to not being trained sufficiently due to training time and hyperparameter limitations. AVRIL performs in between, which can be attributed to the exploitation of a local minimum of the loss-function. However, all policies still need a lot of improvement to reliably master maritime navigation as seen from the performance metrics. Especially, the awareness of shallow waters needs to be improved.

9.2.2 Rewards

AIRL and AVRIL also learned a reward function. This section compares the properties and limitations of both without providing an in-depth analysis. For more details, please refer to the respective chapters. As the reward distribution of the best performing AVRIL configuration collapsed towards a standard normal distribution and was uninformative w.r.t. the qualitative structure of the reward distribution, the reward distribution of the second best parameter configuration is compared at this point.

First, the imitation performance of all three policies can also be evaluated using the reward functions. The higher the cumulated rewards when acting in the test scenarios the better the imitation performance according to the learned reward function. Figure 9.10 visualises the rewards obtained by the experts and all three policies w.r.t. the reward function learned by AIRL and the mean of the reward distribution learned by AVRIL. For the AIRL reward function one can see that the policies trained by BC and AVRIL on average obtain higher rewards but still do not act as good as the experts according to the reward function. The AIRL-policy obtains less cumulated reward and also exhibits a greater variance in the cumulated rewards, which links to the already mentioned spinning behaviours. This confirms the previous observation that BC and AVRIL trained better performing policies in this projects experiments. For the mean of the AVRIL reward distribution, all three policies achieve cumulated rewards in a similar range. The experts show much greater variance in the obtained cumulative mean of the reward distribution. As already analysed, this reward distribution is not interpretable as it has collapsed to a standard normal distribution.



(a) AIRL Reward

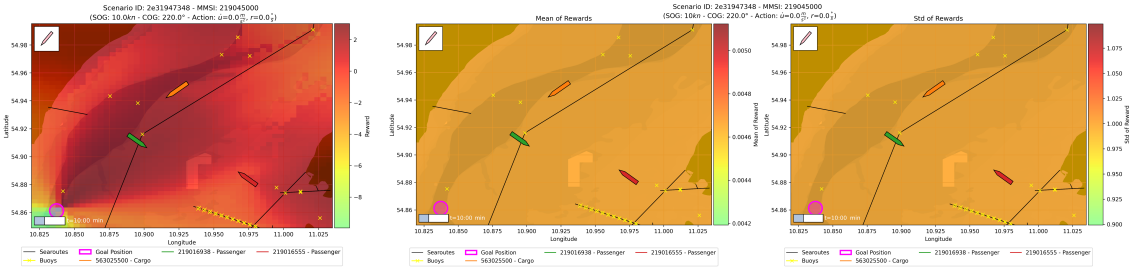
(b) AVRIL Reward

Figure 9.10: Cumulated expert vs. learner rewards under environment-aware AIRL and AVRIL in test AIS scenarios

The comparison of both reward functions is based on the insights gained in Chapters 7 and 8. The most important aspects are:

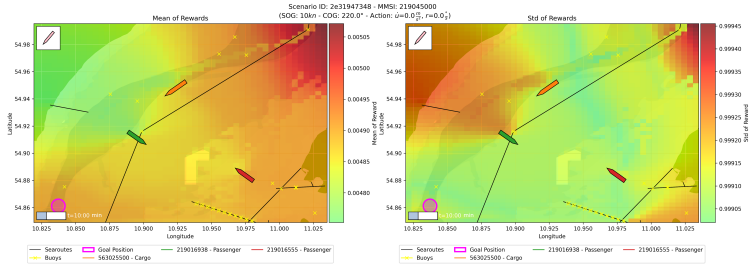
- **Interpretability:** In the conducted experiments, only AIRL learned an interpretable reward function. AVRILs reward distribution collapsed towards a standard normal distribution (see Figure 9.11). The reward distribution associated with the lowest position error policy did not even provide any qualitative structure in its reward distribution. Another policy-reward combination provided a reward distribution that contained qualitative structure but still collapsed to a standard normal distribution.
- **Navigational goals:** The navigational goals are not well represented by both learned reward functions. AIRL mainly focuses on the proximity to the goal-location and rarely issues different rewards based on the presence of environmental features. For AVRIL, only qualitative statements about the structure can be made, as the full distribution is informationless. The reward distribution shows dependencies to some environmental features but is too inconsistent from scenario to scenario to derive comprehensible insights.

- **Limitations:** Both rewards can be improved. In the case of AIRL computation times were the main limitating factor, while AVRIL suffered from a hyperparameter selection that prioritised good performing policies. Overcoming these issues can potentially lead to more interpretable results.



(a) AIRL Reward

(b) Pattern-less AVRIL Reward



(c) AVRIL Reward

Figure 9.11: Scenario 324b2acf7d - Comparison of environment-aware AIRL and AVRIL reward functions

These insights show that AIRL and AVRIL both struggle at learning interpretable and EA-Navigation-conform reward function. While both show signs of awareness of environmental features, AIRL prioritises the distance to the goal-location and AVRIL produces reward distributions that are just standard normal distributions. Further improvements are required to leverage the potential that has been shown in the GO-Navigation task. For AIRL this means expanding the experiments to longer training times and for AVRIL optimising the hyperparamters w.r.t. a more suitable optimisation objective or adding schedules for the loss-function coefficients to escape the local minimum of the loss-function.

9.2.3 Conclusion

This comparison can be summarised as shown in Table 9.5. The BC-policy shows the best performance in the EA-Navigation task as it has the highest awareness of the environment around it and achieves partial COLREG-compliance. By doing so it also imitates the experts considerably better than the AIRL and AVRIL-policies. AIRL shows the worst task-specific performance but offers great potential if policies could be trained longer and better parameter optimisation could be achieved. The learned reward functions by AIRL and AVRIL do not meet the high expectations set out by the GO-Navigation experiments. The AIRL reward function is interpretable but does not offer many insights into the COLREG-relevant aspects. More computation can also benefit this reward function. The AVRIL reward distribution shows signs of environmental awareness but is very inconsistent in the representation of the experts' navigational motives. Additionally, its reward function is informationless, as it collapsed towards a standard normal distribution.

This leads to the conclusion that BC is the best choice for EA-Navigation considering the setting of the experiments. However, further improvements may also lead to AIRL and

		BC	AIRL	AVRIL
Policy	Task-specific performance	+	-	o
	Imitation of expert	+	-	o
	Optimality w.r.t. rewards	+	o	o
Reward	Task representation	n/a	-	-
	Interpretability	n/a	o	- -
	Connection to policy	n/a	+	-

Table 9.5: Evaluation of BC, AIRL and AVRIL for env.-aware navigation

AVRIL being reasonable choices.

9.3 Discussion

This thesis evaluates three different methods - BC, AIRL and AVRIL - for learning policies in maritime navigation tasks. A key objective is to determine if these methods can also learn interpretable reward functions that encode the navigational goals of the experts. Therefore, this section shall discuss the advantages and disadvantages of all three methods based on the experience gained from the conducted experiments.

In the two navigational tasks all three algorithms showed different results in the conducted experiments. While in the simpler GO-Navigation task the two IRL methods managed to learn policies that outperform the baseline BC approach, they failed to do so in the more complex EA-Navigation task. Still, AIRL and AVRIL offer great potential also in the EA-Navigation task, if certain limitations were to be overcome. Especially, AIRL ,through its exploratory nature, can learn policies that are able to succeed in complex tasks given enough computational time. However, through this exploratory property this approach might show not the best imitation of experts as was seen in parts of the experiments. BC and AVRIL have advantages in this regard as they solely rely on offline expert data. All algorithms, regardless of the navigation task, also showed that they can act near-optimal w.r.t. the learned reward function.

In terms of the learned reward functions the results are again split. In GO-Navigation AIRL managed to learn a reward function that allowed to interpret the navigational gaols of the experts but was dependent on the hyperparameter choice. The reward distribution of AVRIL suffered from high uncertainty which diminishes the interpretability, but the structure of the distribution still represented the navigational goals well. In EA-Navigation both failed at learning rewards that could fully explain the navigational goals. However, possible improvements have been identified and the results from the GO-Navigation task indicate that both methods can excel further at learning rewards. Nevertheless, AIRL is the only method that truly uses only the reward for policy learning which gives its reward function much more credibility.

Apart from the performance related aspects of the three algorithms, some algorithmic constraints also require consideration and may motivate the usage of one or another algorithm. The relevant aspects are:

- **Hyperparameters:** All three methods differ in their number of hyperparameters and their sensitivity to hyperparameter choices. BC requires the definition of just a few parameters and is generally robust to different parameter settings. AVRIL does also not require the definition of many parameters, however the correct choice of those

is more important in order to obtain an informative reward distribution. In contrast, AIRL involves considerably more parameters, split between the reward and policy learning components. This is especially true, if the policy is learned by an algorithm like PPO, which has a lot of parameters. Finding parameter settings that yield good performance is also more difficult for AIRL, since poor choices can lead to unstable training. Consequently, BC is more advantageous in terms of hyperparameter selection and tuning.

- **Training time:** The time required to train good performing policies is vastly different between the offline and online methods. BC and AVRIL are comparably fast to train due to their offline nature. In contrast, AIRL requires considerably more time to train, since it has to iteratively sample new demonstrations from the environment and backpropagate loses through a total of three neural networks. This makes it less efficient, leading to higher training times. As a result, BC and AVRIL are preferable in terms of training times.
- **Data requirements:** BC and AVRIL are offline methods while AIRL is an online method. Accordingly, the methods also exhibit different requirements towards the available expert data. BC and AVRIL perform best, when a lot of expert data is available. Ideally, this data should be preprocessed so that all relevant traits, which should be learned, are well represented. AIRL, on the other hand, can work with just a few expert demonstrations, because it can explore good and bad behaviours, also in unseen sceanrios, while acting inside the environment. Consequently, AIRL can be applied to a wider range of problems, while the two offline methods achieve their best results when enough high-quality expert data is available.

Table 9.6 summarises the results of this discussion by rating the properties of all three algorithms w.r.t. the already introduced categories.

		BC	AIRL	AVRIL
Policy	Task-specific performance	+	+ +	+
	Imitation of expert	+	o	+
	Optimality w.r.t. rewards	+	+	+
Reward	Task representation	n/a	+	+
	Interpretability	n/a	+	- -
	Connection to policy	n/a	+	-
Algorithm	Parameter tuning	+ +	- -	-
	Training time	+	- -	+
	Data requirements	-	+	-

Table 9.6: Overall evaluation of BC, AIRL and AVRIL

The properties of all three algorithms have now been thoroughly evaluated and discussed. However, one question remains: Under which circumstances should one choose an offline IL algorithm like BC, an online IRL method like AIRL, or a classical RL approach that requires the manual definition of a reward function?

The answer to this question is highly dependent on the setup of the problem. The two main factors influencing the decision are the availability of high-quality expert data and

the complexity of the task. In this context, task complexity refers to the issue of encoding human expert knowledge into a reward function. It is further assumed, that an environment is available in which IRL and RL algorithms can interact. If such an environment is not available, offline methods provide clear advantages.

In a setting where plenty of high-quality expert data is available offline methods like BC are probably preferable, as they train faster and are more robust in parameter selection. The *compounding error* should also be minimal, given that the expert demonstrations sufficiently cover the states and action the agent may encounter. In case the *compunding error* proves to be an issue, a policy learned by BC can also be further optimised using an IRL algorithm like AIRL to excel in unseen scenarios.

If no expert data is available, one is forced to adopt RL methods to learn policies. In this case the developer or researcher has no choice other than to embed his domain knowledge into an expressive reward function and spend considerable effort in training and hyperparameter optimisation. This is feasible for small and simple problems, however, the larger the complexity of the task, the more difficult this gets.

In fact, it might be worth to generate a few high-quality expert demonstrations and apply IRL algorithms like AIRL. Especially, in tasks that are difficult to encode in a reward function, IRL methods show big advantages as they are able to learn meaningful behaviour from small samples of data. Additionally, the reward function can be analysed to assess the true objectives of the agents. Also in terms of training times, such an approach is not worse than a plain RL approach.

The summary of this is, that offline IL methods like BC should be used if a lot of high-quality expert data is available. RL methods are the right choice if no expert data is available or the task is very simple. IRL methods like AIRL excel in the area in between. It is most suitable for cases where either only insufficient expert data is available, a policy shall be further optimised w.r.t. unseen scenarios or a definition of a reward function proves to be to difficult for a human.

10 Conclusion

This chapter concludes this thesis by first revisiting the objectives and achievements of the project. The limitations of the proposed approaches and methods are summarized and suggestions for future work are provided. Finally, a conclusion is drawn.

10.1 Achievements

This thesis set out to explore the potential of IRL for decision making in ASN, using AIS-data to understand if the IRL methodology is a viable approach to remove the human definition of reward functions on the path towards full autonomy on the sea. More specifically, this work aimed at meeting three objectives:

1. Formulation of the ASN problem as an IRL problem using AIS-data
2. Development of IRL-based solutions for ASN
3. Evaluation of developed IRL-based solutions for ASN

The first objective has been met as the thesis proposes and evaluates different possibilities to translate AIS-data into a format suitable for IRL-based learning algorithms, which is a key novelty of this thesis. Among the proposed methods, two proved to be most effective. Transferring the environmental information into a graph-based description proved to be computationally efficient and allowed policies and reward functions to learn meaningful behaviours. The actions, which were originally not a part of the data, could be extracted efficiently and accurately from the AIS-data by applying a Kalman-filter.

The second objective has also been achieved by developing three different algorithms - BC, AIRL and AVRIL - for two maritime navigation tasks. BC served as performance baseline, while AIRL and AVRIL are reward learning methods. The two maritime navigation tasks were the simple GO-Navigation and the more advanced EA-Navigation tasks.

The last objective aimed at evaluating the policies and reward functions of the developed solutions. In GO-Navigation, all three methods were able to learn policies that reached a defined goal-position. Evaluation of the policies demonstrated that the IRL methods, AIRL and AVRIL, outperformed the baseline BC approach. In the more advanced task of EA-Navigation the baseline BC approach was superior in COLREG-compliance and environmental awareness compared to AVRIL and AIRL. However, AIRL still provides room for further improvement through extended training and better choice of parameters. AVRIL can be further improved by avoiding local minima of the loss-function. Additionally, the learned reward functions were analysed using suitable visualisation tools. The reward function learned by AIRL proved to be interpretable but mainly encoded the waypoint-following objective, while other navigational motives of the experts were less represented in the reward. AVRIL on the other hand, produced reward distributions with large uncertainties in the GO-Navigation task, making them not interpretable. In EA-Navigation the reward distributions collapsed to a standard normal distribution, indicating that further optimisations are necessary.

Achieving these three objectives demonstrated the viability of IRL-based solutions for ASN tasks, bringing the field of ASN closer to fully autonomous and COLREG-compliant navigation systems without the need to manually define reward functions. However, some limitations still need to be addressed on this journey to overcome some of the issues encountered in this project.

10.2 Limitations and Future Work

Some limitations of the taken approaches and chosen methods have also been identified and function as foundation for future work. The following sections introduce those limitations and propose ideas for future improvements.

Data limitations

The quantity of available AIS-data was sufficient for this project. However, many valid trajectories feature merely linear motion without considerable turning or accelerating actions. This dataset imbalance encourages policies to prioritise just moving straight. To avoid this issue a closer inspection of the AIS-data is necessary to achieve a more equal distribution between scenarios with linear movements and scenarios with actual evasive actions.

Additionally, trajectories that contained planned encounters with *pilot* vessels were part of the dataset. In hindsight, those trajectories should have been removed from the data, since it is hard to infer the vessel the *pilot* vessel is trying to approach.

Data representation limitations

Extracting and representing the information from the AIS-data in suitable format is also subject to some limitations. Especially the observation definition has aspects that might require improvement.

A fix of the introduced error in relation to the inference of actions using the Kalman-filter is the number one priority when continuing with this work. Not having the unreasonably large yaw rates in the data, should allow the policies and reward functions to learn better and faster. Nevertheless, the error did not have a significant effect on the results of this thesis.

The share of trajectories with groundings was considerably high for all policies trained for the EA-Navigation task. This indicates that the representation of depths in the graph-based observation is not ideal to learn behaviours that avoid shallow waters. One approach to improve on this, is to describe the seabed as a two-dimensional grid around the OVs position, where each cell expresses the depth at that specific location. Instead of using a GNN as feature extractor a CNN could be used. This would provide a denser representation of the seafloor.

In the current definition of the graph-based observation O_{GRA}, the proprioceptive features about the OV itself are part of the vessel graph. When the graph is processed by the GNNs, this information dissipates through aggregation with the nodes and edges from the TVs. This is evident in policies trained by AIRL which tend to apply unreasonable speeds and yaw rates to the OV. Therefore, it is sensible to add a separate feature vector to the observation that just contains proprioceptive information like the OVs COG or SOG.

The mentioned points are limitations of the current graph-based observation definition. However, during the project some more ideas have been developed w.r.t. the observation definition that might be worth investigating in the future to improve performance of policies.

The graph-based observation O_{GRA} consists of a separate graph for each feature class of the environment (e.g. vessels or buoys). Instead of keeping these separate, a heterogenous graph that models the dependencies and interactions of all those elements might be an interesting opportunity to enhance performance.

Another option to enhance the information contained in the graph-based observation O_{GRA}, is to incorporate human domain knowledge. For example, adding a feature indicating a collision or running aground of the OV should ease learning collision avoiding

behaviours in the IRL setting. Similarly, adding features encoding the obligations of the COLREGs might also be a valuable addition.

Algorithmic limitations

Some limitations and improvements are also associated with the chosen algorithms and their implementation.

First, the two IRL algorithms evaluated in this thesis are only a small share of available algorithms from the literature. Evaluating different algorithms might provide further improvements in policy performance and reward interpretability. Especially the VILD [90] and IRLEED [91] algorithms are promising as they allow to determine different expertise levels of demonstrators. This aspect is interesting to define factors contributing to good seamanship.

Not only the choice of algorithms has been restricted in the thesis. Also the choice of network architectures provides further avenues for exploration. First, different graph layers and configurations of those graph layers are worth exploring to further optimise performance. Different architectures per feature extractor instead of the same for every feature extractor can also improve performance through enhanced representational power. However, this also increases the number of parameters. Second, the current architectures base their prediction just on a state from one timestep. However, navigational decisions are best based on a sequence of previous states. To model this, recurrent network architectures operating on the concatenated outputs of the feature extractors could be used to model temporal dependencies. Long Short Term Memorys (LSTMs) are a good starting point for such investigations.

AIRL is a powerful method, but to fully excel in the EA-Navigation task some improvements are needed. As this algorithm is mainly restricted by training times and hyperparameter choices, improving those aspects may lead to great performance improvements. In particular, training the policies and reward functions for more epochs should lead to convergence to better results. Additionally, optimising the hyperparameters in less training time restrictive ranges can lead to considerable improvements.

The main training time constraint originates from the RL algorithm used. PPO requires iterative sampling from the environment and the expensive learning of a value function to compute advantage estimates. By choosing an off-policy RL algorithm like SAC computation times can be improved. However, training stability might be compromised.

The AIRL method can also be modified by augmenting the reward function. Reward terms that are easily encodable by human experts can be added to the reward function. Those terms could either penalise collisions or reward compliance to the COLREGs. This approach might ease the learning of expressive reward functions and therefore well performing policies.

Last, the main limitation of AVRIL is that the learned reward distribution does not contain any meaningful information due to its high variance or collapsing towards a standard normal distribution. To address this issue, a different hyperparameter optimisation objective can be chosen in order to achieve higher importance of the reward learning term of the loss-function in conjunction with expressive policies. Additionally, changing the coefficients of the loss-function during training via scheduler can help to escape the local minima of the reward function. Ultimately, this should lead to better policies and more expressive reward distributions. An additional area for research would be the use of a different prior in the regularisation objective of AVRIL. This might also help to learn aligning the reward distribution to something more informative.

10.3 Conclusion

Overall, this thesis demonstrates that IRL methods are a promising and viable approach to solve the ASN problem in an end-to-end fashion. Policies trained by IRL methods are able to outperform baseline BC approaches in simple navigational tasks and the learned reward functions prove to be a powerful tool to understand the navigational motives. However, some more improvements to the preprocessing of the data, the observation definitions and the algorithms are needed to also excel in more advanced navigational tasks and to achieve robust, safe and COLREG-compliant autonomy at sea using IRL methods.

Bibliography

[1] *The Metro in Copenhagen is driverless*. en. URL: https://intl.m.dk/about-the-metro/facts/trains/ (visited on 02/17/2025).

[2] *Waymo - Self-Driving Cars - Autonomous Vehicles - Ride-Hail*. en. URL: https://waymo.com/ (visited on 02/17/2025).

[3] Christopher Whitt et al. "Future Vision for Autonomous Ocean Observations". English. In: *Frontiers in Marine Science* 7 (Sept. 2020). DOI: 10.3389/fmars.2020.00697.

[4] *The Future of Maritime Autonomous Surface Ships (MASS)*. Tech. rep. IALA (International Organization for Marine Aids to Navigation), 2024. URL: https://www.iala.int/content/uploads/2024/02/The-Future-of-Mass-2024-Portrait-simples-pages-for-website-corrected.pdf (visited on 02/17/2025).

[5] *Autonomous ships The next step*. Tech. rep. Rolls-Royce plc, 2016. URL: https://www.rolls-royce.com/~/media/Files/R/Rolls-Royce/documents/%20customers/marine/ship-intel/rr-ship-intel-aawa-8pg.pdf (visited on 02/17/2025).

[6] *Convention on the International Regulations for Preventing Collisions at Sea, 1972 (COLREGs)*. URL: https://www.imo.org/en/About/Conventions/Pages/COLREG.aspx (visited on 02/17/2025).

[7] I. B. Hagen et al. "MPC-based Collision Avoidance Strategy for Existing Marine Vessel Guidance Systems". en. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, QLD: IEEE, May 2018, pp. 7618–7623. DOI: 10.1109/icra.2018.8463182.

[8] Bjorn-Olav H. Eriksen and Morten Breivik. "MPC-Based mid-level collision avoidance for asvs using nonlinear programming". en. In: *2017 IEEE Conference on Control Technology and Applications (CCTA)*. Mauna Lani Resort, HI, USA: IEEE, Aug. 2017, pp. 766–772. DOI: 10.1109/ccta.2017.8062554.

[9] Mohamed Abdelaal, Martin Fränzle, and Axel Hahn. "Nonlinear Model Predictive Control for trajectory tracking and collision avoidance of underactuated vessels with disturbances". en. In: *Ocean Engineering* 160 (July 2018), pp. 168–180. DOI: 10.1016/j.oceaneng.2018.04.026.

[10] Thanapong Phanthong et al. "Application of A* algorithm for real-time path re-planning of an unmanned surface vehicle avoiding underwater obstacles". en. In: *Journal of Marine Science and Application* 13.1 (Mar. 2014), pp. 105–116. DOI: 10.1007/s11804-014-1224-3.

[11] S. Campbell and W. Naeem. "A Rule-based Heuristic Method for COLREGS-compliant Collision Avoidance for an Unmanned Surface Vehicle". en. In: *IFAC Proceedings Volumes* 45.27 (2012), pp. 386–391. DOI: 10.3182/20120919-3-it-2046.00066.

[12] Yoshiaki Kuwata et al. "Safe Maritime Autonomous Navigation With COLREGS, Using Velocity Obstacles". In: *IEEE Journal of Oceanic Engineering* 39.1 (Jan. 2014), pp. 110–119. DOI: 10.1109/joe.2013.2254214.

[13] D. K.M. Kufoalor, E. F. Brekke, and T. A. Johansen. "Proactive Collision Avoidance for ASVs using A Dynamic Reciprocal Velocity Obstacles Method". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2018, pp. 2402–2409. DOI: 10.1109/iros.2018.8594382.

[14] Bjorn-Olav H. Eriksen et al. “Radar-based maritime collision avoidance using dynamic window”. en. In: *2018 IEEE Aerospace Conference*. Big Sky, MT: IEEE, Mar. 2018, pp. 1–9. DOI: 10.1109/aero.2018.8396666.

[15] Einvald Serigstad, Bjørn-Olav H. Eriksen, and Morten Breivik. “Hybrid Collision Avoidance for Autonomous Surface Vehicles”. en. In: *IFAC-PapersOnLine* 51.29 (2018), pp. 1–7. DOI: 10.1016/j.ifacol.2018.09.460.

[16] Roman Smierzchalski. “Evolutionary trajectory planning of ships in navigation traffic areas”. en. In: *Journal of Marine Science and Technology* 4.1 (Sept. 1999), pp. 1–6. DOI: 10.1007/s007730050001.

[17] Tomasz Praczyk. “Neural anti-collision system for Autonomous Surface Vehicle”. en. In: *Neurocomputing* 149 (Feb. 2015), pp. 559–572. DOI: 10.1016/j.neucom.2014.08.018.

[18] Agnieszka Lazarowska. “Ship’s Trajectory Planning for Collision Avoidance at Sea Based on Ant Colony Optimisation”. en. In: *The Journal of Navigation* 68.2 (Mar. 2015), pp. 291–307. DOI: 10.1017/s0373463314000708.

[19] Yanzhuo Xue et al. “Automatic simulation of ship navigation”. en. In: *Ocean Engineering* 38.17-18 (Dec. 2011), pp. 2290–2305. DOI: 10.1016/j.oceaneng.2011.10.011.

[20] Clément Pêtrès, Miguel-Angel Romero-Ramirez, and Frédéric Plumet. “Reactive path planning for autonomous sailboat”. In: *2011 15th International Conference on Advanced Robotics (ICAR)*. June 2011, pp. 112–117. DOI: 10.1109/icar.2011.6088585.

[21] Qingyang Xu, Chengjin Zhang, and Li Zhang. “Deep convolutional neural network based unmanned surface vehicle maneuvering”. In: *2017 Chinese Automation Congress (CAC)*. Oct. 2017, pp. 878–881. DOI: 10.1109/cac.2017.8242889.

[22] Yuanyuan Qiao et al. *Survey of Deep Learning for Autonomous Surface Vehicles in the Marine Environment*. Jan. 2023. DOI: 10.48550/arXiv.2210.08487.

[23] Joohyun Woo, Chanwoo Yu, and Nakwan Kim. “Deep reinforcement learning-based controller for path following of an unmanned surface vehicle”. en. In: *Ocean Engineering* 183 (July 2019), pp. 155–166. DOI: 10.1016/j.oceaneng.2019.04.099.

[24] Le Pham Tuyen et al. “Deep reinforcement learning algorithms for steering an underactuated ship”. en. In: *2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. Daegu: IEEE, Nov. 2017, pp. 602–607. DOI: 10.1109/mfi.2017.8170388.

[25] Joel Jose, Md Shadab Alam, and Abhilash Sharma Somayajula. *Navigating the Ocean with DRL: Path following for marine vessels*. Oct. 2023. DOI: 10.48550/arXiv.2310.14932.

[26] Andreas Bell Martinsen. “End-to-end training for path following and control of marine vehicles”. eng. MA thesis. Ntnu, 2018. URL: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2559484 (visited on 02/18/2025).

[27] Andreas B. Martinsen and Anastasios M. Lekkas. “Straight-Path Following for Underactuated Marine Vessels using Deep Reinforcement Learning”. en. In: *IFAC-PapersOnLine* 51.29 (2018), pp. 329–334. DOI: 10.1016/j.ifacol.2018.09.502.

[28] Andreas B. Martinsen and Anastasios M. Lekkas. “Curved Path Following with Deep Reinforcement Learning: Results from Three Vessel Models”. en. In: *OCEANS 2018 MTS/IEEE Charleston*. Charleston, SC: IEEE, Oct. 2018, pp. 1–8. DOI: 10.1109/oceans.2018.8604829.

[29] Yin Cheng and Weidong Zhang. “Concise deep reinforcement learning obstacle avoidance for underactuated unmanned marine vessels”. en. In: *Neurocomputing* 272 (Jan. 2018), pp. 63–73. DOI: 10.1016/j.neucom.2017.06.066.

[30] Joohyun Woo and Nakwan Kim. “Collision avoidance for an unmanned surface vehicle using deep reinforcement learning”. en. In: *Ocean Engineering* 199 (Mar. 2020), p. 107001. DOI: 10.1016/j.oceaneng.2020.107001.

[31] Ingunn Johanne Vallestad. “Path Following and Collision Avoidance for Marine Vessels with Deep Reinforcement Learning”. eng. MA thesis. Ntnu, 2019. URL: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2625707 (visited on 02/18/2025).

[32] Siyu Guo et al. “An Autonomous Path Planning Model for Unmanned Ships Based on Deep Reinforcement Learning”. en. In: *Sensors* 20.2 (Jan. 2020), p. 426. DOI: 10.3390/s20020426.

[33] Thomas Nakken Larsen et al. *Risk-based implementation of COLREGs for autonomous surface vehicles using deep reinforcement learning*. Nov. 2021. DOI: 10.48550/arXiv.2112.00115.

[34] Eivind Meyer et al. *Taming an autonomous surface vehicle for path following and collision avoidance using deep reinforcement learning*. Dec. 2019. DOI: 10.48550/arXiv.1912.08578.

[35] Eivind Meyer et al. *COLREG-Compliant Collision Avoidance for Unmanned Surface Vehicle using Deep Reinforcement Learning*. June 2020. DOI: 10.48550/arXiv.2006.09540.

[36] Wei Guan, Zhewen Cui, and Xianku Zhang. “Intelligent Smart Marine Autonomous Surface Ship Decision System Based on Improved PPO Algorithm”. en. In: *Sensors* 22.15 (Jan. 2022), p. 5732. DOI: 10.3390/s22155732.

[37] Luman Zhao and Myung-Il Roh. “COLREGs-compliant multiship collision avoidance based on deep reinforcement learning”. en. In: *Ocean Engineering* 191 (Nov. 2019), p. 106436. DOI: 10.1016/j.oceaneng.2019.106436.

[38] Thomas Nakken Larsen et al. “Comparing Deep Reinforcement Learning Algorithms’ Ability to Safely Navigate Challenging Waters”. English. In: *Frontiers in Robotics and AI* 8 (Sept. 2021). DOI: 10.3389/frobt.2021.738113.

[39] Alexandra Vedeler and Narada Warakagoda. “Generative Adversarial Immitation Learning for Steering an Unmanned Surface Vehicle”. en. In: *Proceedings of the Northern Lights Deep Learning Workshop* 1 (Feb. 2020), p. 6. DOI: 10.7557/18.5147.

[40] Jonathan Ho and Stefano Ermon. *Generative Adversarial Imitation Learning*. June 2016. DOI: 10.48550/arXiv.1606.03476.

[41] John Schulman et al. *Trust Region Policy Optimization*. Apr. 2017. DOI: 10.48550/arXiv.1502.05477.

[42] Pieter Abbeel and Andrew Y. Ng. “Apprenticeship learning via inverse reinforcement learning”. en. In: *Twenty-first international conference on Machine learning - ICML ’04*. Banff, Alberta, Canada: ACM Press, 2004, p. 1. DOI: 10.1145/1015330.1015430.

[43] Mao Zheng et al. *Research on autonomous collision avoidance of merchant ship based on inverse reinforcement learning*. en. 2020. DOI: 10.1177/1729881420969081.

[44] Takefumi Higaki, Hirotada Hashimoto, and Hitoshi Yoshioka. “Investigation and Imitation of Human Captains’ Maneuver Using Inverse Reinforcement Learning”. en. In: *Journal of the Japan Society of Naval Architects and Ocean Engineers* 36 (2022), pp. 137–148. DOI: 10.2534/jjasnaoe.36.137.

[45] Brian D Ziebart et al. “Maximum Entropy Inverse Reinforcement Learning”. In: *AAAI Conference on Artificial Intelligence* (2008). DOI: 10.5555/1620270.1620297. URL: https://cdn.aaai.org/AAAI/2008/AAAI08-227.pdf.

[46] Takefumi Higaki and Hirotada Hashimoto. “Human-like route planning for automatic collision avoidance using generative adversarial imitation learning”. en. In: *Applied Ocean Research* 138 (Sept. 2023), p. 103620. DOI: 10.1016/j.apor.2023.103620.
[47] John Schulman et al. *Proximal Policy Optimization Algorithms*. Aug. 2017. DOI: 10.48550/arXiv.1707.06347.
[48] Piyabhum Chaysri et al. “Unmanned surface vehicle navigation through generative adversarial imitation learning”. en. In: *Ocean Engineering* 282 (Aug. 2023), p. 114989. DOI: 10.1016/j.oceaneng.2023.114989.
[49] Lingyu Li, Yong Ma, and Defeng Wu. “Underactuated MSV path following control via stable adversarial inverse reinforcement learning”. en. In: *Ocean Engineering* 299 (May 2024), p. 117368. DOI: 10.1016/j.oceaneng.2024.117368.
[50] Justin Fu, Katie Luo, and Sergey Levine. *Learning Robust Rewards with Adversarial Inverse Reinforcement Learning*. Aug. 2018. DOI: 10.48550/arXiv.1710.11248.
[51] *Python 3.12 documentation*. en. URL: https://docs.python.org/3/ (visited on 05/01/2025).
[52] Jason Ansel et al. “PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation”. en. In: *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*. La Jolla CA USA: Acm, Apr. 2024, pp. 929–947. DOI: 10.1145/3620665.3640366.
[53] Mark Towers et al. *Gymnasium: A Standard Interface for Reinforcement Learning Environments*. Nov. 2024. DOI: 10.48550/arXiv.2407.17032.
[54] Matthias Fey and Jan Eric Lenssen. *Fast Graph Representation Learning with PyTorch Geometric*. Apr. 2019. DOI: 10.48550/arXiv.1903.02428.
[55] Charles R. Harris et al. “Array Programming with NumPy”. In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2.
[56] Sean Gillies et al. *Shapely*. Apr. 2025. URL: https://github.com/shapely/shapely.
[57] John D. Hunter. “Matplotlib: A 2D Graphics Environment”. In: *Computing in Science & Engineering* 9.3 (May 2007), pp. 90–95. DOI: 10.1109/mcse.2007.55.
[58] Lukas Biewald. *Experiment Tracking with Weights and Biases*. Software available from wandb.com. 2020. URL: https://www.wandb.com/.
[59] Richard S. Sutton and Andrew Barto. *Reinforcement learning: an introduction*. en. Second edition. Adaptive computation and machine learning. Cambridge, Massachusetts London, England: The MIT Press, 2020. ISBN: 978-0-262-03924-6.
[60] Nathan Gavenski et al. *A Survey of Imitation Learning Methods, Environments and Metrics*. July 2024. DOI: 10.48550/arXiv.2404.19456.
[61] Hado van Hasselt. *Lecture 9: Policy Gradients and Actor Critics*. en. 2021. URL: https://storage.googleapis.com/deepmind-media/UCL%20x%20DeepMind%202021/Lecture%209-%20Policy%20gradients%20and%20actor%20critics.pdf (visited on 06/08/2025).
[62] *Trust Region Policy Optimization — Spinning Up documentation*. URL: https://spinningup.openai.com/en/latest/algorithms/trpo.html (visited on 03/28/2025).
[63] *Proximal Policy Optimization — Spinning Up documentation*. URL: https://spinningup.openai.com/en/latest/algorithms/ppo.html (visited on 03/28/2025).
[64] Timothy P. Lillicrap et al. *Continuous control with deep reinforcement learning*. July 2019. DOI: 10.48550/arXiv.1509.02971.
[65] *Deep Deterministic Policy Gradient — Spinning Up documentation*. URL: https://spinningup.openai.com/en/latest/algorithms/ddpg.html#references (visited on 03/28/2025).

[66] Scott Fujimoto, Herke van Hoof, and David Meger. *Addressing Function Approximation Error in Actor-Critic Methods*. Oct. 2018. DOI: 10.48550/arXiv.1802.09477.

[67] *Twin Delayed DDPG — Spinning Up documentation*. URL: https://spinningup.openai.com/en/latest/algorithms/td3.html (visited on 03/28/2025).

[68] Tuomas Haarnoja et al. *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor*. Aug. 2018. DOI: 10.48550/arXiv.1801.01290.

[69] *Soft Actor-Critic — Spinning Up documentation*. URL: https://spinningup.openai.com/en/latest/algorithms/sac.html (visited on 03/28/2025).

[70] Boyuan Zheng et al. *Imitation Learning: Progress, Taxonomies and Challenges*. Oct. 2022. DOI: 10.48550/arXiv.2106.12177.

[71] Michael Bain and Claude Sammut. "A framework for behavioural cloning". en. In: *Machine Intelligence 15*. Oxford University PressOxford, Jan. 2000, pp. 103–129. DOI: 10.1093/oso/9780198538677.003.0006.

[72] Dean A. Pomerleau. "ALVINN: An Autonomous Land Vehicle in a Neural Network". In: *Proceedings of the 2nd International Conference on Neural Information Processing Systems*. Nips'88. Morgan Kaufmann, 1988, pp. 305–313. DOI: 10.5555/2969735.2969771.

[73] Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. *A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning*. Mar. 2011. DOI: 10.48550/arXiv.1011.0686.

[74] Mark Beliaev et al. *Imitation Learning by Estimating Expertise of Demonstrators*. June 2022. DOI: 10.48550/arXiv.2202.01288.

[75] Siddharth Reddy, Anca D. Dragan, and Sergey Levine. *SQIL: Imitation Learning via Reinforcement Learning with Sparse Rewards*. Sept. 2019. DOI: 10.48550/arXiv.1905.11108.

[76] Ian J. Goodfellow et al. *Generative Adversarial Networks*. June 2014. DOI: 10.48550/arXiv.1406.2661.

[77] Chelsea Finn et al. *A Connection between Generative Adversarial Networks, Inverse Reinforcement Learning, and Energy-Based Models*. Nov. 2016. DOI: 10.48550/arXiv.1611.03852.

[78] Faraz Torabi, Garrett Warnell, and Peter Stone. *Generative Adversarial Imitation from Observation*. June 2019. DOI: 10.48550/arXiv.1807.06158.

[79] Yunzhu Li, Jiaming Song, and Stefano Ermon. *InfoGAIL: Interpretable Imitation Learning from Visual Demonstrations*. Nov. 2017. DOI: 10.48550/arXiv.1703.08840.

[80] Stephen Adams, Tyler Cody, and Peter A. Beling. "A survey of inverse reinforcement learning". en. In: *Artificial Intelligence Review* 55.6 (Aug. 2022), pp. 4307–4346. DOI: 10.1007/s10462-021-10108-x.

[81] Andrew Y. Ng and Stuart J. Russell. "Algorithms for Inverse Reinforcement Learning". In: *Proceedings of the Seventeenth International Conference on Machine Learning*. Icml '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., June 2000, pp. 663–670. DOI: 10.5555/645529.657801.

[82] Nathan D. Ratliff, J. Andrew Bagnell, and Martin A. Zinkevich. "Maximum margin planning". en. In: *Proceedings of the 23rd international conference on Machine learning - ICML '06*. Pittsburgh, Pennsylvania: ACM Press, 2006, pp. 729–736. DOI: 10.1145/1143844.1143936.

[83] Umar Syed and Robert E Schapire. "A Game-Theoretic Approach to Apprenticeship Learning". In: *Advances in Neural Information Processing Systems*. Vol. 20. Curran Associates, Inc., 2007. URL: https://proceedings.neurips.cc/paper_files/

paper/2007/hash/ca3ec598002d2e7662e2ef4bdd58278b-Abstract.html (visited on 06/07/2025).

[84] Gergely Neu and Csaba Szepesvari. *Apprenticeship Learning using Inverse Reinforcement Learning and Gradient Methods*. June 2012. DOI: 10.48550/arXiv.1206.5264.

[85] Deepak Ramachandran and Eyal Amir. "Bayesian Inverse Reinforcement Learning". en. In: *Proceedings of the 20th International Joint Conference on Artifical Intelligence*. Ijcai'07 (2007), pp. 2586–2591. DOI: 10.5555/1625275.1625692.

[86] Jaedeug Choi and Kee-eung Kim. "MAP Inference for Bayesian Inverse Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. Vol. 24. Curran Associates, Inc., 2011. URL: https://papers.nips.cc/paper_files/paper/2011/hash/3a15c7d0bbe60300a39f76f8a5ba6896-Abstract.html (visited on 06/06/2025).

[87] Alex J. Chan and Mihaela van der Schaar. *Scalable Bayesian Inverse Reinforcement Learning*. Mar. 2021. DOI: 10.48550/arXiv.2102.06483.

[88] Abdeslam Boularias, Jens Kober, and Jan Peters. "Relative Entropy Inverse Reinforcement Learning". en. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, June 2011, pp. 182–189. URL: https://proceedings.mlr.press/v15/boularias11a.html (visited on 06/07/2025).

[89] Chelsea Finn, Sergey Levine, and Pieter Abbeel. *Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization*. May 2016. DOI: 10.48550/arXiv.1603.00448.

[90] Voot Tangkaratt et al. *VILD: Variational Imitation Learning with Diverse-quality Demonstrations*. Sept. 2019. DOI: 10.48550/arXiv.1909.06769.

[91] Mark Beliaev and Ramtin Pedarsani. *Inverse Reinforcement Learning by Estimating Expertise of Demonstrators*. Dec. 2024. DOI: 10.48550/arXiv.2402.01886.

[92] Ilya Kostrikov et al. *Discriminator-Actor-Critic: Addressing Sample Inefficiency and Reward Bias in Adversarial Imitation Learning*. Oct. 2018. DOI: 10.48550/arXiv.1809.02925.

[93] *AIS transponders*. URL: https://www.imo.org/en/OurWork/safety/navigation/ais.aspx (visited on 05/18/2025).

[94] Ties Emmens et al. "The promises and perils of Automatic Identification System data". en. In: *Expert Systems with Applications* 178 (Sept. 2021), p. 114975. DOI: 10.1016/j.eswa.2021.114975.

[95] *Revised Guidelines For The Onboard Operational Use Of Shipborne Automatic Identification Systems (Ais)*. Dec. 2015. URL: https://wwwcdn.imo.org/localresources/en/OurWork/Safety/Documents/AIS/Resolution%20A.1106(29).pdf (visited on 05/18/2025).

[96] Abbas Harati-Mokhtari et al. "Automatic Identification System (AIS): Data Reliability and Human Error Implications". en. In: *Journal of Navigation* 60.3 (Sept. 2007), pp. 373–389. DOI: 10.1017/s0373463307004298.

[97] *Automatic Identification System (AIS) Overview | Navigation Center*. URL: https://www.navcen.uscg.gov/automatic-identification-system-overview (visited on 04/05/2025).

[98] *Main Ports (Locations Only)*. en. URL: http://marine-analyst.eu/dev.py?N=simple&O=570&titre_chap=&titre_page=portlocations&maxlat=58&maxlon=13.16&minlon=6.94&minlat=54.25&visit=570 (visited on 04/05/2025).

[99] Thor I. Fossen. “Kinematics”. en. In: *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons, Ltd, 2011, pp. 15–44. ISBN: 978-1-119-99413-8. DOI: 10.1002/9781119994138.ch2.

[100] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. eng. The MIT Press, 2006. ISBN: 0-262-20162-3 0-262-30380-9 978-0-262-20162-9 978-0-262-30380-4.

[101] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. Jan. 2017. DOI: 10.48550/arXiv.1412.6980.

[102] Peter Nicholas Hansen. “Situational Awareness for Autonomous Marine Vessels”. en. Doctoral Thesis. Kongens Lyngby: Technical University of Denmark, 2023.

[103] Benjamin Sanchez-Lengeling et al. “A Gentle Introduction to Graph Neural Networks”. en. In: *Distill* 6.9 (Sept. 2021), e33. DOI: 10.23915/distill.00033.

[104] Justin Gilmer et al. *Neural Message Passing for Quantum Chemistry*. June 2017. DOI: 10.48550/arXiv.1704.01212.

[105] Weihua Hu et al. *Strategies for Pre-training Graph Neural Networks*. Feb. 2020. DOI: 10.48550/arXiv.1905.12265.

[106] Petar Veličković et al. *Graph Attention Networks*. Feb. 2018. DOI: 10.48550/arXiv.1710.10903.

[107] Shengyi Huang et al. *The 37 Implementation Details of Proximal Policy Optimization*. Tech. rep. 2022. URL: https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/ (visited on 04/15/2025).

[108] John Schulman et al. *High-Dimensional Continuous Control Using Generalized Advantage Estimation*. Oct. 2018. DOI: 10.48550/arXiv.1506.02438.

[109] Aasa Feragen. *Lecture notes on explainable AI: Saliency maps*. Lecture Notes. Technical University of Denmark, 2024.

[110] *GINEConv — pytorch_geometric documentation*. URL: https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.nn.conv.GINEConv.html#torch_geometric.nn.conv.GINEConv (visited on 06/22/2025).

[111] *NNConv — pytorch_geometric documentation*. URL: https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.nn.conv.NNConv.html#torch_geometric.nn.conv.NNConv (visited on 06/22/2025).

[112] *GATConv — pytorch_geometric documentation*. URL: https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.nn.conv.GATConv.html#torch_geometric.nn.conv.GATConv (visited on 06/22/2025).

A AIS-Data

Instance	Description	Data Type	Values
Scenario ID	Unique scenario identifier	`string`	
Start Time	Start time of scenario in UTC	`string`	
End Time	End Time of Scenario in UTC	`string`	
Start Epoch	Start UTC-timestamp of scenario	`float`	
End Epoch	End UTC-timestamp of scenario	`float`	
Sam. Time	Sampling time of vessel state	`integer`	10
Lat. Limits	Latitudinal range of scenario	(`float`, `float`)	
Lon. Limits	Longitudinal range of scenario	(`float`, `float`)	
Num. Vessels	Number of vessels in scenario	`integer`	1-13

Table A.1: Scenario metadata features

Vessel Metadata			
Instance	**Description**	**Data Type**	**Values**
MMSI	MMSI of vessel	`int`	
Ship Type	AIS-conform ship type	`string`	
Width	Vessel width in meters	`float`	2.0-62.0
Length	Vessel length in meters	`float`	8.0-446.92
Draught	Vessel draught in meters	`float`	0.1-16.5
Nav. Status	AIS-conform nav. status	`int`	0-15
Vessel State (Trajectory is list of states)			
Instance	**Description**	**Data Type**	**Values**
Time	UTC-timestamp of state	`float`	
Lat.	Latitudinal position of vessel	`float`	
Lon.	Longitudinal position of vessel	`float`	
SOG	SOG of vessel in knots	`float`	2.0-40.4
COG	COG of vessel in degree	`float`	0.0-360.0

Table A.2: Vessel trajectory features

Instance	Description	Data Type	Values
Multiple instances per Seachart			
Bounds	Scenario bounds in (Lat.,Lon.)	`polygon`	
Depth	Depth polygon in (Lat.,Lon.)	`polygon`	
	Depth range in meters	(`float`, `float`)	0.0-100.0
Route	Route line in (Lat.,Lon.)	`line`	
	Route code	`string`	
Buoy	Buoy point in (Lat.,Lon.)	`point`	
	Buoy code	`string`	
Land	Land polygon in (Lat.,Lon.)	`polygon`	

Table A.3: Seachart features

B Observation Parameters

Observation specific		
Symbol	**Parameter**	**Value**
m_L	Number of Shore-Nodes per Polygon	5
m_D	Number of Depth-Nodes per km^2	0.1
Scaling		
Symbol	**Parameter**	**Value**
ρ_{h_d}	Vessel draught offset	10 m
χ_{h_d}	Vessel draught scaling	10 m
ρ_u	Surge velocity offset	0 kn
χ_u	Surge velocity scaling	25 kn
ρ_d	Distance offset	10 km
χ_d	Distance scaling	10 km
ρ_l	Vessel length offset	200 m
χ_l	Vessel length scaling	200 m
ρ_w	Vessel width offset	50 m
χ_w	Vessel width scaling	50 m
$\rho_{u_{rel}}$	Rel. velocity offset	25 kn
$\chi_{u_{rel}}$	Rel. velocity scaling	25 kn
ρ_{h_w}	Depth offset	50 m
χ_{h_w}	Depth scaling	100 m
$\min_v$	Minimum of Vessel Type	0
$\max_v$	Maximum of Vessel Type	11
$\min_s$	Minimum of Navigation Status	0
$\max_s$	Maximum of Navigation Status	15

Table B.1: Parameters of graph-based observation O_{GRA}

C Action Parameters

Action specific specific		
Symbol	**Parameter**	**Value**
σ_N^2	North position variance	50 m
σ_E^2	East position variance	50 m
σ_u^2	Surge velocity variance	0.01 m s^{-1}
σ_ψ^2	Course variance	0.01°
$\sigma_{\dot{u}}^2$	Acceleration variance	0.3 m s^{-2}
σ_r^2	Yaw rate variance	7.5 ° s^{-1}
Scaling		
Symbol	**Parameter**	**Value**
$\chi_{\dot{u}}$	Acceleration scaling	0.3 m s^{-2}
χ_r	Yaw rate scaling	7.5 ° s^{-1}

Table C.1: Parameters of acceleration-yaw rate action A_{kf}

Please note, that these parameters correspond to the erroneous version used throughout the report.

D Scaling functions

Linear scaling

$$\hat{v} = \text{clip}\left(\frac{v-\rho}{\chi},\ -1,\ 1\right) \tag{D.1}$$

where ρ is the offset and χ is the scaling factor.

Min-Max scaling

$$\hat{v} = \text{clip}\left(2 \cdot \left(\frac{v - \text{min}}{\text{max} - \text{min}}\right) - 1,\ -1,\ 1\right) \tag{D.2}$$

where max and min are the maximum and minimum values of the feature.

Power scaling

$$\hat{v} = \text{sign}\left(\text{clip}\left(\frac{v}{\chi},\ -1,\ 1\right)\right) \cdot \left[\text{abs}\left(\text{clip}\left(\frac{v}{\chi},\ -1,\ 1\right)\right)\right]^{0.5} \tag{D.3}$$

where χ is the scaling factor.

E Graph layers

This section shall provide further insights into the definitions of the graph layers used in this report. For a general understanding of GNNs please refer to [103]. The implementation uses pytorch-geometric [54].

GINEConv

$$x_i' = h_\Theta \left((1+\epsilon) \cdot x_i + + \sum_{j \in \mathcal{N}(i)} \text{ReLU}(x_j + e_{j,i}) \right) \tag{E.1}$$

where x_i is the node feature vector, $e_{i,j}$ is the edge feature vector, h_Θ is a neural network and ϵ is a learnable parameter. The definition is based on [110].

NNConv

$$x_i' = l_\theta(x_i) + \sum_{j \in \mathcal{N}(i)} x_j \cdot h_\Theta(e_{i,j}) \tag{E.2}$$

where x_i is the node feature vector, $e_{i,j}$ is the edge feature vector, h_Θ is a neural network and l_θ is a linear transformation. The definition is based on [111].

GATConv

$$x_i' = \alpha_{i,i}\ l_s(x_i) + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j}\ l_t(x_i) \tag{E.3}$$

$$\alpha_{i,j} = \frac{\text{exp}\left(\text{LeakyReLU}(a_s^T\ l_s(x_i) + a_t^T\ l_t(x_j) + a_e^T\ l_e(e_{i,j})\right)}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} \text{exp}\left(\text{LeakyReLU}(a_s^T\ l_s(x_i) + a_t^T\ l_t(x_k) + a_e^T\ l_e(e_{i,k})\right)} \tag{E.4}$$

where x_i is the node feature vector, $e_{i,j}$ is the edge feature vector, l_s, l_t and l_e are linear transformations and a_s, a_t and a_e are attention weights. The definition is based on [112].

ConcatConv

$$x_i' = l_\theta(x_i) + \text{aggregation}_{j \in \mathcal{N}(i)}(h_\Theta(\text{concat}(e_{i,j},\ x_j))) \tag{E.5}$$

where x_i is the node feature vector, $e_{i,j}$ is the edge feature vector, h_Θ is a neural network, l_θ is a linear transformation and aggregation is a aggregation function of type *max*, *mean* or *sum*. This layer is a custom implementation.

Technical
University of
Denmark

Ørsteds Plads, Building 343
2800 Kgs. Lyngby
Tlf. 4525 1700

electro.dtu.dk