Towards Autonomous Inland Shipping: Reinforcement-Learning-Based Control with Data-Driven Validation Schemes

Niklas Paulig

Dissertation to achieve the academic degree

Doctor rerum politicarum (Dr. rer. pol.)

Submitted to the "Friedrich List" Faculty of Transport and Traffic Sciences
of the Dresden University of Technology

Submitted on: November 5, 2025

Reviewer:
Prof. Dr. Ostap Okhrin (Dresden University of Technology)
Prof. Dr. Rudy R. Negenborn (Delft University of Technology)

Abstract

Inland waterway transport (IWT) promises lower greenhouse gas emissions and external costs than road or rail. Nevertheless, its market share remains modest because dense traffic, narrow fairways, aging infrastructure, labor shortages, and increasingly erratic water levels constrain reliability and competitiveness. This thesis investigates whether deep reinforcement learning (DRL) paired with high-quality Automatic Identification System (AIS) data can automate the control of inland waterway crafts reliably. First, we formulate guidance and control for inland vessels as a hierarchical sequential-decision problem. A bootstrapped DRL follower is shown to track complex, curved river paths under full environmental influences and shallow water hydrodynamics with significantly lower cross-track error than a tuned Proportional-Integral-Derivative (PID) controller under equal conditions. Second, we introduce a two-agent architecture that couples insights from the follower with a spatiotemporal DRL planner that proposes real-time rule-compliant waypoints. Extensive testing on real-world data demonstrates safe collision avoidance and compliance with COLREG-like inland rules across high traffic densities. Third, recognizing that realistic evaluation hinges on clean behavioral data, we contribute the α-method, an unsupervised AIS-trajectory extractor that replaces expert-dependent heuristics with vessel-size-scaled quantile filters. Applied to a North and Baltic Sea traffic data set, it consistently yields long and realistic trajectories and is released as the open-source Python package pytsa. Summarizing, these results show that when fed reliable AIS-derived scenarios, DRL can deliver robust, low-latency guidance and control tailored to inland constraints, fostering viable and safe autonomous IWT.

Zusammenfassung

Der Binnenschiffsverkehr verspricht geringere Treibhausgasemissionen und externe Kosten als der Straßen- oder Schienenverkehr. Dennoch bleibt sein Marktanteil beschränkt, da dichter Verkehr, enge, teils komplexe Fahrrinnengeometrie, veraltete Infrastruktur, Arbeitskräftemangel und zunehmend unregelmäßige Wasserstände die Zuverlässigkeit und Wettbewerbsfähigkeit einschränken. Diese Arbeit untersucht, ob Deep Reinforcement Learning (DRL) in Verbindung mit aufbereiteten Daten des Automatischen Identifikationssystems (AIS) die Steuerung von Binnenschiffen zuverlässig automatisieren kann. Zunächst formulieren wir die Steuerung und Kontrolle von Binnenschiffen als hierarchisches sequentielles Entscheidungsproblem. Ein spezialisierter DRL-Pfadfolger zeigt, dass er komplexen, gekrümmten Flussläufen unter vollständigen Umwelteinflüssen und in seichtem Wasser mit deutlich geringeren Querfehlern verfolgen kann als ein abgestimmter Proportional-Integral-Derivative (PID)-Regler. Anschließend führen wir eine Zwei-Agenten-Architektur ein, die die Erkenntnisse des Pfadfolgers mit einem räumlich-zeitlichen DRL-Planer koppelt, der regelkonforme Wegpunkte in Echtzeit vorschlägt. Umfangreiche Tests mit realen Daten belegen eine sichere Kollisionsvermeidung und die Einhaltung von COLREG-ähnlichen Kollisionsvermeidungsregeln auch bei hoher Verkehrsdichte. Weiterhin erkennen wir, dass zur Überprüfung solcher Algorithmen und zur realisitischen Einschätzung maritimer Verkehrslagen generell sauber aufbereitete AIS Trajektorien von wichtiger Bedeutung sind. Daher stellen wir die α-Methode vor, einen vollständing datengetriebenen AIS-Trajektorien-Extraktor, der bisher vorherrschende Heuristiken durch quantile Filter ersetzt, die die Zugehörigkeit von Datenpunkten zu einer Trajektorie anhand der Manövrierfähigkeit des zu Grunde liegenden Schiffes abschätzt. Angewandt auf einen Verkehrsdatensatz der Nord- und Ostsee liefert er durchweg lange und realistische Trajektorien und wird als Open-Source-Python-Paket pytsa veröffentlicht. Diese Ergebnisse zeigen, dass DRL bei Verwendung zuverlässiger, aus AIS-Daten abgeleiteter Szenarien robuste, latenzarme Leit- und Steuerungsfunktionen liefern kann, die auf die Einschränkungen der Binnenschifffahrt zugeschnitten sind und eine praktikable, sichere und autonome Binnenschifffahrt fördern.

Acknowledgements

This thesis is the result of just over four years of exciting research, vivid experiences, and profound challenges that have significantly shaped my academic and professional growth. I would like to express my deepest gratitude to my supervisor, Prof. Dr. Ostap Okhrin, for his inexhaustible curiosity, tireless dedication, and enviable wealth of knowledge, without whom this scientific journey would not have been possible.

I am also grateful to all my colleagues at the Chair of Statistics and the Institute of Transport and Economics for the shared memories, Lidl lunches, and numerous activities. In particular, I would like to thank my long-standing colleague and friend Martin Waltz for always lending a sympathetic ear and enriching our breaks with countless hours of table football and deep political discussions. I would also like to thank my colleague Fabian Hart for the memorable trips to Karlsruhe and the many shared laughs. My thanks also go to my colleagues at the BAW for broadening my scientific and maritime knowledge.

I am deeply indebted to my family for supporting me throughout this journey with their loving care. Finally, my heartfelt gratitude goes to my ever-loving wife Elly, who guided me through the darkest days and was always there for me, even when I was not. I love you.

Contents

List of Figures

List of Tables

Part I

Introduction

Chapter 1

Introduction

1 Inland Waterway Transport: Significance and Challenges

Inland waterway transport (IWT) plays a vital role in the global freight network, offering socio-economic and environmental advantages (Camargo-Díaz et al., 2022). It is often lauded as the most energy-efficient and low-emission mode for heavy cargo, with 40% lower greenhouse gas emissions per ton-kilometer compared to road transport (Calderón-Rivera et al., 2024). Due to the often small market share of IWT in the transport sector, there are substantial academic efforts put into leveraging this potential, such as in Europe (Plotnikova et al., 2022; Drożdż et al., 2023; Van Meir et al., 2022), America (Vilarinho et al., 2024; Hunt et al., 2022), Asia (Trivedi et al., 2021; Voskresenskaya et al., 2018; Duan et al., 2010; Dang et al., 2022; Nguyen and Nguyen, 2020), and Africa (Solomon et al., 2021).

While the aforementioned studies congruently conclude, that IWT bears significant unused potential for ecological and economical passenger and freight transportation, IWT faces persistent challenges that limit its utilization. Infrastructural bottlenecks and natural constraints affect IWT's reliability: Aging locks, canals, and dams hamper system efficiency (Baroud et al., 2014; Mircetic et al., 2017; Defryn et al., 2021), while low water levels due to climate change disrupt schedules and capacity (Riquelme-Solar et al., 2015; Scheepers et al., 2018; Bedoya-Maya et al., 2024). Crew shortages and an aging workforce further strain the sector, as younger workers increasingly avoid careers in inland navigation (Ionescu, 2016; Pfoser et al., 2018). Considering that labor costs account for approximately 26% of total operating expenses (Al Enezy et al., 2017), it becomes increasingly challenging to remain competitive in the freight market. Finally, policy and regulatory frameworks must catch up with technological and market developments. While the EU and other authorities promote inland shipping, implementation lags in some regions due to fragmented governance and slow adoption of innovations (Pfoser et al., 2018; Rogerson et al., 2020). The IWT sector's promise comes with challenges spanning environmental resilience, economic viability, infrastructure modernization, and workforce sustainability. Addressing these challenges is essential to integrate IWT as a fully integrated transport system.

2 Reinforcement Learning and Autonomous Vessel Control

One proposed way of addressing the persistent challenges in IWT is *smart shipping*, where automated technologies and Artificial Intelligence (AI) successively replace human-controlled systems and decision making (Pauwelyn and Turf, 2022; Restrepo-Arias et al., 2022). We will come back to this application soon.

Nowadays, AI is a broadly used term for referring to a system's capacity of solving problems that require human intelligence (Jiang et al., 2022), although methodologically, almost all modern approaches rely on machine learning (ML). ML comprises a range of algorithms that learn to perform specific tasks without being explicitly programmed (Mahesh et al., 2020), but rather by uncovering patterns and structure in the data used for their training. ML is commonly structured into three core branches: supervised learning, unsupervised learning, and reinforcement learning (RL). Supervised learning models are trained on labeled data to approximate a functional relationship between inputs and outputs. In contrast, unsupervised learning seeks to reveal latent structures within unlabeled datasets. Parts of this thesis are concerned with the third pillar, reinforcement learning, which addresses sequential decision-making through trial-and-error interaction with an environment.

At every time step, an RL agent perceives the current state, takes an action, and receives feedback in the form of a reward. The aim is to discover a policy that yields the highest possible total reward in the long run, often using a discount factor to weigh future rewards. RL can be understood as a synthesis of several once-separate research domains. Behavioral psychology contributes the core notions of stimulus $\rightarrow$ response $\rightarrow$ reinforcement, formalized by Skinner (1938), which first framed learning as the gradual shaping of behavior by rewards and punishments. Mathematics enters through dynamic programming and Markov decision processes (MDPs), pioneered by Bellman (1954) and later expanded in operations-research texts, e.g. Howard (1960). These provide the framework that defines what it means for an agent to optimize its sequential actions. Modern RL also leans on statistics and probability, notably inference under uncertainty (Puterman, 2014), which eventually led to the emergence and first successful implementation of deep reinforcement learning (DRL), a combined approach leveraging the approximation powers of neural networks as policies (Mnih et al., 2015), leading to breakthroughs across various domains, from video games (Berner et al., 2019; Vinyals et al., 2019; Wurman et al., 2022) and board games (Silver et al., 2017, 2018) to protein structure prediction (Jumper et al., 2021), nuclear fusion reactor control (Degrave et al., 2022) and fine-tuning of large language models (Bai et al., 2022).

Narrowing the scope to the transportation domain, significant DRL contributions have been made in many fields, mainly in traffic signal control (Chu et al., 2019; Li et al., 2021b), and autonomous driving for cars (Sallab et al., 2017; Zhu et al., 2020), but also for pricing models for toll roads (Pandey and Boyles, 2018; Pandey et al., 2020), or network delay optimization for trains (Šemrov et al., 2016; Mohanty et al., 2020). Despite its success in land transport, RL research on waterborne transport in general still is in its infancy, however centered around automated control (Chun et al., 2021; Guo et al., 2020).

In order to automatically control a vessel, Fossen (2021) defines the need for a three-

component system performing the tasks of *guidance*, *navigation* and *control*. This thesis adopts this terminology. Guidance systems generate reference paths or trajectories to be used by the vessel's motion control system as a target. Navigation systems locate the vessel's position and orientation and sense its state, like velocity or acceleration in its environment. Closing the loop, the control system utilizes the information from the guidance system, to generate the necessary control forces and moments to move the vessel towards its control target. We will focus on guidance and control.

In the literature of vessel automation, *guidance* is implemented as path or trajectory planning. Throughout this thesis we refer to a path as a set of positions and a trajectory as a set of time-position tuples. Central aim of this planning process is the avoidance of static (non-moving) obstacles, like bridges, banks, buoys or platforms and dynamic (moving) obstacles like other traffic participants, such as vessels or swimmers. For open-sea shipping, there already exist solutions that find collision-free paths (Chen et al., 2019; Guo et al., 2020), some even under additional consideration of international maritime collision avoidance rules (COLREGs) (Li et al., 2021a; Xu et al., 2022a). *Control*, on the other hand, refers to path following or propulsion control, executing the plan by adjusting rudder and throttle to keep the vessel on course (Woo et al., 2019; Zhao et al., 2020b; Hart et al., 2023a)

3 Challenges on inland waterways

Path-following and path planning on inland waterways are more complex than at sea because the autonomy system must operate inside a dense-traffic, narrow, shallow, fast-changing "tube" rather than an open plane. The fairway's hard banks or bridge pillars leave only meter-level lateral margins, so path planning algorithms must resolve encounters at a much shorter range and followers must execute changes in plan at higher frequencies. Planners cannot rely on large evasive maneuvers that are acceptable offshore (Orzechowski et al., 2024). River currents accelerate in bends and vary across the cross-section, producing a continually changing set and drift (Tang et al., 2020). Even modern followers that model the flow explicitly still see their cross-track error rise by an order of magnitude when the current strengthens (Vanneste et al., 2022). Confined and shallow water significantly alters ship hydrodynamics, e.g., squat, bank suction, and asymmetric pressure fields reduce yaw authority and thus reaction time, so line-of-sight or proportional-integral-derivative controllers tuned for deep water become unstable or too slow to react (Tran et al., 2023).

Furthermore, testing path planning algorithms in simulation with real-world scenarios requires accurate, high-frequency information about other vessel's properties. While vessel motion is mandatorily recorded via an Automatic Identification System, it is known that this data is often erroneous and unreliable, requiring additional efforts for proper testing (Yan et al., 2020b; Guo et al., 2021a). Finally, path-planning methods that work offshore are expected to return infeasible tracks in rivers due to the geospatial constraints. Therefore, inland-specific planners and followers are required to embed channel geometry, shallow water maneuvering models, fast reaction times and rule-compliant overtaking logic to stay safe in the cluttered fairway.

4 Contributions

This thesis makes two key contributions. First, it shows that DRL is well suited for autonomous inland vessel path-following and path-planning tasks, thanks to its ability to learn complex, adaptive behaviors and compute actions with low inference time. Second, it demonstrates that properly processed shipborne AIS data can substantially enhance the robustness of these tasks by providing high-quality, real-world testing scenarios. The remainder of this thesis is organized as follows. Chapter 2 in Part II provides a concise overview of the three constituent papers and their publication statuses. Chapters 3 through 5 present the full publications detailing the DRL-based path-following method, the two-level path-planning framework and its AIS-based trajectory extraction. Finally, Chapter 6 concludes with a discussion of the main findings and an outlook on future research. The publications in detail:

Chapter 3 investigates the path-following capabilities of DRL systems for autonomous inland vessels, explicitly accounting for environmental disturbances such as strong directional currents and impaired maneuverability in shallow water. Research has addressed only isolated aspects of this problem, such as path-following, without considering shallow-water effects or environmental forces. In this chapter, we first implement an open-source shallow-water-aware ship maneuvering dynamic for simulation and then train a robust bootstrapped DRL algorithm by Waltz and Okhrin (2022), a standard Deep Q-Network (DQN) algorithm by Mnih et al. (2015), and a PID controller to perform the path-following objective with minimal error. The training environment uses a specialized segmental river generator to ensure heterogeneity and diversity in the observation space and thus foster generalizability. The resulting policies are validated by deploying them on real-world data from the Lower- and Middle Rhine. Our tests demonstrate that modern DRL approaches can safely navigate complex and diverse inland waterways, adapt swiftly to changing conditions, like changes in current direction, shallow water passages or impaired sensor measurements. Comparisons to other algorithms further reveal that our DRL controller constantly achieves lower tracking errors than classical RL methods and traditional control-theoretic algorithms.

Chapter 4 builds on the path-following framework of Chapter 3 by integrating a local path planner for dynamic obstacle avoidance. We introduce a two-agent DRL architecture with two consecutive steps: planning and following. The planning agent senses the vessel's environment and identifies static or moving obstacles that interfere with the pre-determined global path. When an imminent collision is detected, it computes a new, collision-free, traffic-rule-compliant set of waypoints as a divergent path and forwards it to the second agent, the path follower. The path follower builds on the aforementioned paper, incorporating water depth and currents in its observation space, and is further extended to include wind and waves. We also generalize rudder actions to a continuous action space to enable more precise control in confined waterways. For the planning agent, we adapt the spatiotemporal recurrent neural network of Waltz and Okhrin (2023) to operate in continuous action spaces. Through simulation studies, including validation against real-world AIS trajectories provided by the European Maritime Safety Agency, we demonstrate that this two-level approach reliably navigates inland vessels in complex, obstacle-rich waterways.

Chapter 5 addresses the problem of extracting coherent vessel trajectories from raw AIS data, an essential prerequisite for any downstream analysis of maritime navigation behavior, risk assessment, or planning. While AIS data is information-rich, it is also unreliable, as sensor faults, receiver outages, or malicious manipulation can produce incoherent movement patterns. Therefore, trajectory extraction from raw AIS records requires a thresholding algorithm that determines which data points per vessel should be retained to form a coherent and credible trajectory. This challenge is typically addressed in the literature using expert-derived, one-size-fits-all heuristics, which inevitably misclassify data points due to the high heterogeneity of maritime traffic participants. We, therefore, introduce the α-method, a procedure that replaces the brittle, hand-tuned thresholds that decide whether AIS data points belong to a coherent trajectory with purely data-derived α-quantiles of speed, course change, turning rate, and inter-message distance, scaled by vessel length, to (1) filter per-vessel anomalies, (2) split tracks at quantile-based breakpoints, and (3) refine for continuity. Unlike prior rule-based methods (Zhao et al., 2018), we rely on a self-contained, data-driven procedure, which we also make publicly available as a Python code base `pytsa` (Paulig, 2024). Tested on 912 days of North and Baltic Sea data (Jan 2020–Jun 2022), it reliably yields long, smooth trajectories with minimal erratic behavior.

Part II

Publications

Chapter 2

Summary

Title: Robust path following on rivers using bootstrapped reinforcement learning
Authors: Niklas Paulig, Ostap Okhrin
Journal: Ocean Engineering
Status: Published in 2024
https://doi.org/10.1016/j.oceaneng.2024.117207

Title: 2-level reinforcement learning for ships on inland waterways: Path planning and following
Authors: Martin Waltz, Niklas Paulig, Ostap Okhrin
Journal: Expert Systems with Applications
Status: Published in 2025
https://doi.org/10.1016/j.eswa.2025.126933

Title: An open-source framework for data-driven trajectory extraction from AIS data - The α-method
Authors: Niklas Paulig, Ostap Okhrin
Journal: Ocean Engineering
Status: Published in 2024
https://doi.org/10.1016/j.oceaneng.2024.119092

Chapter 3

Robust Path Following on Rivers Using Bootstrapped Reinforcement Learning

Niklas Paulig [a], *Ostap Okhrin* [a,b]

[a] *Institute of Transportation Economics, Technische Universität Dresden, 01062 Dresden, Germany*

[b] *Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI) Dresden/Leipzig, Germany*

Published in: *Ocean Engineering* (2024), 298:117207.

Abstract:
This paper develops a Deep Reinforcement Learning (DRL)-based approach for the navigation and control of autonomous surface vessels (ASVs) on inland waterways, where spatial constraints and environmental challenges such as high flow velocities and shallow banks require precise maneuvering. By implementing a state-of-the-art bootstrapped deep Q-learning (DQN) algorithm alongside a novel, flexible training environment generator, we developed a robust and accurate rudder control system capable of adapting to the dynamic conditions of inland waterways. The effectiveness of our approach is validated through comparisons with a vessel-specific Proportional-Integral-Derivative (PID) and standard DQN controller using real-world data from the lower and middle Rhine. It was found that the DRL algorithm demonstrates superior adaptability and generalizability across previously unseen scenarios and achieves high navigational accuracy. Our findings highlight the limitations of traditional control methods like PID in complex river environments, as well as the importance of training in diverse and realistic environments.

Keywords: deep reinforcement learning, path following, restricted waterways, autonomous surface vessel

1 Introduction

According to a survey on the development of the global ocean surface robot market by BIS Research (2018), the market for autonomous vessels is "expected to grow at the rate of 26.7% for the period 2024-2035". For autonomous vessels to be integrated seamlessly into existing hybrid traffic, it is crucial to fulfill their automated tasks with high accuracy. This is especially true for spatially restricted inland waterways such as rivers, bights and canals. The Directorate-General for Mobility and Transport (2023) of the European Commission emphasizes the importance of inland waterway traffic and its development due to decreased costs and increased safety in comparison to other modes of transport. To build on this directive, the present study is one of the first approaches to solving the path-following problem for underactuated vessels on restricted waterways using deep reinforcement learning (DRL) and under consideration of environmental influences. Breivik and Fossen (2004) stated, that, compared to other automated systems, ships on inland waterways face additional challenges due to their environment (e.g., strong directional currents, shallow banks) and underlying physics (e.g., underactuation, highly non-linear maneuvering models), leading to a highly dynamic and stochastic operational environment. To overcome these hurdles, we incorporate water depth, current direction, and speed into the agent's perception, allowing it to navigate tight river turns safely.

In this paper, an ensemble-based DRL algorithm is used to develop a high-precision and generalizable path-following controller for inland transportation vessels. The contribution to the field is as follows:

- We develop a tunable segmental generator to create realistic and diverse training environments specifically for inland waterways. The source code is publicly available as a GitHub repository via `github.com/nikpau/sr-gen`.
- We use a state-of-the-art bootstrapped DQN-based algorithm (Waltz and Okhrin, 2022) to generate robust and generalizable policies for rudder control under varying external environmental disturbances.
- To demonstrate the generalizability and robustness of our approach, we validate the produced policies on real-world data from the middle and lower Rhine.

The rest of the paper is organized as follows. Section 2 recapitulates current literature on the topic of path-following and formalizes the problem. The kinodynamic ship-maneuvering model is detailed in Section 3 while section 4 introduces methodologies and how they are incorporated into the path-follower controller design. The fifth section sets up a benchmark controller for validation and Section 6 applies the path-following results to various maritime scenarios and validates the controller on separate segments of the Rhine river. Section 7 performs a robustness analysis, and the last section concludes.

2 Path-following for ASV

2.1 State of the art

The objective of path-following for ships demands a controller to generate steering commands that enable an underactuated autonomous surface vessel (ASV) to follow a pre-defined path with minimal angular and spatial deviation. The problem formulation for this study will only include rudder angles as control outputs while keeping the engine revolutions constant.

According to Fossen (2021), an onboard path-following system requires three sub-systems to be implemented: *guidance*, *navigation*, and *control*. To autonomously control a vessel, we require to know its current position (navigation), planned trajectory (guidance), and a set of control actions to move towards its current goal (control).

Line-of-sight (LOS) guidance is one common approach in implementing directional awareness of the agent, achieving convergence to the desired path. It has been successfully applied in various problem settings, as in Fossen et al. (2003) and Fossen and Lekkas (2017) with traditional control approaches and Oh and Sun (2010) for a model-predictive-control application. Vector field guidance (VFG) (Nelson et al., 2007a) is a different approach that uses a global vector field encompassing the path to guide the vessel towards it, independent of the magnitude of deviation. Woo et al. (2019) integrated VFG into a combined path-following and collision avoidance method.

After setting up a suitable guidance algorithm, the next step demands a control system. There are two main methodological approaches to solving the path-following problem, analytic control, and reinforcement learning. As part of the analytic control family, proportional-integral-derivative (PID) controllers are well understood, require few computational resources and have successfully been used to develop path-followers in calm and disturbed waters. While Moreira et al. (2007) achieved path-following using a LOS guidance system and PID controller for steering control, Perera et al. (2014) used fuzzy logic to derive, and PID controller to execute sequential actions for path-following but also collision avoidance of a small model vessel. Paramesh and Rajendran (2021) used PID control to navigate a tanker along a given path under the influence of regular waves. More recent advances in control theory allow for different approaches such as non-linear model-predictive-control (Xia et al., 2013; Sandeepkumar et al., 2022), backstepping control (Zhang et al., 2017), or sliding mode control (Liu et al., 2018).

Reinforcement Learning (RL) is based on agent-environment interaction, aimed at learning a correct set of actions given some observed state. Recently, interest in academia in using RL-based motion control has surged due to its ability to tackle problems with high uncertainty, non-linear system dynamics, and self-learning capabilities (Wurman et al., 2022; Kaufmann et al., 2023). Consequently, it operates effectively without the need for prior knowledge of dynamics to fine-tune the controller. Furthermore, by directly utilizing experiential data - encompassing control inputs and the consequent dynamic response of the vehicle - the controller implicitly incorporates considerations for modeling uncertainties or the influence of environmental disturbances (Bellemare et al., 2020). The actions taken by the agent are evaluated based on a hand-crafted reward function, whose goal is to reinforce actions that bring the agent closer to its defined goal, ultimately finding a policy that solves the problem optimally. Various researchers, such as Shen

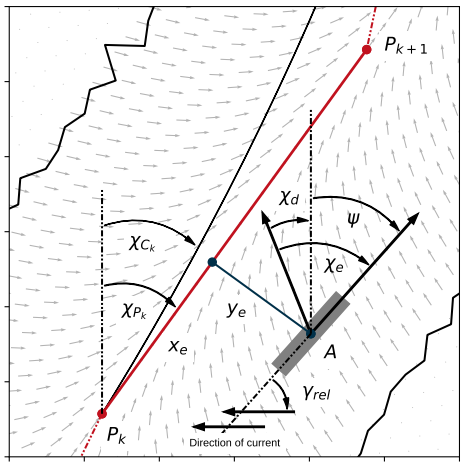


Figure 3.1: Heading control setup

and Guo (2016), Martinsen and Lekkas (2018a) use a family of continuous-action algorithms in which the agent is free to choose any action on the applicable range every time step. While this allows for a highly reactive policy, it is possible to choose actions leading to unrealistic behavior, such as maximally opposing rudder angles on two successive time steps. Discrete action solutions as discussed by Zhao et al. (2019); Amendola et al. (2019, 2020); Martinsen et al. (2020) often share the identical drawback, leading some researchers to block the agent from choosing the next action until the last one was performed subject to physical constraints of the vessel. While this approach is viable in restricting physically impossible movements, it impairs the agent's reactivity during the time of blockage. To mitigate this problem in this study, we opted only to allow the agent to choose from actions within the vessel's physical possibilities. Recent advances in RL research additionally show, that many continuous policies converge to a preference of taking actions at the boundaries of their action space. This allows the policies to be replaced by discrete ones, while still attaining equal performance (Seyde et al., 2021). Therefore, this study uses a discrete action space.

We use a PID controller as a benchmark for our proposed approach, as its assumptions and behavior match well with those of model-free reinforcement learning approaches. Both methods are model-free -they require no prior knowledge of the vessel's dynamics- and are variants of feedback control systems, which are implemented in real-world applications today like model tug-boats (You et al., 2023), or the *Roboat* (Park et al., 2020). Additionally, the computational burden of both approaches is low, allowing for a cost-effective real-time implementation.

2.2 Path following on rivers

Contrary to the open sea, rivers pose additional navigational challenges mainly due to their limited spatial extent, strong directional currents, shallow banks, and small path-curve radii.

Figure 3.1 depicts the heading-control setup used in this study. We assume a given path consisting of a discrete set of K waypoints $P_k = (x_k, y_k)^\top, k \in \{1, \ldots, K\}$, where two consecutive waypoints P_k and P_{k+1} enclose the path heading

$$\chi_{P_k} = \text{atan2}(y_{k+1} - y_k, x_{k+1} - x_k), \tag{3.1}$$

Using such discrete waypoints, however, will produce discontinuous jumps in the desired path heading once the vessel crosses the waypoint in front of it. To infer a continuous path heading during training, this study uses a distance-dependent weighted sum of the current and next path segment heading:

$$\chi_{C_k} = \left\{1 - \frac{x_e}{d(P_k, P_{k+1})}\chi_{P_k}\right\} + \left\{\frac{x_e}{d(P_k, P_{k+1})}\chi_{P_{k+1}}\right\}, \tag{3.2}$$

with $d(P_k, P_{k+1}) = \sqrt{(x_k - x_{k+1})^2 + (y_k - y_{k+1})^2}$ being the Euclidean distance between two succeeding waypoints and the along-track distance x_e given by

$$x_e = (x_A - x_k)\cos(\chi_{P_k}) + (y_A - y_k)\sin(\chi_{P_k}), \tag{3.3}$$

using the current vessel position $A = (x_A, y_A)^\top$. Most vessel-related variables such as the vessel position, the along-track error, cross-track error etc., are time-dependent. For simplicity, and to avoid clutter, we will drop the time index t in this and the next section, i.e. we write $A = (x_A, y_A)^\top$ instead of $A_t = (x_{A,t}, y_{A,t})^\top$.

There are two fundamental metrics to control for in a path-following scenario: *Cross-track-error* (y_e) and *heading-error* (χ_e). The cross-track-error normal to the path can then be found via

$$y_e = (x_A - x_k)\sin(\chi_{C_k}) + (y_A - y_k)\cos(\chi_{C_k}). \tag{3.4}$$

From the cross-track-error, we can construct a vector field after Nelson et al. (2007a) to determine the desired course as

$$\chi_d = \tan^{-1}(cy_e) + \chi_{P_k}, \tag{3.5}$$

where c is a tunable hyperparameter controlling the speed of convergence of the vector field. Using the vessel's current heading ψ and drift angle β (see Figure 3.2), the course error calculates to

$$\chi_e = \chi_d - \psi - \beta. \tag{3.6}$$

The path-following objective is to generate control actions that move both cross-track error and course error to zero.

3 ASV kinodynamics

3.1 Equations of motion

The present study uses the 3-degree-of-freedom MMG model of ship maneuvering (Yasukawa and Yoshimura, 2015) to describe the autonomous vessel's movement in the horizontal plane.

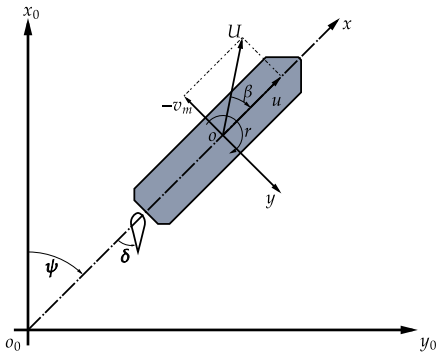


Figure 3.2: Global and local coordinate systems.

The ASV is modeled as a rigid body with a single propeller. This paper uses the coordinate system shown in Figure 3.2. The $o_0 - x_0 y_0 z_0$ coordinate system corresponds to the earth-fixed water surface while the $o - xyz$ system is vessel-fixed with origin o at midship and x, y pointing towards the bow and starboard respectively, z is pointing downwards. The center of gravity is at $(x_G, 0, 0)$ in the vessel-fixed coordinate system; total sway at the center of gravity then is $v = v_m + x_G r$, with v_m being the sway velocity at midships and r the turning rate. Surge velocity is denoted by u, thus total ship velocity is given by $U = \sqrt{u^2 + v_m^2}$, drift angle at midships by $\beta = \tan^{-1}(v_m/u)$ and the heading ψ by the angle between x_0 and x.

The forces acting on the ship are decomposed as follows:

$$\left.\begin{aligned} (m + m_x)\,\dot{u} - (m + m_y)\,v_m r - x_G m r^2 &= X, \\ (m + m_y)\,\dot{v}_m + (m + m_x)\,ur + x_G m \dot{r} &= Y, \\ \left(I_{zG} + x_G^2 m + J_z\right)\dot{r} + x_G m\,(\dot{v}_m + ur) &= N, \end{aligned}\right\} \tag{3.7}$$

where m is the mass of the ASV, m_x and m_y are the added masses in x- and y-axis direction respectively, x_G is the longitudinal coordinate of center of gravity, I_{zG} is the moment of inertia, J_z is the added moment of inertia, and r is the yaw rate.

Total forces of the left-hand-side of (3.7), X, Y, N, are surge force, lateral force and yaw moment around midship and consist of the following parts:

$$\left.\begin{aligned} X &= X_{\mathrm{H}} + X_{\mathrm{R}} + X_{\mathrm{P}}, \\ Y &= Y_{\mathrm{H}} + Y_{\mathrm{R}}, \\ N &= N_{\mathrm{H}} + N_{\mathrm{R}}. \end{aligned}\right\} \tag{3.8}$$

The subscripts $\mathrm{H}, \mathrm{R}, \mathrm{P}$ describe forces acting on the hull, rudder and propeller respectively. Further implementation details are deferred to the original paper by Yasukawa and Yoshimura (2015).

3.2 Environmental forces

According to Fossen (2021, Chap. 2), vessel speed under the influence of currents becomes a relative speed $U = \sqrt{(u - u_c)^2 + (v_m - v_c)^2}$ where u_c and v_c are the current component velocities in longitudinal and lateral direction.

Scale	1/5
Displacement	2500.8 m^3
Length between perpendiculars	64.0 m
Width	11.6 m
Block coefficient	0.81
Draft	4.16 m
Rudder area	4.5 m^2
Propeller diameter	1.76 m

Table 3.1: Principal particulars of a KVLCC2 L64-model tanker

The effects of shallow water on wake fraction, thrust deduction and flow-straightening coefficients are calculated after Amin and Hasegawa (2010) while the effects on hydrodynamic derivatives are adapted using combined formulations from Kijima and Nakiri (1990) and Ankudinov et al. (1990). A summary can be found at Taimuri et al. (2020). The effects on the maneuverability of the vessel are demonstrated in a zigzag and turning maneuver test shown in Figure 3.3. For the zigzag test, the vessel starts with an initial velocity $U_0 = 4.0m/s$, a rudder angle of 0° and an arbitrary course $\bar{\psi} - \beta$ (This study uses $\bar{\psi} - \beta = 0$). The rudder angle is increased by $5.0°s^{-1}$ until it reaches its maximum value (10° or 20°), at which it is held until the vessel's course is changed by the same amount. The rudder direction is then reversed with the same principle. For this test, currents are turned off. The turning maneuver test starts with the same initial conditions as the zigzag test, however, the rudder angle is increased to 35° and held there for the rest of the experiment. For both tests, we see impaired maneuverability for the vessel under shallow water conditions ($h/d = 1.2$), which is to be expected and emphasizes the need for a precise controller on inland waterways.

The open-source implementation of the MMG dynamics used for this study can be found at Paulig (2022).

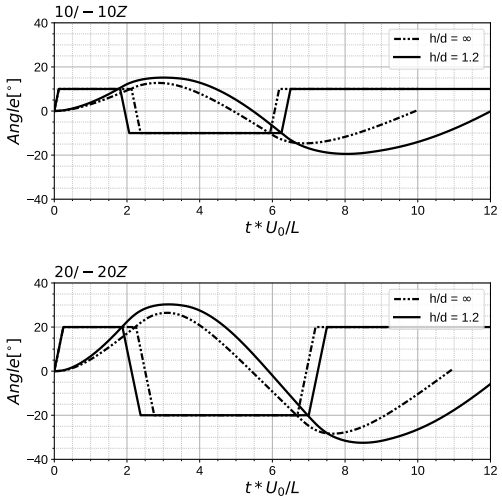
3.3 Vessel model

The vessel type used for simulation is a 1:5 scale model (L-64) of the KVLCC2 Tanker, as it has one of the most well-understood dynamics publicly available. The ship's principal particulars can be found in Table 3.1. We use a 1:5 scaling to mimic the dimensions and behavior of small- to medium-sized inland cargo vessels.

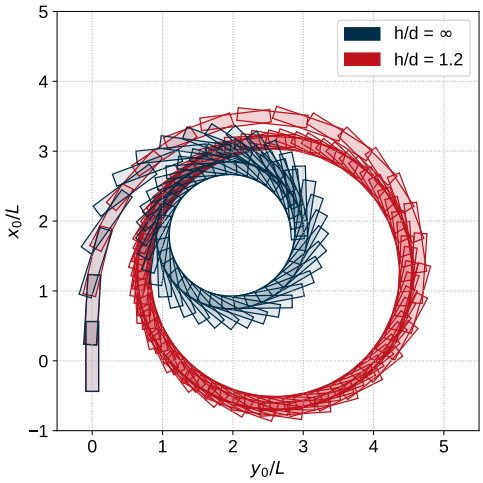
4 Reinforcement Learning framework

4.1 Fundamentals

RL is a subfield of machine learning in which an agent is trained to act in an environment such that it maximizes a reward signal received from the environment (Sutton and Barto, 2018). Formally, the simulated environment is modeled as a Markov Decision Process (MDP)(Puterman, 1994) described by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. At every time step t, given a current state of the environment $s_t \in \mathcal{S}$ the agent executes an action $a_t \in \mathcal{A}$ according to a parameterized policy $\pi_\theta : \mathcal{S} \to \mathcal{A}$. After performing the action, the agent receives a reward $r_t \in \mathcal{R}$ and transitions to



(a) Zigzag maneuvers for 10 and 20-degree rudder angles in shallow and deep waters. Initial velocity $U_0 = 4.0m/s$, steering rate $\Delta\delta = 5.0°/s$.



(b) 35° starboard turning maneuver in deep and shallow water. Initial velocity $U_0 = 4.0m/s$, steering rate $\Delta\delta = 5.0°/s$.

Figure 3.3: Zigzag and turning maneuver tests for water depth h and ship draught d.

the next state s_{t+1} according to the state transition probability distribution $\mathcal{P} : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \to [0, 1]$. The return is defined as the cumulative discounted reward $R_t = \sum_{i=t}^{T} \gamma^{i-t} r_i$ from the current time step until the final time step of the episode $t + T$ with $\gamma \in [0, 1]$ being the discount factor that trades off the importance of immediate and later rewards. All the contributions of the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ for the path-following objective will be specified in Section 4.2.

The goal of RL is to find a policy that maximizes reward in the long-term starting from some initial state. Most current algorithms use a state-action value function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ to assign a value to each state-action pair such that higher values represent pairs leading to a higher long-term reward. $Q^\pi(s, a) = \mathbb{E}_{s\sim\mathcal{P},a\sim\pi}(R_t \mid s_t = s, a_t = a)$ resembles the expected discounted sum of rewards starting from state s, taking action a and following policy π afterwards. The algorithmic foundation for this work is the Q-Learning algorithm (Watkins and Dayan, 1992) that uses the Bellman optimality equations (Bellman, 1957) to solve for the optimal Q-values Q^* satisfying

$$Q^*(s, a) = \mathbb{E}\Big\{ r_t + \max_{a_{t+1}\in\mathcal{A}} Q^*\left(s_{t+1}, a_{t+1}\right) \mid s_t = s, a_t = a \Big\}, \tag{3.9}$$

from which an optimal policy can be derived by $\pi^*(s) = \arg\max_a Q^*(s, a)$. The Q-Learning update rule for a given Q-value estimate $\hat{Q}(s, a)$ and learning rate α is

$$\hat{Q}\left(s_t, a_t\right) \leftarrow \hat{Q}\left(s_t, a_t\right) + \alpha\{\bar{\tau}^T - \hat{Q}\left(s_t, a_t\right)\}, \text{ for } \bar{\tau}^T = r_{t+1} + \gamma \max_{a_{t+1}\in\mathcal{A}} \hat{Q}\left(s_{t+1}, a_{t+1}\right). \tag{3.10}$$

To keep track of the Q-values, their value must be stored for each state-action pair, which is infeasible even for moderately sized environments. To overcome this limitation Mnih et al. (2015) introduced the DQN algorithm that uses neural networks as function approximators for Q-Value estimation, allowing for sufficiently accurate Q-values even in high dimensional domains. The Q-learning update rule in (3.10) however, suffers from an overestimation bias, which is induced

through the fact, that the estimation of the bootstrapped target $\bar{\tau}^T$ uses the maximum over all possible actions. Because all Q-Values are approximations of their true expectation, some estimations are probably higher than the true expected value (Thrun and Schwartz, 1993). This can lead to misjudgment during exploration, as states with falsely attributed high Q-Values are taken into consideration more often, than the ones with falsely attributed low Q-Values. Eventually, this inequality can lead to suboptimal policies. Therefore, several approaches set out to mitigate or lessen this overestimation (Van Hasselt, 2010; Van Hasselt et al., 2016; Osband et al., 2016; Kumar et al., 2020).

The off-policy approach used in this paper was proposed by Waltz and Okhrin (2022), which itself is an extension of the bootstrapping framework from Osband et al. (2016). The general idea is to rely on the ability of bootstrapping to provide measures of accuracy for statistical estimates, which is usually achieved by resampling an original dataset with replacement and calculating the statistics of interest using these bootstrapped samples.

In the DRL setting, this is translated into either maintaining several distinct Q-networks, each with its own target network or using a single network with a shared core and several heads. This work will use the latter approach. The bootstrapping nature is achieved by randomly initializing the network heads and using a binary map to determine which head is to be updated on the current iteration. Additionally, Waltz and Okhrin (2022) propose to replace the maximum over all possible actions in the target with a kernel-based testing procedure. Suppose a network with one common core and $B \in \mathbb{N}$ heads. Furthermore, let κ be a kernel function (in our study we use the Gaussian cumulative distribution function $\Phi(\cdot)$). The target for the b^{th} head then becomes

$$\bar{\tau}^{T,b} = r + \gamma \left[\sum_{a_{t+1} \in \mathcal{A}} \kappa \left\{ T_{\hat{Q}_b}\left(s_{t+1}, a_{t+1}\right) \right\} \right]^{-1} \sum_{a_{t+1} \in \mathcal{A}} \kappa \left\{ T_{\hat{Q}_b}\left(s_{t+1}, a_{t+1}\right) \right\} \hat{Q}_b\left(s_{t+1}, a_{t+1}; \theta_b^-\right), \tag{3.11}$$

where

$$T_{\hat{Q}_b}\left(s_{t+1}, a_{t+1}\right) = \frac{\hat{Q}_b\left(s_{t+1}, a_{t+1}; \theta_b^-\right) - \max\limits_{a_{t+1} \in \mathcal{A}} \hat{Q}_b\left(s_{t+1}, a_{t+1}; \theta_b^-\right)}{\sqrt{\widehat{\mathrm{Var}}\left\{ \hat{Q}_b\left(s_{t+1}, a_{t+1}; \theta_b^-\right) \right\} + \widehat{\mathrm{Var}}\left\{ \hat{Q}_b\left(s_{t+1}, a^*; \theta_b^-\right) \right\}}}, \tag{3.12}$$

is a statistic for testing whether the selected action from head v is not smaller than the maximum estimate for that head, and the currently maximizing action $a^* \in \left\{ a \in \mathcal{A} \mid \hat{Q}_b\left(s_{t+1}, a; \theta_b^-\right) = \max\limits_{a_{t+1} \in \mathcal{A}} \hat{Q}_b\left(s_{t+1}, a_{t+1}; \theta_b^-\right) \right\}$. In the following, we will stick with the naming of Waltz and Okhrin (2022) and call this algorithm *KEBDQN*. Further implementation details are deferred to the original paper. Certainly, deep Q-learning is not the only possible option for solving this particular problem, however, the focus of this study lies not in the comparison of different RL methods, but on the feasibility of using deep RL to control a vessel in spatially restricted and dynamic environments.

4.2 Controller design for inland ASVs

In this section, we describe the state, action and reward structure of the MDP used to model inland waterways for this study. As described in Section 4.1, on every time step, t, the agent -our vessel- observes a state s_t from the environment and chooses to perform action a_t according to its policy π_θ.

State space The state space $s_t = \left(s_t^{d\top} \; s_t^{e\top}\right)^\top$ is assumed to be fully observable and involves two parts: The first part

$$s^d = \left(u_t, \quad v_t, \quad r_t, \quad \delta_t, \quad u_{t-1}, \quad v_{t-1}, \quad r_{t-1}, \quad \delta_{t-1}\right)^\top, \tag{3.13}$$

contains information about surge, sway and yaw rates and the rudder angle, δ, at the current and previous time steps. This way the agent can perceive the changes in dynamics resulting from different environmental conditions, for example, increased sway rates due to cross-current fields, or due to the agent's actions. The second part encodes information about the surroundings of the agent:

$$s^e = \left(\tilde{y}_{e,t}, \quad \tilde{y}_{e,t-1}, \quad \chi_{e,t} \quad \chi_{e,t-1}, \quad \frac{h_t - d}{\max(h)}, \quad \gamma_{rel}\right)^\top, \tag{3.14}$$

where h_t is the current water depth below keel, and d is the ship draught, thus $\frac{h_t-d}{\max(h)}$ is the remaining water under keel normalized by the maximum depth possible in the environment. The current attack angle relative to the bow is γ_{rel}, and $\tilde{y}_{e,t} = c_1 \tanh(y_{e,t})$, with c_1 being a tunable hyperparameter controlling the importance of the cross-track error. The above cross-track-error scaling is done to stabilize training in later stages, as its magnitude exceeds all other observations being measured.

Action space In this study, we follow other researchers (Moreira et al., 2007; Amendola et al., 2019; Zhao et al., 2019; Amendola et al., 2020) and assume constant thrust by fixing the propeller rotation rate to $4.0s^{-1}$ i.e. the agent does not control its velocity, but its rudder angle. There are three possible actions $a_t \in \{\delta_{t-1} - 2°, \delta_{t-1}, \delta_{t-1} + 2°\}$, that either increase or decrease the rudder angle by two degrees or leave it as is. The admissible rudder range is $\delta_t \in \{-20°, -18°, \ldots, 18°, 20°\}$. The choice of a stepwise rudder change rather than choosing between fixed angles avoids generating successive rudder commands of unrealistic magnitude, for example, $\{a_t = -20°, a_{t+1} = 20°\}$, which would lead to structural damage of the rudder.

Reward structure The reward the environment emits acts as an immediate measurement of the quality of the action taken by the agent. To fulfill the path-following objective, minimal spatial and angular deviation from the given path is intended. Therefore, the reward system includes three parts:

$$R_t = \omega_1 R_{y_e,t} + \omega_2 R_{\chi_e,t} + R_{\text{aground},t}. \tag{3.15}$$

The first part rewards closeness to the desired path, while the second guides the agent towards its desired course as dictated by the underlying vector field. The terms are defined as:

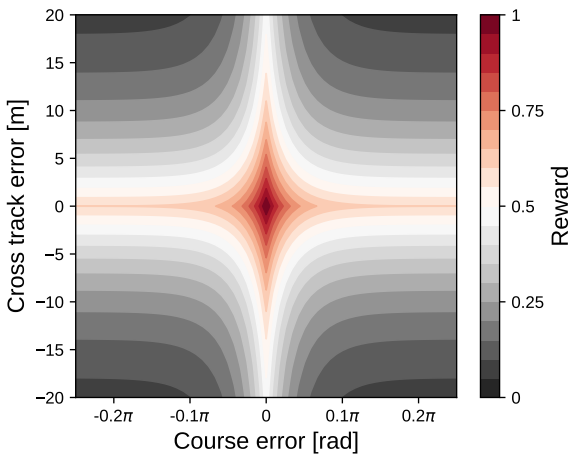


Figure 3.4: Reward contours for the path-following setup

$$R_{y_e,t} = \exp(-c_2|y_{e,t}|), \tag{3.16}$$

and

$$R_{\chi_e,t} = \exp(-c_3|\chi_{e,t}|) \tag{3.17}$$

If the water depth below the agent is less than $1.2d$, the agent will receive a negative reward defined by

$$R_{\text{aground},t} = \begin{cases} -20 & \text{if} \quad h_t < 1.2d \\ 0 & \text{otherwise.} \end{cases} \tag{3.18}$$

The factor of 1.2 is used as a lower bound as the shallow-water correction terms for the hydrodynamic derivatives (Kijima and Nakiri, 1990; Ankudinov et al., 1990) lead to unrealistic vessel behavior below this bound. If the vessel advances to areas where the water depth falls below this threshold, the current episode is terminated. Preliminary testing concluded that values of $c_2 = 0.1$, $c_3 = 10$, $\omega_1 = 0.6$ and $\omega_2 = 0.4$ yielded a reward structure sensitive to cross-track error deviations of more than one ship width. Figure 3.4 shows a contour plot of the reward structure described above. The selection of weights was chosen such, that cross-track deviations are penalized more quickly than course deviations. This was done to allow the vessel to advance through curves and current fields with a non-zero drift angle while still attaining high rewards.

4.3 Training environment generation

Since restricted waterways in general, and rivers in particular feature a wide variety of widths, lengths, riverbed profiles, water depth distributions and current velocities, a robust agent needs to be trained in an equally diverse training environment.

To generate arbitrary rivers we loosely follow the procedures in Fossen (2021, Chap. 5) by using an alternating sequence of straight and curved segments of equal width w_S, as shown in Figure 3.5. A given straight segment is described by the tuple $S^S := (\xi, l)$, while each curved segment is defined by the triple $S^C := (\xi, r_C, \phi)$, where ξ is the starting angle of the

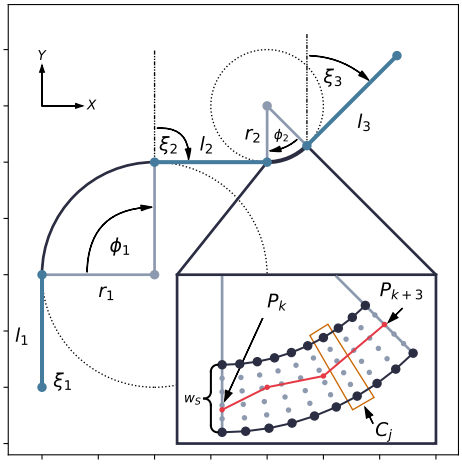


Figure 3.5: Example river generated from five segments as described in Section 4.3. The bottom right view details a curved river segment, showing the width of the segment w_S, a cross-section C_j (see Figure 3.6 for a side-view), as well as an example path comprised of four waypoints starting at P_k and ending at P_{k+3}.

segment against the ordinate, l is the length of the straight segment, r_C is the radius of the circle inscribing the curved segment, and ϕ is the angle by which we want the segment to curve (curvature).

A training environment is now built by chaining n straight and curved segments together in an alternating fashion to form a n-random river

$$\text{Riv}(n) = (S_1^S, S_1^C, S_2^S, S_2^C, \ldots, S_n^S, S_n^C), \tag{3.19}$$

by the following rules: The first angle ξ_1 is initialized arbitrarily, in our study we use $\xi_1 = 0$. All successive angles are calculated by:

$$\xi_k = \xi_1 + \sum_{i=1}^{k-1} \phi_i, \tag{3.20}$$

for $k \in \{1, 2, \ldots, n\}$.

We additionally divide the entire n-random river into p cross-sections $C_j = \{q_{1,j}, \ldots, q_{m,j}\}, j \in \{1, 2 \ldots, p\}$, each holding m supporting points $q_{i,j} = (x_{q_{i,j}}, y_{q_{i,j}})^\top$, $i \in \{1, 2 \ldots, m\}$. The set of all supporting points forms a two-dimensional grid (see Figure 3.5, bottom right), which will be used for current field and water depth sampling. On straight segments, the grid is equidistant such that $d(q_{i,j}, q_{i,j+1}) = d(q_{i+1,j}, q_{i,j})$, while for curved segments, the distance between adjacent supporting points varies depending on the segment's curvature.

For every cross-section C_j, the water depth is sampled according to

$$h_{q_{i,j}} = -h_{\max} \exp\Big\{-\epsilon \cdot d(q_{i,j}, q_j^M)^4\Big\} + \eta, \tag{3.21}$$

with random noise $\eta \sim \mathcal{N}(0, \sigma)$, maximum water depth $h_{\max}$, and ϵ, a parameter controlling the river wall steepness and fairway width. q^M is the middle point of a given cross-section j such

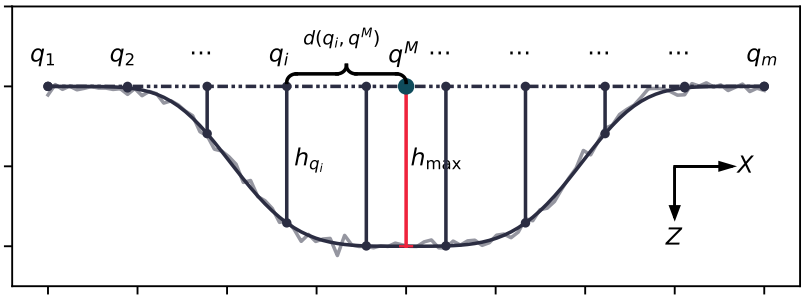


Figure 3.6: Cross-section view through a river segment. The distorted gray line resembles the depth-generation function with added noise.

that $d(q_1, q_j^M) = d(q_m, q_j^M)$.

To induce a current field, for a given maximum current speed ν_{max}, and a cross-section C_j, we set the direction of current for all supporting points in that cross-section to be $\gamma_{q_{\cdot,j}} = \frac{2}{p}\pi j$ radians and the current speed to be $\nu_{q_{\cdot,j}} = \mathsf{f}(\frac{2}{p}\pi j)\nu_{max}$ for all supporting points of C_j. The function $\mathsf{f} : \mathbb{R} \to [-1, 1]$ can be an arbitrary continuous periodic function, this study uses the cosine. By tying the current generation process to the number of cross-sections, two rivers constructed from identical segments also share an identical current field, which is helpful in terms of reproducibility, yet, since the segments are rotated at random on each generation iteration, the likelihood of constructing alike rivers during training decreases exponentially with the number of segments.

4.4 Comparison to related work

Deep RL for path-following has seen diverse methodologies, each with unique formulations of state, action, rewards and training environments. For the state space, Zhao et al. (2019) focus on errors related to course and heading angles, while Woo et al. (2019) combine heading error with cross-track error and steering inputs. Shen and Guo (2016) emphasize heading error and yaw rate, and Martinsen and Lekkas (2018a) extend this to include surge and sway alongside cross-track error. Our approach distinguishes itself by encompassing not only dynamic aspects like surge, sway, yaw rates, and rudder angles but also crucial environmental factors such as water depth, keel clearance, current attack angle, and cross-track error, thereby offering a solution more attuned to the specific demands of navigating inland waterways.

Regarding the action space, Zhao et al. (2019) employs a discrete set $\{-20°, 0°, 20°\}$, which may lead to unrealistically high changes in heading. Woo et al. (2019) uses thrust to control port or starboard turns. Shen and Guo (2016) employs the rudder angle without specifying its range, while Martinsen and Lekkas (2018a) decouples the desired and the possible rudder angle by letting the vessel dynamic control its amount. In our approach, the action space consists of the current rudder angle adjusted by $\pm 2°$ to ensure structural integrity of the vessel.

Shen and Guo (2016), Zhao et al. (2019), Woo et al. (2019) and Martinsen and Lekkas (2018a) all utilize the cross-track and heading error as a reinforcement signal, albeit with different functional dependencies. While Zhao et al. (2019), Martinsen and Lekkas (2018a) and Woo et al. (2019) use negative exponentials, Shen and Guo (2016) uses a step function with a threshold

for assessing the agent's actions. Additionally, Zhao et al. (2019) considers the distance to the next waypoint, and Woo et al. (2019) adds the standard deviation of the cross-track error to the reward signal. Our approach adopts the negative exponential functions for cross-track and heading errors, along with a penalty for running aground, reflecting the challenges found on inland waterways.

Many of the previously mentioned researchers omitted crucial aspects of a realistic training environment in their path-following studies. Notably, Zhao et al. (2019); Woo et al. (2019); Shen and Guo (2016); Martinsen and Lekkas (2018a) did not incorporate water depth into their models, neglecting a critical navigational parameter that significantly influences vessel dynamics. Furthermore, Woo et al. (2019); Shen and Guo (2016) predominantly utilized straight paths or neglected to impose spatial restrictions, potentially oversimplifying the training scenarios, which could lead to less effective policies, as suggested by Hart and Okhrin (2023). In contrast, our approach comprehensively addresses the complexities of real-world, inland, maritime navigation by explicitly considering water depth and current variations and incorporating diverse path geometries, thereby enhancing the practical relevance and robustness of our trained models in dynamic and challenging environments.

4.5 Training

For training, we chose a discretization time-step of $\Delta T = 1s$ and an episode length of 2000 steps equating to roughly 33 minutes in real-time. At the beginning of each episode, a random river is generated as described in 4.3. We construct $n = 5$ straight and curved segments with angles, radii, and lengths drawn uniformly from the following sets

$$\begin{aligned} \phi &\in \{\pm 60^\circ, \pm 61^\circ, \dots, \pm 100^\circ\}, \\ r &\in \{1000, \dots, 5000\}, \\ l &\in \{400, \dots, 2000\}. \end{aligned}$$

The value ranges for r and l are chosen such that they resemble real-world river behavior and avoid the construction of unrealistically sharp turns or too short straights. During training, we sample values from each set with equal probability. We set $w_S = 500m$, $d(q_k, q_{k+1}) = 20m$ and the maximum current velocity $\nu = 1.5ms^{-1}$; two example generated rivers can be found in Figure 3.7.

At the beginning of each episode, the agent-vessel is placed at the outset of the first segment of the constructed river with a heading equal to the current path heading plus some noise

$$\psi_0 = \chi_{P_0} + R, \quad \text{with } R \sim \mathcal{U}(-5^\circ, 5^\circ), \tag{3.22}$$

an initial speed $U_0 = 4.0ms^{-1}$, and a fixed propeller rotation rate of $4.0s^{-1}$. For this study, we use a network with one common core and 10 heads. The core network is a multilayer perceptron with a single hidden layer containing 128 neurons. The heads follow the same structure as the core, with one hidden layer, each containing 128 neurons. During training random batches of

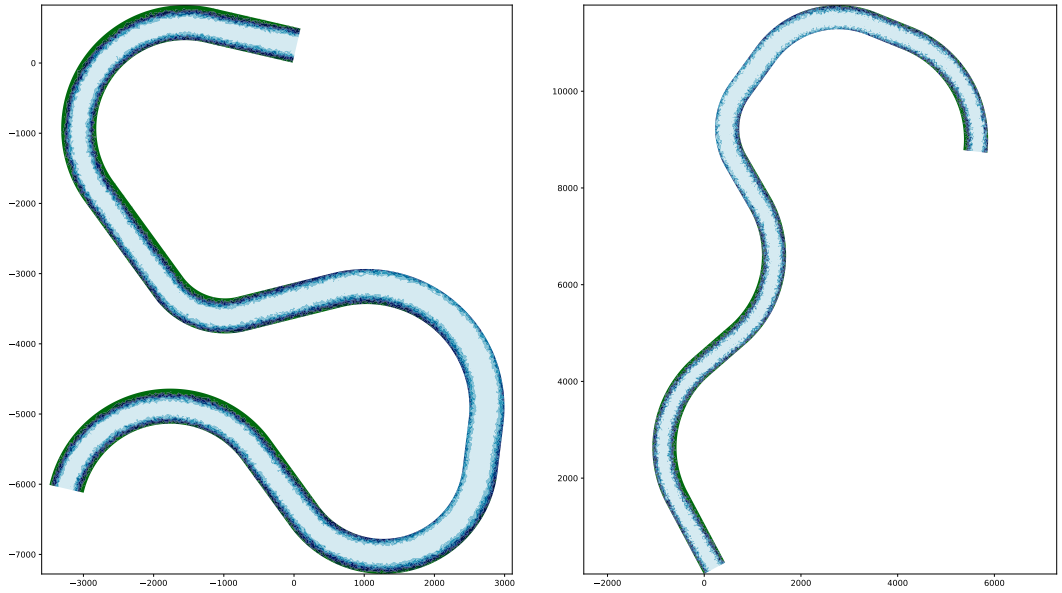
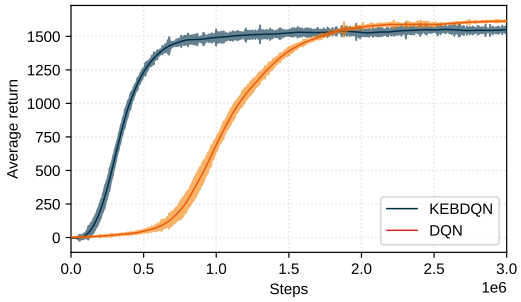


Figure 3.7: Example rivers used for training, generated as in Section 4.3.



Parameter	Value
Number of hidden layers	2
Number of neurons per layer	$[256, 128]$
Batch size	128
Discount factor (γ)	0.99
Loss function	MSE
Replay buffer size	1.0×10^6
Optimizer	Adam
Target network update frequency	1000 steps
Initial exploration rate ($\epsilon_{initial}$)	1.0
Final exploration rate (ϵ_{final})	0.01
Exploration decay time	1.0×10^6 steps

Figure 3.8: Left: Training results running 15 independent seeds for 3×10^6 steps each. The shaded area resembles the 95% point-wise confidence intervals. The theoretical reward limit is 2000. Right: Hyperparameter setup for the DQN.

128 transitions are sampled from a replay buffer of size 10^6, gradient updates are performed by the Adam optimizer (Kingma and Ba, 2015) with a learning rate of $\alpha = 5 \times 10^{-4}$ and a discount rate of $\gamma = 0.99$. Training has been conducted for 3×10^6 steps, the implementation framework for the *KEBDQN* is the `TUD_RL` package (Waltz and Paulig, 2022) written in Python.

For comparison, we also trained a vanilla *DQN* alongside the *KEBDQN*. Figure 3.8 summarizes the training of 15 different seeds per algorithm, as well as the hyperparameter setup for the DQN.

5 PID Benchmark

In preparation for the validation of our approach, we chose a PID rudder controller design for the KVLCC2 tanker from Paramesh and Rajendran (2021) to serve as a performance benchmark. The original PID implementation is tuned to the full-size vessel, therefore the provided gains

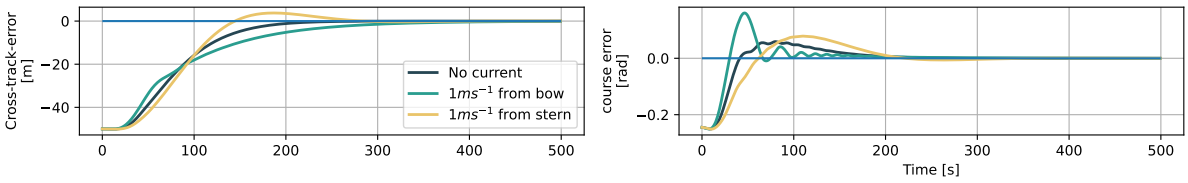


Figure 3.9: PID-response for achieving zero course error.

cannot be used in this paper. To find the best possible PID configuration for comparison against the DRL controller, we use a Particle Swarm Optimization (PSO) procedure with random uniform inertia weights proposed by Eberhart and Shi (2000) to tune our PID controller, as PSO has been proven to be an effective solution for tuning PID controller for path-following applications, also in real-world scenarios (You et al., 2023). The rudder angle at every time step evaluates to:

$$\delta_t = K_p \chi_{e,t} + K_d r_t + K_i \int_0^t \chi_{e,t} \mathrm{d}t. \tag{3.23}$$

Initial gains, $K_p = 2.96, K_d = 19, K_i = 0.03$, have been found via a coarse grid search. The PSO algorithm used the objective function

$$J(t) = \int_0^t \chi_{e,t}^2 \mathrm{d}t \tag{3.24}$$

to solve for $\mathrm{argmin}_{K_p,K_i,K_d} J(t)$ which yielded $K_p = 2.81, K_d = 64, K_i = 0.0$ as gains, thereby reducing the system to a PD controller. We also used a different objective function with an additive term for minimum overshoot, yet the result could not beat the simple procedure from above. The controller response was tested in three different scenarios. In all three, the agent is set into a straight channel with a course error of 14° and a cross-track error of 50 meters. Responses for no current, current to bow and stern can be found in Figure 3.9. Additionally, as with the RL agent, the maximum change in rudder angle is limited to $2°s^{-1}$ to respect the structural integrity of the ASV.

6 Simulation and validation

The policy found from training was simulated on several sections of the lower and middle Rhine as well as on artificial scenarios checking for reactivity under harsh environmental changes. All experiments are enrolled for the *KEBDQN*, PID and DQN approach for comparison.

6.1 Rhine river

The first scenario validates the performance on a near 180° degree turn on the *lower Rhine* close to Düsseldorf harbor $(51.22°N, 6.73°E)$, as it features one of the tightest turns in the lower Rhine. Figure 3.10 shows a map containing the path to be followed, and the trajectories generated by each approach; the corresponding metrics are depicted in Figure 3.11(a).

The second validation scenario was selected on the *middle Rhine*. We chose a segment close

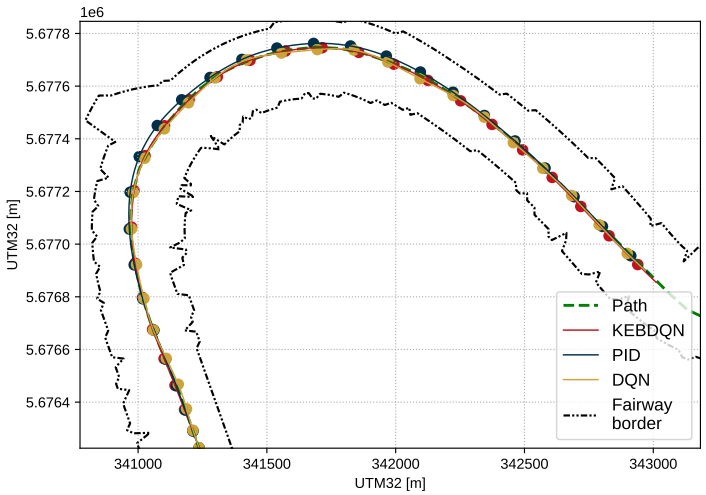
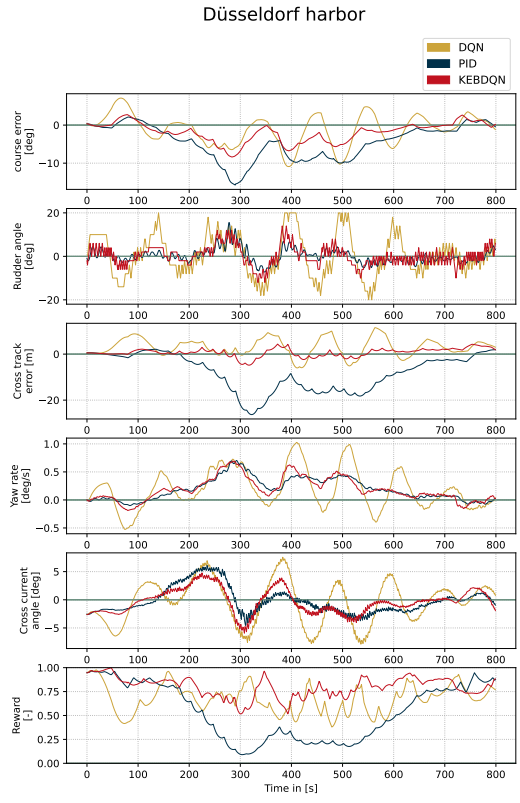
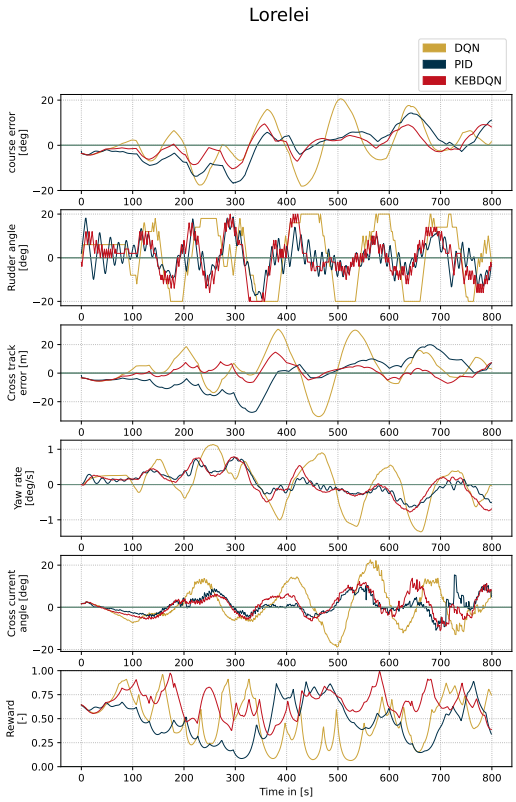


Figure 3.10: 2D map of the 180-degree turn around Düsseldorf harbor. The lines represent the paths taken by our ASV given the respective control approach. The dots represent equitemporally-spaced points with a distance of 30 seconds.



(a) 180 degree curve around Düsseldorf harbor



(b) Zigzag curve around the Lorelei.

Figure 3.11: Time series of relevant metrics for the two scenarios on the Rhine. The reward plot for the PID controller is the reward it would have received if judged by the same reward function as the RL-based controller.

to the *Lorelei* ($50.12°N, 7.73°E$) which features one of the smallest widths on the river together with a fast succession of right and left turns. The results can be seen in Figure 3.11. In both

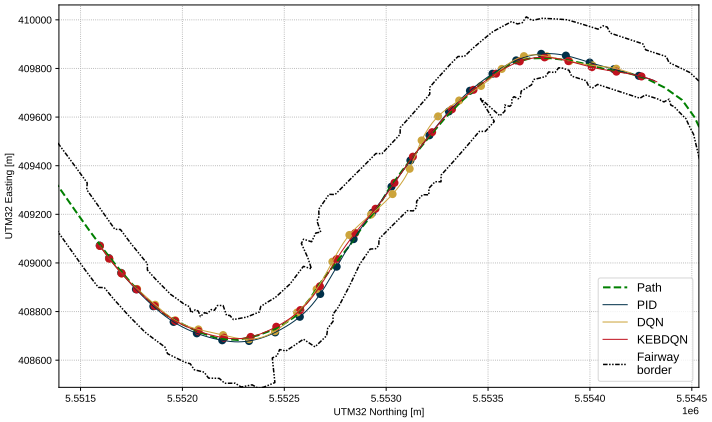


Figure 3.12: 2D map of the starboard turn near the Lorelei.

scenarios, the path was generated by selecting the deepest point for every cross-section through the entire river and smoothing the result using two-dimensional exponential smoothing.

Analyzing the rudder commands generated for both scenarios, we observe relative similarity in magnitude and direction, indicating that the DRL agents were able to learn a similar behavior as exerted by the PID controller.

Inspecting the cross-track error and course error for both scenarios, both controllers are again found to follow akin patterns, however, the DRL controller reacts quicker, thus being able to achieve a maximum cross-track deviation of $4.36m$ compared to $26.30m$ from the PID controller for the Düsseldorf harbor scenario, and $14.67m$ and $27.47m$ respectively for the Lorelei.

One of the major drawbacks of discrete RL-based controllers is the jittering of the rudder angle as seen in the rudder commands in Figure 3.11, however, since the rudder steps in the RL approach are chosen such that the structural limits of the ASV are respected, the jittering is not prone to damaging the rudder of the ASV if this algorithm had been deployed in the real world. In earlier stages of research, we followed other authors (Martinsen and Lekkas, 2018a,b), and added a penalty for changing rudder angles too quickly, concluding that less change in rudder angle came at the cost of losing cross-track-error accuracy. Since we valued accuracy higher than slow rudder change, the penalty term was removed.

6.2 Straight paths

For maneuvers like berthing, docking, and locking or advancing through canals it seems natural to ask the vessel to follow a straight line with very high accuracy. We will test this ability for the PID and DRL controller in a straight-path scenario. We would expect the PID controller to achieve near-perfect convergence to the path, as any other result would indicate a misconfigured set-point. As with the PID calibration, the vessel will be placed in a straight canal 50 and 20 meters starboard to the path with a course error of 14° and 5.7° respectively. Propeller revolutions are fixed to $4.0s^{-1}$ and the initial velocity $U_0 = 2.0ms^{-1}$. We assume no currents and a water depth to draught ratio of roughly $h/d = 2.40$.

The results from Figure 3.13 confirm our initial assumption about the PID controller. In comparison to our DRL approach, the PID converges faster and more accurately, falling below

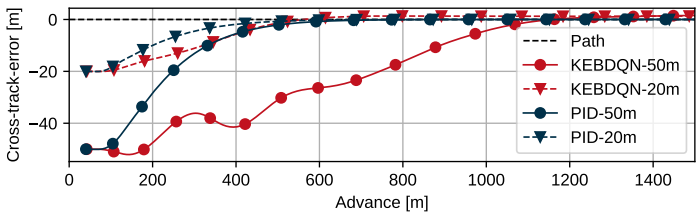


Figure 3.13: Straight-path-following experiment. Consecutive markers are each 30 seconds apart.

one meter of cross-track error after $600m$ of advance. The DRL rate of convergence seems to be dependent on the offset magnitude from the path. In the $20m$ offset scenario, the *KEBDQN* performs similarly to the PID, while for the $50m$ offset the DRL agent has noticeable difficulties returning to the path. We assume, that the agent rarely saw cross-track errors this large at the beginning of an episode during training, therefore no convergence strategy could be developed.

7 Robustness analysis

7.1 Varying revolutions

To verify the robustness of our approach, both controllers were driven up and downstream through the entire lower- and middle Rhine, each in a single episode. We did this once for a propeller revolution rate of $4.0s^{-1}$, which is also the frequency used for training, and another time using $5.0s^{-1}$ to investigate the generalization capabilities of our approach.

The cross-track-error distributions achieved are depicted in Figure 3.15. For the downstream scenario we find acceptable results for both controllers, whereby the DRL solution exerts significantly smaller variance, yet in the downstream scenario, is biased towards starboard, for both $4.0s^{-1}$ and $5.0s^{-1}$. Interestingly, this bias does not appear in the upstream scenarios, ruling out doubt about starboard-biased training, as we would expect to observe a bias towards port for driving upstream. For the upstream scenario, the PID controller appears to be sensitive to changes in ship velocity, with a tendency to become more stable for higher velocities. The inability of the PID controller to stay on course at a slower speed and high current velocities to the bow (the middle Rhine features current velocities of up to $2.4ms^{-1}$, and the lower Rhine up to $1.5ms^{-1}$), is likely due to misconfiguration of the PID controller for such environments. The fundamental problem with PIDs is that it may be impossible to find a set of gains, optimally controlling the rudder in dynamic environments featuring a wide range of external disturbances.

DRL approaches in contrast have the ability to adapt to more general cases, as they can rely on their experience acquired from the training. Although the agents did not see current velocities above $1.5ms^{-1}$, they were trained to react to those from every direction. This may lead to an additional environmental awareness, capable of achieving small cross-track errors across varying external disturbances.

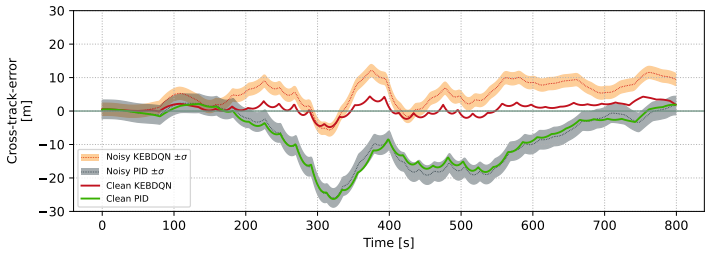


Figure 3.14: Cross-track-error for 10 different attempts per controller with sensor noise. The shaded area for the noisy runs resembles one standard deviation distance from the 10-run average. Clean runs are the same as in Figure 3.11 (a) and have no noise applied to any input.

7.2 Noisy observations

To further explore the robustness of our approach against the PID controller, we decided to compare cross-track-error performance under noisy sensor inputs. Impaired sensor measurements appear regularly in real-world applications, thus posing a valuable platform to evaluate controller behavior.

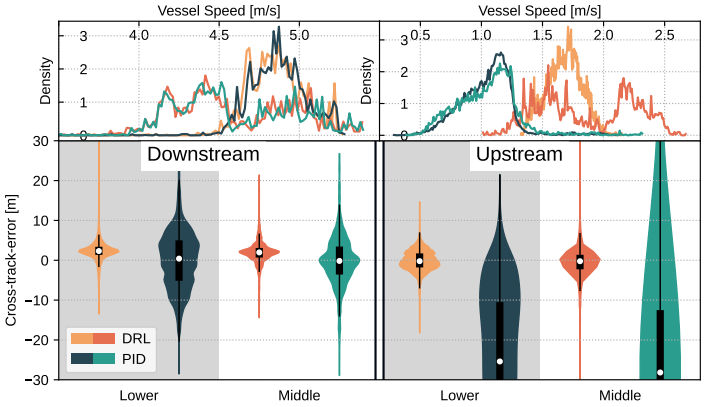
We again chose the unaltered Düsseldorf scenario on the lower Rhine as in Section 6.1 but with added Gaussian noise to the yaw-rate $\bar{r}_t = r_t + \epsilon_r, \epsilon_r \sim \mathcal{N}(0, \bar{\sigma}_r)$ and course error $\bar{\chi}_t^e = \chi_t^e + \epsilon_{\chi_e}, \epsilon_{\chi_e} \sim \mathcal{N}(0, \bar{\sigma}_{\chi_e})$. The standard deviations are calculated from the empirical distributions of yaw rate and course error obtained from driving the *KEBDQN* controller through the entire river in a single episode and had been measured to be $\bar{\sigma}_r = 0.004\,\mathrm{rad}\cdot s^{-1}$ and $\bar{\sigma}_{\chi_e} = 0.052°$. All other sensor inputs for the DRL approach are left unchanged.

The results in Figure 3.14 paint an ambiguous picture. On the one hand, we still observe greater accuracy in cross-track error for the DRL controller, on the other hand, the deviation from the noise-free run is smaller for the PID controller. We also did this for several other scenarios (available upon request), all with similar outcomes. Although the variation in cross-track error for the PID controller under noisy inputs is smaller technically, the accuracy achieved by the DRL controller remains higher.

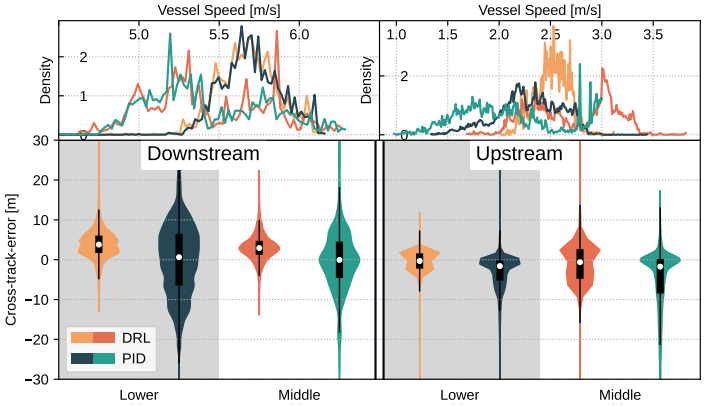
Therefore, we can conclude, that in terms of sensor noise, our PID controller attains lower deviations from its anticipated position contrary to the DRL system. However, the broader picture of robustness can be seen in favor of the DRL controller, since it not only produces smaller absolute cross-track errors -even under noisy inputs-, it also performs well under unseen propeller revolutions and very slow vessel advance rates.

8 Conclusion

It was found that ASV path-following in inland waterways presents distinct challenges not encountered in open sea navigation. The present study specifically addressed these challenges by implementing a state-of-the-art bootstrapped DQN algorithm utilizing a novel and adaptable inland waterway training environment generator, resulting in the development of a robust and adaptable rudder control system for efficient path-following in inland environments. It was concluded that traditional control approaches, such as PID controllers and conventional Deep



(a) Cross-track-error and speed-over-ground distributions at training revolutions of $4.0s^{-1}$. The PID controller episode in the upstream scenario on the middle Rhine did not complete the entire river due to running aground.



(b) Cross-track-error and speed-over-ground distributions for unseen revolutions of $5.0s^{-1}$

Figure 3.15: Cross-track-error and speed-over-ground distributions for the lower- and middle Rhine for the DLR and PID controller at $4.0s^{-1}$ (a) and $5.0s^{-1}$ (b) revolutions. The white dots represent the median of the distribution and the black bars correspond to the interquartile range. The plots at the top, represent the distributions of vessel speeds for each attempt. Kernel densities were estimated using Gaussian kernels with bandwidths calculated after Botev et al. (2010).

Reinforcement Learning algorithms, like the standard DQN, exhibit limited flexibility in adjusting to the dynamic and complex conditions characteristic of riverine settings, particularly under strong upstream hydrodynamic conditions. Despite their capability to generate precise rudder commands for ASV control, these methods necessitate frequent re-training or configuration adjustments to cope with the variable dynamics of constrained waterways. It was observed that the bootstrapped DQN algorithm demonstrated superior adaptability without the need for constant modification. However, it is recommended that future research should consider factors such as maritime traffic and dynamic propeller revolution changes, which were simplified assumptions in this study and could significantly impact ASV navigation in real-world scenarios.

9 Acknowledgements

The authors would like to thank the German Federal Waterways Engineering and Research Institute (BAW, Bundesanstalt für Wasserbau) for providing real-world depth and current data for the lower- and middle Rhine as well as the Center for Information Services and High-Performance Computing at TU Dresden for providing its facilities for high throughput calculations. The authors would also like to extend their gratitude to Martin Waltz and Fabian Hart for their valuable discussions and support throughout this project.

Chapter 4

2-level reinforcement learning for ships on inland waterways: Path planning and following

Martin Waltz [a], *Niklas Paulig* [a], *Ostap Okhrin* [a,b]

[a] *Institute of Transportation Economics, Technische Universität Dresden, 01062 Dresden, Germany*

[b] *Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI) Dresden/Leipzig, Germany*

Published in: *Expert Systems with Applications* (2025), 274:126933

Abstract:
This paper proposes a realistic modularized framework for controlling autonomous surface vehicles (ASVs) on inland waterways (IWs) based on deep reinforcement learning (DRL). The framework improves operational safety and comprises two levels: a high-level local path planning (LPP) unit and a low-level path following (PF) unit, each consisting of a DRL agent. The LPP agent is responsible for planning a path under consideration of dynamic vessels, closing a gap in the current research landscape. In addition, the LPP agent adequately considers traffic rules and the geometry of the waterway. We thereby introduce a novel application of a spatial-temporal recurrent neural network architecture to continuous action spaces. The LPP agent outperforms a state-of-the-art artificial potential field (APF) method by increasing the minimum distance to other vessels by 65% on average. The PF agent performs low-level actuator control while accounting for shallow water influences and the environmental forces winds, waves, and currents. Compared with a proportional-integral-derivative (PID) controller, the PF agent yields only 61% of the mean cross-track error (MCTE) while significantly reducing control effort (CE) in terms of the required absolute rudder angle. Lastly, both agents are jointly validated in simulation, employing the lower Elbe in northern Germany as an example case and using real automatic identification system (AIS) trajectories to model the behavior of other ships.

Keywords: deep reinforcement learning, path planning, path following, autonomous surface vehicle, inland waterway

1 Introduction

Inland waterway (IW) transport is widely regarded as an energy-efficient and low-emitting mode of transportation in terms of greenhouse gases compared to road or rail transportation. Additionally, IW transport offers a significant freight capacity, making it an integral part of a sustainable transport system (Rohács and Simongati, 2007; de Barros et al., 2022). Traditionally, the control of inland vessels is performed by human operators, and recent research began to investigate the use of autonomous surface vehicles (ASVs) for such inland operations (Gan et al., 2022; Vanneste et al., 2022). For a comprehensive understanding of ASV systems, we refer to Liu et al. (2016), Fossen (2021), and Negenborn et al. (2023).

One crucial factor affecting the economic potential of shipping companies is the costs associated with crew members (Al Enezy et al., 2017). These costs can be significantly reduced since ASVs require little to no on-board personnel (Vagale et al., 2021). Furthermore, although IW operations already exhibit relatively low accident rates compared to other transportation modes (Hofbauer and Putz, 2020), the human factor can still pose a significant threat to operational safety. For instance, according to Bačkalov et al. (2023), human failures were responsible for 58% and 19% of accidents in Austria and Serbia, respectively, from the early 2000s to 2017. As emphasized by Gan et al. (2022), IW operations are particularly challenging for human operators due to waterway geometry and potential high traffic densities, further reinforcing the potential benefits of employing ASVs in inland operations.

There are different approaches to categorizing the level of autonomy of autonomous ships (Ringbom, 2019). In this paper, we adopt the definition from Vagale et al. (2021, p. 1296), which defines an ASV as *"a vessel capable of making decisions and operating independently, without human guidance, navigation, and control"*. An ASV has to continuously generate a path which it can subsequently follow. The literature distinguishes between global (GPP) and local path planning (LPP), where GPP addresses the static problem of defining a plan for the entire voyage ignoring kinematic and dynamic constraints, while LPP is an ongoing process based on real-time information to generate a feasible local path (Siegwart et al., 2011; Vagale et al., 2021). Path following (PF) describes the task of following a pre-determined (local) path without considering the visitation time of a particular waypoint (Fossen, 2021). As stated in the beginning, in this paper, we focus on LPP and PF. We refer to the controlled ASV as the *own ship* and to its surrounding ships as *target ships*.

In recent years, the vessel traffic literature has started to embrace the advancements in artificial intelligence (Munim et al., 2020; Zhuge et al., 2023). One notable approach is deep reinforcement learning (DRL, Matsuo et al. 2022), which combines reinforcement learning (RL, Sutton and Barto 2018; Szepesvári 2010) with deep neural networks (Goodfellow et al., 2016). DRL has showcased remarkable success across challenging application domains (Vinyals et al., 2019; Ibarz et al., 2021; Bellemare et al., 2020; Silver et al., 2018; Soler et al., 2024; Kitchat et al., 2024), including vessel control tasks (Heiberg et al., 2022; Hart et al., 2023a). In RL, an agent acquires knowledge by engaging in iterative interactions with an environment, where it learns to make decisions through trial-and-error. For instance, in the context of LPP, the agent represents the own ship and is required to adapt its heading based on the presence of nearby

target ships and the geometric characteristics of the waterway. By defining an appropriate feedback mechanism, known as a reward, the agent receives penalties for causing collisions or running aground.

In contrast to many traditional control methods, DRL does not require prior knowledge or a complete model of the environment. Instead, this essential information is inherently embedded in the experiences the agent accumulates through training. Moreover, due to the use of deep neural networks as powerful function approximators, DRL can generalize across environmental situations, facilitating adaptation to previously unencountered scenarios (Kang et al., 2019).

While there have been applications of DRL to LPP and PF tasks on open waters, which are reviewed in Section 2, research on IWs is strongly limited. Regarding the LPP task on IWs, to the best of our knowledge, only Vanneste et al. (2022) have explored the application of DRL. However, their study is restricted to static obstacles and does not address the dynamic collision avoidance (COLAV) problem. Our study fills this gap by thoroughly considering the unique challenges associated with IW operations, including dealing with dynamically moving target ships, navigating through narrow waterways, and accounting for the effects of water depth on vessel dynamics. For the PF task on IWs, the closest work we identified is Paulig and Okhrin (2024), who applied the Deep Q-Network (DQN, Mnih et al. 2015) modification proposed in Waltz and Okhrin (2022) to achieve strong PF performances on various paths under different current influences. We extend their work by also considering the impact of waves and winds on the vessel, thereby further enhancing the comprehensiveness and robustness of the control system.

In summary, our study makes the following contributions:

- We introduce the first holistic control architecture for ASVs on IWs based on DRL. Our framework comprises separate agents dedicated to the LPP and PF tasks. The architecture accounts for dynamically moving target ships, environmental disturbances such as winds, waves, and currents, while also adhering to existing traffic rules and considering the geometry of the waterway.

- To accommodate continuous action spaces, we transfer the spatial-temporal recurrent network architecture from Waltz and Okhrin (2023) to actor-critic frameworks, and employ it for the LPP agent.

- To assess the effectiveness of our approach, we conduct extensive testing on both agents in a variety of challenging scenarios that are representative of the most difficult practical occurrences. In particular, we focus on advanced overtaking maneuvers and situations involving strong environmental forces. Furthermore, we validate the entire architecture by using real trajectories obtained from the automatic identification system (AIS).

- We numerically compare the performance of the two agents with strong baselines in the form of an artificial potential field (APF) (Liu et al., 2023; Wang et al., 2019) method for the LPP task and a PID controller (Paramesh and Rajendran, 2021) for the PF task. The performance metrics used for the comparison build on Jadhav et al. (2023).

- To ensure reproducibility and facilitate further research, we have made the source code

for this paper publicly accessible in Waltz and Paulig (2022). In addition, we have open-sourced our trajectory extraction pipeline from AIS data in Paulig (2023). By providing these resources, we aim to encourage the inclusion of real AIS data in the validation process of ASV control systems.

This paper is structured as follows: Section 2 provides additional information about traffic rules, sensor systems, and PF and planning algorithms. The newly proposed architecture is visualized and described in Section 3, while Section 4 details the relevant theory used in this work. Sections 5 and 6 contain detailed descriptions of the LPP and PF modules, respectively. The results and validation scenarios are shown in Section 7, and Section 8 concludes the paper.

2 Background and related work

2.1 Traffic rules

While path planning is an extensively studied problem in robotics (Siciliano et al., 2008), path planning for vessels presents unique challenges due to the incorporation of traffic rules specific to each waterway. The International Regulations for Preventing Collisions at Sea (COLREGs, International Maritime Organization 1972) form the basis of these rules, governing the required behavior of ships during encounters. Additionally, national regulations exist that further define the traffic rules for specific waters. Although our architecture is generally applicable for any IW, as a use case, we focus in this paper on the lower part of the Elbe river in northern Germany. The relevant regulation in this area is the *Seeschifffahrtsstraßen-Ordnung* of the Bundesministerium für Digitales und Verkehr (1998). Of particular importance to us are two specific rules within this regulation, which are also representative of IW regulations of other countries. First, overtaking should be conducted on the portside of the vessel being overtaken. Second, the vessel being overtaken should facilitate the overtaking maneuver as much as possible. For the convenience of the reader, the exact wording of the regulation in German together with its translation into English is provided in Appendix A.

2.2 ASV sensors

ASVs rely on navigational information on their current state, including position, velocity, accelerations, and environmental forces such as current and wind speed. These quantities stem from various sensors such as radar, LIDAR, sonar, visual sensors, infrared sensors, inertial measurement units, or the global positioning system (Liu et al., 2016). However, not every seagoing vessel is necessarily equipped with each of these sensors, and the sensor data might be noisy or erroneous and requires advanced state estimation techniques (Lefeber et al., 2003; Motwani et al., 2013). Moreover, data about target ships is received via AIS, mandatory equipment since the end of 2004 for all cargo ships of certain sizes and all passenger ships (International Maritime Organization, 2023). Furthermore, information about the waterway geometry and depth are provided via Electronic Navigational Charts (Blindheim and Johansen, 2021), which are digital maps usually available to seafarers.

2.3 Path following algorithms

The PF task describes the derivation of low-level control commands from a given local path. First, the controller needs to derive a guidance law from the received set of waypoints (Breivik and Fossen, 2009; Fossen, 2021). In this study, we use the vector-field guidance (VFG, Nelson et al. 2007b) detailed in Section 4.2. Second, the controller converts the directional awareness established through the guidance law to low-level actuator commands that minimize spatial and angular deviation from the desired path and the guidance signal. The actuators are typically propellers or rudders to control the vessel's speed and course, respectively. In this paper, we focus on using rudder adjustments as the primary actuator command. Steering changes are preferred over speed adjustments due to fuel considerations and better visual and radar observability of the course changes by other ships (Wang et al., 2017).

Various control algorithms can be employed to translate the guidance signal into an actuator command. Arguably, one of the most popular techniques is the classic proportional-integral-derivative (PID) controller (Paramesh and Rajendran, 2021), which will also serve as a benchmark for our PF module. Further conventional methodologies used in the literature include H_∞ control (Donha et al., 1998), linear quadratic Gaussian control (Sharma et al., 2012), model predictive control (Annamalai et al., 2015), sliding motion control (Liu et al., 2018), backstepping control (Zhang et al., 2017), $\mathcal{L}_1$ adaptive control (Breu and Fossen, 2011), or dynamic surface control (Wan et al., 2020), to mention a few. For a comprehensive comparison of PF control methods for ASVs, we refer to the recent survey of Xu and Guedes Soares (2023).

Moreover, recent research started to investigate the suitability of DRL for PF of ASVs, although primarily concentrated on open water situations while overlooking the joint impact of currents, winds, and waves. An important contribution in this realm is Woo et al. (2019), where the Deep Deterministic Policy Gradient (DDPG, Lillicrap et al. 2015) algorithm was used to successfully control a small-scale vessel to follow a linear path in a real-world experiment. Zhao et al. (2020b) outline a smoothly convergent modification of the DQN algorithm to perform PF on polygonal and sinusoidal paths. Another relevant related work is Martinsen and Lekkas (2018a), in which the DDPG algorithm was used for curved PF of three different vessel classes: a mariner, a container, and a tanker. Gonzalez-Garcia et al. (2020) combine the DDPG algorithm with an adaptive sliding mode control strategy (Castañeda et al., 2021) to let an ASV follow straight and zig-zag-like paths. Further contributions to DRL-based PF of ASVs include Wang et al. (2023), Zhao et al. (2021), and Peng et al. (2023).

2.4 Path planning algorithms

Tam et al. (2009) provide historical perspectives on the development of path planning algorithms for open waters, while recent reviews are available in Vagale et al. (2021), Öztürk et al. (2022), and Yu et al. (2023). The most important methodological approaches in the field include evolutionary algorithms (Tam and Bucknall, 2010), velocity obstacle methods (Kuwata et al., 2013), model predictive control (Johansen et al., 2016), APF methods (Lyu and Yin, 2019), rapidly-exploring random trees (RRT, Zhang et al. 2019), dynamic-window approaches (Serigstad et al., 2018), fast marching methods (Liu and Bucknall, 2015), whale optimization (Han et al., 2022),

and particle swarm optimization (PSO, Ding et al. 2018; Tutsoy et al. 2024).

In addition, there have been recent attempts to perform planning on open waters based on DRL. The first major study on this topic was Cheng and Zhang (2018), in which a concise DRL algorithm for static obstacle avoidance was proposed. Heiberg et al. (2022) designs a DRL agent using Proximal Policy Optimization (Schulman et al., 2017) while relying on advanced collision risk assessment theory. Xu et al. (2022a) apply the DDPG with a modified experience replay mechanism to tackle a path planning and COLAV task. A similar proposal has been made in Zhai et al. (2022), where the authors build on the DQN to construct a COLAV algorithm. Finally, Waltz and Okhrin (2023) outline a spatial-temporal recurrent network architecture for the DQN in maritime LPP. The architecture effectively handles an arbitrary number of target ships and is robust to partial observability. Further contributions for DRL-based approaches to maritime path planning include Chun et al. (2021), Li et al. (2021a), Meyer et al. (2020), and Guo et al. (2020).

In contrast to the extensive literature on algorithms for ASVs on open waters, there is a notable scarcity of work done on vessels operating on IWs, even in the realm of classical algorithms. Zhang et al. (2023b) tackle this gap by employing an anisotropic fast marching algorithm for ASVs, with a particular emphasis on navigating through restricted areas near bridges. Chen et al. (2016) compare the performance of the heuristic search algorithm A* (Hart et al., 1968) and its various extensions for GPP in Dutch IWs. Similar to the concepts introduced in Lyu and Yin (2018), Gan et al. (2022) develop a planning algorithm for inland rivers that incorporates the safety potential field theory to account for static and dynamic obstacles and waterway bank walls. Finally, Cao et al. (2022) detail a modification of the RRT algorithm to perform path planning on the Yangtze River Channel in Zhenjiang.

3 Proposed architecture

Figure 4.1 visualizes the proposed architecture for ASV control on IWs. The vessel controller comprises three key components: a GPP module, an LPP module, and a PF module. Two DRL agents are involved: a high-level agent at the core of the LPP module (in yellow) and a low-level agent for the PF task (in red). The computational flow of the architecture is as follows: The global path generated by the GPP module (in green) serves as input for the LPP module, which in turn produces a local path. The PF module processes this local path and generates an actuator control command. Note that we leave the algorithm or heuristic employed in the GPP module unspecified, although A* or its extensions are reasonable choices (Chen et al., 2016; Singh et al., 2018).

Describing the procedure in more detail, the LPP module is activated every t_{replan} seconds to generate a new local plan based on existing information. In our simulation, we set $t_{\text{replan}} = 30\,\text{s}$, although this parameter can be freely chosen depending on the geometrical difficulty of the waterway segment or the traffic density. To optimize computation time, the local path is replanned only when target ships are near the own ship. Otherwise, a simple linear local path can be constructed to return to the global path. In this case, it suffices to select a waypoint of the global path and linearly connect it to the own ship's position. Figure 4.2 (a) illustrates this

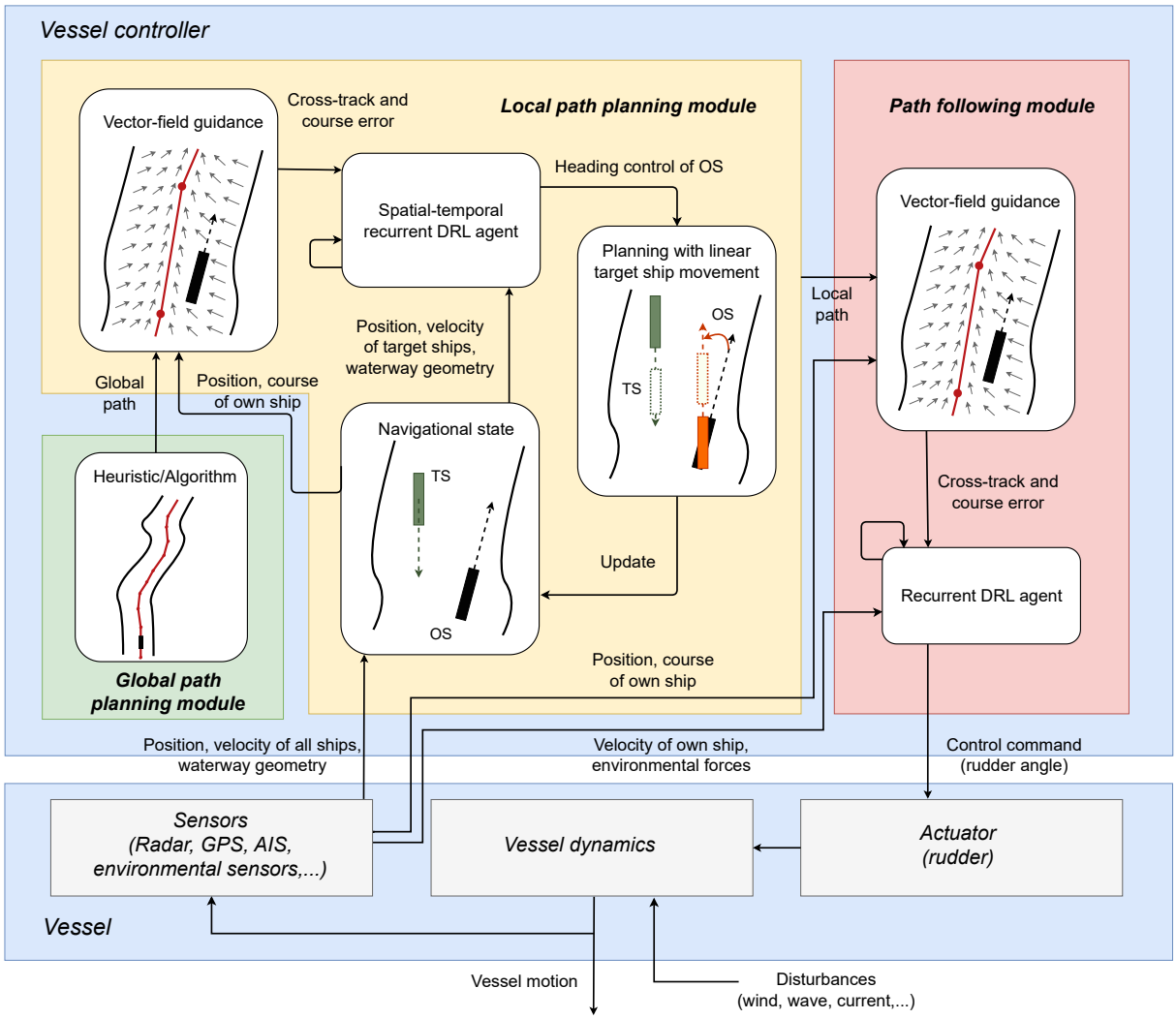
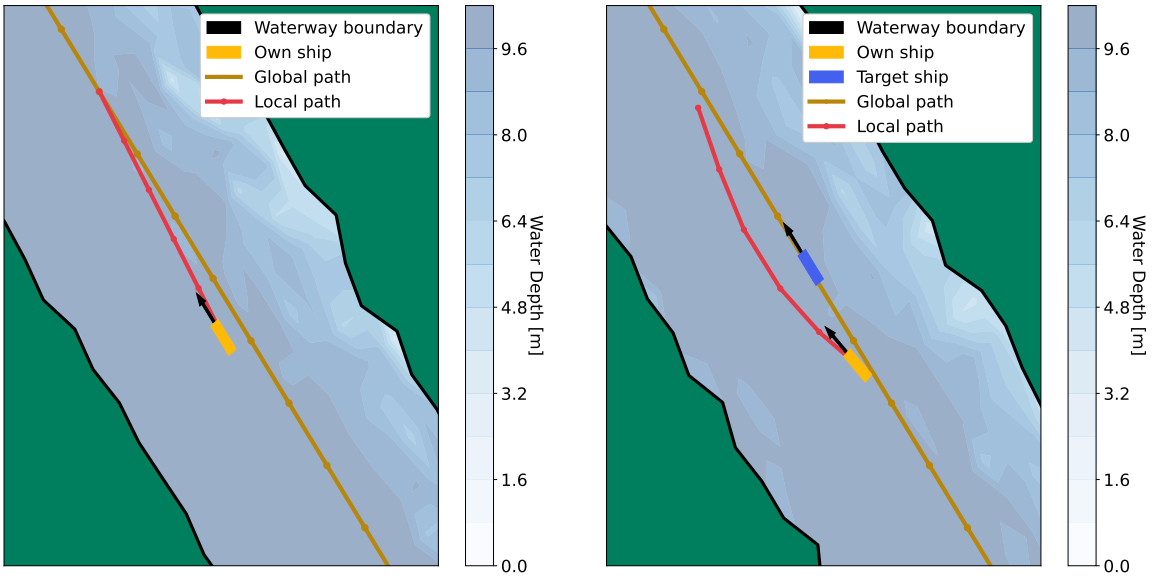


Figure 4.1: The proposed architecture for an ASV based on DRL, with the visualization being inspired by Chen et al. (2016).

procedure, while Figure 4.2 (b) visualizes the functionality of the framework in an overtaking scenario. In the latter case, the own ship recognizes the target ship in front and plans a safe local path accordingly.

When the LPP unit is activated, the DRL agent within it processes information from two sources: a cross-track and course error signal, indicating deviation from the global path, and navigational data from the sensor suite. The sensor data includes the positions and velocities of target ships as well as information about the waterway's geometry. Based on this information, the DRL agent generates a heading change command for the own ship, and the positions and velocities of all vessels are updated according to the vessel's model detailed in Section 4. The own ship's position is stored as a waypoint of the local path, and the error signals to the global path are updated. This process is repeated until a sufficient number of waypoints has been generated. We emphasize that the LPP procedure always occurs in simulation, even when deployed on a real ship, as path planning does not involve any actuator control.

For simplicity, control commands for the target ships are not incorporated during these planning iterations. Therefore, the LPP agent assumes that all target ships move linearly



(a) Linear local path

(b) Overtaking maneuver

Figure 4.2: The simplification of the LPP procedure if no target ships are present (left), and an overtaking maneuver (right).

and maintain a constant course and speed, regardless of their actual behavior. In practice, this simplification can be replaced with a more sophisticated intent estimation or trajectory prediction unit (Huang et al., 2020; Ma et al., 2021), which can be seamlessly integrated into the architecture.

Finally, the PF module is activated every t_{control} seconds, where it receives the local path and calculates the cross-track and course error using the VFG approach. Combined with velocity information and observations of environmental forces, an actuator command is generated to navigate the vessel along the desired path. In our simulation, we set the realistic value of $t_{\text{control}} = 5\,\text{s}$. In addition, we include Algorithm 1 to provide a concise summary of the overall computational flow.

Algorithm 1: 2-Level ASV control using DRL

Pre-Departure:
plan global path using any heuristic or algorithm

Procedure:
while *not arrived* **do**
  if *time to replan locally (every* t_{replan} *seconds)* **then**
    if *no target ships around* **then**
      plan linear path back to global path
    else
      plan local path using LPP agent (Section 5)
    end
  if *time to control actuators (every* t_{control} *seconds)* **then**
    adjust rudder angle using PF agent (Section 6)
end

4 Theory

4.1 Vessel dynamics

In this study, we employ the 3-degree-of-freedom Maneuvering Modeling Group (MMG) model introduced by Yasukawa and Yoshimura (2015) to simulate the dynamic behaviour of a 1:5 scale replica of the commonly used KVLCC2 tanker. The principal particulars of the tanker can be found in Paulig and Okhrin (2024). Two coordinate systems are considered in our research, which are illustrated in Figure 4.3. The first system, denoted as $\{n\}$, follows the *North-East-Down* convention. It describes the navigational status of the vessel using the vector $\eta = (x_n, y_n, \psi)^\top$. The variables x_n and y_n represent the north and east positions of the vessel, respectively, relative to the origin point o_n. The variable ψ is the heading angle of the ship and represents the angle between the x_n-axis and the x_b-axis of the second coordinate system, denoted as $\{b\}$. The $\{b\}$ reference frame is a *body-fixed* frame that is centered at the midship position of the vessel. Within this coordinate system, the longitudinal axis is denoted as x_b, and the transversal axis is denoted as y_b. The velocity of the vessel is described by the vector $\nu = (u, v, \tilde{r})^\top$, consisting of the surge velocity u, sway velocity v, and yaw rate $\tilde{r}$. Following Fossen (2021), the total speed of the vessel is $U = \sqrt{u^2 + v^2}$ and the course angle is $\chi = \psi + \arctan(v/u)$.

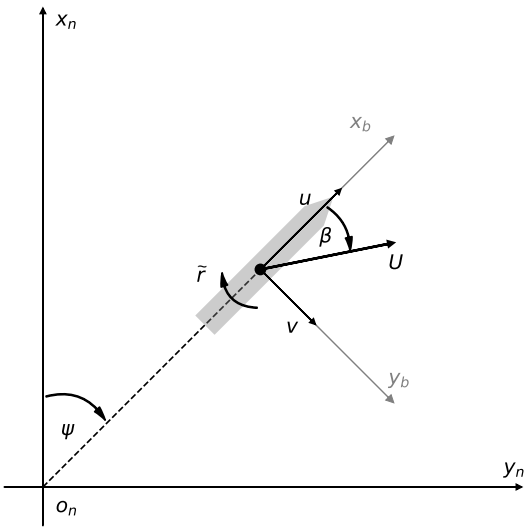


Figure 4.3: Coordinate systems considered in this work; see Yasukawa and Yoshimura (2015).

Unlike the majority of prior studies in the field of ASVs (Heiberg et al., 2022; Fan et al., 2022; Zhai et al., 2022), our research incorporates environmental forces, namely winds, waves, and currents, into our simulations. These forces play a critical role in real-life operations. Consequently, the vessel's movement is governed by the following set of equations:

$$\begin{aligned}
(m + m_{x_b})\dot{u} - (m + m_{y_b})v\tilde{r} - x_G m\tilde{r}^2 &= X = X_H + X_R + X_P + X_{WI} + X_{WA}, \\
(m + m_{y_b})\dot{v} + (m + m_{x_b})u\tilde{r} + x_G m\dot{\tilde{r}} &= Y = Y_H + Y_R + Y_{WI} + Y_{WA}, \\
(I_{zG} + x_G^2 m + J_z)\dot{\tilde{r}} + x_G m(\dot{v} + u\tilde{r}) &= N_m = N_H + N_R + N_{WI} + N_{WA},
\end{aligned} \tag{4.1}$$

where X is the surge force, Y denotes the lateral force, and N_m signifies the yaw moment around the midship. The first order derivative of a variable x with respect to time is denoted $\dot{x}$. The variables m, m_{x_b}, and m_{y_b} represent the mass of the ASV, the added masses in the x_b and y_b directions, respectively. Additionally, x_G is the longitudinal coordinate of the center of gravity in $\{b\}$, I_{zG} represents the moment of inertia, and J_z denotes the added moment of inertia. The set of equations (4.1) encompasses five distinct force components: hull (H), rudder (R), propeller (P), wind (WI), and wave (WA). It is important to note that the propeller component only affects the surge force X. The equations for the hull, rudder, and propeller components are derived in Yasukawa and Yoshimura (2015), which also provides relevant hydrodynamic derivatives and further parameter values. Importantly, the rudder components X_R, Y_R, and N_R depend on the rudder angle δ, which is used to control the ship.

To compute the wind forces depending on the wind speed V_{wi} and wind angle β_{wi}, we use the wind coefficient approximation for symmetrical ships described in Fossen (2021, Chapter 10.1). Regarding wave forces and moments, we follow Taimuri et al. (2020) and Sakamoto and Baba (1986) to compute the respective quantities depending on the wave amplitude ζ_{wa}, wave angle β_{wa}, wave period T_{wa}, and wave length λ_{wa}. The most significant environmental force for vessels operating on IWs are currents, represented by the current speed V_c and the current angle β_c in the global frame $\{n\}$. Following Fossen (2021, Chapter 6.7), these are not considered via an additional force or moment term in (4.1) but by directly specifying the velocity vector of the vessel relative to the currents. Consequently, the total speed of the ship in the presence of currents becomes $\sqrt{(u-u_c)^2+(v-v_c)^2}$, where u_c and v_c are the longitudinal and lateral components of the currents in $\{b\}$, respectively.

Furthermore, another crucial yet often neglected environmental characteristic for vessels operating on IWs is the influence of the water depth H on the vessel dynamics. We follow Taimuri et al. (2020) and consider the effects of shallow waters on wake fraction, thrust deduction, and flow-straightening coefficient as proposed by Amin and Hasegawa (2010), while the hydrodynamic derivatives in shallow waters are approximated following Kijima and Nakiri (1990) and Ankudinov et al. (1990).

In our simulation, we discretize the dynamics in (4.1) using a step size of 5 seconds, aligning with the control frequency t_{control} of the PF unit, and use the ballistic method of Treiber and Kanagaraj (2015) to update the velocity and the position vector of the vessel. Throughout the paper, we refer to a variable x at time t as x_t. We provide an overview of the used abbreviations and notation in Appendix B for the convenience of the reader.

4.2 Vector-field guidance

Both our LPP and PF units use VFG signals to ensure accurate path tracking. The concept of VFG originated from unmanned aerial vehicle control (Nelson et al., 2007b) and is visualized in Figure 4.4. The objective is to establish a vector field that guides the vessel back to its intended path, with the steepness of the field determined by a proportional gain parameter, denoted as $k > 0$. When given two consecutive waypoints, P_k and P_{k+1}, the angle from P_k to P_{k+1} in the global frame $\{n\}$ is denoted as χ_{P_k}. The vessel's cross-track and along-track errors are denoted as y_e and x_e respectively. Computation details for these quantities can be found in Fossen (2021,

Chapter 12.4).

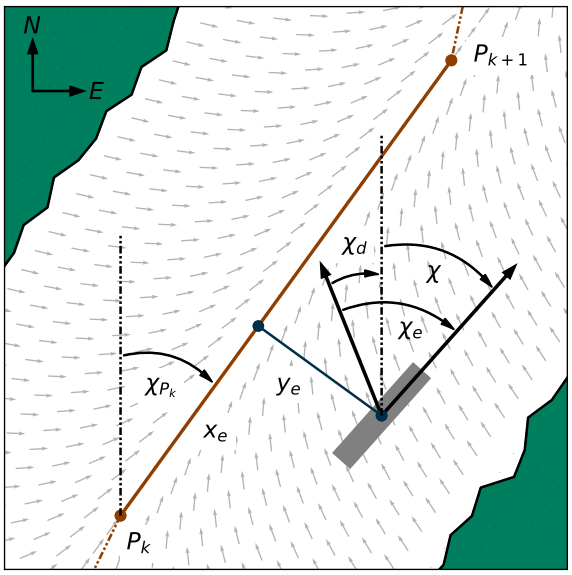


Figure 4.4: Visualization of the induced vector field; inspired by Paulig and Okhrin (2024).

The VFG method defines the desired course of the vessel as $\chi_d = \chi_{P_k} - \chi^\infty \frac{2}{\pi} \arctan(k \cdot y_e)$, where $\chi^\infty \in \left(0, \frac{\pi}{2}\right]$. In this study, we set $\chi^\infty = \frac{\pi}{2}$, effectively reducing the guidance mechanism to a proportional line-of-sight guidance law (Fossen and Pettersen, 2014). The formulation introduces a course error, $\chi_e = \chi_d - \chi$, which represents the deviation between the desired course and the actual course of the ship, χ. To address sudden changes in the desired course when a waypoint is passed, we adopt the approach proposed by Paulig and Okhrin (2024) and redefine the desired course to allow for a weighted transition between the current and subsequent path segments, resulting in smoother course adjustments. In particular, we set:

$$\chi_d = \left\{ \left[1 - \frac{x_e}{d(P_k, P_{k+1})} \right] \chi_{P_k} + \frac{x_e}{d(P_k, P_{k+1})} \chi_{P_{k+1}} \right\} - \arctan(k \cdot y_e), \tag{4.2}$$

where $d(P_k, P_{k+1})$ represents the Euclidean distance between waypoints P_k and P_{k+1}.

4.3 Collision risk assessment

Estimating the collision risk with nearby target ships is a core task of ASVs (Öztürk and Cicek, 2019). In the literature, two concepts hold special importance. The first concept, the *ship domain*, was originally introduced by Toyoda and Fujii (1971) and specifies a safe area around a vessel that should not be entered by any other ship. While various shapes of ship domains have been explored in the literature (Szlapczynski and Szlapczynska, 2017), we adopt a symmetric domain configuration that allows for one ship length of space in front of the USV's bow, and one ship width to the USV's starboard, stern, and port side. Furthermore, we define a collision event when the midship position of a target ship is at or inside our own ship's ship domain.

The second concept is the *closest point of approach* (CPA) (Lenart, 1983), which includes the measures distance (DCPA) and time (TCPA) to the CPA. The CPA quantifies the criticality

of a scenario by assuming that both the own ship and the target ship maintain their speed and course. Several recent studies have built upon these two concepts and defined specific collision risk metrics to assess the severity of a situation (Mou et al., 2010; Ha et al., 2021; Waltz and Okhrin, 2023).

4.4 Reinforcement learning

Reinforcement learning is a fundamental pillar of artificial intelligence that involves an agent learning through iterative trial-and-error interactions with an environment (Sutton and Barto, 2018). In this paradigm, the agent's interactions with the environment are formalized as a Markov decision process (MDP) (Puterman, 1994), represented by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. Here, $\mathcal{S}$ denotes the state space of the system, $\mathcal{A}$ represents the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the state transition probability function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a bounded reward function, and $\gamma \in [0, 1)$ is a discount factor that balances the importance of immediate and future rewards. The sum of discounted rewards is called *return*. The agent's objective within this framework is to optimize for a policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, a mapping from states to probability distributions over actions, that maximizes the expected return.

The methods to achieve this task are commonly divided into *value-based* and *policy gradient* algorithms (Sutton and Barto, 2018). Value-based methods typically learn a policy-dependent action-value for each state-action pair (s, a), denoted $Q^\pi(s, a)$, which represents the expected return when being in state s, executing action a, and following policy π afterward. Crucially, under fulfillment of regularity conditions, an optimal policy π^* can be derived by behaving greedily to the optimal action-value function Q^*, where the latter fulfills $Q^*(s, a) = \max_\pi Q^\pi(s, a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$ (Puterman, 1994). Policy gradient algorithms, on the other hand, directly optimize a parametrized policy π^θ with respect to a performance measure such as the expected return under policy π^θ over the initial state distribution. The set θ is the parameter set of the policy π^θ, for example, the biases and weights of a neural network. Lastly, several state-of-the-art algorithms belong to the class of *actor-critic* algorithms, which are policy gradient algorithms that use action-value estimates during the gradient step. In particular, the policy is referred to as the *actor*, and the action-value function is the *critic* (Fujimoto et al., 2018).

4.5 RL algorithm LSTM-TD3

In practical applications, fully-observed systems, where the agent has access to the complete state vector, are uncommon due to sensor noise, delays, or other disturbances (Meng et al., 2021). Recognizing this reality, we extend the MDP formalism to encompass partially-observable MDPs (Kaelbling et al., 1998) by introducing an observation space $\mathcal{O}$ and an observation function $\mathcal{Z} : \mathcal{S} \times \mathcal{A} \times \mathcal{O} \to [0, 1]$. Consequently, at each time step t, the system resides in a state $s_t \in \mathcal{S}$, the agent receives an observation $o_t \in \mathcal{O}$ generated according to $\mathcal{Z}$, and then selects an action $a_t \in \mathcal{A}$. The system transitions to the next state $s_{t+1} \in \mathcal{S}$ based on $\mathcal{P}$ and provides a reward r_t determined by $\mathcal{R}$. During implementation, the state input $s \in \mathcal{S}$ for an action-value function and policy is replaced by an observation $o \in \mathcal{O}$. For example, in our PF module, the observation vector includes the deviation to the local path and the currently acting environmental forces, while the action is a change in the rudder angle.

In this study, we use the Long Short-Term Memory-based Twin Delayed Deep Deterministic Policy Gradient (LSTM-TD3) algorithm proposed by Meng et al. (2021) for both the LPP and PF agents. Building upon prior work by Lillicrap et al. (2015) and Fujimoto et al. (2018), the LSTM-TD3 is an actor-critic algorithm maintaining an actor μ with parameter set θ_μ and two critics Q_1 and Q_2 with parameter sets θ_1 and θ_2, respectively. Crucially, all networks incorporate LSTM layers (Hochreiter and Schmidhuber, 1997), which process the information of several times steps and offer a robust solution to address the challenges posed by partial observability. The optimization of the critics uses a DQN-style update (Mnih et al., 2015) with the clipped double Q-Learning target proposed in Fujimoto et al. (2018), while the actor is optimized based on the deterministic policy gradient theorem of Silver et al. (2014).

While the algorithm is the same, the architecture of the neural networks differ for our LPP and PF agents and will be explained in detail in the upcoming Sections 5 and 6.

5 Local path planning module

5.1 Configuration of the RL agent

Observation space

The LPP agent constitutes the core of the LPP module, as illustrated in Figure 4.1. Its responsibility is to generate a reliable local plan while taking into account surrounding target ships, traffic rules, and the waterway geometry. This module does not consider environmental forces such as winds, waves, and currents, as those fall under the responsibility of the PF unit.

The observation for the LPP agent at time t, denoted as o_t^{LPP}, is defined by stacking three vector components: $o_{\mathrm{OS,t}}^{\mathrm{LPP}}$, which summarizes information about the own ship; $o_{\mathrm{IW,t}}^{\mathrm{LPP}}$, which describes the navigational area; and $o_{\mathrm{TS,t}}^{\mathrm{LPP}}$, which delivers information about the surrounding target ships. Specifically, we have:

$$o_t^{\mathrm{LPP}} = \left(\left(o_{\mathrm{OS},t}^{\mathrm{LPP}}\right)^\top, \left(o_{\mathrm{IW},t}^{\mathrm{LPP}}\right)^\top, \left(o_{\mathrm{TS},t}^{\mathrm{LPP}}\right)^\top \right)^\top. \tag{4.3}$$

Own ship observation. For the observation about the own ship, $o_{\mathrm{OS,t}}^{\mathrm{LPP}}$, the following features are included:

$$o_{\mathrm{OS},t}^{\mathrm{LPP}} = \left(\frac{U_{\mathrm{OS},t}}{U_{\mathrm{scale}}}, \frac{[\psi_{\mathrm{OS},t} - \chi_{P_k,t}]_{-\pi}^{\pi}}{\pi}, \frac{[\chi_{e,t}^{\mathrm{global}}]_{-\pi}^{\pi}}{\pi}, \frac{y_{e,t}^{\mathrm{global}}}{y_{\mathrm{scale}}} \right)^\top, \tag{4.4}$$

where $U_{\mathrm{OS},t}$ represents the speed of the own ship at time t, $\psi_{\mathrm{OS},t}$ is the heading of the own ship, $\chi_{P_k,t}$ is the angle between the two active waypoints of the own ship (as shown in Figure 4.4), $y_{e,t}^{\mathrm{global}}$ is the cross-track error on the global path, and $\chi_{e,t}^{\mathrm{global}}$ is the course error derived from the VFG method. The superscript *global* indicates that these features are specific to the LPP agent and are computed with respect to the global path. The VFG gain parameter is set to $k^{\mathrm{LPP}} = 0.001$. The function $[\cdot]_a^{a+2\pi} : \mathbb{R} \to [a, a+2\pi)$ is used to transform an angle to a desired domain; see Waltz and Okhrin (2023). Additionally, we have scaling parameters $U_{\mathrm{scale}} = 3\,\mathrm{m/s}$ and $y_{\mathrm{scale}} = 64\,\mathrm{m}$, where 64 meters corresponds to one length between perpendiculars (L_{pp}) of the downscaled KVLCC2 tanker.

Waterway observation. To ensure the agent can plan without running aground, it requires information about the navigational area. We construct a vector, denoted as $o^{\text{LPP}}_{\text{IW},t}$, which contains proximities to the navigational boundary in 10 different directions; see Figure 4.5:

$$o^{\text{LPP}}_{\text{IW},t} = \left(1 - \frac{d^H(\gamma_1)}{d^H_{\text{scale}}}, \ldots, 1 - \frac{d^H(\gamma_{10})}{d^H_{\text{scale}}}\right)^\top. \tag{4.5}$$

The value γ_i represents the i-th component, where $i = 1, \ldots, 10$, in radians of the vector with degree values $(0, 20, 45, 90, 135, 180, 225, 270, 315, 340)^\top$. We carefully selected these angles to provide the agent with a comprehensive view of its surroundings while avoiding the need for a high-dimensional feature vector that would require specifying angles, for example, for every degree around the clock. The function $d^H : [0, 2\pi) \to [0, d^H_{\text{scale}}]$ calculates the distance to either the coastline or the global path of the opposing traffic for a given angle.

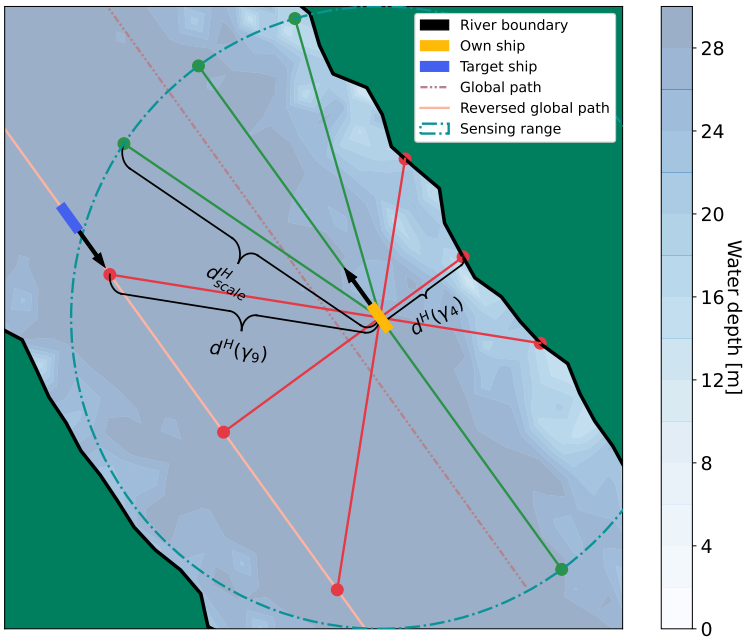


Figure 4.5: Visualization of $o^{\text{LPP}}_{\text{IW},t}$, representing the LPP units' awareness of the geometry of the waterway. The reversed global path is the path used to generate opposing traffic.

During implementation, we examine for each angle up to 50 logarithmically scaled distances within a maximum range of $d^H_{\text{scale}} = 1\,\text{NM}$, leading to a normalization of the fractions in (4.5) to the interval $[0, 1]$. We determine if a particular point is below the required water depth for the tanker or if it lies on the opposite side of the opposing traffic's path. Additionally, we follow the approach in Heiberg et al. (2022) by reversing the signs of the fractions to provide the agent with information about the proximity rather than the distance to either the coastline or the opposing path.

Target ship observation. For the observation of the i-th target ship at time t, denoted as

$o_{\mathrm{TS},i,t}$, the following features are considered:

$$o_{\mathrm{TS},i,t} = \left(\frac{d^i_{\mathrm{OS},t} - D(\alpha^i_{\mathrm{OS},t})}{d_{\mathrm{scale}}}, \frac{[\alpha^i_{\mathrm{OS},t}]^{\pi}_{-\pi}}{\pi}, \frac{[\psi_{i,t} - \chi_{P_k,t}]^{\pi}_{-\pi}}{\pi}, \frac{U_{i,t} - U_{\mathrm{OS},t}}{U_{\mathrm{scale}}}, \sigma_{i,t}, \frac{t^{\mathrm{cpa}}_{i,t}}{t_{\mathrm{norm}}}, \frac{d^{\mathrm{cpa},*}_{i,t}}{d_{\mathrm{norm}}} \right)^{\top}. \tag{4.6}$$

Here, $d^i_{\mathrm{OS},t}$ represents the Euclidean distance between the own ship and the i-th target ship, $\alpha^i_{\mathrm{OS},t}$ is the relative bearing of ship i from the perspective of the own ship, $\psi_{i,t}$ and $U_{i,t}$ are the heading and speed of the target ship, respectively. The function $D : [0, 2\pi) \to \mathbb{R}$ computes the ship domain around the own ship for a certain angle. Thus, the distance between the target ship's midpoint and the own ship's ship domain is provided to the agent, directly reflecting our definition of a collision event. The binary variable $\sigma_{i,t}$ indicates whether the other vessel is traveling in the same direction as the own ship and is defined as follows:

$$\sigma_{i,t} = \begin{cases} -1 & \text{if} \quad |[\psi_{i,t} - \psi_{\mathrm{OS},t}]^{\pi}_{-\pi}| \geq \pi/2, \\ 1 & \text{else.} \end{cases} \tag{4.7}$$

The collision risk with the other ship is captured by $t^{\mathrm{cpa}}_{i,t}$, the TCPA between the own ship and ship i, and the variable $d^{\mathrm{cpa},*}_{i,t}$. The latter follows Waltz and Okhrin (2023) and is defined as $d^{\mathrm{cpa},*}_{i,t} = \max\left[0, d^{\mathrm{cpa}}_{i,t} - D(\alpha^{i,\mathrm{cpa}}_{\mathrm{OS},t})\right]$, where $d^{\mathrm{cpa}}_{i,t}$ is the regular DCPA and $\alpha^{i,\mathrm{cpa}}_{\mathrm{OS},t}$ is the relative bearing of the ship i from the perspective of the own ship at the CPA. Thus, we consider the ship domain in the computation of the DCPA metric similar to as we do it for the present distance to the target ship. The scaling parameters $t_{\mathrm{norm}} = 300\,\mathrm{s}$ and $d_{\mathrm{norm}} = 0.25\,\mathrm{NM}$ are used.

The complete target ship observation vector is constructed as:

$$o^{\mathrm{LPP}}_{\mathrm{TS},t} = \left((o_{\mathrm{TS},1,t})^{\top}, \ldots, (o_{\mathrm{TS},N_t,t})^{\top} \right)^{\top}, \tag{4.8}$$

where N_t is the number of present target ships at time step t. The target ships within this vector are sorted in descending order based on their distance to the own ship. Furthermore, a target ship is only included if its distance to the own ship is less than $d_{\mathrm{scale}} = 0.5\,\mathrm{NM}$, which is considered a reasonable range for practical operations on IWs.

Neural network architecture

The varying number of surrounding ships, denoted as N_t, introduces a challenge for conventional feed-forward neural networks, which typically rely on a fixed input size. To overcome this challenge, we leverage the spatial-temporal recurrent neural network architecture proposed by Waltz and Okhrin (2023) and integrate it into actor-critic methods. This adaptation empowers the model to effectively handle continuous action spaces. The schematic representation of our proposed architecture is illustrated in Figure 4.6. In the figure, the concatenation operator is denoted as $\bowtie$, the fully connected (FC) layers have 64 neurons, and the number of hidden units in the LSTM layers is 64.

In particular, we incorporate the spatial-temporal recurrent structure into both the actor and the two critics of the LSTM-TD3 algorithm. The spatial recurrent component loops over surrounding vessels, while the temporal recurrent component loops over time steps. This design

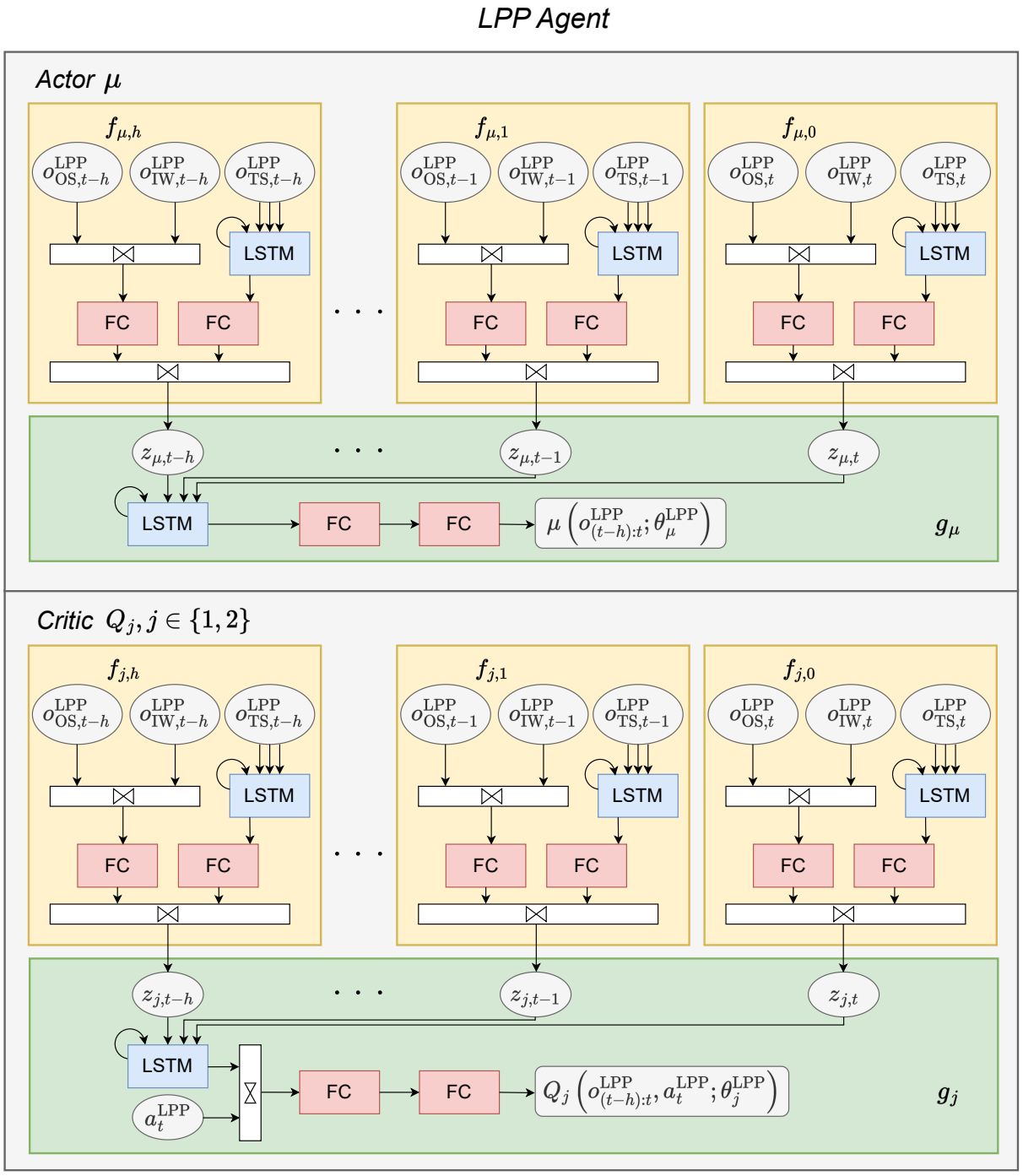


Figure 4.6: Neural network architecture for the LPP agent; adapted from Waltz and Okhrin (2023). We apply a ReLU activation after each FC layer, except for the last one of the actor and the critic, which use a tanh and linear activation, respectively.

choice equips our LPP agent with resilience against partial observability, while allowing it to adapt seamlessly to scenarios involving varying numbers of target ships.

The actor's architecture closely mirrors that presented in Waltz and Okhrin (2023), except that we apply a hyperbolic tangent activation at the last layer to generate actions within the range $[-1,1]^{|\mathcal{A}|}$, where $|\mathcal{A}|$ denotes the cardinality of the action space. Further details on the

action space are provided in Section 5.1. Formally, the actor network can be described as follows:

$$\begin{aligned} z_{\mu,t-l} &= f_{\mu,l}\left(o^{\text{LPP}}_{\text{OS},t-l}, o^{\text{LPP}}_{\text{IW},t-l}, o^{\text{LPP}}_{\text{TS},t-l}; \theta_{f_{\mu,l}}\right) \quad \text{for} \quad l = 0, \dots, h, \\ \mu\left(o^{\text{LPP}}_{(t-h):t}; \theta^{\text{LPP}}_{\mu}\right) &= g_{\mu}\left(z_{\mu,t-h}, \dots, z_{\mu,t-1}, z_{\mu,t}; \theta_{g_\mu}\right), \end{aligned} \tag{4.9}$$

where h is the history length, the functions $f_{\mu,l}$ with parameter sets $\theta_{f_{\mu,l}}$ for $l = 0, \dots, h$ are the spatial recurrent components, and the function g_μ with parameter set θ_{g_μ} represents the temporal recurrency. The notation $o^{\text{LPP}}_{(t-h):t} = \cup_{l=0}^{h} o^{\text{LPP}}_{t-l}$ is used to indicate that the actor network is a function of the past observations, while the complete parameter set of the actor is $\theta^{\text{LPP}}_{\mu} = \left(\cup_{l=0}^{h} \theta_{f_{\mu,l}}\right) \cup \theta_{g_\mu}$.

Similarly, the critics' architecture follows the principles outlined in Waltz and Okhrin (2023), with the difference that we concatenate the output of the temporal LSTM with the actions supplied by the actor. Formally, we have for critic Q_j with $j \in \{1, 2\}$:

$$\begin{aligned} z_{j,t-l} &= f_{j,l}\left(o^{\text{LPP}}_{\text{OS},t-l}, o^{\text{LPP}}_{\text{IW},t-l}, o^{\text{LPP}}_{\text{TS},t-l}; \theta_{f_{j,l}}\right) \quad \text{for} \quad l = 0, \dots, h, \\ Q_j\left(o^{\text{LPP}}_{(t-h):t}, a^{\text{LPP}}_{t}; \theta^{\text{LPP}}_{j}\right) &= g_j\left(z_{j,t-h}, \dots, z_{j,t-1}, z_{j,t}, a^{\text{LPP}}_{t}; \theta_{g_j}\right), \end{aligned} \tag{4.10}$$

with the spatial recurrent functions $f_{j,l}$ for $l = 0, \dots, h$, parametrized with sets $\theta_{f_{j,l}}$, and the function g_j with parameter set θ_{g_j}, which represents the temporal recurrent component. The critics evaluate the action a^{LPP}_t provided by the actor, and the complete parameter set of critic Q_j is denoted $\theta^{\text{LPP}}_j = \left(\cup_{l=0}^{h} \theta_{f_{j,l}}\right) \cup \theta_{g_j}$.

If there are no target ships present, we artificially create a no-risk ship with $o_{\text{TS},1,t} = (1, -1, 1, -1, 1, -1, 1)^\top$ to fulfill the requirement of having at least one target ship for the recurrence loop in the network architecture.

Action space

We define a one-dimensional action space ($|\mathcal{A}| = 1$) for the LPP unit, which represents a change in the own ship's heading. At time step t, the agent generates an action $a^{\text{LPP}}_t \in [-1, 1]$ that is used to update the own ship's heading $\psi_{\text{OS},t}$ according to the following formula:

$$\psi_{\text{OS},t+1} = \psi_{\text{OS},t} + a^{\text{LPP}}_t \cdot a^{\text{LPP}}_c, \tag{4.11}$$

where $a^{\text{LPP}}_c = 10°$. To ensure realistic and feasible paths, we only apply the agent's action every four time steps. Given our simulation step size of 5 seconds, this means that the planner can adjust the heading by a maximum of ten degrees every 20 seconds.

Reward function

Designing an appropriate reward function is crucial in RL applications as it provides feedback to the agent's actions. By incorporating insights from Waltz and Okhrin (2023) and Paulig and Okhrin (2024), we have identified five reward components that facilitate LPP on IWs. These components address global PF, COLAV, traffic rule adherence, and comfort considerations.

The first two components focus on following the global path. We introduce a cross-track error-based reward, denoted as $r^{\text{LPP}}_{y_e,t}$, and a course error-based component, denoted as $r^{\text{LPP}}_{\chi_e,t}$. They are defined as follows:

$$r^{\text{LPP}}_{y_e,t} = \exp\left(-k^{\text{LPP}}_{y_e} \cdot \frac{|y^{\text{global}}_{e,t}|}{y_{e,\text{norm}}}\right), \quad r^{\text{LPP}}_{\chi_e,t} = \exp\left(-k^{\text{LPP}}_{\chi_e} \cdot |[\chi^{\text{global}}_{e,t}]^{\pi}_{-\pi}|\right), \tag{4.12}$$

where the power weights are $k^{\text{LPP}}_{y_e} = 2$ and $k^{\text{LPP}}_{\chi_e} = 4$, and the normalization length is $y_{e,\text{norm}} = 128\,\text{m}$, which corresponds to $2L_{pp}$ of the KVLCC2 replica ship. These values are chosen to appropriately weigh the importance of the respective error terms in the reward computation.

The third reward component, denoted as $r^{\text{LPP}}_{\text{coll},t}$, penalizes collision events with other ships and leaving the navigational area. We, therefore, define the binary variables $\sigma_{\text{ground},t}$ and $\sigma_{\text{lane},t}$, which take the value one if the ship runs aground or crosses the global path of the opposing traffic, respectively, and zero otherwise. Both situations indicate leaving the navigational area and are therefore considered collision events. Further, we define $\sigma_{\text{coll},i,t}$ as a binary variable that takes the value one if the own ship has a collision with target ship i at time step t, and zero otherwise.

In addition, research on COLAV for ASVs on open waters has shown the benefit of including a distance-based collision reward component (Xu et al., 2022a). We also include such a signal into $r^{\text{LPP}}_{\text{coll},t}$ since it has the advantage of permanently yielding feedback for the RL agent, also in non-collision events. In particular, we will impose an elliptical collision reward around a target ship, where we encourage the agent to maintain a larger longitudinal distance from the target ship while allowing for smaller lateral distances, as the latter is necessary for overtaking maneuvers. This behavior is specified via the function $f : [0, 2\pi) \times \mathbb{R} \to \mathbb{R}$ as follows:

$$f(\alpha, d) = \exp\left\{\frac{-[d \cdot \sin(\alpha)]^2}{e^2_{\text{norm}}}\right\} \cdot \exp\left\{\frac{-[d \cdot \cos(\alpha)]^2}{n^2_{\text{norm}}}\right\}, \tag{4.13}$$

where $e_{\text{norm}} = 3B$ and $n_{\text{norm}} = 1L_{pp}$, with B being the width of the ship. On this basis, we introduce $r^{\text{LPP}}_{\text{coll},t}$ as:

$$r^{\text{LPP}}_{\text{coll,t}} = k_{\text{coll}} \cdot \left(\sigma_{\text{ground,t}} + \sigma_{\text{lane,t}} + \textstyle\sum_{i=1}^{N_t} \sigma_{\text{coll,i,t}}\right) - \max_{i=1,\ldots,N_t} f\left[\alpha^{\text{OS}}_{i,t}, d^i_{\text{OS},t} - D\left(\alpha^i_{\text{OS},t}\right)\right] \tag{4.14}$$

where the collision weight k_{coll} is set to -10, the variable $\alpha^{\text{OS}}_{i,t}$ represents the relative bearing of the own ship from the perspective of target ship i, and $\alpha^i_{\text{OS},t}$ is the relative bearing of target ship i from the perspective of the own ship.

The fourth reward component, $r^{\text{LPP}}_{\text{rule,t}}$, concerns the compliance with traffic rules, which is investigated for each target ship separately. We define the binary variable $\sigma_{\text{rule},i,t}$ that takes value one if at time step t a traffic rule with respect to target ship i is violated. A traffic rule violation consists of three components in this study, which must be fulfilled simultaneously. First, the target ship needs to be close enough to be relevant. Second, it needs to travel in the same direction as the own ship. Third, one of the two behavioral rules we enforce during simulation is violated. As described in Section 2.1, we require that other ships should be overtaken on their

port side, corresponding to §23 (1) of Bundesministerium für Digitales und Verkehr (1998), and that if the own ship is being overtaken by another vessel, it should facilitate the maneuver by making space for the target ship. The latter requirement represents §23 (2) of the regulation, and we introduce the variable $\sigma_{\text{spd},t}$ to check for its fulfillment. The variable takes the value one if all target ships traveling in the same direction as the own ship are faster than the own ship, and zero otherwise.

In summary, we define $r^{\text{LPP}}_{\text{rule,t}} = k_{\text{rule}} \cdot \sum_{i=1}^{N_t} \sigma_{\text{rule,i,t}}$, where we set the constant to $k_{\text{rule}} = -2$, and define the binary violation variable $\sigma_{\text{rule,i,t}}$ as follows:

$$\sigma_{\text{rule,i,t}} = \begin{cases} 1 & \text{if} \quad \left\{ d^i_{\text{OS,t}} \leq g(\alpha^{\text{OS}}_{i,t}) \right\} \wedge \left\{ |[\psi_{i,t} - \psi_{\text{OS},t}]^{\pi}_{-\pi}| < \frac{\pi}{2} \right\} \\ & \wedge \left\{ \left[(U_{\text{OS,t}} > U_{i,t}) \wedge \left(\frac{\pi}{2} \leq \alpha^{\text{OS}}_{i,t} \leq \pi \right) \right] \vee \left[\sigma_{\text{spd,t}} \wedge \left(\frac{3}{2}\pi \leq \alpha^{\text{OS}}_{i,t} < 2\pi \right) \right] \right\} \\ 0 & \text{else.} \end{cases} \tag{4.15}$$

The function $g : [0, 2\pi) \to \mathbb{R}$ calculates a bearing-dependent distance to determine if the target ship is within a range that requires consideration of traffic rules. Specifically, we select a lateral distance of 0.25 NM and a longitudinal distance of 0.5 NM from the target ship, and $g(\cdot)$ is defined to linearly interpolate between the four resulting corner points.

The fifth component is a comfort reward, $r^{\text{LPP}}_{\text{comf,t}}$, and should prevent the agent from frequently selecting large heading changes, resulting in a stable and smooth local path. Thus, we define: $r^{\text{LPP}}_{\text{comf,t}} = -\left(a^{\text{LPP}}_t\right)^2$. Finally, we aggregate all five reward components into a single scalar via:

$$r^{\text{LPP}}_t = r^{\text{LPP}}_{y_e,t}\omega^{\text{LPP}}_{y_e} + r^{\text{LPP}}_{\chi_e,t}\omega^{\text{LPP}}_{\chi_e} + r^{\text{LPP}}_{\text{coll,t}}\omega^{\text{LPP}}_{\text{coll}} + r^{\text{LPP}}_{\text{rule,t}}\omega^{\text{LPP}}_{\text{rule}} + r^{\text{LPP}}_{\text{comf,t}}\omega^{\text{LPP}}_{\text{comf}}, \tag{4.16}$$

where we experimentally determined the weights as follows: $\omega^{\text{LPP}}_{y_e} = \frac{4}{19} \approx 0.211$, $\omega^{\text{LPP}}_{\chi_e} = \frac{1}{19} \approx 0.053$, $\omega_{\text{coll}} = \omega^{\text{LPP}}_{\text{rule}} = \frac{6}{19} \approx 0.316$, and $\omega^{\text{LPP}}_{\text{comf}} = \frac{2}{19} \approx 0.105$.

5.2 Training environment

Waterway generation

In the following, we denote a real-valued uniform distribution with support $[a, b]$ as $\mathcal{U}(a, b)$, an integer-valued uniform distribution with support $[a, b]$ as $\mathcal{DU}(a, b)$, and an exponential distribution with expectation β as $Exp(\beta)$. To establish the simulation environment, we first sample a global path by interchanging straight and curved segments following the method described in Fossen (2021, Chapter 12). The length of each straight river interval is generated according to $400\,\text{m} + \mathcal{DU}(0, 32) \cdot 50\,\text{m}$, and the radii of the curves are sampled from $1000\,\text{m} + \mathcal{DU}(0, 4000) \cdot 1\,\text{m}$. Additionally, the curve angles are generated from $60^\circ + \mathcal{DU}(0, 40) \cdot 1^\circ$. We randomly sample left or right curves, respectively. Furthermore, we construct another path by imposing a fixed offset of 200 meters to the global path. The second path is referred to as reversed global path and is responsible for generating the opposing traffic.

Once we have established the global path, we proceed to create an IW by sampling water depths around it, following Paulig and Okhrin (2024). Initially, we sample a maximum water depth in meters from $\text{clip}\,[Exp(35), 20, 100]$ with the clipping operation: $\text{clip}(x, l, u) = \min\,[u, \max(x, l)]$. The generated waterway has a maximum width of 500 meters. To introduce

some variability to the depth data, we incorporate noise by adding realizations from $\mathcal{U}(-2, 2)$ in meters. We resample a generated waterway at the beginning of every fifth episode to keep the computational effort low.

Target ship generation

To initialize the target ships, we incorporate insights from the recent work conducted by Hart and Okhrin (2024). Our aim is to create training scenarios that pose significant challenges to the agent, increasing the risk of collisions with other ships. In each episode, we randomly sample $N \sim \mathcal{DU}(0, 10)$ target ships, encompassing a range of speeds, including both slower and faster vessels than the own ship, as well as ships traveling in opposing directions. The base vessel speed in our simulation is $U_{\text{base}} = 3\,\text{m/s}$, and at each episode, we randomly sample the own ship's speed from $\mathcal{U}(0.8, 1.2) \cdot U_{\text{base}}$.

To initiate target ship i, where $i = 1, \ldots, N$ if $N > 0$, there is a 15% probability of sampling a faster ship with a speed of $U_i \sim \mathcal{U}(1.3, 1.5) \cdot U_{\text{base}}$. In this case, the vessel i is initiated behind the own ship and travels in the same direction. The initial distance to the own ship along the global path, denoted d_i, is generated via $d_i \sim \mathcal{U}(0.3, 0.7) \cdot 1\,\text{NM}$, ensuring the vessel will create a threat for the own ship in the future. In the remaining 85% of cases, vessel i is slower with $U_i \sim \mathcal{U}(0.4, 0.8) \cdot U_{\text{base}}$ and is initiated in front of the own ship. In this case, the vessel i is randomly assigned to travel in the same or opposing direction. When traveling in the opposing direction, we sample $d_i \sim \mathcal{U}(1.1, 1.9) \cdot 1\,\text{NM}$ and place vessel i on the reversed global path. Otherwise, we have $d_i \sim \mathcal{U}(0.3, 0.7) \cdot 1\,\text{NM}$.

To introduce further variability and prevent the target ships from consistently spawning exactly on their respective paths, we introduce small positional noise to the target vessel's initial positions. An episode ends if a maximum of 150 steps has been reached or if the agent is more than 0.5 NM away from the global path. A screenshot of the full environment including waterway and target ships is shown in Figure 4.7. The own ship is depicted in red and travels in the south-east direction in this case, while the target ships are grey if they travel in the same direction as the own ship, and golden otherwise.

Target ship behavior modeling

One of the key considerations in simulating AVSs is the control mechanism for the target ships (Zhou et al., 2019). In our study, we focus on the non-communicative scenario with no explicit exchange of intentions or planned trajectories among vessels. In many existing works on ASVs, the assumption of linearly moving target ships with no course and speed changes is adopted (Guo et al., 2020; Fan et al., 2022; Xu et al., 2022a,b; Sawada et al., 2021). However, while trained policies based on this assumption can generalize to non-linear target ship motions, as shown by Waltz and Okhrin (2023), the linearity and non-reactiveness assumptions are unrealistic in practical scenarios, particularly on IWs.

To address this limitation, we employ a simple rule-based controller for the target ships during the training of the LPP agent. The basis of this controller is the VFG method, as outlined in Section 4.2. Additionally, the target ships are capable of performing basic overtaking maneuvers, in compliance with §23(1) of the collision regulations outlined in Bundesministerium

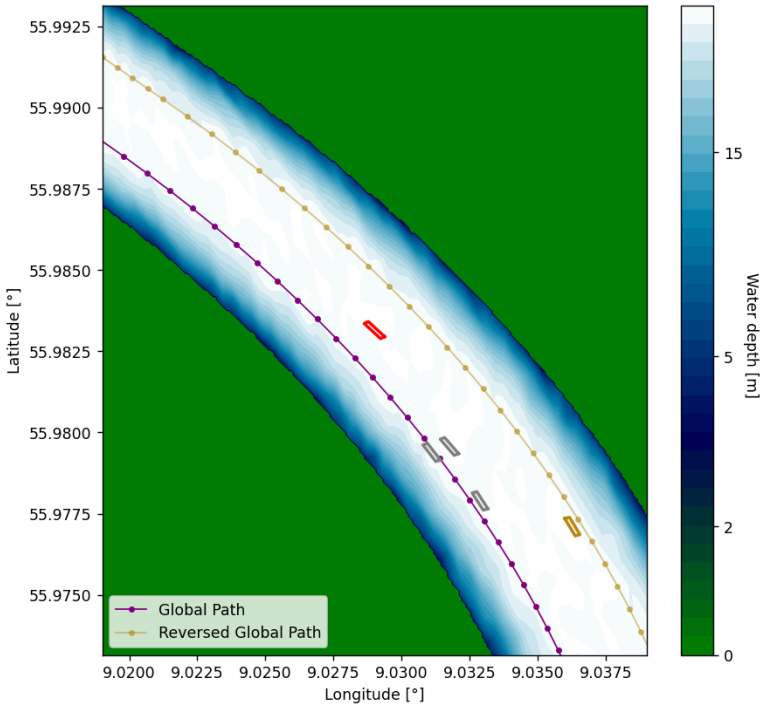


Figure 4.7: Screenshot of the simulation environment for the LPP agent (red) with four target ships (grey in the agent's direction, golden otherwise). The latitude and longitude values are artificial and serve as orientation.

für Digitales und Verkehr (1998). In head-on scenarios, they avoid collisions by executing a turn to starboard. For a detailed description of the complete controller, please refer to Algorithm 2 in Appendix C. Importantly, considering the recent work of Akdağ et al. (2022), we incorporate a 20% probability of generating a non-cooperative target vessel which does not perform any COLAV maneuvers.

6 Path following module

6.1 Configuration of the RL agent

Observation space

Following the overall architecture in Figure 4.1, the PF unit takes as input the local path generated from the LPP module and controls the rudder angle under consideration of the environmental disturbances. The PF module has no information about the IW geometry or the target ships and is solely responsible for tracking the given local path.

The observation of the PF agent at time t is denoted as o_t^{PF} and consists of a component describing the status of the own ship, denoted $o_{\mathrm{OS},t}^{\mathrm{PF}}$, and a component regarding the environmental forces, denoted $o_{\mathrm{Env},t}^{\mathrm{PF}}$. Thus, we have $o_t^{\mathrm{PF}} = \left(\left(o_{\mathrm{OS},t}^{\mathrm{PF}}\right)^\top, \left(o_{\mathrm{Env},t}^{\mathrm{PF}}\right)^\top\right)^\top$.

Own ship observation. The component describing the status of the own ship is defined as:

$$o_{\mathrm{OS},t}^{\mathrm{PF}} = \left(\frac{u_{\mathrm{OS},t}}{u_{\mathrm{scale}}}, \frac{v_{\mathrm{OS},t}}{v_{\mathrm{scale}}}, \frac{\tilde{r}_{\mathrm{OS},t}}{\tilde{r}_{\mathrm{scale}}}, \frac{\dot{\tilde{r}}_{\mathrm{OS},t}}{\dot{\tilde{r}}_{\mathrm{scale}}}, \frac{\delta_{\mathrm{OS},t}}{\delta_{\max}}, \frac{y_{e,t}^{\mathrm{local}}}{y_{\mathrm{scale}}}, \frac{[\chi_{e,t}^{\mathrm{local}}]_{-\pi}^{\pi}}{\pi} \right)^{\top}. \tag{4.17}$$

Here, $u_{\mathrm{OS},t}$, $v_{\mathrm{OS},t}$, $\tilde{r}_{\mathrm{OS},t}$, $\dot{\tilde{r}}_{\mathrm{OS},t}$, $\delta_{\mathrm{OS},t}$ are the surge velocity, sway velocity, yaw rate, change in yaw rate, and rudder angle of the own ship, respectively. The scaling constants are: $u_{\mathrm{scale}} = 3\,\mathrm{m/s}$, $v_{\mathrm{scale}} = 0.2\,\mathrm{m/s}$, $\tilde{r}_{\mathrm{scale}} = 0.002\,\mathrm{rad/s}$, $\dot{\tilde{r}}_{\mathrm{scale}} = 8 \cdot 10^{-5}\,\mathrm{rad/s^2}$, and $\delta_{\max} = 20°$. The variables $y_{e,t}^{\mathrm{local}}$ and $\chi_{e,t}^{\mathrm{local}}$ are the cross-track error and the course error from the VFG method for the local path, which uses a gain parameter of $k^{\mathrm{PF}} = 0.01$.

Environmental force observation. Further, we define the environmental observation component as follows:

$$\begin{aligned} o_{\mathrm{Env},t}^{\mathrm{PF}} = \Bigg(& \frac{V_{c,t}}{V_{\mathrm{c,norm}}}, \frac{[\beta_{c,t} - \psi_{\mathrm{OS},t}]_{-\pi}^{\pi}}{\pi}, \frac{V_{wi,t}}{V_{wi,\mathrm{norm}}}, \frac{[\beta_{wi,t} - \psi_{\mathrm{OS},t}]_{-\pi}^{\pi}}{\pi}, \frac{[\beta_{wa,t} - \psi_{\mathrm{OS},t}]_{-\pi}^{\pi}}{\pi}, \\ & \frac{\zeta_{wa,t}}{\zeta_{wa,\mathrm{norm}}}, \frac{T_{wa,t}}{T_{wa,\mathrm{norm}}}, \frac{\lambda_{wa,t}}{\lambda_{wa,\mathrm{norm}}}, \frac{H_t}{H_{\mathrm{norm}}} \Bigg)^{\top}, \end{aligned} \tag{4.18}$$

where $V_{c,t}$, $\beta_{c,t}$, $V_{wi,t}$, $\beta_{wi,t}$ are the current and wind's speed and angle of attack, respectively, at the position of the own ship at time t. Moreover, $\beta_{wa,t}$, $\zeta_{wa,t}$, $T_{wa,t}$, $\lambda_{wa,t}$, are the wave's angle, height, period, and length. Finally, we included the water depth H_t at the own ship's position into the observation vector since it also affects the dynamics via the shallow water corrections outline in Section 4.1. The scaling parameters are set to: $V_{c,\mathrm{norm}} = 0.5\,\mathrm{m/s}$, $V_{wi,\mathrm{norm}} = 15\,\mathrm{m/s}$, $\zeta_{wa,\mathrm{norm}} = 2\,\mathrm{m}$, $T_{wa,\mathrm{norm}} = 7\,\mathrm{s}^{-1}$, and $\lambda_{wa,\mathrm{norm}} = H_{\mathrm{norm}} = 100\,\mathrm{m}$.

Neural network architecture

We use the network architecture outlined in Hart et al. (2023b) for both actor μ and the critics Q_j, $j \in \{1, 2\}$ of the PF unit, which is similar to the original proposal of Meng et al. (2021). The architecture is visualized in Figure 4.8, where the FC layers use 128 neurons and the number of hidden units of the LSTM layers is 128. In particular, the actor μ, parametrized by $\theta_{\mu}^{\mathrm{PF}}$, processes the observations $o_{(t-h):t}^{\mathrm{PF}} = \cup_{l=0}^{h} o_{t-l}^{\mathrm{PF}}$ to yield an action $\mu\left(o_{(t-h):t}^{\mathrm{PF}}; \theta_{\mu}^{\mathrm{PF}}\right)$. The critic Q_j, $j \in \{1, 2\}$, which is parametrized by the set θ_j^{PF}, evaluates a given action a_t^{PF} to produce an action-value estimate $Q_j\left(o_{(t-h):t}^{\mathrm{PF}}, a_t^{\mathrm{PF}}; \theta_j^{\mathrm{PF}}\right)$. Following Meng et al. (2021), the networks process the past observations $o_{t-h}^{\mathrm{PF}}, \ldots, o_{t-1}^{\mathrm{PF}}$ through an LSTM layer, which constitutes a *memory extraction* unit. The current observation o_t^{PF} is separately processed by FC layers, which Meng et al. (2021) refer to as the *current feature extraction* unit.

Action space

The PF agent controls the rudder angle of the own ship. Building on the observation defined in the last subsection, the agent computes an action $a_t^{\mathrm{PF}} \in [-1, 1]$ which adjusts the rudder angle as follows:

$$\delta_{\mathrm{OS},t+1} = \mathrm{clip}\left(\delta_{\mathrm{OS},t} + a_t^{\mathrm{PF}} \cdot a_c^{\mathrm{PF}}, -\delta_{\max}, \delta_{\max}\right), \tag{4.19}$$

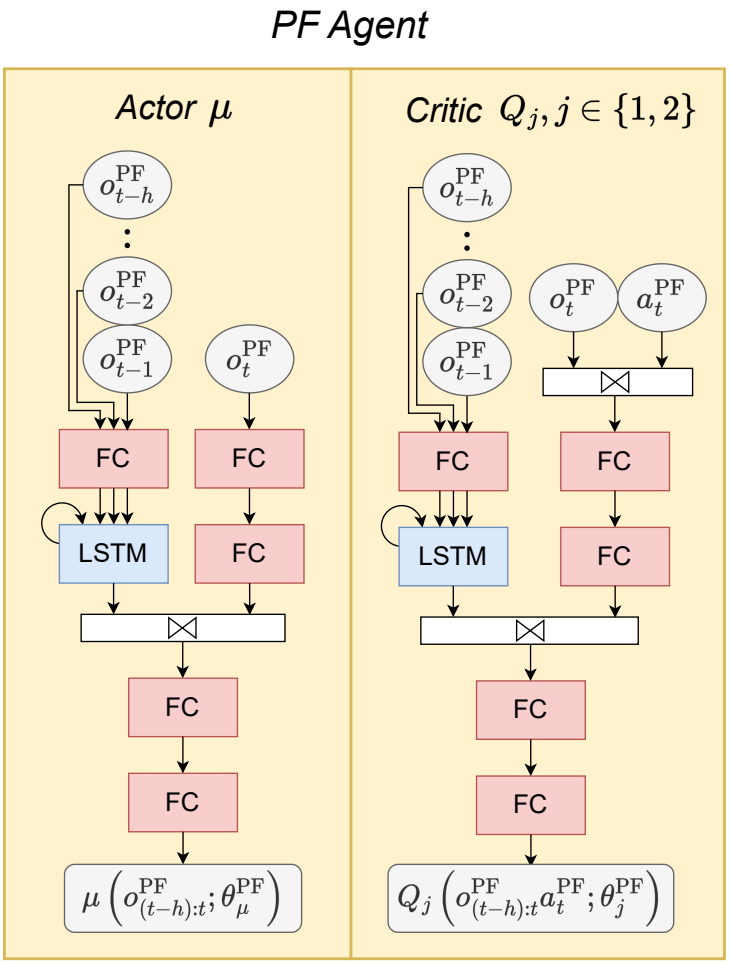


Figure 4.8: Neural network architecture used for the PF agent; adapted from Hart et al. (2023b). We apply a ReLU activation after each FC layer, except for the last one of the actor and the critic, which use a tanh and linear activation, respectively.

where the clipping operation ensures the absolute value of the rudder angle does not exceed $\delta_{\max} = 20°$. Further, we set $a_c^{\text{PF}} = 5°$, which results in combination with our control frequency of $t_{\text{control}} = 5\,\text{s}$ in realistic rudder changes.

Reward function

The reward function for the PF unit builds on Paulig and Okhrin (2024) and consists of three components: a cross-track error reward, $r^{\text{PF}}_{y_e,t}$, a course error component, $r^{\text{PF}}_{\chi_e,t}$, and a comfort reward, $r^{\text{PF}}_{\text{comf},t}$. Similar to the LPP agent's reward function of Section 5.1, we define:

$$r^{\text{PF}}_{y_e,t} = \exp\left(-k^{\text{PF}}_{y_e} |y^{\text{local}}_{e,t}|\right), \quad r^{\text{PF}}_{\text{comf},t} = -\left(a^{\text{PF}}_{t}\right)^2, \tag{4.20}$$

$$r^{\text{PF}}_{\chi_e,t} = \begin{cases} k_{\text{turn}} & \text{if } |\chi^{\text{local}}_{e,t}| \geq \frac{\pi}{2} \\ \exp\left(-k^{\text{PF}}_{\chi_e} |\chi^{\text{local}}_{e,t}|\right) & \text{else.} \end{cases} \tag{4.21}$$

We set the constants $k^{\text{PF}}_{y_e} = 0.05$, $k^{\text{PF}}_{\chi_e} = 5$, and $k_{\text{turn}} = -10$. The condition with the large negative penalty for the course error is included to prevent the agent from completely turning

around and following the path in the wrong direction. The reward components are aggregated to the reward for the PF agent at time t, denoted r_t^{PF}, as follows:

$$r_t^{\mathrm{PF}} = r_{y_e,t}^{\mathrm{PF}} \omega_{y_e}^{\mathrm{PF}} + r_{\chi_e,t}^{\mathrm{PF}} \omega_{\chi_e}^{\mathrm{PF}} + r_{\mathrm{comf,t}}^{\mathrm{PF}} \omega_{\mathrm{comf}}^{\mathrm{PF}}, \tag{4.22}$$

where a small grid search yielded the equal weights: $\omega_{y_e}^{\mathrm{PF}} = \omega_{\chi_e}^{\mathrm{PF}} = \omega_{\mathrm{comf}}^{\mathrm{PF}} = \frac{1}{3}$.

6.2 Training environment

The IW generation for the PF unit is identical to the one described in Section 5.2. However, in this module, we need to sample the current, wind, and wave conditions since it is the PF unit's primary responsibility to account for these. The respective angles of attack in radians are separately sampled from $\mathcal{U}(0, 2\pi)$. Moreover, we sample a current speed from $\mathrm{clip}\left[Exp(0.2), 0, 0.5\right] \cdot 1\,\mathrm{m/s}$ and a wind speed from $\mathcal{U}(0, 15) \cdot 1\,\mathrm{m/s}$. The wave height, length, and period are sampled from $\mathrm{clip}\left[Exp(0.1), 0.01, 2\right] \cdot 1\,\mathrm{m}$, $\mathrm{clip}\left[Exp(20), 1, 100\right] \cdot 1\,\mathrm{m}$, and $\mathrm{clip}\left[Exp(1), 0.5, 7\right] \cdot 1\,\mathrm{s}^{-1}$, respectively. Furthermore, we introduce zero-mean Gaussian noise to each value when queried by the own ship, enhancing the robustness of the policy.

The termination conditions for an episode are as follows: if more than 500 steps have elapsed, if the agent strays more than 400 meters away from the local path, or if the water depth at the own ship's position becomes insufficient, rendering the shallow water approximations infeasible.

7 Results and validation

7.1 Training details

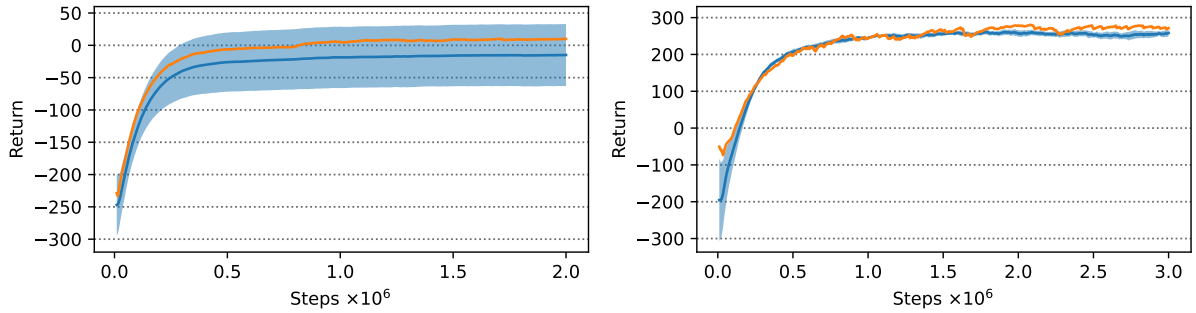
The LPP and PF agents are separately trained for $2 \cdot 10^6$ and $3 \cdot 10^6$ steps, respectively. The remaining hyperparameters are the same for both agents and are outlined in Table 4.1. We thereby primarily followed the configuration of Hart et al. (2023b). However, an exception is the history length, which is set to $h = 2$ for computational reasons, following the setup of Waltz and Okhrin (2023). All experiments were conducted using Python 3.8.6 (Van Rossum and Drake, 2009) and the PyTorch deep learning library (Paszke et al., 2019) version 1.10.0. The computational hardware employed for these experiments consisted of Intel(R) Xeon(R) CPUs E5-2680 v3 (12 cores) running at 2.50 GHz. We make the source code to this study publicly available in a GitHub repository in Waltz and Paulig (2022) to ensure full reproducability. In addition, we note that while the training of the DRL agents is computationally demanding due to the time-consuming optimization of the neural networks, the inference after training is extremely fast. Therefore, the proposed approach is highly practical, as the real-time computation of control commands during operations requires only a few milliseconds.

During the training process, we average every 5000 steps the test return, which is the sum of rewards, of 3 evaluation episodes. To enhance clarity, we apply exponential smoothing to these values. Additionally, we perform the experiment ten times under different random initialization of the neural networks. This enables us to compute confidence intervals that better represent the training performance, which are shown in Figure 4.9. The blue line in each of the plots is the respective mean over ten independent runs, and the light blue area around it represents the

Hyperparameter	Value
Batch size	32
Discount factor	0.99
History length (h)	2
Learning rate (actor)	0.0001
Learning rate (critic)	0.0001
Loss function	Mean squared error
Max. replay buffer size	$5 \cdot 10^5$
Min. replay buffer size	5,000
Optimiser	Adam (Kingma and Ba, 2014)
Policy update delay	2
Soft update rate	0.001
Target policy smoothing noise	0.2
Target policy smoothing noise clip	0.5

Table 4.1: List of hyperparameters. For a detailed description of each parameter we refer to Fujimoto et al. (2018) and Meng et al. (2021).

95% pointwise confidence interval. The orange line is the run whose final policy is validated in the upcoming section. The LPP unit's training plot has a larger variance due to its dependence on five distinct reward components. In contrast, the PF unit's training is less variable since PF is a more straightforward control task, minimizing spatial and angular deviation from the local path.



(a) LPP unit

(b) PF unit

Figure 4.9: The two plots show the test return's development during training for the two agents.

In the upcoming sections, we separately validate the proposed LPP and PF modules on simulation examples. Afterward, we employ the complete architecture described in Section 3 in validation scenarios on real AIS data.

7.2 Validation: Local path planning module

Setup

We developed a comprehensive procedure to thoroughly evaluate the performance of the LPP unit. The procedure consists of six distinct setups, which are among the most challenging and safety-critical scenarios encountered on IWs:

1. overtaking a vessel train,
2. overtaking an overtaker,
3. overtaking under oncoming traffic,
4. overtaking an overtaker under oncoming traffic,
5. getting overtaken,
6. and navigating along static obstacles.

The inclusion of static obstacles along the ship's path is particularly important to assess the LPP agent's generalization capabilities, as the agent has not been exposed to such obstacles during training. We analyze the behavior of the DRL agent separately in each of these six setups on a straight waterway segment, a left curve, and a right curve, respectively. This results in a total of 18 different scenarios being studied.

Moreover, we compare the performance of the DRL planning agent for the straight waterway segment with a state-of-the-art APF method, building on the recent proposals of Liu et al. (2023) and Wang et al. (2019). The detailed functionality of the APF approach is described in Appendix D. We choose three performance metrics to compare the two methods thoroughly. Following Jadhav et al. (2023), the first metric is the controller effort (CE) over a trajectory, which measures the average commanded heading change of the respective planning method. Formally, we define:

$$\mathrm{CE}_{\mathrm{LPP}} = \frac{1}{\Delta_\psi \cdot T} \sum_{t=1}^{T} |\psi_{\mathrm{OS},t} - \psi_{\mathrm{OS},t-1}|, \tag{4.23}$$

where T is the length of the trajectory. Note that the CE is normalized by Δ_ψ, which is the maximum possible heading change between consecutive time steps. As described in Section 5.1, this value is 10° for the DRL agent, while we reduced it to 2° for the APF method, which heavily increased the latter method's performance. Generally, the CE should be as small as possible to generate smooth local paths that allow for fuel efficient operations.

The second performance metric for the LPP task is the mean cross-track error (MCTE), which is defined by Jadhav et al. (2023) as follows:

$$\mathrm{MCTE}_{\mathrm{LPP}} = \frac{1}{L_{pp} \cdot T} \sum_{t=0}^{T} |y_{e,t}^{\mathrm{global}}|, \tag{4.24}$$

which includes a normalization by the length between perpendiculars $L_{pp} = 64\,\mathrm{m}$ of the vessel. A smaller MCTE indicates a planned path close to the global path, reflecting strong tracking capabilities.

Finally, incorporating safety considerations into our evaluation, we introduce a third performance metric named MinDist. This metric is the minimum distance encountered with respect to any target ship along the trajectory. Formally, we have:

$$\mathrm{MinDist} = \frac{1}{L_{pp}} \cdot \min_{t \in \{0,\dots,T\}} \min_{i \in \{0,\dots,N_t\}} \left[d_{\mathrm{OS},t}^{i} - D\left(\alpha_{\mathrm{OS},t}^{i}\right) \right], \tag{4.25}$$

where we consider the ship domain of the own ship similar to the target ship observation in (4.6). Generally, a sufficiently large MinDist indicates a safe operation.

Analysis

The trajectories for the straight case of the DRL agent and the APF method are visualized in Figures 4.10 and 4.11, respectively. The black trajectories in these figures correspond to the respective planning agent, while the colorized ones are the target ships controlled by Algorithm 2. The purple and grey dotted lines are the global and reversed global paths. Further, Table 4.2 contains the corresponding performance metrics. Additionally, Figure 4.12 provides plots of the cross-track and course errors to the global path, the selected action by the DRL agent, and the distance to the target ships. The results for the curved scenarios alongside the initial speed configurations of each vessel can be found in Appendix E.

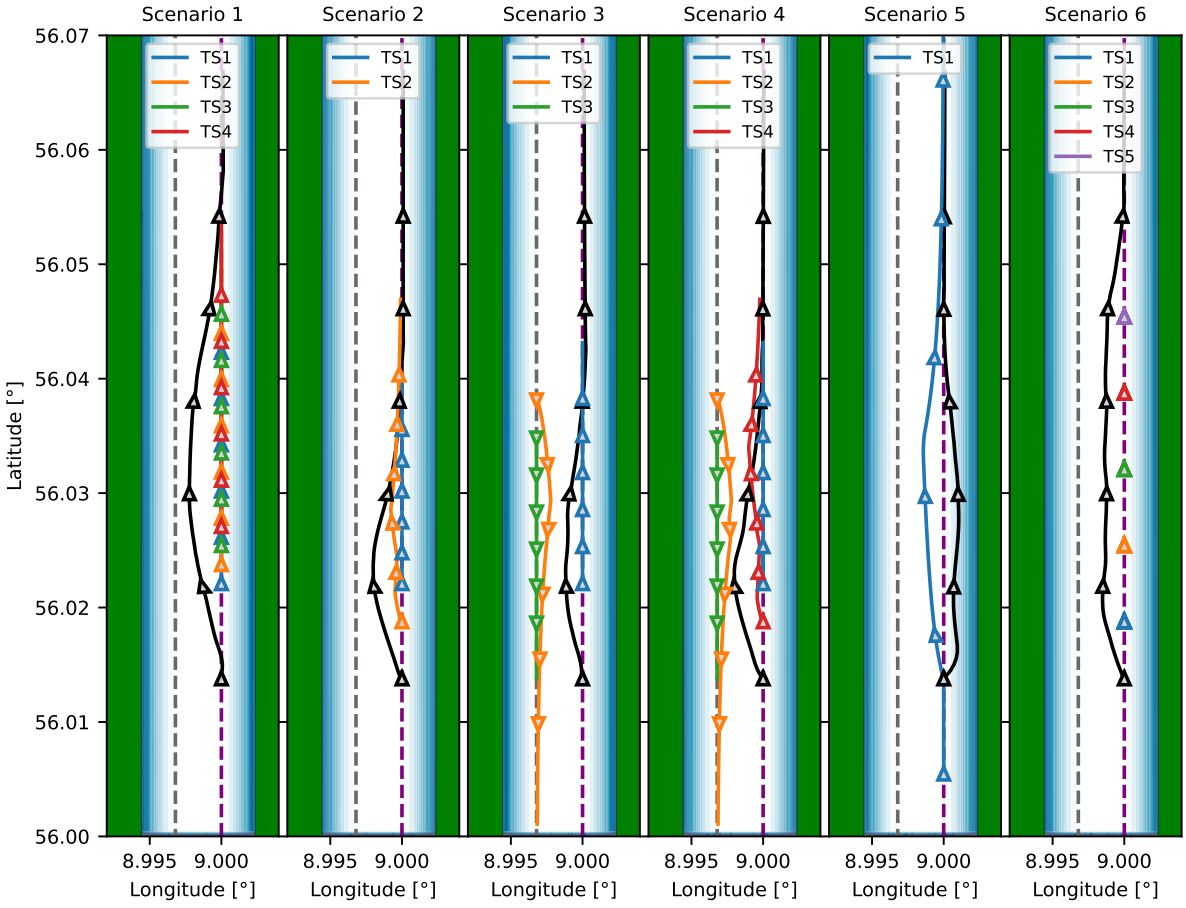


Figure 4.10: Trajectories of the LPP validation scenarios of the DRL agent on a straight river segment. Note that the latitude and longitude values are artificial and serve as orientation.

Focusing on Scenario 1 in Figures 4.10 and 4.12, we can observe the successful overtaking maneuver performed by the LPP agent on a vessel train consisting of four target ships. Initially, the agent steers to the port side to avoid a collision with the nearest target ship and then overtakes each vessel one by one until the entire vessel train has been passed. Importantly, this behavior complies with §23(1) of the regulations specified in Bundesministerium für Digitales

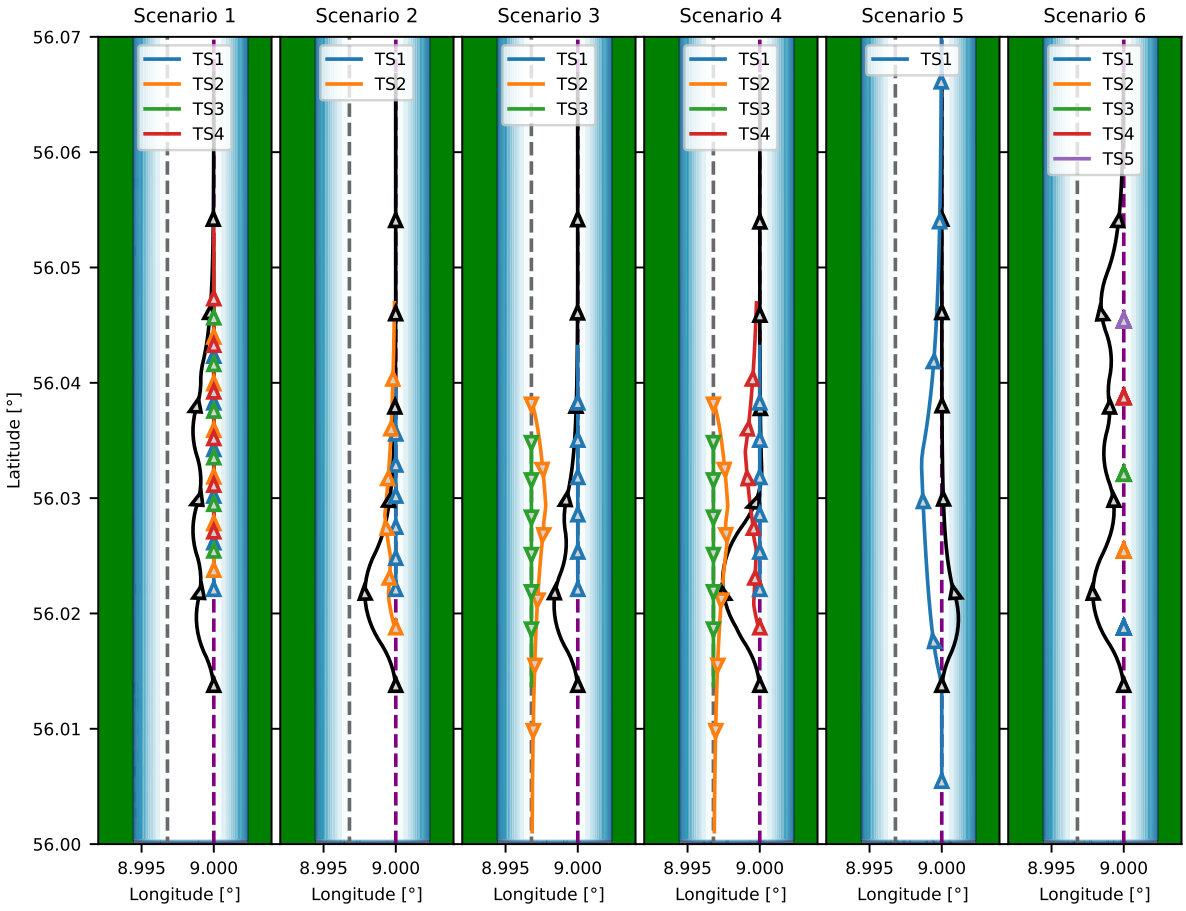


Figure 4.11: Trajectories of the LPP validation scenarios of the APF method on a straight river segment. Note that the latitude and longitude values are artificial and serve as orientation.

und Verkehr (1998), as the agent performs overtaking maneuvers on the target vessel's port side. Once the overtaking maneuver is completed safely, the agent returns to the global path until the cross-track and course errors are close to zero again. Moreover, the action selection during the maneuver is relatively moderate since the agent avoids large consecutive heading changes.

Similar successful results can be observed in the overtaking cases of Scenarios 2 to 4, demonstrating the LPP agent's proficiency in executing advanced maneuvers while maintaining appropriate safe distances from the target ship. Furthermore, the agent's action selection demonstrates a balanced approach in these scenarios, indicating the successful integration of the comfort reward component. In Scenario 5, we can observe the agent's compliance with §23(2), as it enables overtaking of the target ship by slightly moving towards starboard, which directly reflects the traffic rule reward component defined in (4.15). Of particular note is Scenario 6, where the agent effectively navigates around static obstacles while maintaining a minimum distance of approximately 50 meters.

Comparing the DRL agent to the APF method, we observe that the APF trajectories are more unstable and clearly display the undesired change between stronger and weaker repulsive forces when passing target ships; see, for instance, Scenario 6 of Figure 4.11. Remarkably, as shown in Table 4.2, this unstable behavior leads to a slightly smaller average MCTE of the APF method in comparison to the DRL agent. However, this circumstance comes at the expense of

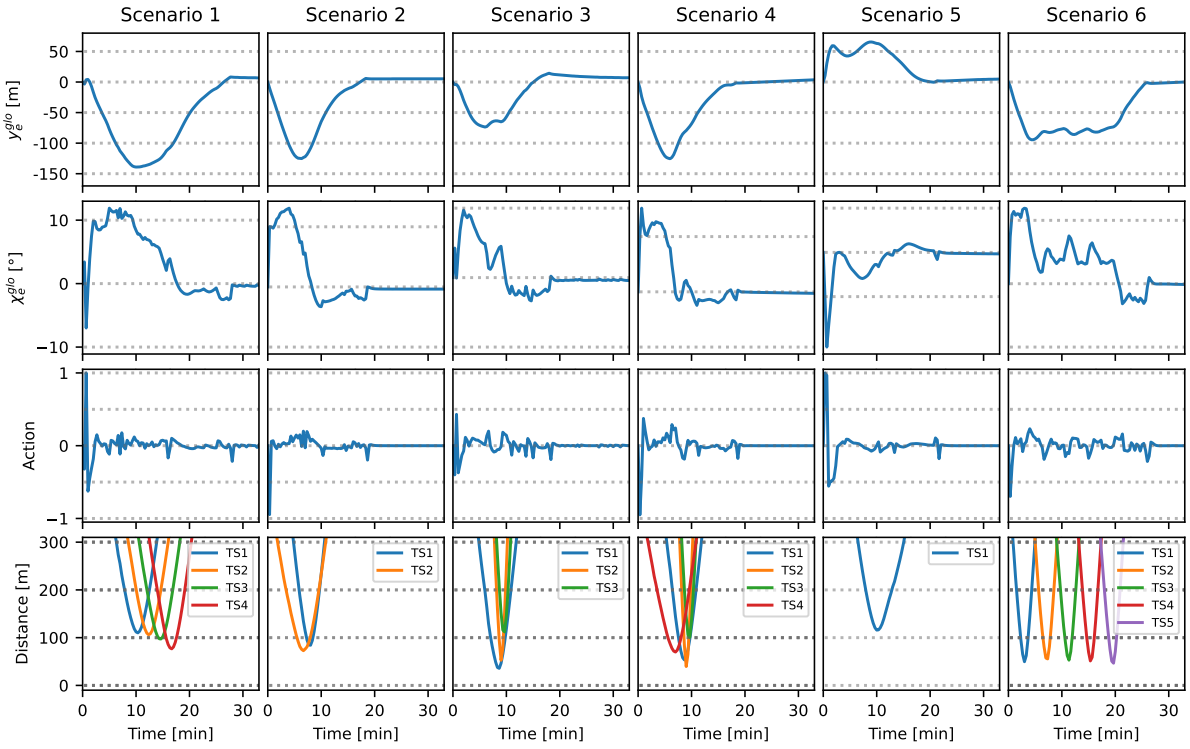


Figure 4.12: Global cross-track and course error, selected actions, and distances to the target ships during validation of the LPP agent on a straight waterway segment.

the increased criticality of the APF trajectories. Generally, although the APF approach does not produce collisions, the MinDist of the DRL controller is on average 65% larger. Further following Table 4.2, the CE is massively reduced by the DRL approach, which is in line with the visually unstable behavior of the APF method.

Scenario	CE_{LPP}		$MCTE_{LPP}$		MinDist	
	DRL	APF	DRL	APF	DRL	APF
1	0.067	0.565	0.954	0.576	1.197	0.434
2	0.040	0.283	0.523	0.388	1.136	0.639
3	0.052	0.343	0.381	0.349	0.558	0.423
4	0.054	0.374	0.502	0.456	0.618	0.655
5	0.064	0.195	0.390	0.172	1.810	0.928
6	0.064	0.710	0.801	0.855	0.727	0.589
Average	0.057	0.412	0.592	0.466	1.008	0.611

Table 4.2: Performance comparison of the DRL agent with the APF method for the LPP task. All metrics are dimensionless.

Lastly, we emphasize that the hyperparameters of the APF method have been carefully tuned to increase its overall performance during the six scenarios on the straight waterway segment. However, deploying this APF configuration without further tuning on the curved waterway segments is not feasible, so we limit comparing the DRL and APF methods to the straight waterway case. On the contrary, our DRL agent can seemingly handle different waterway

curvatures without further adaptations, as shown in Appendix E.

7.3 Validation: Path following module

Setup

We evaluate the performance of our PF agent by subjecting it to various environmental conditions, including the three major forces: currents, winds, and waves. Each force is tested separately in both moderate and extreme scenarios, resulting in a total of six validation scenarios. In each scenario, the agent is tasked with following a straight path initially unaffected by environmental forces. Subsequently, a force field perpendicular to the path is introduced, followed by another segment with zero forces. Finally, the force direction is reversed in the last segment. This testing procedure enables us to assess the agent's adaptability to different environmental conditions. Moderate scenarios are defined as those with a current speed of $V_c = 0.25\,\mathrm{m/s}$, a wind speed of $V_{wi} = 5\,\mathrm{m/s}$, and wave heights of $\zeta_{wa} = 0.5\,\mathrm{m}$. Extreme scenarios are these with $V_c = 1\,\mathrm{m/s}$, $V_{wi} = 20\,\mathrm{m/s}$, and $\zeta_{wa} = 1.5\,\mathrm{m}$, reflecting extremely challenging navigation conditions.

For comparison purposes, we follow Paramesh and Rajendran (2021) and Paulig and Okhrin (2024) and include a PID controller for the rudder angle, which was optimized using the PSO approach outlined in Eberhart and Shi (2000); see Appendix F for details. It should be emphasized that the PID controller is specifically optimized for the validation scenarios, while the RL agent's training encompasses a broader range of scenarios.

Furthermore, we follow Jadhav et al. (2023) and consider the CE and the MCTE to quantitatively compare the performance of the two methods. However, contrary to Section 7.2, the CE for PF measures the average absolute rudder angle, while the MCTE in this task is computed with respect to the local instead of the global path. Formally, we set:

$$\mathrm{CE}_{\mathrm{PF}} = \frac{1}{\delta_{\max} \cdot T} \sum_{t=0}^{T} |\delta_{\mathrm{OS},t}|, \qquad \mathrm{MCTE}_{\mathrm{PF}} = \frac{1}{B \cdot T} \sum_{t=0}^{T} |y_{e,t}^{\mathrm{local}}|, \tag{4.26}$$

where $B = 11.6\,\mathrm{m}$ is the width of the downscaled KVLCC2 tanker.

Analysis

The results are presented in Figures 4.13 and 4.14, where each column refers to a testing scenario for a separate environmental force, while the other two forces are set to zero. From top to bottom, the rows depict the local cross-track error, the local course error, the surge and sway velocities, the yaw rate, and the rudder angle, respectively. Afterward, the rows include the specific attributes for each force; for example, the speed and angle of attack of the currents. In addition, Table 4.3 displays the CE and MCTE performance metrics.

Generally, the DRL agent demonstrates remarkable performance across all six scenarios, effectively adapting to force fields by promptly adjusting the rudder angle to steer back to the desired path. As a result, it successfully maintains minimal cross-track and course error. Notably, the inclusion of a comfort reward component has proven to be beneficial, as the agent generally exhibits smooth and moderate changes in the rudder angle. In contrast, the PID

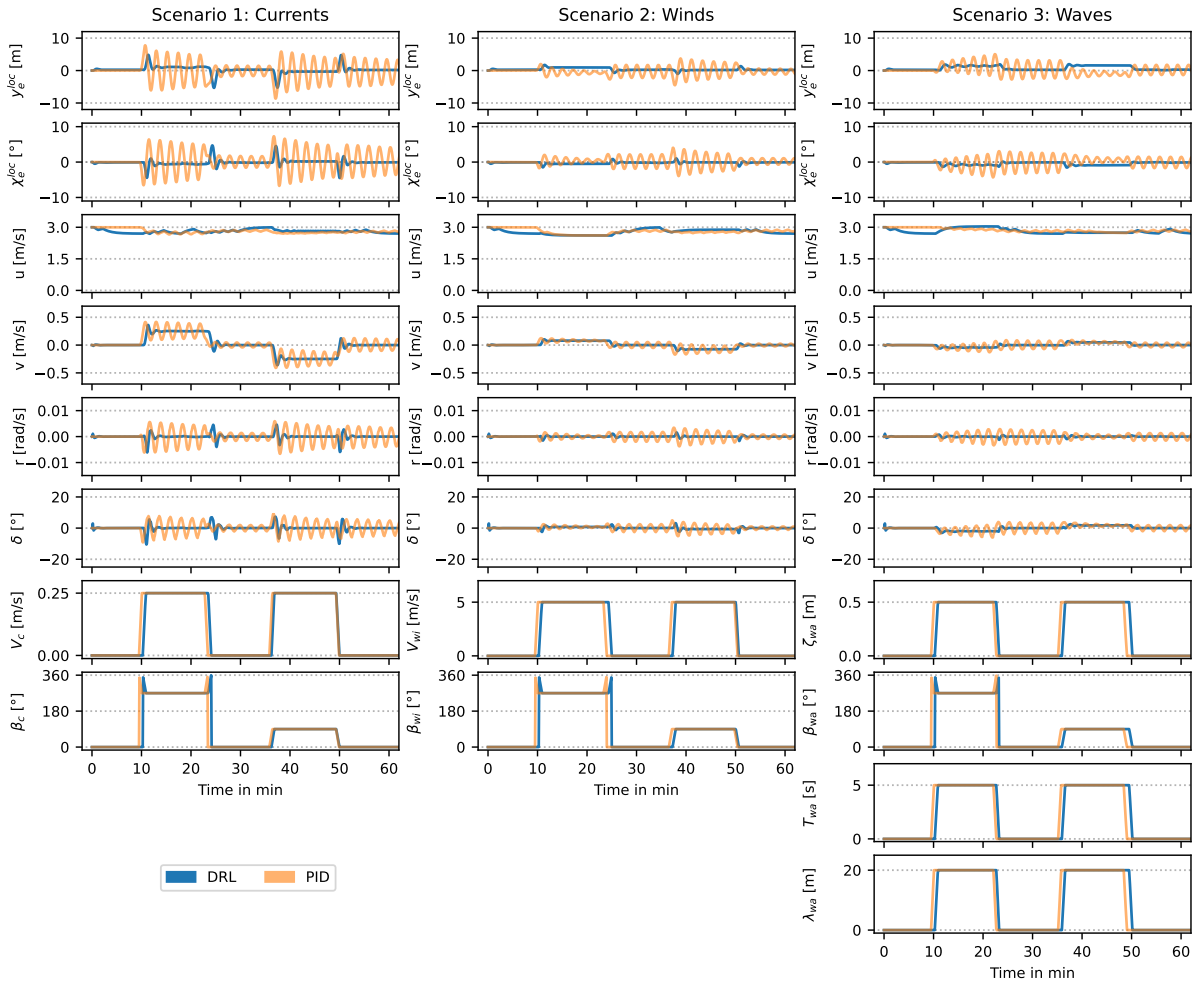


Figure 4.13: Validation results for PF under moderate environmental conditions.

Scenario	CE_{PF}		$MCTE_{PF}$	
	DRL	PID	DRL	PID
1	0.032	0.137	0.056	0.202
2	0.021	0.058	0.036	0.087
3	0.041	0.078	0.067	0.105
4	0.114	0.386	0.498	0.615
5	0.165	0.311	0.460	0.488
6	0.298	0.419	0.165	0.613
Average	0.112	0.232	0.214	0.352

Table 4.3: Performance comparison of the DRL agent with the PID controller for the PF task. All metrics are dimensionless.

controller exhibits the typical undesired oscillation behavior, which persists even after applying the PSO technique. The challenging aspect lies in the heterogeneity of the scenarios, where the transition dynamics of the environment vary significantly under different force fields. This variability likely poses a difficulty for the PID controller in achieving stable and consistent performance. Following Table 4.3, the DRL agent strongly outperforms the PID controller both

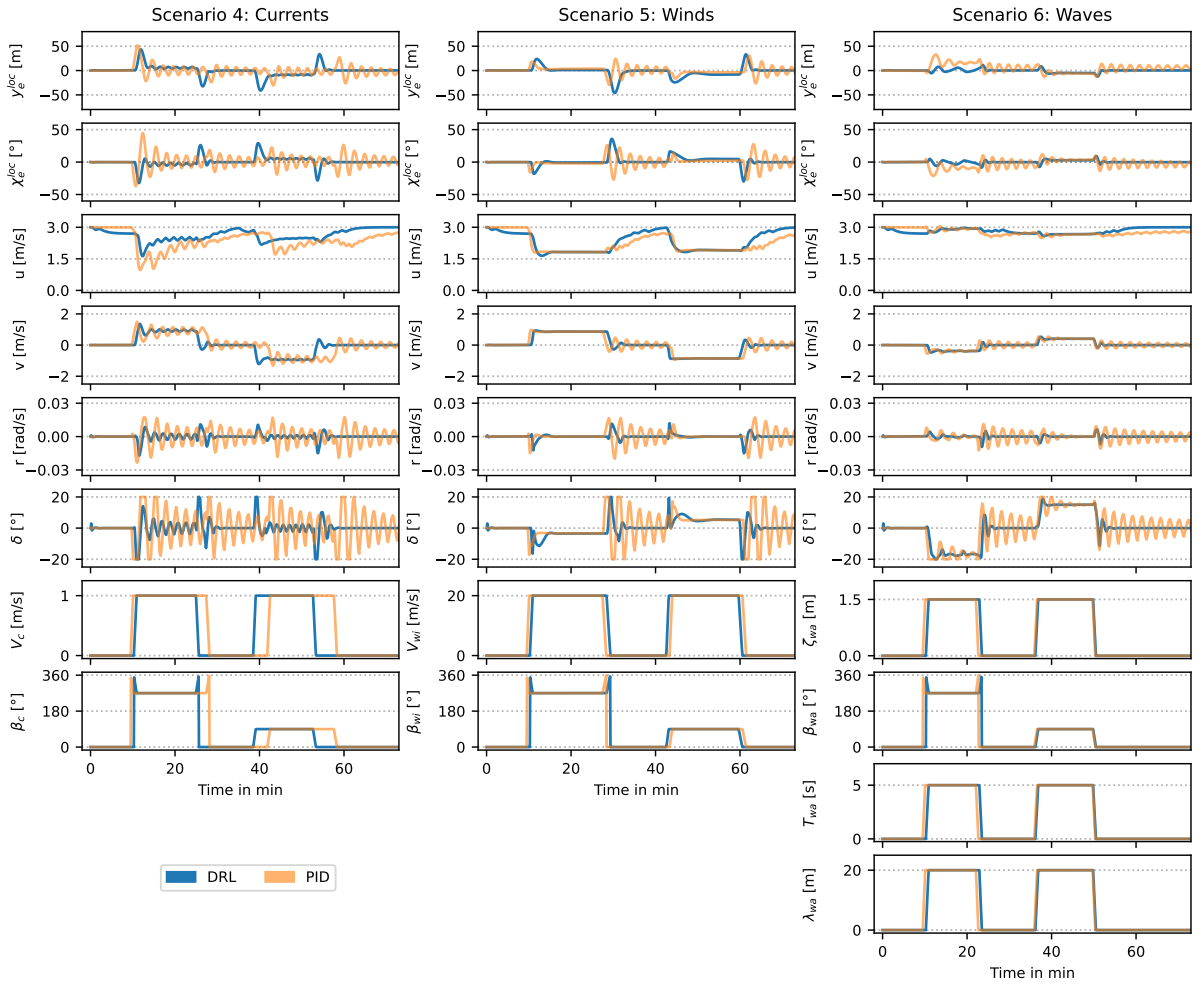


Figure 4.14: Validation results for PF under extreme environmental conditions.

in terms of CE and MCTE. In particular, the CE of the DRL is, on average, approximately 48% of the CE of the APF method, while the MCTE of the DRL method is simultaneously only circa 61% of the MCTE of the APF approach.

7.4 Validation: Complete architecture

After validating the LPP and PF units in separation, in this subsection, we aim to conduct a comprehensive evaluation of the complete architecture introduced in Section 3. We deploy it in simulation on the lower part of the river Elbe in northern Germany. To ensure a realistic simulation, we specifically choose the date of January 29, 2022, and utilize the actual AIS vessel trajectories and environmental disturbances observed on that day. It is worth noting that the presence of the storm named *Malik* in Middle Europe on that date adds an extra challenge to ASV. We manually specify a global path from Lighthouse Tinsdal to the Elbe estuary close to Cuxhaven, which is illustrated in Figure 4.15. Further information regarding the data sources and the trajectory interpolation of the target ships are deferred to Appendix G.

To test the architecture's generalization capabilities, we perform the same test for three different base speeds (in m/s) of the own ship: $U \in \{3, 4, 5\}$, even though both agents have only been trained on $U = 3\,\mathrm{m/s}$. Figure 4.16 presents the angular and spatial deviations of the local

and global paths, respectively, while selected COLAV maneuvers are displayed in Figure 4.17.

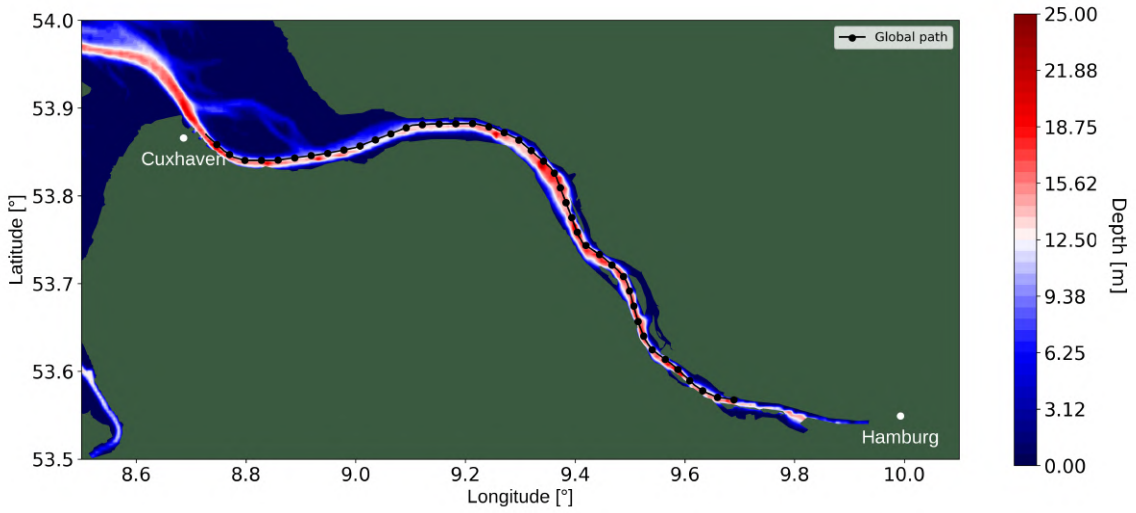


Figure 4.15: Global path for validation of the complete architecture based on AIS data.

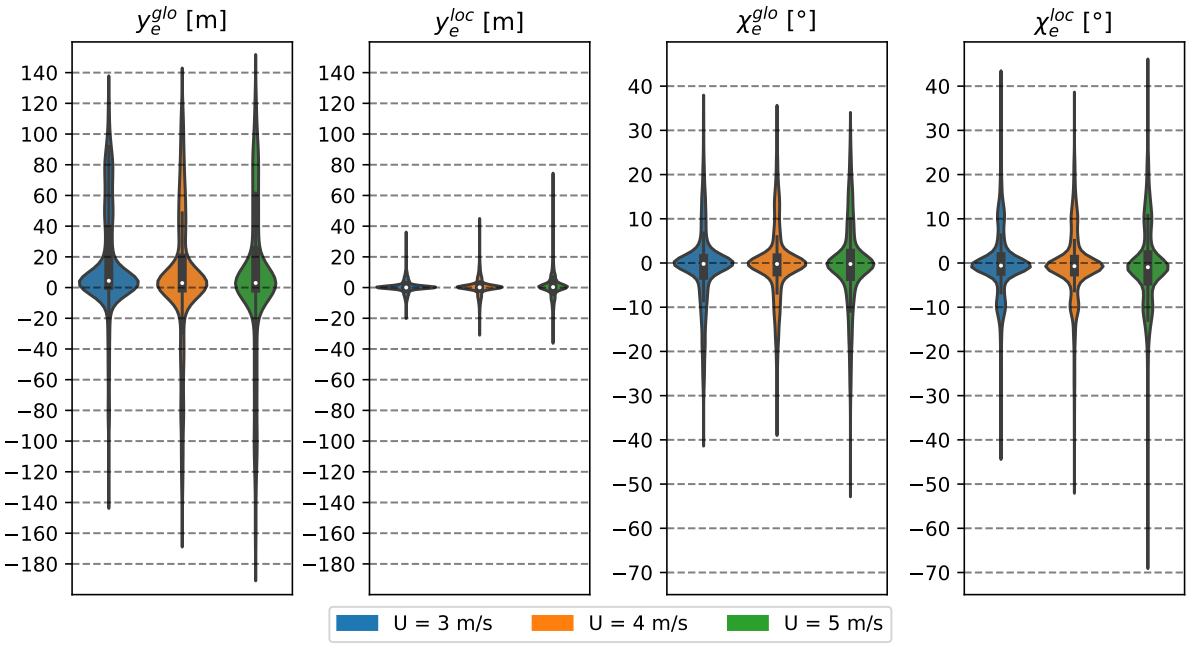
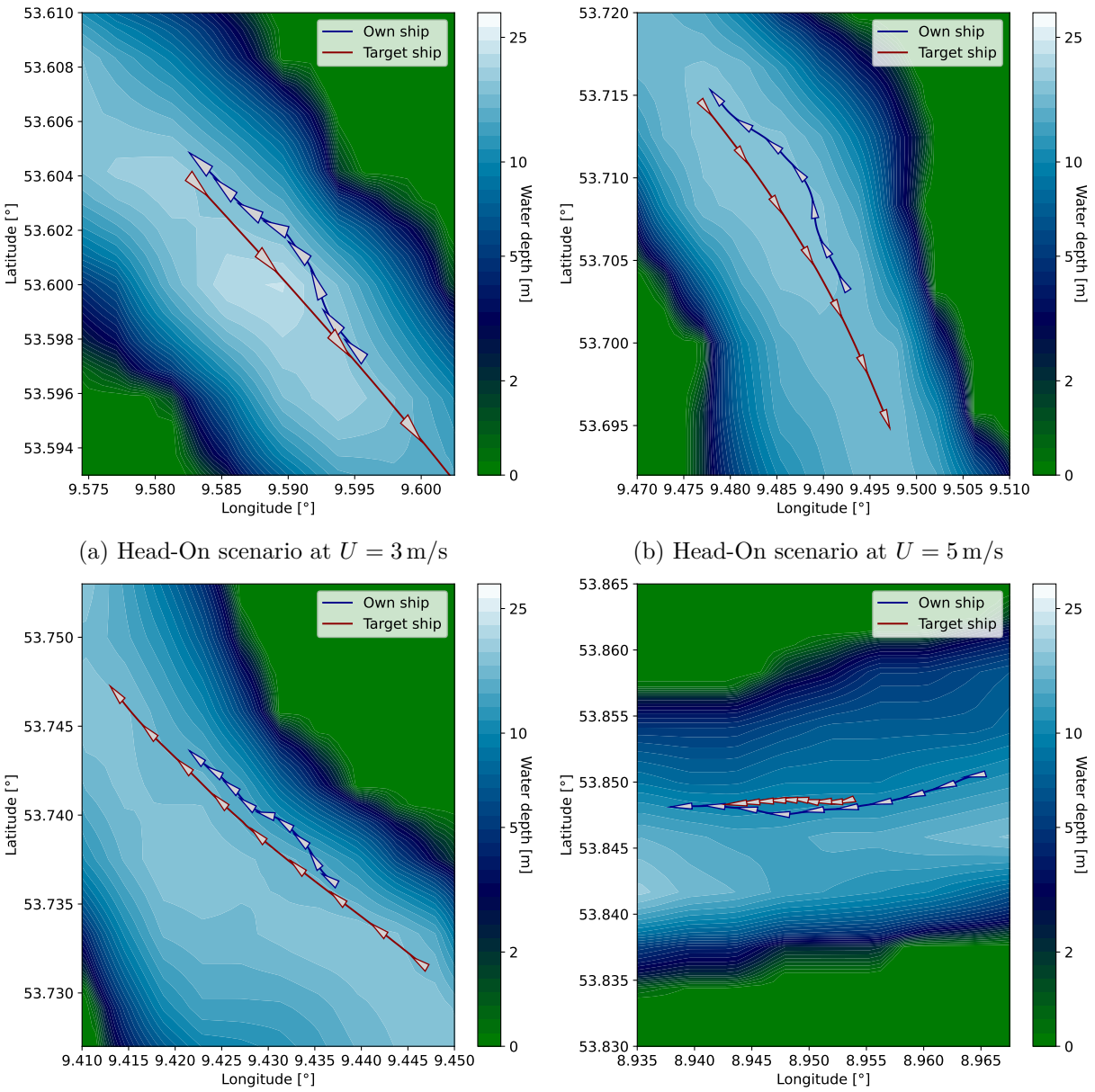


Figure 4.16: Empirical distributions of the global and local cross-track and course errors over the complete journey from Lighthouse Tinsday to the Elbe estuary close to Cuxhaven.

Upon analyzing Figure 4.16, we observe that the maximum spatial deviation from the global path reaches approximately 200 meters, while the vast majority of deviations are smaller than 40 meters in absolute value. These deviations are relatively low, considering the fairway width of the Elbe is typically between 600 and 2000 meters in the selected segment. The local cross-track error, which measures the deviation from the planned local path, is naturally of smaller magnitude and exceeds 20 meters only in a few selected cases. Additionally, we notice a slight increase in the spatial deviation from both the local and global paths as the speed increases. However, this increase remains moderate and demonstrates the successful generalization ability of both

agents. Moreover, the angular deviations show resilience to changes in speed and, importantly, do not differ significantly between the global and local levels. This is likely attributed to the replanning of the local path, which presents the PF agent with a new path to react to.



(a) Head-On scenario at $U = 3\,\mathrm{m/s}$

(b) Head-On scenario at $U = 5\,\mathrm{m/s}$

(c) Enabling overtaking at $U = 3\,\mathrm{m/s}$

(d) Overtaking at $U = 4\,\mathrm{m/s}$

Figure 4.17: Encounter scenarios based on real depth and AIS data using the complete architecture for ASV control.

Analyzing the maneuvers depicted in Figure 4.17, it becomes evident that the framework possesses the capability to execute COLAV actions effectively. The model successfully performs starboard turns in head-on scenarios (Panels (a) and (b)), creates space for overtaking vessels (Panel (c)), and overtakes slower vessels on their portside (Panel (d)). These observations align with the validation results presented in Section 7.2, further highlighting the architecture's suitability for real-world scenarios. Notably, the maneuvers prove successful for all three selected speeds.

While the architecture demonstrates strong performance in various validation scenarios, it is essential to note that collisions have occurred in rare cases during the AIS testing routine. However, these collisions arise due to the nature of the testing format, where the ASV is deployed into scenarios with static target ship trajectories. Consequently, the target ships remain unaware of the presence of the own ship, as illustrated in Figure 4.17. Consider, for instance, Panel (c) of Figure 4.17, where the own ship makes room for a faster target ship. If the target ship suddenly executes a starboard turn while being alongside the own ship, a collision will occur regardless of the own ship's reaction. As stated by Lyu and Yin (2019), it is impossible to avoid a collision if the target ship intentionally moves towards the own ship.

7.5 Discussion and practical challenges

The validation scenarios outlined in Sections 7.2, 7.3, and 7.4 have thoroughly assessed the effectiveness of the LPP and PF agents, both individually and in tandem, establishing a comprehensive evaluation of our autonomous system. Notably, the validation using AIS data in Section 7.4 delves into the behavior of our two-level system within the context of human vessel trajectories, demonstrating successful handling of various real-world encounter situations. However, we recognize the practical challenges associated with the interaction between autonomous systems and traditional, human-controlled vessels in practical operations (Kim et al., 2022). For instance, an autonomous vessel needs to accurately estimate the intention of the conventional ship, while the conventional similarly needs to predict the future trajectory of the ASV (Porathe and Rødseth, 2019).

Recently, Rødseth et al. (2023) presented several options for enhancing safety in encounters between conventional ships and ASVs, including the consideration of broadcasting intentions as a form of communication. We emphasize that such a broadcasting mechanism can be easily integrated into our architecture since the acquired information could be used during the planning iterations of the LPP agent. Generally, as emphasized by Akdağ et al. (2022), collaboration through communication between vessels is a critical component of practical operations.

Finally, real-time applications can present unseen and unexpected uncertainties, such as abrupt alterations in weather patterns, unanticipated vessel maneuvers due to emergencies or navigational errors, or technical malfunctions in sensor equipment. Although our agents are trained in diverse scenarios and DRL offers strong generalization abilities, the sim-to-real transfer of autonomous systems should not be underestimated (Zhao et al., 2020a).

8 Conclusion

The use of ASVs for IW transportation shows promise in creating a sustainable and economically attractive transportation system. Our study introduces a two-level architecture for ASVs operating on IWs based on DRL, employing separate agents for LPP and PF. We consider relevant environmental disturbances, adhere to traffic rules, and validate our approach using simulated and real AIS trajectories.

We acknowledge the limitations of our work, which highlight avenues for future research. Firstly, we do not address limited visibility or sensor faults, which are critical high-risk sce-

narios in actual operations. Secondly, as emphasized in Section 7.5, seafarers usually exchange communication signals when operating on IWs. Such information can be incorporated into our LPP unit, thereby removing the simplifying assumption of linear target ship movement during the planning iterations. Lastly, our focus on simulation experiments calls for validating the proposed architecture in real-world waterways. Addressing these limitations will enable further advancements in developing ASVs for IW transportation, enhancing safety and effectiveness while paving the way for a sustainable transport system.

Acknowledgements

The authors thank the members of the RL Dresden Group, especially Martin Treiber and Fabian Hart, for their constructive feedback on this work. Furthermore, the authors thank Thor Fossen for his insightful answers regarding the consideration of environmental forces in the simulation. The authors also acknowledge the Center for Information Services and High Performance Computing at TU Dresden for providing the resources for high-throughput calculations. Further, the authors thank the European Maritime Safety Agency for providing the AIS data for the validation scenarios. Finally, the authors express their gratitude to the Weiße Flotte Dresden for sharing their extensive experience in vessel operations. Their detailed explanations of the functionalities of the inland vessel 'Gräfin Kosel' have greatly enriched this research. Niklas Paulig was funded by BAW - Bundesanstalt für Wasserbau (Mikrosimulation des Schiffsverkehrs auf dem Niederrhein), Germany.

A Appendix: German vessel traffic rules

The following paragraph is an excerpt from the German regulation for shipping lanes (Bundesministerium für Digitales und Verkehr, 1998), which we consider in our research. We first state the original legal text in German, followed by the official English translation.

§23 Überholen

(1) Grundsätzlich muß links überholt werden. Soweit die Umstände des Falles es erfordern, darf rechts überholt werden.

(2) Das überholende Fahrzeug muß unter Beachtung von Regel 9 Buchstabe e und Regel 13 der Kollisionsverhütungsregeln die Fahrt so weit herabsetzen oder einen solchen seitlichen Abstand vom vorausfahrenden Fahrzeug einhalten, daß kein gefährlicher Sog entstehen kann und während des ganzen Überholmanövers jede Gefährdung des Gegenverkehrs ausgeschlossen ist. Das vorausfahrende Fahrzeug muß das Überholen soweit wie möglich erleichtern.

§23 Overtaking

(1) As a rule, an overtaking vessel shall pass the vessel being overtaken on the latter vessel's port side. If the circumstances of the case so require, the overtaking vessel may pass the vessel being overtaken on the latter vessel's starboard side.

(2) The overtaking vessel, acting in compliance with the provisions of Rule 9(e) and Rule 13 of the International Regulations for Preventing Collisions at Sea, 1972, as amended, shall slacken her speed so much, respectively, shall give the vessel being overtaken such a wide berth that no dangerous suction or wash can develop and that no vessel proceeding in the opposite direction will be put at any risk for the entire duration of the overtaking process. The vessel being overtaken shall facilitate the overtaking vessel's action to the greatest possible extent.

B Appendix: Nomenclature

AIS	Automatic identification system	δ	Rudder angle
APF	Artificial potential field	δ_{max}	Maximum rudder angle
ASV	Autonomous surface vehicle	Δ_ψ	Maximum heading change between time steps
CE	Controller effort	$\dot{\hat{r}}_{scale}$	Normalization constant
COLAV	Collision avoidance	$\mathcal{DU}$	Discrete uniform distribution
COLREGs	The International Regulations for	e_{norm}	Normalization constant
	Preventing Collisions at Sea	Exp	Exponential distribution
CPA	Closest point of approach	$F_{att,t}$	APF attractive force
DCPA	Distance to CPA	$F_{e,t}$	East component of APF total forces
DQN	Deep Q-Network	$F_{n,t}$	North component of APF total forces
DRL	Deep reinforcement learning	$F_{rep,t}$	APF repulsive force
FC	Fully connected	F_t	APF resulting total force
GPP	Global path planning	$f_{j,l}(\cdot, \theta_{f_{j,l}})$	Spatial recurrent network part for the
IW	Inland waterway		critic with parameter set $\theta_{f_{j,l}}$
LPP	Local path planning	$f_{\mu,l}(\cdot, \theta_{f_{\mu,l}})$	Spatial recurrent network part for the
LSTM	Long short-term memory		actor with parameter set $\theta_{f_{\mu,l}}$
MCTE	Mean cross-track error	γ	Discount factor
MDP	Markov decision process	γ_i	i-th lidar beam
PF	Path following	$g_j(\cdot, \theta_{g_j})$	Temporal recurrent network part for the
PID	Proportional-integral-derivative		critic with parameter set theta g_j
PSO	Particle swarm optimization	$g_\mu(\cdot, \theta_{g_\mu})$	Temporal recurrent network part for the
RL	Reinforcement learning		actor with parameter set theta g_{mu}
RRT	Rapidly exploring random tree	η	$= (x_n, y_n, \psi)^\top$
TCPA	Time to CPA	h	History length
VFG	Vector field guidance	H	Water depth
		H_{norm}	Normalization constant
a	Action	I_{zG}	Moment of inertia
$\mathcal{A}$	Action space	J_z	Added moment of inertia
a_{lat}	Length of the minor axis of APF	k_{coll}	Collision reward weight
	ellipse	k_{rule}	Rule adherence reward weight
a_{lon}	Length of the major axis of APF	k_{turn}	Maximum course error penalty
	ellipse	$k^{\text{LPP}}, k^{\text{PF}}$	VFG gain parameters
$a_c^{\text{LPP}}, a_c^{\text{PF}}$	Action multiplier	$k_{\chi_e}^{\text{LPP}}$	Power weight for
a_t^{LPP}	LPP action		course error reward for LPP
a_t^{PF}	PF action	$k_{y_e}^{\text{LPP}}$	Power weights for
$\alpha_{\text{OS},t}^i$	Relative bearing of ship i from the		cross-track error reward for LPP
	perspective of the own ship	$k_{\chi_e}^{\text{PF}}$	Power weight for
$\alpha_{i,t}^{\text{OS}}$	Relative bearing of the own ship from		course error reward for PF
	the perspective of ship i	$k_{y_e}^{\text{PF}}$	Power weight for
$\alpha_{\text{OS},t}^{i,\text{cpa}}$	Relative bearing of ship i at the CPA		cross-track error reward for PF
β_c	Current attack angle	k_a, k_r	APF force weights
β_{wa}	Wave attack angle	K_p, K_i, K_d	PID gains
β_{wi}	Wind attack angle	λ_{wa}	Wave length
B	Ship breadth	$\lambda_{wa,\text{norm}}$	Normalization constant
χ	Course angle	L_{pp}	Ship length between perpendiculars
χ_d	Desired course	m	Mass of ASV
χ_e	Course error	m_{x_b}	Added mass in x_b direction
χ^∞	VFG parameter	m_{y_b}	Added mass in y_b direction
χ_{P_k}	Path segment angle	μ	Actor network
d_{norm}	Scaling constant	$\mu(\cdot, \theta_\mu^{\text{LPP}})$	Actor network with combined
d^*, d_0, d_l	Distance constants		parameter set θ_μ^{LPP}
$d_{i,t}^{\text{cpa}}$	DCPA to target ship i	$\{n\}, \{b\}$	Earth and body frame
$d_{\text{OS},t}^i$	Distance from agent to ship i	N_H	Yaw moment force on ship hull
$d_{\text{OS},t}^{\text{G}}$	Distance from own ship to goal point	N_R	Yaw moment force on ship rudder
d_{scale}	Maximum considered distance from own to	N_m	Yaw moment around midship
	target ships	N_{WA}	Wave yaw forces
$d_{i,t}^{\text{cpa}}$	DCPA to the i-th target ship	N_{WI}	Wind yaw forces

ν	$= (u, v, \tilde{r})^\top$	t_q	Query time for AIS data extraction
n_{norm}	Longitudinal ship domain	t_{replan}	LPP replan interval
	reward normalization constant	T_{wa}	Wave period
$\vec{n}_{\text{OG}}$	APF normal vector from	$T_{wa,\text{norm}}$	Normalization constant
	agent in direction of goal (OG)	θ_μ^{LPP}	Actor network parameter set
$\vec{n}_{\text{OT}}$	APF normal vector from	θ_j^{LPP}	Critic network parameter set
	agent in direction of target ship (OT)	$\theta_{f_{j,l}}$	Parameter set for spatial
$\vec{n}_{\text{OT}}$	APF normal vector from		recurrent network part for the critic
	target ship in direction of agent	$\theta_{f_{\mu,l}}$	Parameter set for spatial
$\vec{n}_{\text{OT}\perp}$	APF unit vector perpendicular to portside unit		recurrent network part for the actor
	vector from the own ship to the target ship	θ_{g_j}	Parameter set for temporal
o	Observation		recurrent network part for the critic
$\mathcal{O}$	Observation space	θ_{g_μ}	Parameter set for temporal
			recurrent network part for the actor
$o_{\text{IW},t}^{LPP}$	Navigational area information	u	Surge
$o_{\text{OS},t}^{LPP}$	Own ship information	$\mathcal{U}$	Uniform distribution
$o_{\text{TS},t}^{LPP}$	Target ship information	u_c	Longitudinal component of current velocity
o_t^{LPP}	Agent observation	u_{scale}	Normalization constant
$o_{\text{Env},t}^{PF}$	Environmental observation	U	Vessel speed ($\sqrt{u^2+v^2}$)
$o_{\text{OS},t}^{\text{PF}}$	Own ship observation	U_{base}	Validation base speed of the agent
o_t^{PF}	PF observation	$U_{i,t}$	Speed of target ship i
$o_{\text{TS},i,t}$	Observation for target ship i	$U_{\text{OS},t}$	Own ship speed
$\mathcal{P}$	State transition probability function	U_{scale}	Speed scaling constant
P_k	k-th waypoint of a path	v	Sway
ψ	Heading angle	v_c	Lateral component of current velocity
$\psi_{d,t}$	APF desired heading	v_{scale}	Normalization constant
$\psi_{\text{OS},t}$	Own ship heading	V_c	Current velocity
π	Policy function	$V_{c,\text{norm}}$	Normalization constant
$Q_j(\cdot,\cdot,\theta_j^{\text{LPP}})$	Critic network with combined	V_{wi}	Wind speed
	parameter set θ_j^{LPP}	$V_{wi,\text{norm}}$	Normalization constant
Q^π, Q_1, Q_2	Action-value function	$\omega_{\chi_e}^{\text{LPP}}, \omega_{coll}^{\text{LPP}}, \omega_{\text{comf}}^{\text{LPP}}$	Reward component weight
$\mathcal{R}$	Reward function	$\omega_{y_e}^{\text{LPP}}, \omega_{\text{rule}}^{\text{LPP}}, \omega_{\text{comf}}^{\text{LPP}}$	Reward component weight
$\tilde{r}$	yaw rate	$\omega_{\chi_e}^{\text{PF}}, \omega_{y_e}^{PF}, \omega_{\text{comf}}^{\text{PF}}$	Reward component weight
$\tilde{r}_{\text{scale}}$	Normalization constant	X	Surge force
$\dot{\tilde{r}}$	First derivative of yaw rate	X_H	Forces on ship hull in surge direction
$\dot{\tilde{r}}_{\text{scale}}$	Normalization constant	X_P	Forces on ship propeller in surge direction
$r_{\chi_e,t}^{\text{LPP}}$	Course error reward	X_R	Forces on ship rudder in surge direction
$r_{\text{coll},t}^{\text{LPP}}$	Collision penalty	X_{WA}	Wave forces on ship in surge direction
$r_{\text{comf},t}^{\text{LPP}}$	LPP comfort reward	X_{WI}	Wind forces in surge direction
$r_{\text{rule},t}^{\text{LPP}}$	Rule adherence reward	x_b	Ship-fixed longitudinal axis
$r_{y_e,t}^{\text{LPP}}$	Cross-track error reward	x_e	Along-track error
$r_{\chi_e,t}^{\text{PF}}$	Course error reward	x_G	Longitudinal coordinate of
$r_{\text{comf},t}^{\text{PF}}$	Comfort reward		ASV's center of gravity
$r_{y_e,t}^{\text{PF}}$	Cross-track error reward	x_n	Northing
$\tilde{r}_{\text{scale}}$	Normalization constant	y_b	Ship-fixed transversal axis
$\sigma_{\text{ground},t}$	Binary indicator for grounding	y_e	Cross-track error
$\sigma_{i,t}$	Direction indicator	$y_{e,\text{norm}}$	Cross-track error reward
$\sigma_{\text{coll},i,t}$	Collision indicator		normalization constant
$\sigma_{\text{lane},t}$	Binary indicator for lane crossing	Y	Sway force
$\sigma_{\text{off},j}$	Binary indicator for leaving path	Y_H	Forces on ship hull in sway direction
$\sigma_{\text{rule},t}$	Rule adherence indicator	Y_R	Forces on ship rudder in sway direction
$\sigma_{\text{spd},t}$	Target ship speed check indicator	Y_{WA}	Wave forces in sway direction
s	State	Y_{WI}	Wind forces in sway direction
$\mathcal{S}$	State space	y_n	Easting
T	Length of trajectory	y_{scale}	Normalization constant
$t_{i,t}^{\text{cpa}}$	TCPA to the i-th target ship	$\mathcal{Z}$	Observation function
t_{control}	PF activation interval	ζ_{wa}	Wave amplitude
t_{norm}	Scaling constant	$\zeta_{wa,\text{norm}}$	Normalization constant

C Appendix: Target ship control

In Algorithm 2, we specify the behavior of the target ships during the training and validation phases of the LPP agent. The returned angle from the algorithm is assigned as the new heading for the controlled target ship. The phrase *closest ship* in the algorithm refers to the surrounding ship with the closest Euclidean distance between the vessel's midpoints.

Algorithm 2: Rule-based heading control for target ships on IWs

Input:
B, L_{pp}: width and length between perpendiculars of the controlled ship
χ_d: desired course from VFG (with gain parameter $k = 0.001$)
χ_{P_k}: path angle between current waypoints
$\tilde{d}$: distance to closest ship
U_0, U_1: speed of the controlled ship and its closest surrounding ship
$t^{\text{CPA}}, d^{\text{CPA}}$: time and distance to CPA with the closest ship
α: relative bearing of the controlled ship from the perspective of its closest surrounding ship
σ: boolean, True if closest ship travels in reversed direction, False otherwise

Procedure:

```
# Turn right in case of opposing traffic
```
if $(t^{\text{CPA}} > 0) \wedge (d^{\text{CPA}} < 2B) \wedge (\tilde{d} \leq 10Lpp) \wedge \sigma$ **then**
  return $\chi_d + 5°$

```
# Overtake left
```
else if $(U_0 > U_1) \wedge (\tilde{d} \leq 10Lpp) \wedge (135° \leq \alpha \leq 315°)$ **then**
  if $y_e > 0$ **then**
    return $\psi_{dc} - 8°$
  else
    return $\chi_{P_k} - 8° \cdot \exp\left[(y_e/5B)\log(4)\right]$
  end

```
# Default VFG guidance
```
else
  return χ_d
end

D Appendix: Baseline: Artificial potential field method

We select an APF approach as a baseline method for the LPP task. The approach is based on the recent proposals of Liu et al. (2023) and Wang et al. (2019), which we slightly adapt to improve the method's performance on IW scenarios. According to the original proposal from the robotics domain of Khatib (1986), an APF can be constructed by superpositioning attractive and repulsive forces. The attractive forces $F_{\text{att},t}$ at time step t pull the planning agent toward the desired goal position, while the repulsive forces $F_{\text{rep},t}$ push the agent away from obstacles to avoid collisions. The resulting total force $F_t = F_{\text{att},t} + F_{\text{rep},t}$ is used to derive a desired heading as follows:

$$\psi_{d,t} = \arctan\left(F_{e,t}/F_{n,t}\right), \tag{D.1}$$

where $F_{n,t}$ and $F_{e,t}$ are the north and east components of F_t, respectively. Based on the desired heading $\psi_{d,t}$, the new heading can be set via:

$$\psi_{\text{OS},t+1} = \psi_{\text{OS},t} + \text{clip}\left(\psi_{d,t} - \psi_{\text{OS},t}, -\Delta_\psi, \Delta_\psi\right), \tag{D.2}$$

where $\psi_{\mathrm{OS},t}$ is the heading of the own ship of time t and Δ_ψ is the maximum possible heading change between time steps, considering the limited maneuverability of the vessel. Figure D.1 visualizes the attractive and repulsive forces imposed in this study, which we describe in detail in the following.

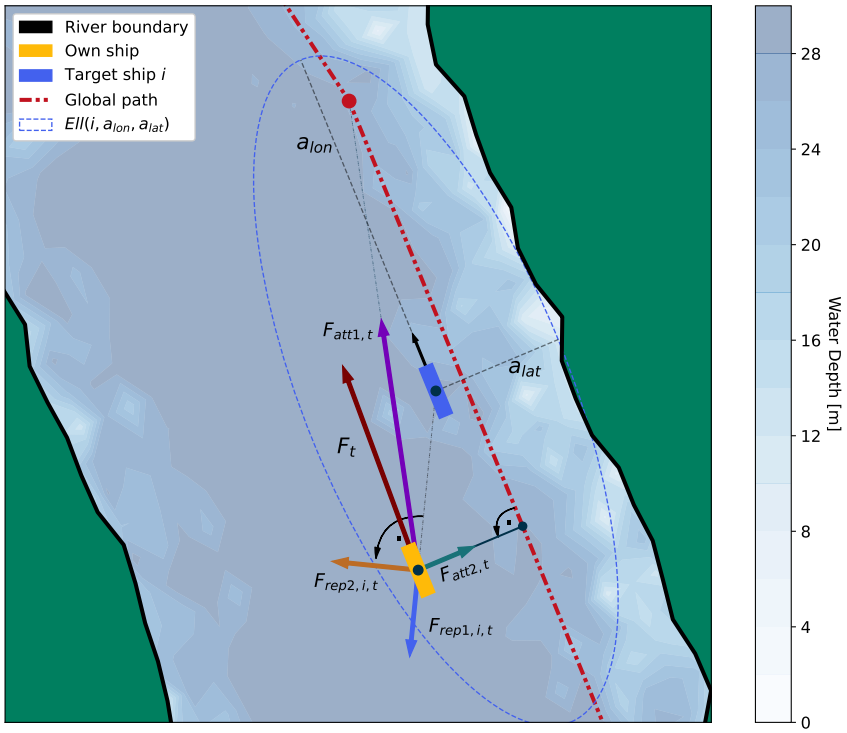


Figure D.1: Visualization of the forces of the APF method building on Liu et al. (2023) and Wang et al. (2019).

Following Liu et al. (2023), we construct the attractive forces as a sum of two components: $F_{\mathrm{att},t} = F_{\mathrm{att1},t} + F_{\mathrm{att2},t}$. The first component, $F_{\mathrm{att1},t}$, is the goal-oriented attractive force:

$$F_{\mathrm{att1},t} = \begin{cases} k_{a1} \cdot d^{\mathrm{G}}_{\mathrm{OS},t} \cdot \vec{n}_{\mathrm{OG}} & \text{if} \quad d^{\mathrm{G}}_{\mathrm{OS},t} < d^*, \\ k_{a1} \cdot d^* \cdot \vec{n}_{\mathrm{OG}} & \text{else}, \end{cases} \tag{D.3}$$

where k_{a1} and d^* are constants, $d^{\mathrm{G}}_{\mathrm{OS},t}$ is the Euclidean distance between the own ship and the goal point at time t, and $\vec{n}_{\mathrm{OG}}$ is the unit vector pointing from the own ship toward the goal. We thereby define the goal as a future waypoint of the global path, which is updated as the vessel progresses. The second component, $F_{\mathrm{att2},t}$, is responsible for pulling the vessel back toward the path to the goal if the cross-track error is too large. Formally, we have:

$$F_{\mathrm{att2},t} = \begin{cases} k_{a2} \cdot |y_{e,t}| \cdot \vec{n}_{\mathrm{OP}} & \text{if} \quad y_{e,t} \geq d_l, \\ 0 & \text{else}, \end{cases} \tag{D.4}$$

where k_{a2} and d_l are constants, $y_{e,t}$ is the cross-track error of the own ship at time t, and $\vec{n}_{\mathrm{OP}}$ is the unit vector from the own ship perpendicular to the linear path between the initial vessel

position and the goal; see Liu et al. (2023) and Figure D.1.

Denoting the repulsive force with each target ship i at time t as $F_{\text{rep},i,t}$, the overall repulsive force is attained via summation:

$$F_{\text{rep},t} = \sum_i F_{\text{rep},i,t}. \tag{D.5}$$

Similar to the attractive forces, we construct the repulsive force with relation to target ship i as the sum of two components. The first component, $F_{\text{rep1},i,t}$, is the conventional repulsive force pushing the vessel away from the obstacle (Khatib, 1986), while the second component, $F_{\text{rep2},i,t}$, being orthogonal to $F_{\text{rep1},i,t}$, is inspired by Liu et al. (2023) and ensures that overtaking occurs on the portside of the target ships.

Crucially, we impose two constraints that must be fulfilled to achieve a non-zero repulsive force from an obstacle. The first follows the proposal of Wang et al. (2019) and requires the own ship to be in an ellipse around a target ship, allowing the repulsive force to differ in the lateral and longitudinal direction of the target vessel. Note that this approach is similar to how we specified the collision reward component of the LPP agent in Section 5.1. The second constraint is that the TCPA between the own ship and the target ship at time t, denoted $t_{i,t}^{\text{cpa}}$, has to be larger or equal to zero. We found this second constraint crucial to improving the performance of the APF methods on our IW scenarios since it eliminates repulsive forces after a vessel has been overtaken. Aggregating these thoughts, we specify:

$$F_{\text{rep},i,t} = \begin{cases} F_{\text{rep1},i,t} + F_{\text{rep2},i,t} & \text{if} \quad [p_{\text{OS},t} \in \text{Ell}(i, a_{\text{lon}}, a_{\text{lat}})] \wedge (t_{i,t}^{\text{cpa}} \geq 0), \\ 0 & \text{else}, \end{cases} \tag{D.6}$$

where $p_{\text{OS},t}$ is the current north-east position of the own ship, and $\text{Ell}(i, a_{\text{lon}}, a_{\text{lat}})$ is the set of points inside the ellipse around target ship i. The semi-major axis of the ellipse has length a_{lon} and points in the direction of the heading of the target ship, while the semi-minor axis has length a_{lat}.

Following Khatib (1986), we thereby set:

$$F_{\text{rep1},i,t} = k_{r1} \cdot \left(\frac{1}{d_{\text{OS},t}^i} - \frac{1}{d_0} \right) \cdot \frac{1}{\left(d_{\text{OS},t}^i \right)^2} \cdot \vec{n}_{\text{TO}}, \tag{D.7}$$

where k_{r1} and d_0 are constant coefficients, and $d_{\text{OS},t}^i$ is the Euclidean distance between the own ship and target ship i at time t. The unit vector $\vec{n}_{\text{TO}}$ points from the target ship toward the own ship. Lastly, we define:

$$F_{\text{rep2},i,t} = \begin{cases} k_{r2} \cdot \vec{n}_{\text{OT}\perp} & \text{if} \quad |[\psi_{i,t} - \psi_{\text{OS},t}]_{-\pi}^{\pi}| < \frac{\pi}{2}, \\ 0 & \text{else}, \end{cases} \tag{D.8}$$

where k_{r2} is a constant, $\vec{n}_{\text{OT}\perp}$ is the unit vector that is perpendicular to portside on the unit vector from the own ship to the target ship. The variables $\psi_{i,t}$ and $\psi_{\text{OS},t}$ are the heading of the target ship i and the own ship, respectively. Hence, the condition in (D.8) requires that the vessels travel in opposing directions. Otherwise, the overtaking-related force $F_{\text{rep2},i,t}$ is set to

zero.

We empirically determine all parameters via a grid search, leading to the values: $\Delta_\psi = 2°$, $d^* = 0.5\,\mathrm{NM}$, $d_l = 50\,\mathrm{m}$, $k_{\mathrm{a1}} = 1$, $k_{\mathrm{a2}} = 0.1$, $k_{\mathrm{r1}} = 0.1$, $k_{\mathrm{r2}} = 0.1$, $a_{\mathrm{lat}} = d_0 = 0.5\,\mathrm{NM}$, and $a_{\mathrm{lon}} = 0.04\,\mathrm{NM}$.

E Appendix: Further validation: Local path planning agent

In the following, we provide further details on the discussed validation scenarios from Section 7.2, including the results for curved waterway segments of the LPP agent. As emphasized in Section 7.2, the APF method cannot be directly transferred to curved scenarios since its hyperparameters have been carefully tuned for the straight waterway case. Table E.1 includes the initial speed configuration of the target ships, while the speed of the own ship is set to 3 m/s in all cases. Furthermore, Figures E.1 and E.3 display the trajectories of the DRL planning agent for right and left curves, respectively. The black trajectories in these figures correspond to the DRL agent, while the colorized ones are the target ships controlled by Algorithm 2. The purple and grey dotted lines are the global and reversed global paths. Moreover, we display additional metrics during the curved scenarios in Figures E.2 and E.4, respectively.

Scenario	Waterway	U_1 [m/s]	U_2 [m/s]	U_3 [m/s]	U_4 [m/s]	U_5 [m/s]
1	straight	1.50	1.50	1.50	1.50	1.50
2	straight	1.05	1.65	-	-	-
3	straight	1.20	2.10	1.20	-	-
4	straight	1.20	2.10	1.20	1.65	-
5	straight	4.50	-	-	-	-
6	straight	0.00	0.00	0.00	0.00	0.0
7	right curve	1.50	1.50	1.50	1.50	1.50
8	right curve	1.05	1.65	-	-	-
9	right curve	1.20	2.10	1.20	-	-
10	right curve	1.20	2.10	1.20	1.65	-
11	right curve	4.50	-	-	-	-
12	right curve	0.00	0.00	0.00	0.00	0.0
13	left curve	1.50	1.50	1.50	1.50	1.50
14	left curve	1.05	1.65	-	-	-
15	left curve	1.20	2.10	1.20	-	-
16	left curve	1.20	2.10	1.20	1.65	-
17	left curve	4.50	-	-	-	-
18	left curve	0.00	0.00	0.00	0.00	0.0

Table E.1: Initial target ship speeds for the validation scenarios in the LPP task. The variable U_i, $i = 1, \ldots, 5$, is the speed of target ship i.

Similar to the findings for the straight waterway segment, the LPP agent demonstrates the successful execution of all necessary maneuvers while effectively maintaining safe distances and avoiding consecutive large changes in heading. It is worth highlighting that the agent's need to navigate through curved paths in the scenarios depicted in Figures E.1 and E.3 is also reflected

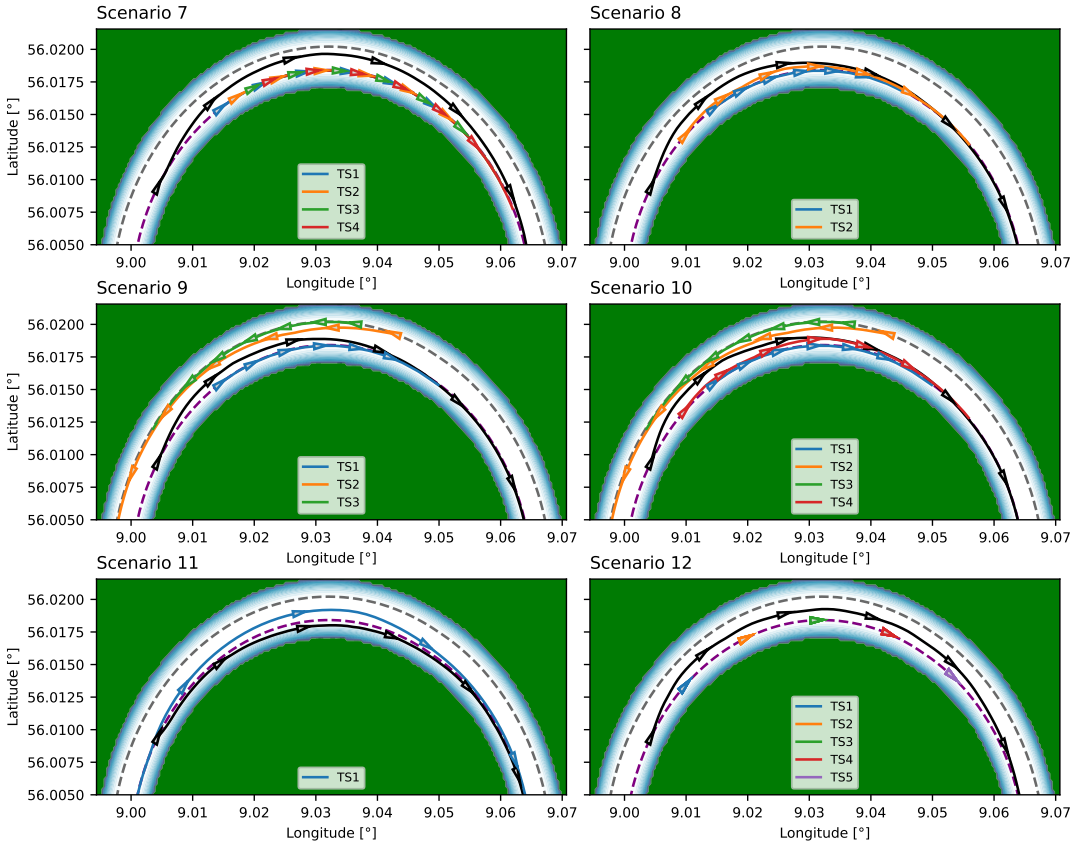


Figure E.1: Trajectories of the validation scenarios of the LPP agent on a right curve. Note that the latitude and longitude values are artificial and serve as orientation.

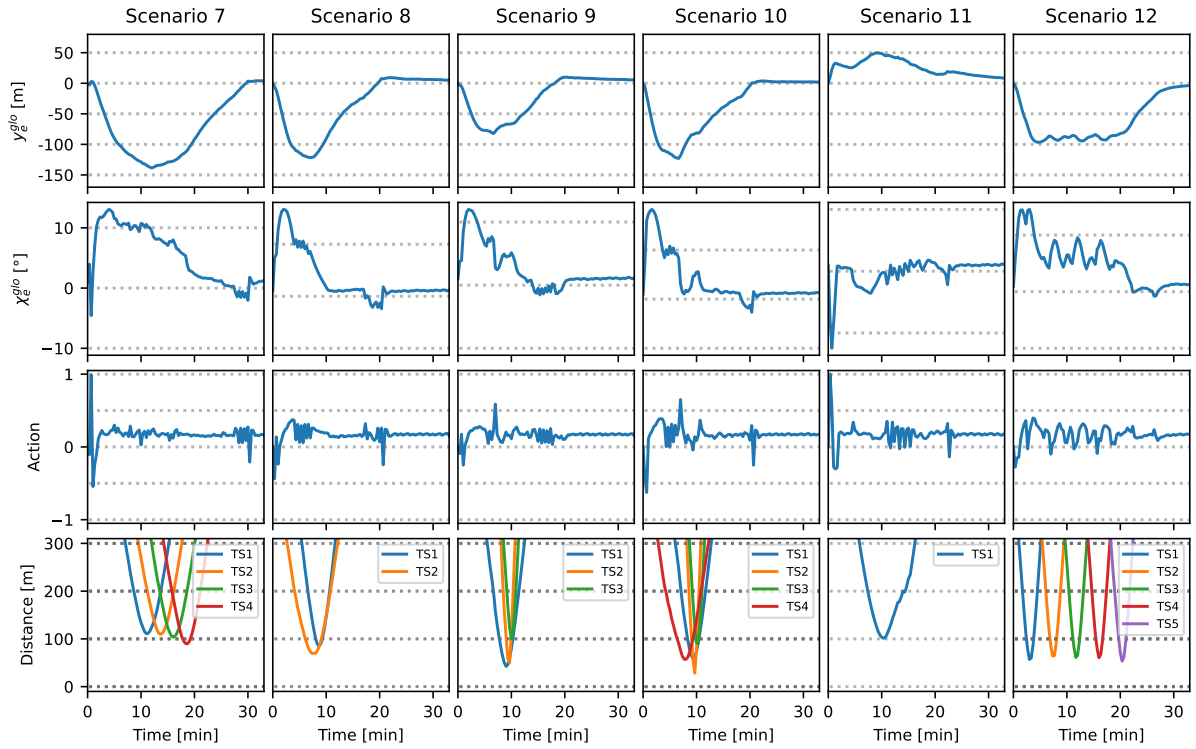


Figure E.2: Global cross-track and course error, selected actions, and distances to the target ships during validation of the LPP agent on a right curve.

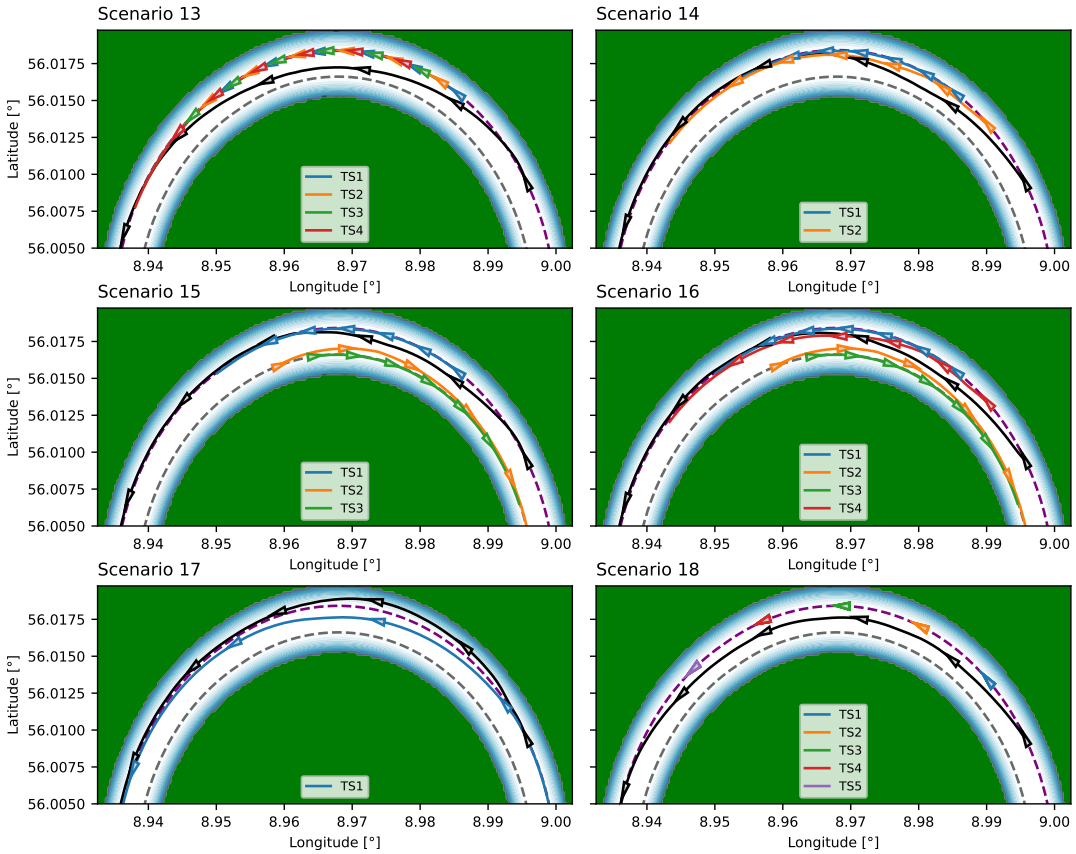


Figure E.3: Trajectories of the validation scenarios of the LPP agent on a left curve. Note that the latitude and longitude values are artificial and serve as orientation.

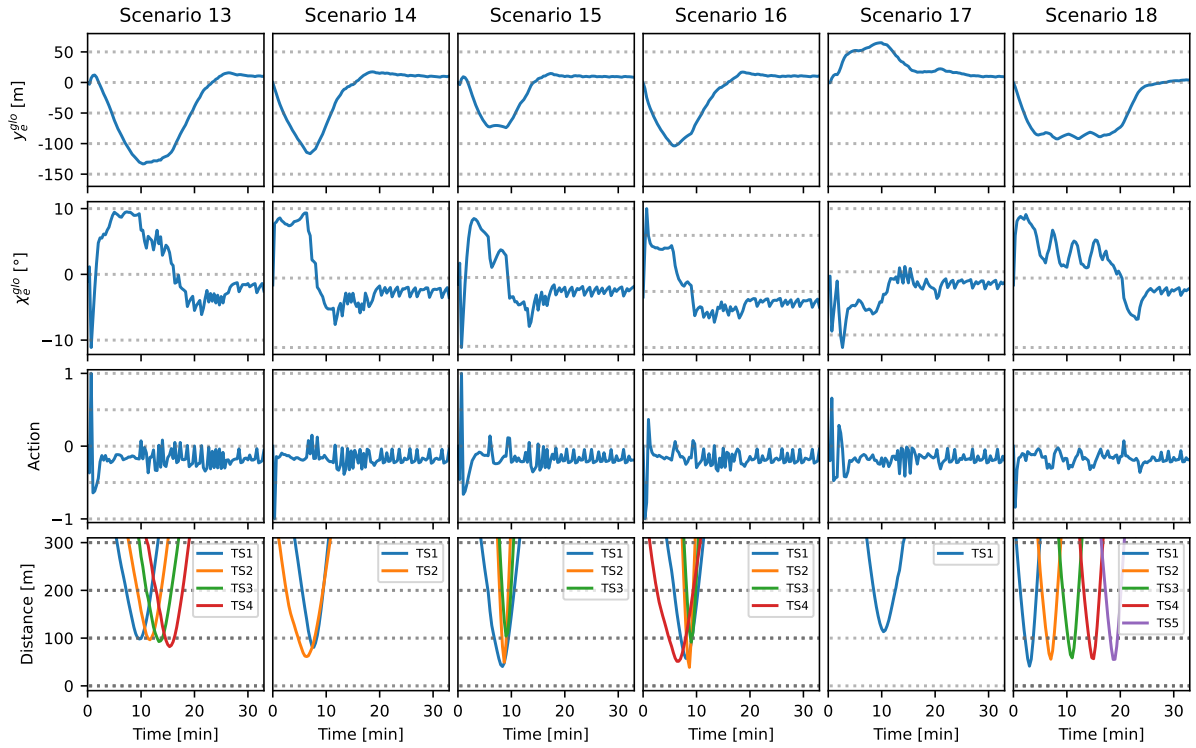


Figure E.4: Global cross-track and course error, selected actions, and distances to the target ships during validation of the LPP agent on a left curve.

in the average action values as shown in Figures E.2 and E.4, respectively. In these figures, we observe a slightly positive average action in the right curve scenario, while a slightly negative average action is observed in the left curve scenario.

F Appendix: Optimization of the PID controller

Following Paramesh and Rajendran (2021), we select a PID controller as a benchmark for our PF agent. In general, the PID-controlled rudder angle is computed as follows:

$$\tilde{\delta}_{t+1} = K_p \cdot \chi_{e,t} + K_i \cdot \sum_{\tilde{t}=0}^{t} \chi_{e,\tilde{t}} + K_d \cdot \tilde{r}_t, \tag{F.1}$$

$$\delta_{t+1} = \text{clip}\left[\text{clip}\left(\tilde{\delta}_{t+1}, \delta_t - a^{\text{PF}}, \delta_t + a^{\text{PF}}\right), -\delta_{\max}, \delta_{\max}\right], \tag{F.2}$$

where $\delta_0 = 0°$ and K_p, K_i, and K_d are the gain parameters. The two clipping operations ensure that the rudder angle does not change by more than $a^{\text{PF}} = 5°$ between time steps and does not exceed $\delta_{\max} = 20°$ in absolute value. Note that these constraints also apply to our PF agent. On this basis, we use the PSO approach of Eberhart and Shi (2000) to solve the following optimization problem:

$$\min_{K_p,K_i,K_d} \sum_{j=1}^{6} \left(10^7 \cdot \sigma_{\text{off},j} + \sum_{t=0}^{T_j} \chi_{j,e,t}^2\right), \tag{F.3}$$

where the sum goes over the six validation scenarios shown in Figures 4.13 and 4.14, respectively. The binary variable $\sigma_{\text{off},j}$ is set to one if the controlled vessel deviates significantly from the specified path and actually leaves the river during the corresponding scenario. Otherwise, $\sigma_{\text{off},j} = 0$. The episode length for scenario j is denoted as T_j, and if the vessel remains within the river throughout the episode, we have $T_j = 750$. Additionally, $\chi_{j,e,t}$ represents the course-error with respect to the global path at time t in scenario j. In summary, the optimization objective is to minimize the sum of squared course-errors while ensuring that deviations from the global path are not excessive enough to result in the vessel leaving the river.

Regarding the parametrization of the PSO method, we select a population size of 20 particles whose initial K_p, K_d, and K_i values are sampled from $\mathcal{U}(0.25, 3.75)$, $\mathcal{U}(10, 30)$, and $\mathcal{U}(0.025, 0.075)$, respectively, while the velocities where set to 0.05, 1, and 0.05. The algorithm was run over 1000 iterations and the inertia weight was linearly decayed from 0.9 to 0.4, following the suggestions of Eberhart and Shi (2000).

G Appendix: Details on the validation data

We consider the segment of the river Elbe from Lighthouse Tinsdal to the Elbe estuary close to Cuxhaven. The environmental data was gathered from the E.U. Copernicus Marine Service. In particular, current data stems from E.U. Copernicus Marine Service (2023a), wind data from E.U. Copernicus Marine Service (2023b), and wave data from E.U. Copernicus Marine Service (2023c).

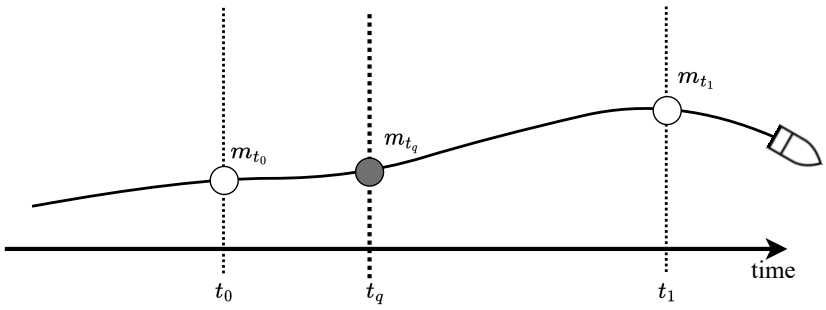


Figure G.1: Interpolation between two AIS messages of two ships, to receive information at the query time t_q; inspired by Rong et al. (2022).

The AIS data was kindly provided by the European Maritime Safety Agency. The collected records can be categorized into two groups:

1. *Static and voyage-related information:* This includes details such as the vessels' call sign, IMO number, estimated time of arrival (ETA), maximum draught, vessel type, and cargo type.

2. *Dynamic information:* This includes real-time data such as the vessels' current position (latitude, longitude), navigational status (under way, at anchor, etc.), Maritime Mobile Service Identity (MMSI), speed over ground (SOG), course over ground (COG), and rate of turn.

The static information is recorded every 6 minutes, while dynamic information is recorded at various intervals ranging from 2 to 180 seconds, depending on the vessels' dynamic conditions. For a detailed listing, please refer to International Telecommunication Union (2014, p. 8).

AIS records may contain incomplete, erroneous, or duplicate information (Last et al., 2014; Tu et al., 2017). Incomplete or faulty data can often be attributed to impaired measurement or transmission equipment onboard the vessels, while duplicate records occur when more than one receiving base station captures and records the vessel's message. To ensure accurate trajectory interpolation, we employ filtering techniques to eliminate duplicates. We compare the MMSI, timestamp, and receiving base station for each vessel, retaining only the first record if a vessel reports multiple times within a two-second window. Additionally, we remove records that have missing or incomplete information regarding position, SOG, COG, or MMSI. To facilitate the route extraction process, we sort the messages in ascending order based on time.

The LPP module relies on information about surrounding target ships at a specific time point, denoted as t_q. However, in cases where there is no exact record for t_q, interpolation between messages from different time points becomes necessary. This issue is illustrated in Figure G.1, which depicts two AIS messages, m_{t_0} and m_{t_1}, recorded at times t_0 and t_1 respectively, with $t_0 < t_q < t_1$. Leveraging ideas from Rong et al. (2022), we estimate the message at the query time, m_{t_q}, using a cubic spline. Importantly, we fit separate univariate splines for the northing, easting, course over ground, and speed over ground of each vessel. In cases where there are insufficient data points, we resort to linear interpolation for the respective quantity. The complete trajectory extraction pipeline is publicly available in Paulig (2023).

Chapter 5

An open-source framework for data-driven trajectory extraction from AIS data - the α-method

Niklas Paulig [a], *Ostap Okhrin* [a,b]

[a] *Institute of Transportation Economics, Technische Universität Dresden, 01062 Dresden, Germany*

[b] *Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI) Dresden/Leipzig, Germany*

Published in: *Ocean Engineering* (2024), 312:119092.

Abstract:
Ship trajectories from Automatic Identification System (AIS) messages are important in maritime safety, domain awareness, and algorithmic testing. Although the specifications for transmitting and receiving AIS messages are fixed, it is well known that technical inaccuracies and lacking seafarer compliance lead to severe data quality impairment. This paper proposes an adaptable, data-driven, maneuverability-dependent, α-quantile-based framework for decoding, constructing, splitting, and assessing trajectories from raw AIS records to improve transparency in AIS data mining. Results indicate the proposed filtering algorithm robustly extracts clean, long, and uninterrupted trajectories for further processing. An open-source Python implementation of the framework is provided.

Keywords: AIS, data-driven, trajectory extraction, big data, open-source

5.1 Introduction and background

The emergence of the AIS represents a crucial advancement in maritime technology with implications for navigational safety and maritime domain awareness. The International Maritime Organization (IMO) played a crucial role in shaping the global adoption of AIS, incorporating mandatory requirements into the Safety of Life at Sea (SOLAS) regulation V/19. The required use of AIS systems for all ships of more than 300 gross tonnages on international voyages and

cargo ships of more than 500 gross tonnages on national voyages became effective on 31 December 2004 (IMO, 2003).

Since then, a large group of researchers analyzed trajectories extracted from AIS records for various research domains such as anomaly detection (Pallotta et al., 2013; Zhang et al., 2016; Zhen et al., 2017; Rong et al., 2020; Wolsing et al., 2022), collision avoidance and risk assessment (Mou et al., 2010; Silveira et al., 2013; Chen et al., 2018; Rong et al., 2022; Zhang et al., 2023a), pattern classification (Amigo Herrero et al., 2019; Sanchez Pedroche et al., 2020; Duan et al., 2022; Luo et al., 2023) or path planning (He et al., 2019; Xu et al., 2019; Gu et al., 2023; Waltz et al., 2023).

Trajectories, however, cannot be constructed directly from raw AIS messages as they contain notable inconsistencies (Bailey, 2005; Harati-Mokhtari et al., 2007) such as erroneous positions, time, speed or course, Maritime Mobile Service Identity (MMSI) misuse or sharing or intentional transceiver deactivation, rendering them useless for further analysis. Trajectory extraction usually consists of two consecutive steps. First, filtering the raw data on an individual message level to eliminate erroneous messages. Second, the cleaned messages are combined into meaningful, coherent trajectories. A shared challenge for these steps lies in determining the diverse thresholds crucial for filtering individual messages or entire trajectories, especially since the range of many of the possible parameters to consider, like velocity, acceleration or turning rate depends on the maneuvering capabilities of individual ships, of which many different share the national and international waters.

In this paper, we introduce a framework (Figure 5.1) for maneuverability-dependent trajectory extraction, characterized by its near-complete reliance on data-driven methodologies with minimal reliance on predefined threshold values. The framework aims to cover the entire process of trajectory extraction from raw AIS messages, beginning with the collection and decoding phase (marked with green in Figure 5.1), MMSI grouping and individual message exclusion (marked with Purple), trajectory and sub-trajectory determination (Blue), and post-determination refinement (Yellow). In the following, relevant section headings will be marked with their corresponding framework color. In particular, our work contributes to the field as follows:

- *Generalizability:* Due to the pure data-driven approach, the framework is universally usable, regardless of the geographical region under consideration.
- *Expanded methodology:* This article goes beyond the methodology of Zhao et al. (2018), which splits trajectories into sub-tracks based on velocity and time differences between individual messages by additionally considering turning rates, distances, and the maneuverability of the ship by using its length as a proxy.
- *Trajectory assessment:* We introduce the concept of *average absolute change of course*, which allows for flexible assessment of extracted trajectories in terms of maneuvering behavior.
- *Implementation:* Our end-to-end framework is publicly available as an open-source package (Paulig, 2024) for the Python programming language.

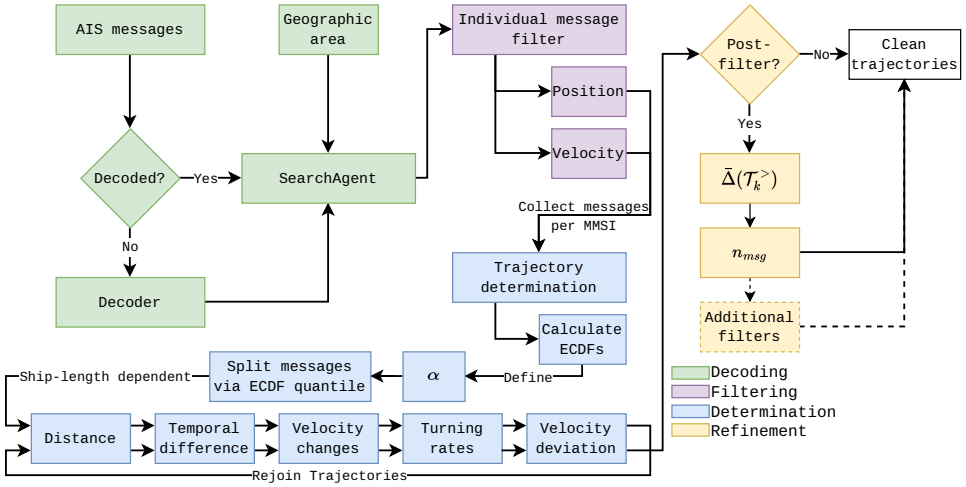


Figure 5.1: Flow chart diagram of the trajectory extraction framework presented in this article. Data collection and decoding is discussed in Section 5.4, Filtering in Section 5.5, Determination in Section 5.6 and post-filtering in Section 5.9.

Central to our approach is utilizing the α-quantiles of empirical distributions constructed from different features in the data, whose values will be used to distinguish coherent from incoherent trajectories. The parameter α can be adjusted to control the statistical power of the extracted trajectories.

To test the framework, a comprehensive analysis was conducted on a dataset spanning 912 days of AIS records from January 2020 to June 2022. The geographical scope of this investigation encompasses the North Sea and regions of the Baltic Sea, with the primary objective of determining robust methodologies for extracting long, uninterrupted, and clean trajectories from the dataset. By *clean*, we understand the absence of erratic and highly volatile movement sequences of navigational parameters (SOG, COG, Position). Our experiment shows that the fully data-driven paradigm is suitable to effectively remove erroneous messages and construct meaningful trajectories out of the remaining messages.

The rest of the paper is organized as follows. Section 5.2 recapitulates the different methodologies in trajectory extraction so far, Section 5.3 briefly revisits AIS records and their structure, and Section 5.4 details the obtained data set and Section 5.5 explains the individual message exclusion processes. Applications and comparisons of the framework are conducted in Sections 5.7 and 5.8. Section 5.6 introduces the split-point procedure based on empirical distribution functions. Section 5.9 proposes tools for assessing spatial features of the extracted trajectories, while Section 5.10 discusses the obtained results. Section 5.11 concludes. Additionally, Appendix J provides exemplary usage of the accompanying Python package.

5.2 Related works

Concerning the methodology for trajectory extraction, a notable observation emerges. While a confluence of approaches exists, an agreed-upon standard within the scientific community is still absent.

Various procedures have been proposed to identify and eliminate abnormal individual AIS

records. One filtering category applies to abnormal position reports, specifically those outside designated data areas. Zhang et al. (2018), Sang et al. (2015), Yuan et al. (2019), and Chen et al. (2020) have all explicitly addressed this issue. Another set of rules revolves around abnormal velocities. Let SOG be a message's speed over ground, then Yuan et al. (2019) suggests to exclude messages outside of $2kn < SOG < 20kn$, while Sang et al. (2015) drops the current message if the absolute difference between the last and the current velocity is outside the interval $[0.3, 0.8]$. Zhang et al. (2018) incorporates ship-type-specific maximum speeds, and Chen et al. (2020) sets a threshold at 30 knots for velocity-based exclusion. Abnormal Course over Ground (COG) is addressed by Yuan et al. (2019), Sang et al. (2015), and Chen et al. (2020), each proposing unique criteria, such as valid COG ranges and comparisons with theoretical tactical diameters. Abnormal Rate of Turn is considered by Zhang et al. (2018) and Yuan et al. (2019), with Zhang et al. (2018) specifying a maximum rotational limit based on vessel characteristics. Zhang et al. (2018) also introduces conditions for abnormal acceleration, restricting values within $[-1, 1]$ knots per second, while abnormal jerk, the derivative of acceleration, is constrained by Zhang et al. (2018) to values beyond $\pm 15 m/s^3$. Lastly, the issue of repeated data records is addressed by Yuan et al. (2019), albeit with unspecified criteria based on time judgments.

Following the initial filtration of individual messages, trajectories are typically constructed by sequentially linking messages emanating from the same source, identified by the MMSI (Mao et al., 2018; Zhao et al., 2018; Yuan et al., 2019; Guo et al., 2021b; Capobianco et al., 2021). Subsequently, an additional layer of trajectory-based filtering is applied to the generated trajectories. Several criteria have been proposed for filtering and splitting trajectories to enhance the accuracy and granularity of maritime data. Yuan et al. (2019), Zhao et al. (2018), and Mao et al. (2018) have contributed specific conditions for these purposes, while Yuan et al. (2019) suggests filtering trajectories based on the number of observations, proposing a threshold of 20 observations per track. Zhao et al. (2018) introduces a more stringent criterion, recommending a minimum of 100 observations for a track to be considered, whereas Mao et al. (2018) sets a threshold of 500 observations for tracks where the SOG is not equal to zero, thereby emphasizing the importance of trajectories during active movement. Additionally, Mao et al. (2018) proposes a condition to avoid high-complexity trajectories, employing the average cosine of the angles between three consecutive message points as an indicator, with a threshold set at 0.8.

Moreover, the concept of splitting trajectories into sub-tracks is addressed by Zhao et al. (2018). This involves three key conditions: a speed change greater than 15 knots between consecutive messages, a time interval exceeding 10 minutes between messages, and a permissible deviation between reported and generated time within $[-5, 5]$ seconds, covering 96% of the value range. These conditions collectively provide a comprehensive approach to delineating sub-tracks within longer trajectories, thereby extracting meaningful and discrete segments of vessel movement from AIS records.

5.3 AIS specifications

Over two decades ago, the ITU (2001) defined 27 different message types broadcast to serve many purposes, such as dynamic and static information about a voyage, safety-relevant information,

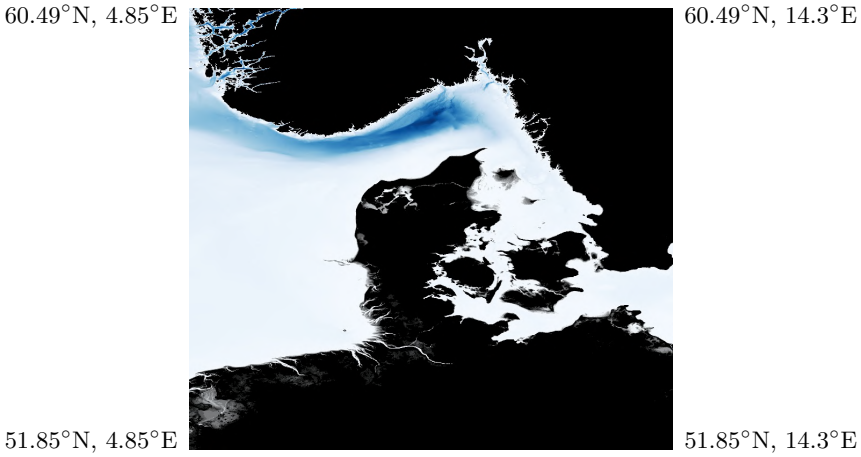


Figure 5.2: Geographical extent of the data set.

or UTC inquiries.

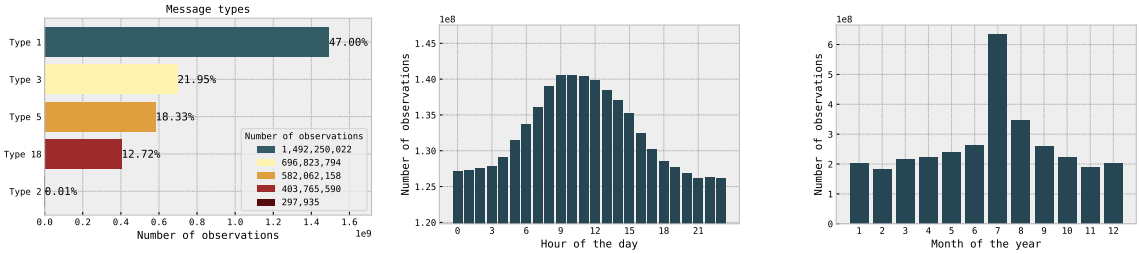
AIS broadcasts are sent from two types of transceivers (Class A and Class B), dependent on ship size, purpose, and local regulations; where Class A transceivers meet the performance standards and carriage requirements adopted by IMO, while Class B only partially fulfill them. Class A stations autonomously transmit their position (message types 1/2/3) every 2-10 seconds, depending on the vessel's speed and course changes, and at intervals of every three minutes or less when the vessel is anchored or moored. Additionally, these stations broadcast the vessel's static and voyage-related information (message type 5) every 6 minutes.

Like Class A stations, Class B reports their position every three minutes or less when the vessel is anchored or moored. However, their position reports (messages 6 and 8) occur less frequently and with reduced power. Likewise, they transmit the vessel's static data (messages 18/24) every 6 minutes, excluding voyage-related information. Due to the limited range and reliability, Class B reports are responsible for the small minority of records collected by base stations.

5.4 Raw data and decoding

The AIS data used in this study has been provided by the European Maritime Safety Agency (EMSA) and spans the geographical coordinates from 51.85°N to 60.49°N latitude and 4.85°E to 14.3°E longitude (refer to Figure 5.2). Due to its geographical extent, the data set contains messages received by base stations of the following countries: Belgium, Denmark, Estonia, Finland, France, Germany, Iceland, Ireland, Latvia, Lithuania, Netherlands, Norway, Poland, Russia, Sweden, and the United States military bases in Europe. The temporal scope encompasses the period from January 1, 2020, to June 30, 2022, which leads to approximately 3.2 billion (3 175 199 499) messages encoded as raw `AIVDM / AIVDO` sentences (refer to Table 5.1) (ITU, 2001), originating from 101 166 unique MMSIs. Dynamic messages, including types 1/2/3/18, constitute around 81.7% of the dataset, while the remaining messages pertain to static voyage reports of type 5, as presented in Figure 5.3a.

Descriptively, intra-day message patterns demonstrate an area-specific and predictable



(a) Count of messages by type.

(b) Absolute message count for each hour of the day, as calculated from receiving timestamps [UTC].

(c) Absolute message counts per month of the year, as calculated from receiving timestamps.

Figure 5.3: Descriptive statistics for the entire data set. Note that for the year 2022, the months from July until December are not part of the data set, and thus are not part of the absolute counts above.

Message Type	AIVDM sentence(s)
Type 1	`!AIVDM,1,1,,,13‘dUP0P000GqBjMw‘im0wvD0000,0*01`
Type 5	`!AIVDM,2,1,7,B,539g?6T00000@8i6221HU=E8LU>222222222220j1h5334@P04hTQCADR,0*15`
	`!AIVDM,2,2,7,B,0EQC‘888888880,2*27`

Table 5.1: Example of raw AIVDM sentences from the data set for a type 1 and a type 5 AIS message. Note, that type 1 messages consist of a single sentence, while type 5 messages are transmitted in two distinct sentences. A sentence always starts with the introducer "`!AIVDM`".

"workday" trend, with a heightened message frequency observed during Coordinated Universal Time (UTC) daytime hours (see Figure 5.3b). However, monthly messages present an atypical surge in message count during July (see Figure 5.3c), whose origin remains unknown.

The data set was provided as one-day chunks in the `.csv` format, with messages of types 1/2/3/18 as one file and messages of type 5 in a separate file. The variables of the raw data set can be found in Table 5.2. The raw `AIVDM / AVIDO` sentences were decoded using the open-source `pyais` (Morien, 2023) Python package according to IMO regulations (IMO, 2010). In the decoding process of the AIS messages, an initial filtering stage was implemented to address the issue of duplicate data. This phenomenon arises when a vessel's AIS report is within the coverage area of multiple base stations, leading to the same message being received from different locations. To mitigate this, any messages that were identical in content and received from more than one base station within a time frame shorter than the AIS system's minimum sending

Variable name	Description
`timestamp`	UTC timestamp of message reception
`message_id`	Message type
`latitude`	Latitude of the message
`longitude`	Longitude of the message
`raw_message`	`AIVDM` sentence
`MMSI`	Maritime Mobile Service Identifier
`originator`	Country abbreviation of the receiving base station

Table 5.2: Variables of the raw data set.

frequency $f_{min} = 2s$ were excluded.

5.5 Individual message exclusion

5.5.1 Position reports

In line with methodologies outlined by Zhang et al. (2018), Sang et al. (2015), Yuan et al. (2019), and Chen et al. (2020), we systematically exclude messages with geospatial coordinates falling outside the defined spatial region of the data set. This process also conveniently addresses the issue of erroneous positional data. By setting logical boundaries for latitude (lat) and longitude (lon) – specifically, $51.85° \leq \text{lat} \leq 60.49°$ and $4.85° \leq \text{lon} \leq 14.3°$ – any data points falling outside these ranges are automatically discarded. Notably, the data set sourced from the EMSA had already undergone this preliminary filtering step; therefore, applying this spatial boundary criterion resulted in removing zero additional messages from the data set.

5.5.2 Velocity reports

The data feature an extensive range of reported SOGs, of which only a fraction is relevant for trajectory extraction. The analysis of the SOG data from 2021, as presented in Figure 5.4, indicates a notable number of observations registering a speed of approximately 102 knots. This observation aligns with established standards in the AIS protocol, in which a reported speed of 102.3 knots is conventionally used to signify that actual speed data is unavailable. This value does not represent an actual speed measurement but serves as a placeholder in the data set when speed information cannot be provided or is not applicable.

Figure 5.4 also reveals that a small fraction of messages ($\approx 0.3\%$) report velocities exceeding 30 knots. This observation is significant given the economic ramifications of high-speed maritime travel, such as elevated fuel consumption and accelerated attrition of materials. As also documented by Notteboom and Cariou (2009), commercial vessels rarely surpass the 30-knot threshold due to these constraints. Primarily, only specific categories of ships, such as high-speed crafts and certain military vessels, regularly operate beyond this speed limit.

In light of these insights, in concordance with Chen et al. (2020), this study opts to exclude all messages reporting speeds above 30 knots, thus keeping 99.7% of the data. This decision is grounded in the understanding that such high speeds are atypical for most commercial maritime traffic and, thus, may not represent typical operational behaviors and could potentially stem from errors in the ship's transceiving equipment.

A significant proportion of the recorded messages ($> 71\%$) indicated speeds below one knot. This prevalence of low-speed signals in the data set can be attributed to several factors, the most notable of which is the inclusion of non-moving, i.e., anchored or moored vessels. Slow-moving construction vessels, which are mandated to transmit AIS records regardless of their motion status, also contribute to these messages. For instance, Figure 5.5 illustrates an instance of a vessel at anchor with its AIS transceiver operational. While minimal, the motion of this vessel is still captured and transmitted as part of the AIS data. Such behavior, characterized by minimal or no actual navigational movement, should be considered for exclusion from data

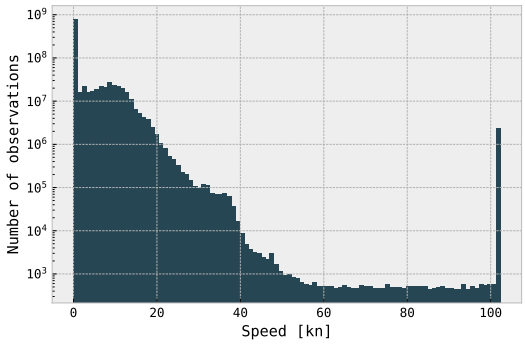


Figure 5.4: Histogram of speeds for all messages received in 2021 ($n \approx 1.07 \times 10^9$). Note that the ordinate is log-transformed to improve visualization due to the large range of counts.

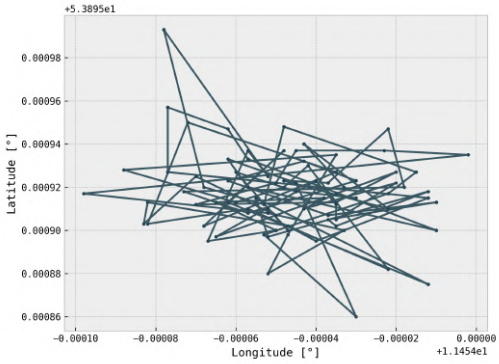


Figure 5.5: Trajectory of the vessel with MMSI 211XXXX90 laying at anchor in the harbor of Wismar, Germany.

analyses focusing on vessel trajectories. For this reason, in this study, we remove messages whose SOG is less than one knot.

5.6 Trajectory construction

After the preliminary cleaning, the next step is to assemble the remaining AIS records into trajectories using a two-step approach. In the first step, we identify potential *split-points* within continuous vessel trajectories, defined as pairs of consecutive messages that appear to be more appropriately classified as belonging to separate trajectories. The separation criteria are based on empirical quantiles of several metrics detailed below. In the second step, the concept of *absolute change of course* of a trajectory is introduced to provide a flexible examination tool of the split trajectories.

A series of definitions are introduced below to facilitate comprehension and minimize confusion in the subsequent sections of this paper.

5.6.1 Definitions

Let m represent a single AIS message. Define the set $\mathcal{T}_k = \{m \mid \text{mmsi}(m) = k\}$ as the collection of all messages from a vessel with MMSI being equal to k, where $\text{mmsi}(m)$ is a mapping of the

AIS message to its corresponding MMSI. For this study, we focus on a vessel's trajectory, i.e., ordered time-position tuples represented by the ordered set:

$$\mathcal{T}_k^{>} = \{m_1, m_2, \ldots, m_p\},\ \text{unix}(m_1) \leq \text{unix}(m_2) \leq \cdots \leq \text{unix}(m_p),\ \forall m \in \mathcal{T}_k. \tag{5.1}$$

This set consists of the vessel's transmitted messages, ordered by their UNIX timestamps in seconds, extracted from the messages via the $\text{unix}(m)$ function. Furthermore, define $\text{lat}(m)$, $\text{lon}(m)$, and $\text{sog}(m)$ functions returning latitude, longitude, and SOG, respectively, from a message m.

Some analyses require the computation of velocities obtained from spatial and temporal data. Given the Earth's oblate spheroid geometry, distance estimation between geographical coordinates cannot be performed using Euclidean geometry. Predominantly, two methods are employed: the Haversine formula and the Vincenty (1975) algorithm, offering superior precision (Mahmoud and Akkari, 2016). However, for the objectives of this study, such a level of accuracy is unnecessary, predominantly due to the significantly larger error margins inherent in original GPS data (Jankowski et al., 2021). In addition, the Haversine formula's computational efficiency is notably higher, making it a more pragmatic choice. It is defined as

$$\text{hav}(m_1, m_2) = 2r \arcsin\left[\sqrt{\sin^2\left(\frac{\Delta_\phi}{2}\right) + \cos\{\text{lat}(m_1)\}\cos\{\text{lat}(m_2)\}\sin^2\left(\frac{\Delta_\lambda}{2}\right)}\right], \tag{5.2}$$

with lateral and longitudinal differences $\Delta_\phi = \text{lat}(m_2) - \text{lat}(m_1)$, $\Delta_\lambda = \text{lon}(m_2) - \text{lon}(m_1)$, and $r = 6371km$ being earth's volumetric radius. The formulas' output is a distance in meters, while latitude and longitude are provided in radians.

Furthermore, let

$$\overline{SOG}_{m_i}^{m_{i+1}} = \frac{\text{sog}(m_i) + \text{sog}(m_{i+1})}{2}, \tag{5.3}$$

be the average reported speed over the ground for two consecutive messages $\text{sog}(m_i)$ and $\text{sog}(m_{i+1})$, and

$$\widehat{SOG}_{m_i}^{m_{i+1}} = \frac{\text{hav}(m_i, m_{i+1})}{\text{unix}(m_{i+1}) - \text{unix}(m_i)} \cdot \frac{1852}{3600}, \tag{5.4}$$

be the estimated SOG in knots calculated from the change in position over time. The scalar constant $1852/3600$ is the conversion factor that translates ms^{-1} to kn.

5.6.2 Trajectory splitting

Consistent with the abovementioned research, we use the MMSI to assign AIS messages to their originating vessels and order these messages using the UTC timestamp to formulate a trajectory. Nevertheless, this methodology exhibits significant shortcomings without further processing, as it often results in trajectories with numerous undesirable characteristics, which will be outlined in this section. Accordingly, this study advances the trajectory segmentation methodology proposed by Zhao et al. (2018), applying it to ascertain the continuity of a trajectory by evaluating whether two successive AIS messages should be attributed to the same navigational path via a *split-point* approach. Therefore, α-quantiles of various distributions, detailed in the next Sections, are computed to decide whether to split the trajectory at a particular point. Additionally,

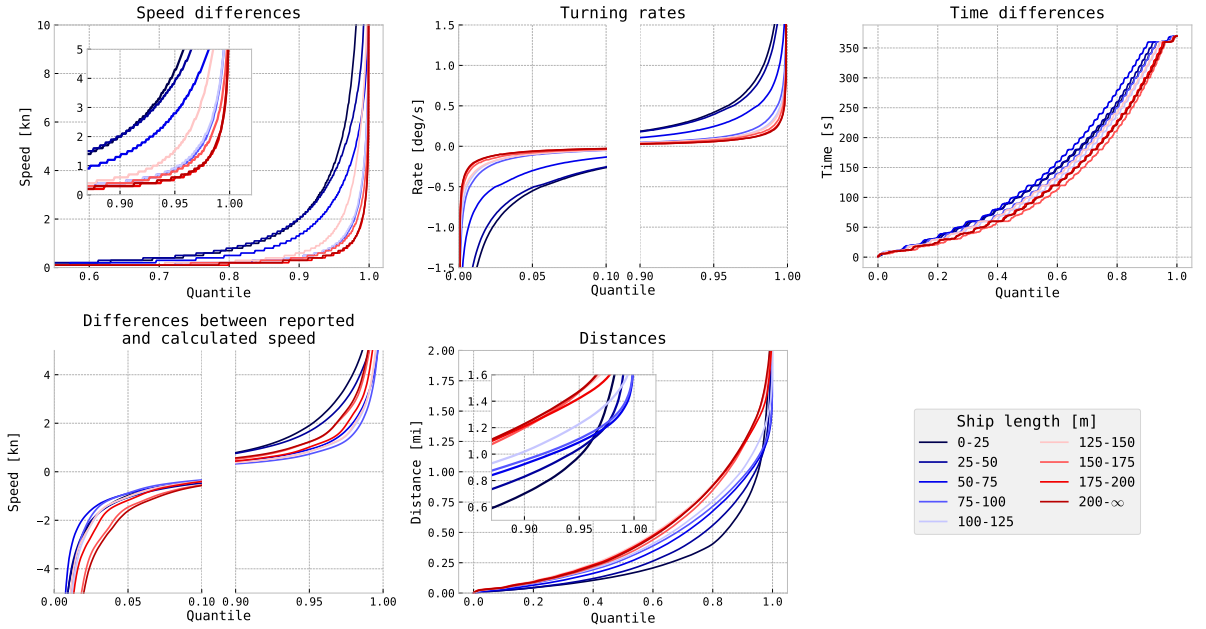


Figure 5.6: Length-dependent quantile values for the five different metrics used to determine split points in the constructed trajectories. Note that the time difference quantiles are not used ship-length dependent, and are only displayed for completeness.

we incorporate the ideas from Zhang et al. (2018), and Chen et al. (2020) by explicitly taking the vessel maneuverability into account, by using the ship length as a proxy. Preliminary testing concluded that other proxies such as the ship type work worse, as there are ship types such as `CARGO` spanning the entire range of ship lengths and widths, therefore not representing maneuverability accurately. Ship mass, while being a promising proxy, was also discarded, as critical information for its calculation are either unreliable in the data (draft) or not part of it (block coefficient). Furthermore, Hansen et al. (2022) related ship lentgths to thier heading change per meter and concluded that ship length is a valid proxy for maneuverability. To get a useful segmentation of the length, we introduce 9 distinct bins $[0, 25, 50, 75, 100, 125, 150, 175, 200, \infty]m$. We acknowledge, that these boundaries have no physical justification, but are a practical tradeoff between accuracy and computational complexity. Empirical distributions are then calculated for each of the bins. The procedure is consolidated in Algorithm 3 and a visual representation of the length-dependent quantile values can be found in Figure 5.6. Throughout the next subsections, it is assumed that if a trajectory is split multiple times such that any of the sub-trajectories only consists of one message, this sub-trajectory is discarded.

Temporal difference

The most natural metric to investigate for split points is the temporal difference between two successive messages of one trajectory $\text{unix}(m_{i+1}) - \text{unix}(m_i)$. Revisiting the AIS specifications, the defined transmission interval for dynamic AIS messages ranges from two to ten seconds for class A equipment ($2 - 30$ seconds for class B equipment) for vessels not at anchor and 180 seconds for vessels at anchor. Therefore, it is expected to find an accumulation of temporal gaps around these thresholds in the data, which could then be used to split trajectories. A

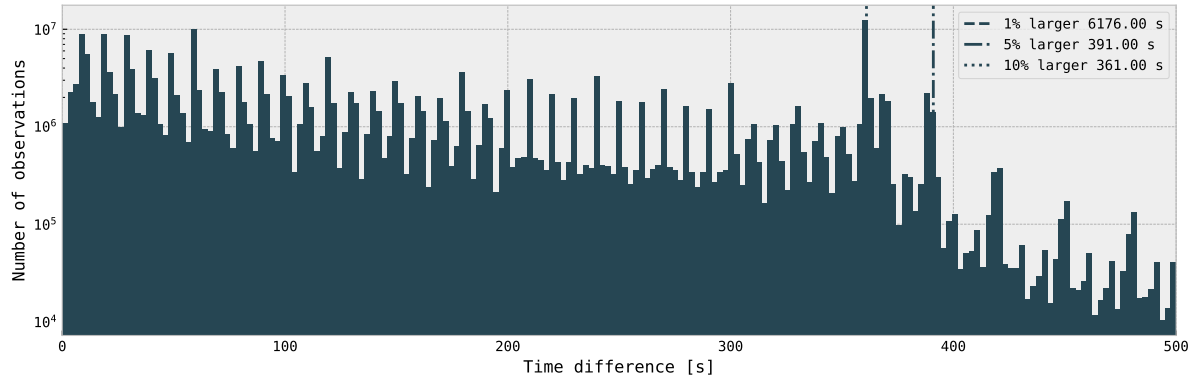


Figure 5.7: Histogram of the gap size for two consecutive dynamic messages in one trajectory. Gap sizes larger than 500s are truncated for better visualization. The histogram was built with speed-filtered data.

histogram of the temporal differences between two successive messages inside each trajectory for 2021 is depicted in Figure 5.7 to verify the assumptions. A brief discussion on why one year of data was used in the calculation is provided in Appendix I. It is noteworthy that we abstained from determining the distribution for each length interval, as the temporal distribution is not influenced by the ship's length but rather by technical or human factors such as delay or intent. The distribution clearly shows that our initial assumptions are untenable, as temporal differences vary significantly in the interval from $[2, 360]s$ with an exceptional peak around the $360s$ mark, whose origin remains undisclosed. The only message types in the AIS specification transmitted at $360s$ intervals are safety-relevant messages, none in the data set, and static voyage reports, which had not been considered.

The findings of this study suggest that it is not advisable to segment vessel trajectories based on the transmission intervals prescribed by AIS standards. Instead the calculated values from the distribution of Figure 5.7 are used, depending on α.

To determine all following empirical distribution functions, we only consider the value of the respective metric if the two messages under consideration are less than the 95%-percentile of temporal gaps, namely $391s$, apart. We do this to explicitly exclude artificial outliers that are so far apart temporally that the significance of the obtained values is questionable.

Velocity changes

Given the defined transmission interval for dynamic AIS messages, the variance in reported speeds between two adjacent records is anticipated to be reasonably small. This expectation is based on the premise that substantial acceleration is atypical for commercial vessels. Therefore, a threshold value must determine whether two consecutive messages belong to the same trajectory.

Following the procedure from the preceding paragraph, the absolute speed change distribution for all vessel length intervals for 2021 was obtained. The results are illustrated in Table 5.3 and show a clear pattern of receding speed and thus acceleration values for increasing ship lengths, reflecting the different maneuvering capabilities of different ship types.

Length bin	α			Accel. range	Length bin	α			Accel. range	Length bin	α			Accel. range
	0.1	0.05	0.01	at $\alpha = 0.05$		0.1	0.05	0.01	at $\alpha = 0.05$		0.1	0.05	0.01	at $\alpha = 0.05$
$[0, 25)m$	2.3	4.7	14.4	$[0.012, 2.350]$	$[75, 100)m$	0.4	0.9	3.4	$[0.002, 0.450]$	$[150, 175)m$	0.4	0.8	2.8	$[0.002, 0.400]$
$[25, 50)m$	1.9	3.7	9.0	$[0.009, 1.850]$	$[100, 125)m$	0.4	0.9	3.5	$[0.002, 0.450]$	$[175, 200)m$	0.3	0.5	1.8	$[0.001, 0.250]$
$[50, 75)m$	1.2	2.5	6.2	$[0.006, 1.250]$	$[125, 150)m$	0.5	1.3	5.9	$[0.003, 0.650]$	$[200, \infty)m$	0.3	0.6	1.9	$[0.002, 0.300]$

Table 5.3: Cutoff quantile values for the speed of two consecutive messages in miles for selected values of α across the different length intervals. The lower acceleration values are calculated by dividing the cutoff value by the 5%-percentile of the temporal difference distribution ($391s$). The upper acceleration values are obtained by dividing by the minimum AIS sending frequency of $2s$. Units of acceleration are kn/s.

Length bin	$[1-\alpha/2, \alpha/2]$			Length bin	$[1-\alpha/2, \alpha/2]$		
	0.1	0.05	0.01		0.1	0.05	0.01
$[0, 25)m$	$[-0.619, 0.473]$	$[-1.067, 0.840]$	$[-1.949, 1.600]$	$[75, 100)m$	$[-0.113, 0.106]$	$[-0.218, 0.200]$	$[-0.436, 0.408]$
$[25, 50)m$	$[-0.521, 0.402]$	$[-0.880, 0.700]$	$[-1.557, 1.300]$	$[100, 125)m$	$[-0.100, 0.092]$	$[-0.183, 0.169]$	$[-0.365, 0.340]$
$[50, 75)m$	$[-0.305, 0.251]$	$[-0.510, 0.459]$	$[-0.860, 0.785]$	$[125, 150)m$	$[-0.099, 0.091]$	$[-0.173, 0.160]$	$[-0.324, 0.300]$

Length bin	$[1-\alpha/2, \alpha/2]$		
	0.1	0.05	0.01
$[150, 175)m$	$[-0.083, 0.080]$	$[-0.148, 0.142]$	$[-0.148, 0.142]$
$[175, 200)m$	$[-0.062, 0.060]$	$[-0.114, 0.111]$	$[-0.218, 0.210]$
$[200, \infty)m$	$[-0.065, 0.062]$	$[-0.113, 0.109]$	$[-0.205, 0.200]$

Table 5.4: Cutoff quantile intervals for the turning rate of two consecutive messages in $[^\circ/s]$ for selected values of α across the different length intervals.

Turning rates

It is generally not anticipated that significant deviations in a ship's turning rate will occur during navigation, as rapid directional shifts as those in Figure 5.8 not only increase material wear and tear but also contravene the IMO's routing rationale advocating for safe and straightforward routing (IMO, 2019). Consequently, Table 5.4 presents the tuning rates for two consecutive messages using all trajectories from 2021 and three selected α values. Turning rates are calculated by dividing the difference of COG changes of two consecutive messages by their temporal difference. As with the differences in speed, we also see a clear length-dependent pattern of inversely related turning rates and ship lengths.

The relatively low turning rates may be attributed to several factors that underline the principles of safe and efficient maritime navigation. As mentioned, the IMO's guidelines advocate for straightforward and safe routing. These guidelines are designed to minimize the risk of accidents, enhance the safety of the ship's voyage, and protect the marine environment. Ships often use predetermined routes optimized for safety and efficiency, reducing the need for frequent or sharp turns. Further, many modern ships have advanced navigation systems like autopilots and GPS-based tracking. These systems are programmed to execute changes in course in a controlled and gradual manner, adhering closely to the calculated optimal route. Using such technology helps maintain the consistency of movements and avoid abrupt directional changes. Thus, the small range of turning rates reflects the combined impact of ship design, navigational technology, regulatory compliance, and environmental strategy aimed at ensuring a smooth, safe, and efficient voyage.

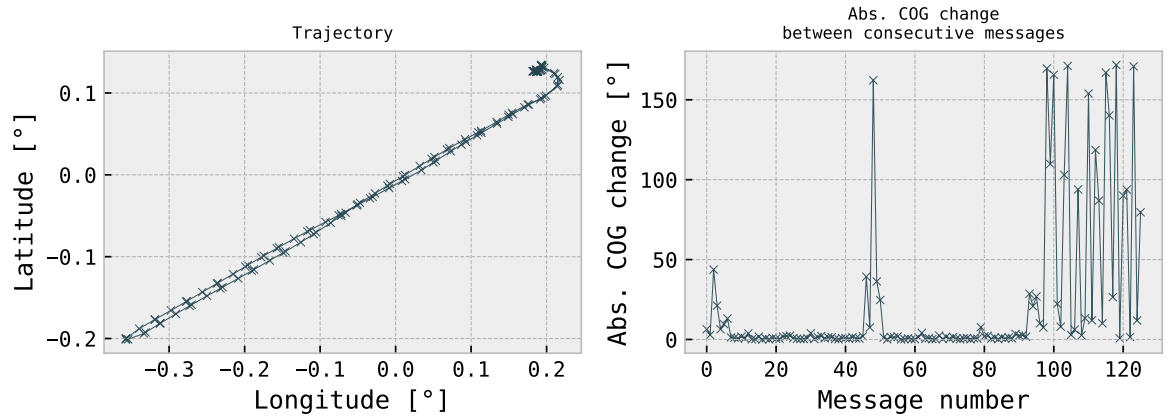


Figure 5.8: Example of a vessel's trajectory that will be cut into several sub-trajectories by the heading change split-point threshold. The route exerts a ferry-like pattern where most of its trajectory is inconspicuous except for the turning points (around message numbers 50 and 100), where the ferry presumably performs a turn-around maneuver or is moored.

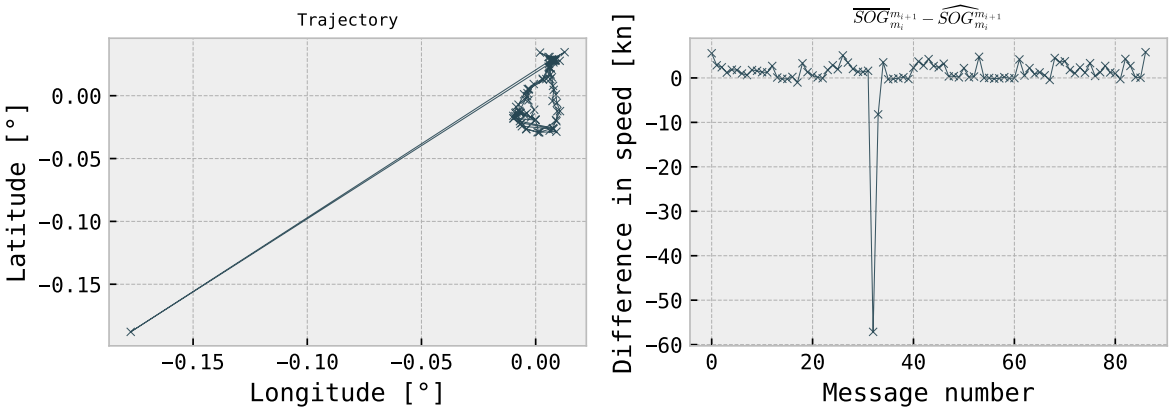


Figure 5.9: Erroneous positional outlier detected via the difference between the average reported SOG from the messages and the calculated speed from positional and temporal data. Latitude and longitude had been normalized for data protection

Length bin	$[1-\alpha/2, \alpha/2]$			Length bin	$[1-\alpha/2, \alpha/2]$		
	0.1	0.05	0.01		0.1	0.05	0.01
$[0, 25)m$	$[-1.434, 2.332]$	$[-3.237, 4.022]$	$[-8.412, 6.699]$	$[75, 100)m$	$[-1.074, 0.822]$	$[-2.701, 1.507]$	$[-10.197, 3.338]$
$[25, 50)m$	$[-1.426, 1.806]$	$[-3.233, 3.240]$	$[-9.794, 5.381]$	$[100, 125)m$	$[-1.294, 0.975]$	$[-3.353, 1.861]$	$[-12.804, 4.014]$
$[50, 75)m$	$[-1.032, 0.995]$	$[-2.110, 1.876]$	$[-6.892, 3.633]$	$[125, 150)m$	$[-1.294, 0.987]$	$[-3.041, 1.948]$	$[-11.222, 4.233]$

Length bin	$[1-\alpha/2, \alpha/2]$		
	0.1	0.05	0.01
$[150, 175)m$	$[-2.023, 1.393]$	$[-5.134, 2.780]$	$[-19.192, 5.877]$
$[175, 200)m$	$[-1.534, 1.188]$	$[-4.032, 2.433]$	$[-15.734, 5.155]$
$[200, \infty)m$	$[-2.176, 1.371]$	$[-5.907, 2.850]$	$[-21.707, 6.044]$

Table 5.5: Cutoff quantile intervals for the difference between reported and calculated speed of two consecutive messages in $[kn]$ for selected values of α across the different length intervals.

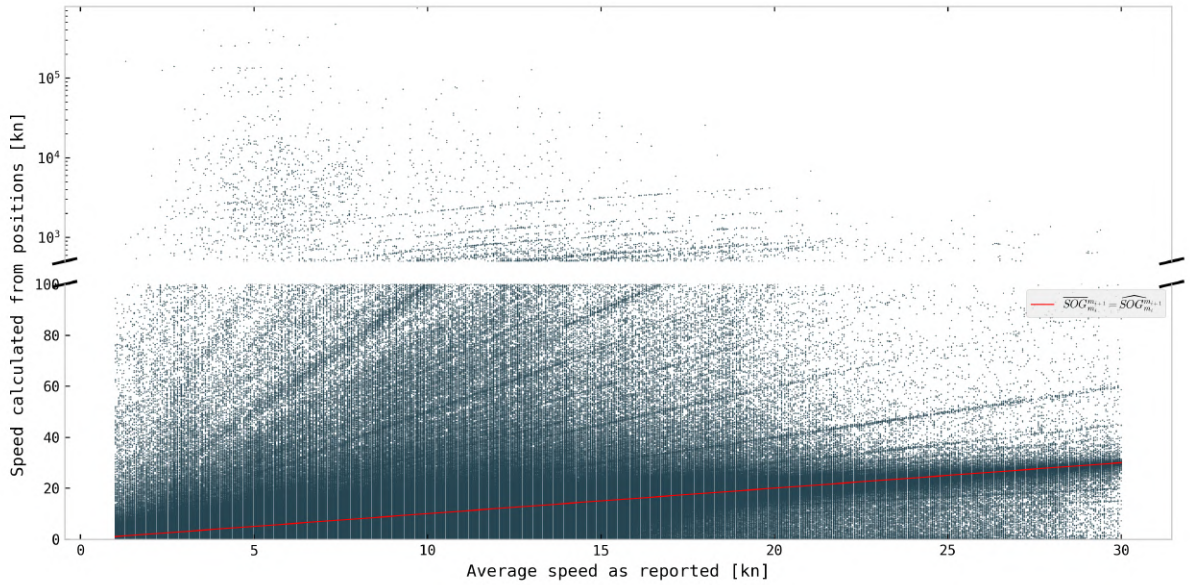


Figure 5.10: Scatter plot of the average observed SOG ($\overline{SOG}_{m_i}^{m_{i+1}}$) and the calculated SOG from positional and temporal data ($\widehat{SOG}_{m_i}^{m_{i+1}}$) for all trajectories of July 2021.

Reported against positional velocity

Another deficiency of trajectories that cannot be captured with the three metrics above are erroneous position reports that lead to significant outliers, as seen in Figure 5.9. In those cases, there is usually no anomaly detectable in terms of speed, course, or temporal differences, which is why this study follows Zhao et al. (2018) and uses the difference between $\overline{SOG}_{m_i}^{m_{i+1}}$ and $\widehat{SOG}_{m_i}^{m_{i+1}}$ to determine the outliers. Figure 5.10 provides context on the magnitude of differences across the admitted speed range. It can be observed that there are substantial deviations of different magnitudes in both directions. As with the other metrics, some of the determined threshold values can be viewed in Table 5.5. The skewness of the intervals is expected, as $\overline{SOG}_{m_i}^{m_{i+1}}$ is bounded at $30kn$ due to the speed filtering employed in Section 5.5.2, while $\widehat{SOG}_{m_i}^{m_{i+1}}$ is unbounded.

Distance between messages

The final metric to be considered in our analysis is the distance between two successive AIS messages. The speed filtering process described in Section 5.5.2 involves removing trajectory

Length bin	α			Length bin	α			Length bin	α		
	0.1	0.05	0.01		0.1	0.05	0.01		0.1	0.05	0.01
$[0, 25)m$	0.727	1.001	1.969	$[75, 100)m$	0.922	1.095	1.352	$[150, 175)m$	1.148	1.424	1.879
$[25, 50)m$	0.771	1.004	1.574	$[100, 125)m$	0.992	1.205	1.517	$[175, 200)m$	1.162	1.391	1.775
$[50, 75)m$	0.895	1.072	1.332	$[125, 150)m$	1.160	1.409	1.808	$[200, \infty)m$	1.177	1.444	1.994

Table 5.6: Cutoff quantile values for the distance of two consecutive messages in miles for selected values of α across the different length intervals.

segments where the SOG falls below one knot. This exclusion results in significant spatial gaps between successive messages within a trajectory, potentially leading to substantial temporal gaps if the vessel continues sailing. If the vessel increases its SOG to over one knot, its messages appear again in the data. In these cases, the discrepancy between the calculated speed (derived from positional and temporal data) and the average reported speed might be insufficiently large to be identified as split points based on their percentile criterion.

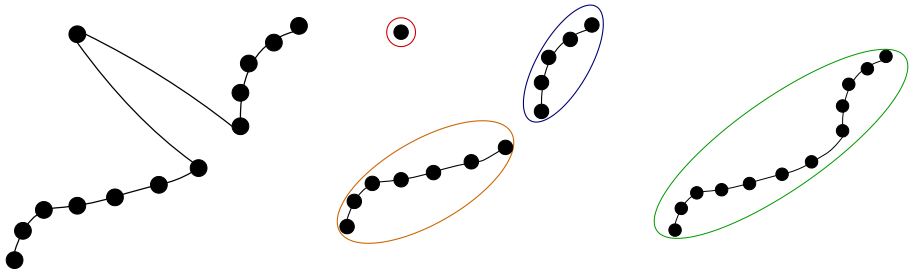
To additionally capture these special cases, we also label two consecutive messages a split-point if their distance in miles, as calculated via the Haversine formula, is greater than the $(1 - \alpha)$ quantile of the distance distribution obtained for the year 2021, which are depicted in Figure 5.6.

5.6.3 Re-joining trajectories

As pointed out by Zhao et al. (2018), rejecting singular anomalous points may lead to an accidental separation of a trajectory that logically belongs together. The example in Figure 5.11 illustrates such a case. The original trajectory (Figure 5.11a) consists of a single anomalous point that should be removed from the data. Our split-point procedure, therefore, looks at consecutive overlapping pairs of messages and decides to split them based on the empirical distribution functions of the above-derived metrics. In the example, this leads to the trajectory being split into three separate trajectories, of which one only consists of a single message (Figure 5.11b). Since single-message-trajectories cannot sensibly count as such, they get removed automatically. In an attempt to rejoin the two remaining trajectories, the last message of the first trajectory and the first message of the second trajectory are again judged by the split-point procedure. If no split point is found, the two trajectories are rejoined into one (Figure 5.11c).

5.7 Applications of the split-point procedure to data

To demonstrate the effectiveness of the proposed split-point methodology, Figure 5.12 showcases a collection of trajectories derived from segments of the *Øresund* and harbors of Copenhagen in Denmark and Malmö and Helsingborg in Sweden, both with and without the application of the split-point procedure. In the absence of this procedure (as depicted in Figure 5.12a), numerous trajectories exhibit anomalously large distances between consecutive messages. This results in unrealistic jumps in the trajectory paths, leading to instances where the trajectories erroneously appear to traverse over land. This behavior can arise if multiple vessels transmit messages from a single MMSI, which is a common occurrence (Wu et al., 2017; Zhao et al., 2018; Yan et al., 2020a), if trajectories are interrupted by the SOG-filter constraint or if vessels deliberately



(a) Original trajectory. (b) After split-point procedure applied. (c) After re-joining.

Figure 5.11: Exemplary rejoin procedure. The split-point method splits the original, erroneous trajectory (a) into three separate ones, of which one only has a single message (b). Trajectories are judged again by the derived thresholds from the split-point method and possibly get rejoined (c)—illustration inspired by Zhang et al. (2018).

Algorithm 3: Split-point procedure.

Data: Trajectory to split $\mathcal{T}^{>}$, Ship length l, Length-dependent empirical quantile functions for change in speed $\hat{q}_l^{\Delta_{SOG}}$, tuning rate $\hat{q}_l^{ROT}$, time difference $\hat{q}_l^{\Delta_t}$, distance $\hat{q}_l^{dist}$, and reported against calculated speed $\hat{q}_l^{\overline{SOG}-\widehat{SOG}}$, quantile threshold α.

`/* Set the thresholds */`

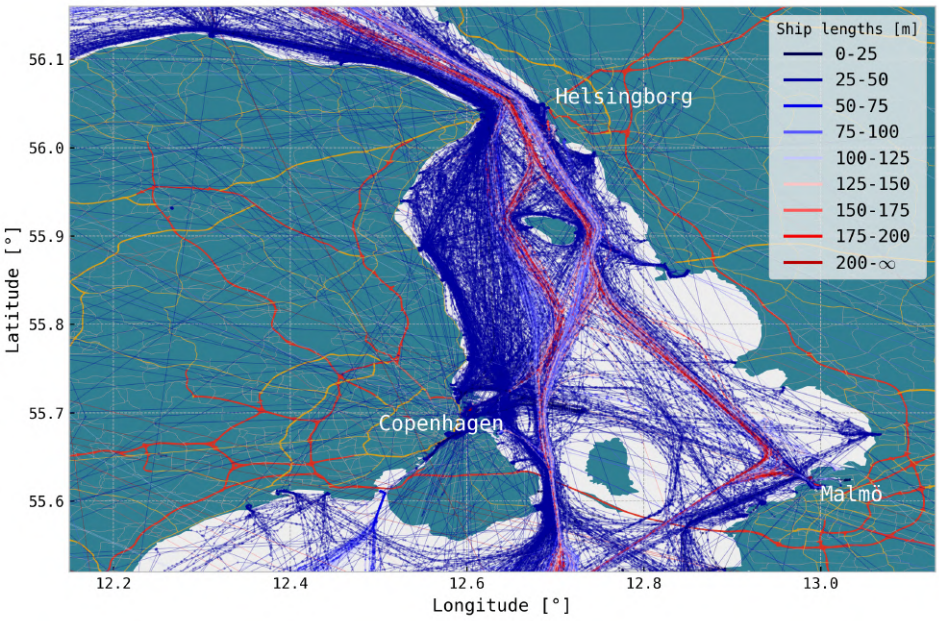
$s \leftarrow \hat{q}_l^{\Delta_{SOG}}(1-\alpha)$
$r \leftarrow [\hat{q}_l^{ROT}(\alpha/2), \hat{q}_l^{ROT}(1-\alpha/2)]$
$t \leftarrow \hat{q}_l^{\Delta_t}(1-\alpha)$
$d \leftarrow \hat{q}_l^{dist}(1-\alpha)$
$b \leftarrow [\hat{q}_l^{\overline{SOG}-\widehat{SOG}}(\alpha/2), \hat{q}_l^{\overline{SOG}-\widehat{SOG}}(1-\alpha/2)]$
for $(m_i, m_{i+1}) \in \mathcal{T}^{>}$
  if $\text{sog}(m_{i+1}) - \text{sog}(m_i) > s$ *(Section 5.6.2)*
    or $\frac{\text{cog}(m_{i+1})-\text{cog}(m_i)}{\text{unix}(m_{i+1})-\text{unix}(m_i)} \notin r$ *(Section 5.6.2)*
    or $\text{unix}(m_{i+1}) - \text{unix}(m_i) > t$ *(Section 5.6.2)*
    or $\text{hav}(m_{i+1}, m_i) > d$ *(Section 5.6.2)*
    or $\overline{SOG}_{m_i}^{m_{i+1}} - \widehat{SOG}_{m_i}^{m_{i+1}} \notin b$ *(Section 5.6.2)*
  then
    Split the trajectory between m_i and m_{i+1}
  end
end

disable their AIS transceiver and re-enable it at a different location. In the filtered version, as depicted in Figure 5.12b, the application of the split-point procedure effectively eliminates these anomalies, thereby accurately reconstructing the underlying structure of the waterway network and, due to the different coloring by the length of the ship, one can also gain valuable insights into the distribution of ship sizes along the waterway, enabling a more comprehensive analysis of traffic patterns and the potential identification of areas with specific navigational challenges or congestion points based on vessel dimensions.Quantitative information about the procedure can be found in Figure 5.12c.

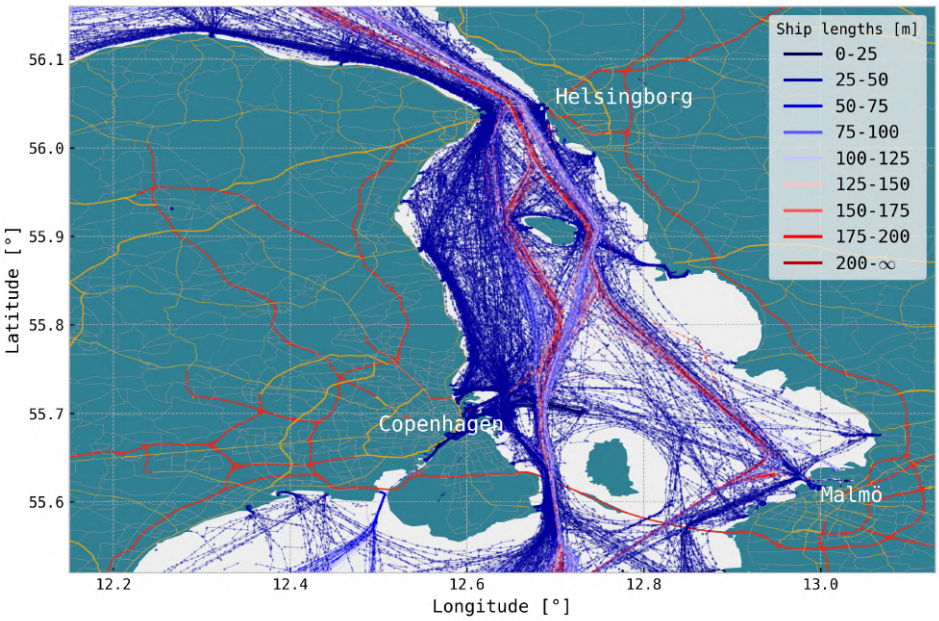
The split-point procedure also allows for significant insights into the navigational behaviors of vessels across different regions and types. This technique is particularly effective in generating ship-type dependent density maps, which illustrate the varying concentrations of vessel traffic along established routes and can also be used to discover new routes. This procedure allows key navigational routes and traffic flow dynamics specific to various ship types to be analyzed effectively. The methodology's utility and effectiveness are demonstrated in the Appendix I in Figures H.16 and H.17, which present detailed density maps of vessel traffic routes in the area of this research.

On an individual trajectory level Figure 5.13 demonstrates the effects of choosing different values of α on an unfiltered trajectory by showing how different values of α impact the number of splits in the trajectory. As α decreases, the trajectory becomes less interrupted with fewer splits. This demonstrates the trade-off between the level of data retention and the degree of realism, highlighting how lower α values result in a more generalized path with increasingly unreliable jumps in the metrics between two consecutive messages. The properties of trajectories extracted with a given value of α are easily interpretable by guaranteeing that all consecutive messages within the trajectory lie inside the navigational bounds of $(1 - \alpha) \cdot 100\%$ of all vessels used to determine the thresholds. This ensures that the extracted trajectories represent common navigational patterns, making them reliable and consistent with the overall traffic flow while filtering out outliers or unusual movements.

Moreover, due to its data-driven nature, the split-point methodology is highly generalizable and can be effectively applied to any geographical region. Whether dealing with complex networks of inland waterways, coastal regions, or open ocean environments, the underlying principles of the split-point procedure remain consistent and adaptable. The flexibility of this approach allows for its application in diverse maritime contexts, making it a valuable tool for global maritime navigation analysis but also as a precursor for algorithmic testing procedures that rely on realistic real-world trajectories. Practitioners can leverage the accompanying Python package, as detailed in Appendix J, to use and adapt Algorithm 3 and generate results similar to those presented, regardless of the geographical region under study. This generalizability ensures that the insights gained from this methodology are not confined to the specific regions of the North Sea but are applicable worldwide.



(a) Trajectories without split-point filtering.



(b) Trajectories after applying the split-point procedure from Section 5.6.2 with $\alpha = 0.03$.

Number of Raw Messages	26 612 047	Percentage of Duplicates	0.46%	Number of Trajectories	722 624
Number of Split-Points	1 444 724	Number of Rejoined Tracks	20 558	Average Trajectory length	4.53 *nm*

(c) Quantitative information on the split-point procedure.

Figure 5.12: Effects of the split-point filter applied to an area in the *Øresund* around Copenhagen in Denmark. Data from 10/08/2024 - 19/08/24 is used.

5.8 Comparison

When comparing trajectory extraction processes from AIS data, it is essential to recognize the inherent limitations in quantitatively evaluating the performance of different methods. As mentioned in the introduction, AIS data, while a valuable source of real-time maritime information,

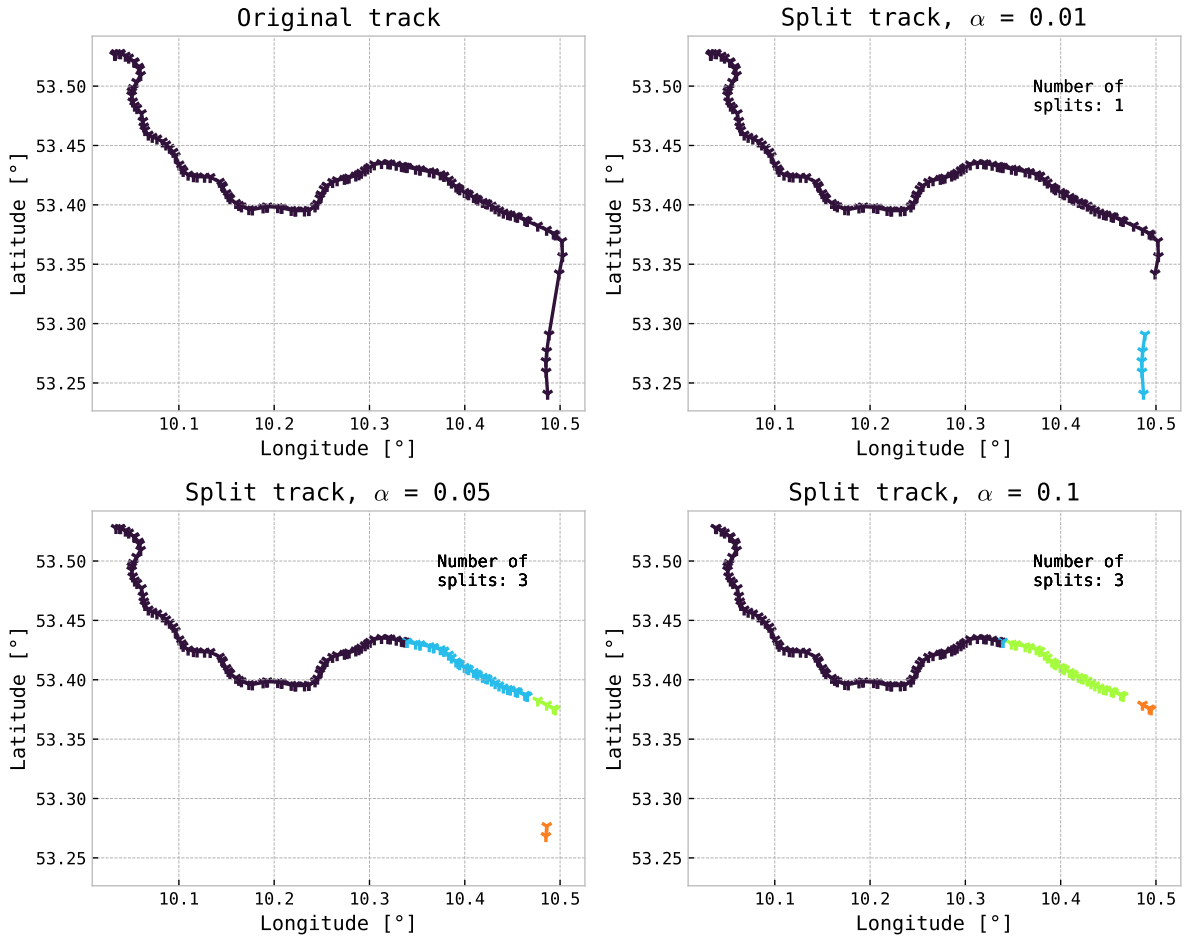


Figure 5.13: Results of our proposed split-point procedure under different values of α, where lower values split less aggressively at the cost of larger margins of accepted values between messages. Differently colored parts of a trajectory indicate sub-trajectories. On the rightmost illustration, the lower-right part of the trajectory gets filtered out entirely as it was decomposed into several single-message trajectories, which automatically get discarded. The most significant positional gap in the original trajectory is $3.11nm$.

is subject to various errors and inconsistencies. These can include inaccuracies in positional data due to GPS errors, data loss from signal gaps, or intentional manipulation of AIS transmissions. Consequently, no definitive *ground truth* can be used as a baseline for comparison.

Given this lack of an absolute reference, direct quantitative comparisons of trajectory extraction methods become problematic. Traditional metrics like accuracy or error rates, which rely on a known correct answer, are not applicable in this context. Instead, comparisons between methods must often be based on qualitative assessments or indirect metrics that reflect the performance of the extraction processes under realistic conditions.

Considering the above restrictions, we compare our approach to two commonly used extraction methods, namely the ones by Zhao et al. (2018) and Guo et al. (2021b). Both are popular approaches, whereby Guo et al. (2021b) is a pure exclusion process that does not split trajectories but removes erroneous messages. Zhao et al. (2018), on the other hand, acts as the base of our proposed framework, as it also splits trajectories. It is to be mentioned that the approach from Guo et al. (2021b) also features an extended trajectory interpolation scheme, which we will not implement in our comparison because neither the approach of Zhao et al. (2018) nor ours interpolate filtered trajectories, rendering the comparison skewed.

For our comparison, we will investigate the maximum jump in metrics such as time, distance, and velocity inside each trajectory, examine the values, and view them in a maritime context. For all compared methods we start with a time-sorted collection of AIS messages coming from one MMSI.

Setup Zhao et al. (2018) uses three steps to filter messages. (1) All messages whose absolute difference between receiving time and sending time is more than $5s$. (2) For a pair of consecutive messages m_1, m_2, inside a trajectory, a split point is defined if $\widehat{SOG}_{m_1}^{m_2} > 15kn$ or $\text{unix}(m_2) - \text{unix}(m_1) > 600s$. (3) All sub-tracks constructed in (2) are rejoined similarly as described in Section 5.6.3, with the exception that only the condition $\widehat{SOG}_{m_1}^{m_2} > 15kn$ is taken for a decision.

The exclusion process of Guo et al. (2021b) consists of two steps. (1) Turning rates are calculated in the same way as in Section 5.6.2, and a turning rate limit Δc_{lim} is defined. If the turning rate of any two messages, m_1 and m_2, is higher than the limit, the message m_2 is discarded. (2) If for any two messages, $\widehat{SOG}_{m_1}^{m_2} > v_{lim}kn$, for a velocity limit v_{lim}, message m_2 is discarded. Unfortunately, the authors did not provide specific values for Δc_{lim} and v_{lim} but rather reference the ship's maneuverability under consideration and the speed limit in the area.

Example trajectory A 223-messages-long trajectory with distinct jumps driven by a $19m$ long fishing vessel, roughly $80km$ off the coast of the Netherlands, is investigated. The grey line in Figure 5.14 shows the raw trajectory. Since there is no speed limit this far off the coast, we set $v_{lim} = 30kn$, and for $c_{lim} = 2°$ which is a bit more the 99-percentile of the turning rate distribution for vessels with lengths $[0, 25)m$ as calculated in Section 5.6.2.

Visually comparing the different approaches in Figure 5.14 reveals that no other approach was able to either effectively filter out or separate the trajectory at the two large spatial jumps that span distances of $37.85nm$ and $24.15nm$ each. As seen in Table 5.7, the approach of Guo et al. (2021b) discarded a single message or 0.45% of the messages, the one of Zhao et al. (2018) discarded 54.71% of the messages mainly due to mismatching sending and receiving timestamps.

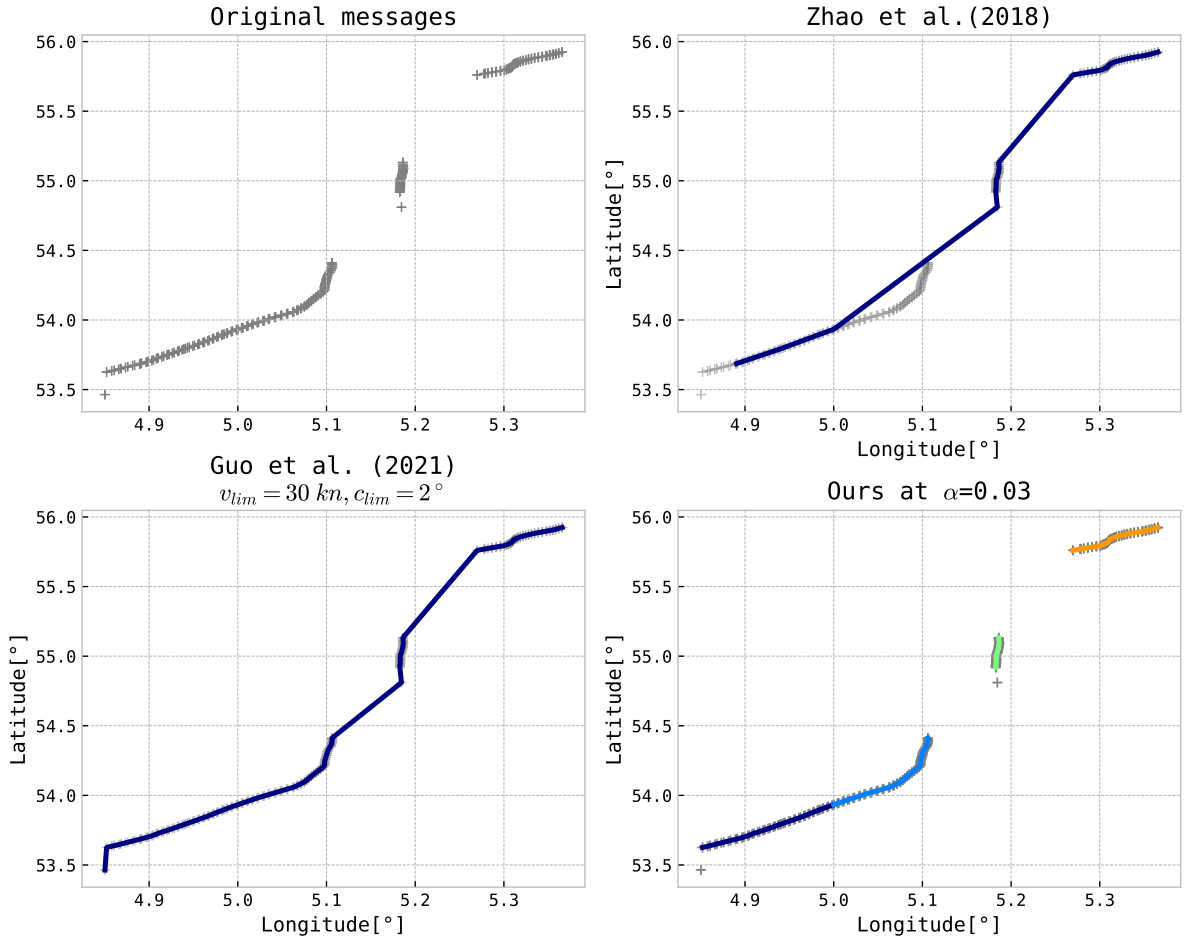


Figure 5.14: Method comparison for different trajectory extraction approaches.

	# of discarded messages	# of split points	max. turning rate $[°/s]$	max. velocity change $[kn/s]$	max. distance $[nm]$
Zhao et al. (2018)	1	0	0.58	0.04	53.06
Guo et al. (2021b)	122	-	5.10	0.10	37.87
Proposed	2	4	0.4	0.04	1.57

Table 5.7: Maximum values attained for consecutive messages inside a trajectory or any of its split sub-trajectories.

No splitting was performed. Guo et al. (2021b) additionally allowed for a quite extensive turning rate of $5.10°/s$.

In contrast, the proposed method demonstrates a balanced approach with minimal message loss and more optimal spatial filtering. As highlighted in Table 5.7, the proposed approach discarded just two messages, introduced four split points to avoid large spatial jumps, and maintained a realistic maximum turning rate of $0.4°/s$. These results indicate that the proposed method achieves accurate and efficient trajectory filtering, ensuring superior spatial consistency and precision.

While this approach demonstrates superior performance in the specific scenario analyzed, it is important to acknowledge that there may be better choices for some applications. The setup process for the proposed method is considerably more complex and time-consuming compared to the more straightforward approaches of Zhao et al. (2018) and Guo et al. (2021b). This complexity arises from the need to pre-compute all threshold values from large amounts of data, which may require substantial computational resources. For scenarios where quick deployment or lower computational overhead is prioritized, the simplicity and ease of setup offered by other methods may outweigh the benefits of our more intricate approach. Thus, while the proposed method achieves high accuracy and precision, its applicability may be limited in contexts with critical ease of use and rapid implementation. Consequently, both methods are also implemented in the accompanying software package.

5.9 Spatial properties of split trajectories

From the approach in Section 5.6.2 we extracted trajectories, each comprising a reliable and consistent series of messages, notably free from anomalies. Nonetheless, we have no tool to assess the spatial properties of the extracted trajectories, especially concerning our original aim of extracting long, clean, and uninterrupted trajectories.

Therefore, we need a tool that measures these three quantities. Upon closer inspection, we find that all trajectories that had undergone the split-point procedure in Section 5.6.2 are already uninterrupted. Addressing the length and cleanliness of trajectories, we adopt a dual approach to control the length and the spatial properties of the trajectories. Consistent with methodologies employed in prior studies (Yuan et al., 2019; Zhao et al., 2018; Mao et al., 2018), the length of the trajectories is quantified using the count of messages per track, n_{msg}. While effective in gauging trajectory length, this metric does not provide insights into the trajectory's shape or spatial extent.

To bridge this gap, we investigate the convex hull each trajectory forms, or more precisely,

its area. Since the calculation of a convex hull's area relies on Euclidean geometry to apply, we first have to project the spheroid space of Earth to a Euclidean space, which is achieved using the Universal Transverse Mercator (UTM) projection (Krüger, 1912). While no isometric map from the sphere to the plane exists, the UTM projection still provides high accuracy if not used directly at the poles (Karney, 2011). Now, let $U = \{u_1, u_2, \ldots, u_n\}$ be the set of UTM-projected positions of a trajectory. The convex hull $C(U)$ is the set of all convex combinations of positions from U. Formally, it is defined as

$$C(U) = \left\{ \sum_{i=1}^{n} \lambda_i u_i \;\middle|\; \lambda_i \geq 0, \sum_{i=1}^{n} \lambda_i = 1 \right\}, \tag{5.5}$$

and in practice, is obtained using the quickhull algorithm by Barber et al. (1996). In two-dimensional space, the convex hull $C(U)$ forms a regular polygon whose area A^C can be easily obtained.

From investigating the number of messages in and the convex hull area of a trajectory, it is not immediately clear how these metrics connect to its *cleanliness*. To measure this influence, this study adapts the idea of *average complexity* by Mao et al. (2018). In its original form, it is defined as

$$\bar{c}(\mathcal{T}_k^{>}) = |\mathcal{T}_k^{>}|^{-1} \sum_{m \in \mathcal{T}_k^{>}} \frac{\mathbf{p}^\top \mathbf{q}}{\|\mathbf{p}\|_2 \|\mathbf{q}\|_2}, \tag{5.6}$$

with

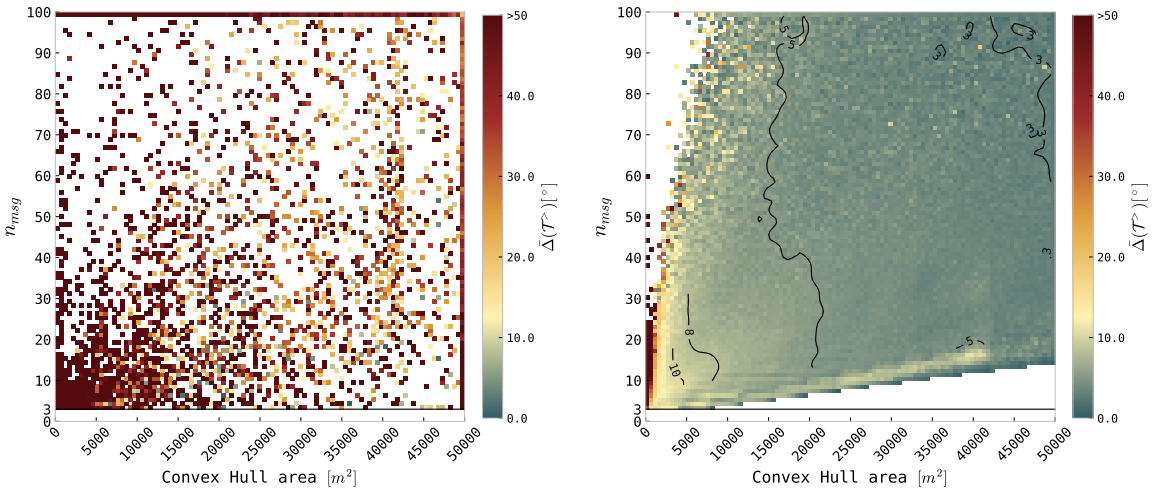
$$\mathbf{p} = \begin{bmatrix} \mathrm{lat}(m_i) - \mathrm{lat}(m_{i-1}) \\ \mathrm{lon}(m_i) - \mathrm{lon}(m_{i-1}) \end{bmatrix}, \quad \mathbf{q} = \begin{bmatrix} \mathrm{lat}(m_{i+1}) - \mathrm{lat}(m_i) \\ \mathrm{lon}(m_{i+1}) - \mathrm{lon}(m_i) \end{bmatrix},$$

which can be interpreted as the average cosine of the angle that the connecting lines of three consecutive messages enclose over the entire trajectory $\mathcal{T}_k^{>}$. The usage of this formula, however, has some drawbacks. In the original paper by Mao et al. (2018), only routes with an average complexity of more than 0.8 were included in the selection of accepted trajectories. We argue that the term complexity in this application is misleading as higher values of *average complexity* correspond to trajectories with smaller average course changes between messages. Additionally, the resulting values can be difficult to interpret in a maritime context as they merely describe a mathematical property of the examined trajectory.

Conveniently, a small adaption allows us to mitigate these shortcomings and construct an easily interpretable metric for assessing the *cleanliness* of a trajectory. We define the *average absolute change of the course* as

$$\bar{\Delta}(\mathcal{T}_k^{>}) = \pi^{-1} \cos^{-1}[\bar{c}(\mathcal{T}_k^{>})] \times 180, \quad \text{for} \quad |\mathcal{T}_k^{>}| > 3, \tag{5.7}$$

which gives us a good intuition about the magnitude of directional changes in one trajectory in degrees. For a value of $\bar{c}(\mathcal{T}_k^{>}) = 0$, i.e., minimum complexity, the average absolute change of the course calculates to 180° reflecting the U-turn behavior, while for $\bar{c}(\mathcal{T}_k^{>}) = 1$ the average absolute change of the course $\bar{\Delta}(\mathcal{T}_k^{>}) = 0°$, encoding a perfectly straight trajectory.



(a) Average values of $\bar{\Delta}(\mathcal{T}^>)$ for unfiltered trajectories.

(b) Average values of $\bar{\Delta}(\mathcal{T}^>)$ for trajectories after split-point filtering.

Figure 5.15: 100×100 pixel map of $\bar{\Delta}(\mathcal{T}^>)$ in degrees as a function of the number of messages per track (n_{msg}) and convex hull area (A^C). Each pixel resembles a bin, showing the average of $\bar{\Delta}(\mathcal{T}^>)$ for all trajectories falling inside it. (a) considers unfiltered trajectories, (b) used the same data but with the split-point procedure applied. White pixels indicate the absence of data. The black line at $n_{\text{msg}} = 3$ is the minimum number of messages a trajectory needs to comprise to calculate $\bar{\Delta}(\mathcal{T}^>)$. The convex hull area is restricted to 5×10^4 for this visualization, as there are no changes in patterns above this threshold.

Figure 5.15 presents a pixel map illustrating the impact of the number of messages (n_{msg}) and the convex hull area (A^C) on $\bar{\Delta}(\mathcal{T}_k^>)$ of trajectories in degrees. This analysis aggregates trajectories from 2021, both before (a) and following (b) the implementation of the split-point procedure, calculating $\bar{\Delta}(\mathcal{T}_k^>)$ for each pixel. Each pixel represents a bin that averages $\bar{\Delta}(\mathcal{T}_k^>)$ for trajectories within specific n_{msg} and A^C ranges. The analysis is limited to $n_{\text{msg}} \leq 100$ and $A^C \leq 5 \times 10^4$ to maintain relevancy, as values beyond these thresholds yield negligible additional insights.

The results indicate that unprocessed trajectories exhibit no discernible pattern, suggesting that n_{msg} and A^C are ineffective metrics for understanding unprocessed trajectory data. Conversely, applying the split-point procedure enhances the diversity of observed routes (fewer white spots) and demonstrates a fast trend towards higher smoothness values, particularly as A^C increases.

5.10 Discussion

This study provides a framework for trajectory extraction from raw, encoded AIVDM sentences. A multi-level filtering and splitting procedure has been proposed to construct long, clean, uninterrupted trajectories. Particular emphasis lies on the data-driven nature of the splitting process of this study. While other approaches (Sang et al., 2015; Zhao et al., 2018; Zhang et al., 2018; Yuan et al., 2019; Chen et al., 2020), too, propose functioning extraction processes, many threshold values used in those studies are still expert-derived, questioning their generalizability beyond

the locally tested data. This study, instead, proposes to generate an empirical cumulative distribution function for the metric under consideration and control the statistical properties of the extracted trajectories freely by using the $1 - \alpha$ quantile bounds to determine the threshold values.

The methodology developed in this study demonstrates effective performance and aligns to establish a self-explanatory, data-driven approach for trajectory extraction. However, it is acknowledged that there exists no specific guidance on the selection of the quantile parameter α, thus the final decision must be carefully considered by the practitioner, especially as many derived thresholds have no specific meaning regarding established AIS transceiving regulations or the maritime domain in general.

5.11 Conclusion

Maritime AIS data is inherently noisy due to technical inaccuracies, seafarer inattention, or lacking compliance. However, trajectories extracted from AIS messages are valuable for maritime surveillance, domain awareness, and algorithmic testing, for which we propose a data-driven, open-source trajectory extraction and refinement framework. The extraction process is based on a split-point procedure that uses quantile boundaries of five empirically derived distribution functions to decide whether two consecutive messages belong to the same trajectory or should be split. We found that this procedure effectively eliminates anomalies in the data without resorting to expert-derived thresholds, making it universally adaptable to any AIS data set. Nevertheless, carefully scrutinizing the quantile boundary and threshold values is recommended, as they may vary depending on data availability. For spatial understanding, we introduced the concept of *average absolute change of course*, $\bar{\Delta}(\cdot)$, which allows us to assess the presence or absence of erratic positional movement in a trajectory and recommended the number of messages per trajectory (n_{msg}) as well as the convex hull area (A^C) as means to control $\bar{\Delta}(\cdot)$. The entire extraction and assessment pipeline, described by this article, is publicly available as an open-source Python package at Paulig (2024).

5.12 Acknowledgments

We thank the Center for Information Services and High-Performance Computing at TU Dresden for providing its facilities for high throughput calculations. We thank the European Maritime Safety Agency for providing the raw AIS records analyzed in this study. Further, we would also like to thank Martin Waltz for his valuable support throughout this project.

H A_v^C for different ship types

Given that $ship\_type(MMSI)$ is a function mapping an MMSI to its corresponding ship type v, and $A^{C_{\mathcal{T}_k^>}}$ is the convex hull area for trajectory $\mathcal{T}_k^>$, the ship-type-average of their convex hull area can be found via

$$\bar{A}_v^C = |\mathcal{V}|^{-1} \sum_{\mathcal{T}_k^> \in \mathcal{V}} A^{C_{\mathcal{T}_k^>}}, \tag{H.1}$$

where $\mathcal{V} = \{\mathcal{T}_k^> | ship\_type(k) = v\}$ is the set of all trajectories belonging to one ship type.

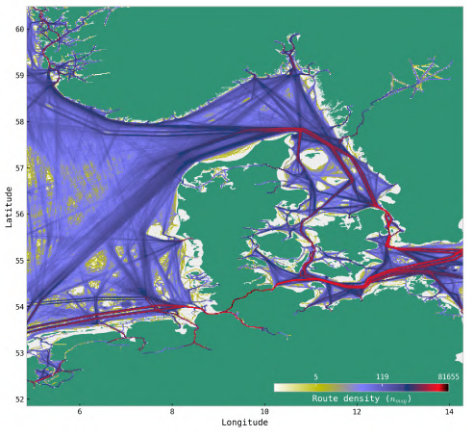
v	$\bar{A}_v^C$ $[m^2]$
CARGO	174 100
TANKER	173 443
NOTAVAILABLE	141 221
TUGTOW	126 509
PASSENGER	115 021
MILITARY	109 111
OTHER	99 771
PLEASURE	93 902
SAILING	79 730
WIG	78 384
FISHING	60 610
HSC	37 618

Table H.8: Average convex hull area for all trajectories with $n_{\text{msg}} > 50$ for all ship types for the year 2021. The ship types are specified according to ITU standards (ITU, 2001). For this analysis, only the base ship type had been distinguished; for example, any tanker (types 80 - 89) is included in the TANKER category. NOTAVAILABLE refers to the default if no ship type has been transmitted. OTHER are all ships not fitting into the above categories.

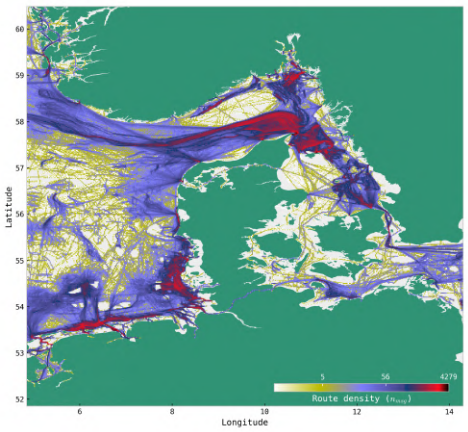
Table H.8 collects these averages. In line with our intuition, cargo ships' routes cover the largest average area of all ship types, as one of their primary goals is to ship goods effectively over long distances. Interestingly, Tug and tow boats cover the 3rd highest convex hull area. This finding might initially seem paradoxical given their role in assistive navigation in ports or terminals, typically associated with short and compressed trajectories. However, this observation can be rationalized by considering that our analysis only includes trajectories with more than 50 messages and a minimum speed of $1kn$. A visual inspection using heatmaps in Figures H.16 and H.17 revealed that Tug and Towboats travel longer routes than expected under these circumstances. On the other hand, High-Speed-Crafts (HSC) are primarily used as high-speed ferries for short passages in commercial contexts, which explains their relatively small area covered.

I Time scale of empirical distributions

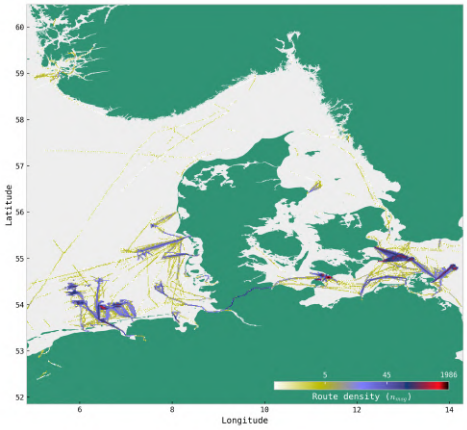
The split-point methodology outlined in Section 5.6.2 requires the computation of five distinct empirical distributions, using their quantiles to establish thresholds. In this process, we explicitly used trajectory data from one year of AIS records as the basis for each distribution. To



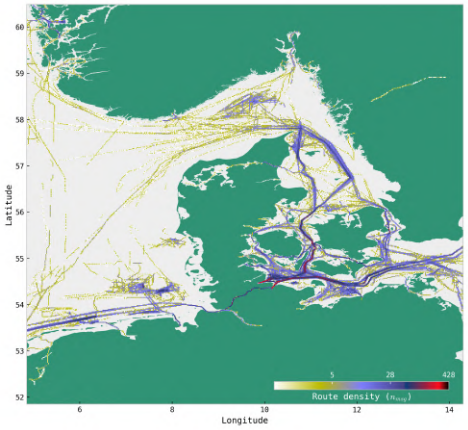
(a) Cargo



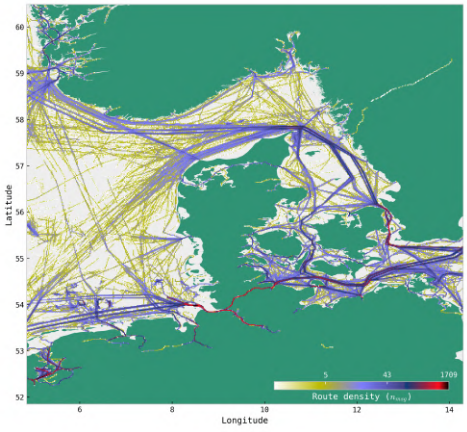
(b) Fishing



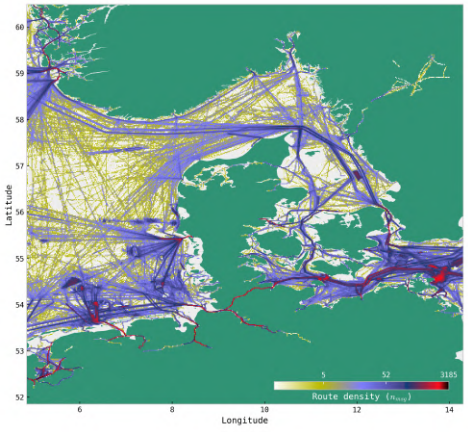
(c) High Speed Crafts



(d) Military

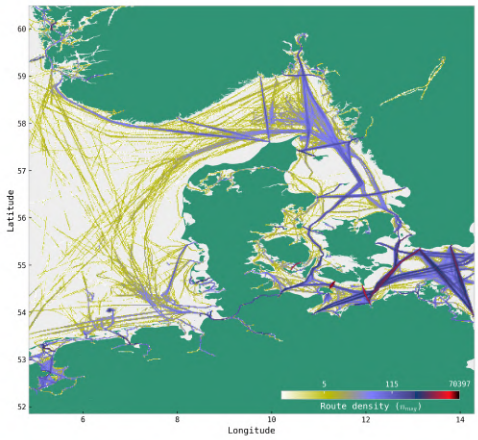


(e) Not available

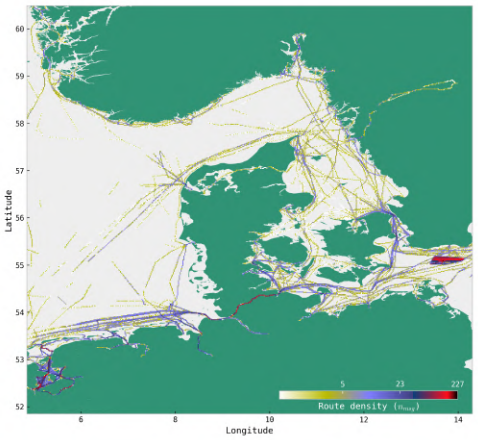


(f) Other

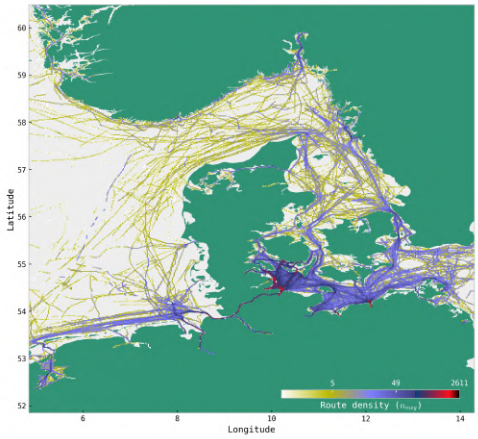
Figure H.16: Heatmap of route densities for different ship types for the year 2021. Plots generated from raw AIS records using the split-point method and $A^C > 3 \times 10^4 m^2$ and $n_{\text{msg}} > 50$-refinement



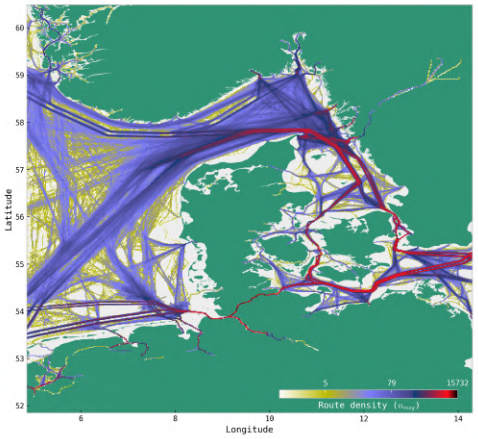
(a) Passenger



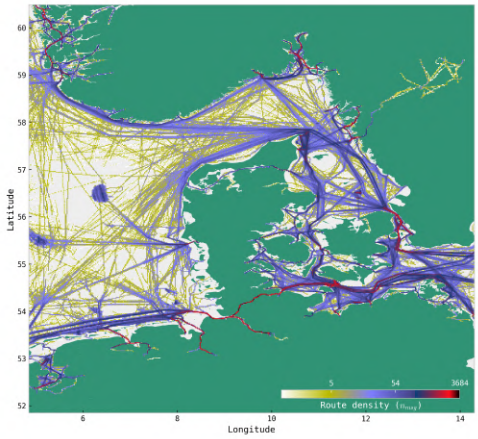
(b) Pleasure



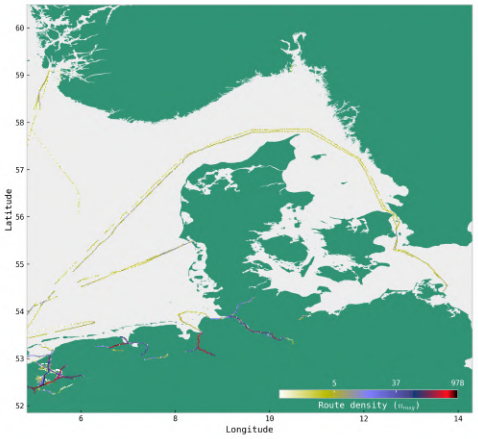
(c) Sailing



(d) Tanker



(e) Tug Tow



(f) Wing in Ground (WIG)

Figure H.17: Continuation of Figure H.16

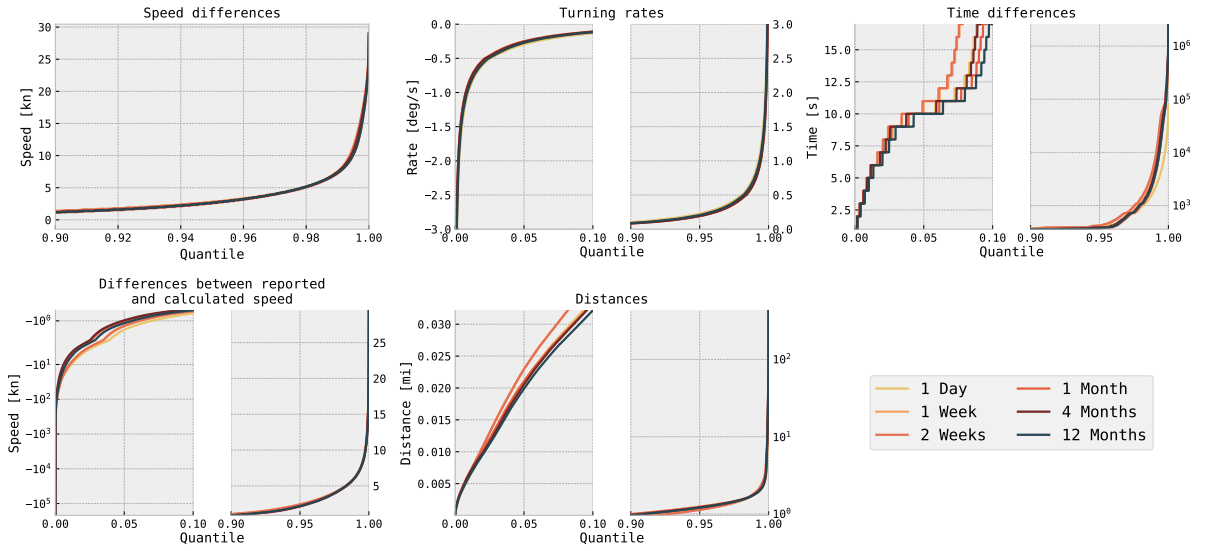


Figure I.18: Comparison of quantile functions for the split-points metrics using different amounts of data. Please note the different ordinate and abscissa scalings.

investigate how much data is needed to accurately calculate the empirical distribution functions I.18 displays the (ship-length-independent) quantile functions for all metrics, each calculated using different amounts of trajectory data (given as days and months). Visually, these distributions appear somewhat similar, leading to the assumption that the data follows identical distributions over various time horizons.

To substantiate this visual assumption, we compare the quantiles calculated from various time horizons against each other using the procedure outlined in Wilcox et al. (2014) using 10^6 samples and 1000 bootstrap replications each. We deliberately chose this setup to test how much information is pertained when using less data compared to one-year data. Contrary to initial expectations, our tests indicate that the distributions of different time horizons differ.

The computed quantile values in Table I.9 only show minor differences in absolute terms, which leaves leeway to the choice of the amount of data used to calculate the quantiles. However, due to the huge sample sizes (over 10^6 observations) that lead to very narrow confidence intervals, almost all quantiles are tested statistically differently. Ultimately, we use one year of data throughout the study to maximize generalizability.

Nonetheless, since variations may emerge for distributions derived from shorter or longer time spans, additional adjustments by practitioners are potentially necessary.

Metric	Quantiles used in Section 5.6.2 Upper and lower quantiles							
	$q = 0.025$				$q = 0.975$			
	$D = 1$	$D = 7$	$D = 30$	$D = 120$	$D = 1$	$D = 7$	$D = 30$	$D = 120$
Difference between reported and calculated speed $[kn]$	-3.577	-3.149	-2.866	-4.078	4.041	4.080	4.053	4.334
Turning rates $[°/s]$	-0.507	-0.466	-0.466	-0.491	0.421	0.376	0.372	0.400
					Only upper quantiles			
					$q = 0.95$			
					$D = 1$	$D = 7$	$D = 30$	$D = 120$
Time difference between messages $[s]$					397.0	420.0	430.0	419.0
Speed changes $[m/s]$					2.800	2.800	2.799	2.700
Distance between consecutive messages $[nm]$					1.147	1.142	1.110	1.123

Table I.9: Quantile values for the metrics used in Section 5.6.2. q is the value of the quantile, D is the number of days of data used to obtain the quantiles.

J Python package implementation details

All pre-processing, filtering, extracting, splitting, assessment and visualization procedures outlined in this article are available as the open-source Python package *PyTSA* (**Py**thon **T**rajectory **S**plitting and Assessment **A**gent, Paulig (2024)). This Section is intended to provide implementation details and practitioners guidelines on how to use *PyTSA* to reproduce the results presented in this research. For installation instructions, please refer to the package documentation.

Decoding raw AIS messages If the data to be analyzed is present as raw `AVIDM/AVIDO` sentences, they must first be decoded. To do this with *PyTSA*, the input files must be provided as `.csv` files sorted such that one group of files only contains dynamic messages of types 1/2/3/18 and the other only messages of type 5 (static voyage reports). Both files must be named identically by the day for which they hold AIS messages, separated by underscores ("YYYY_MM_DD.csv").

Subsequently, decoding can be performed via

```
from pytsa import decode

decode(
    source = "path/to/raw_dir",
    dest = "path/to/decoded_dir",
    njobs = 1
)
```

which will take all `.csv` files in the "source" folder, decode them, and store the decoded files in the `.csv` format in the "dest" folder. For more information on the "njobs" keyword and the structure of the data files, the user is deferred to the package documentation.

```python
import pandas as pd
import pytsa

from pathlib import Path

# Column name for SOG in the decoded
# csv data.
_speed = ...

# Filter out all vessels that are slower than 1 knot
# and faster than 30 knots.
# Any pre-processor is mandated to take in a single
# pandas DataFrame and return a single pandas DataFrame
def speed_filter(df: pd.DataFrame) -> pd.DataFrame:
    return df[(df[_speed] > 1) & (df[_speed] < 30)]

# Bounding Box for the area of study in this research
frame = pytsa.BoundingBox(
    LATMIN = 52.2, # [°N]
    LATMAX = 56.9, # [°N]
    LONMIN = 6.3,  # [°E]
    LONMAX = 9.5,  # [°E]
)
# This can be either a single file or a
# list of files
dynamic_data = Path("/path/to/dynamic.csv")
static_data = Path("/path/to/static.csv")

search_agent = pytsa.SearchAgent(
    msg12318file = dynamic_data,
    msg5file = static_data
    frame = frame,
    preprocessor = speed_filter # Apply the speed filter using the "preprocessor" keyword
)
```

Code Listing J.1: Basic instantiantion of the `SearchAgent` class with a custom pre-processing function.

Trajectory extraction using the `SearchAgent` *PyTSA*'s central object, the `SearchAgent`, provides a convenient tool for constructing, filtering, and splitting trajectories extracted from the messages decoded earlier. The `SearchAgent` class is instantiated using at least three components:

1. `mgs12318file` - A single file path or a list of file paths for dynamic messages
2. `mgs5file` - A single file path or a list of file paths for static messages
3. `frame` - A *PyTSA* `BoundingBox` object determining the spatial extent of the extraction process.

If, instead, it is desired for the data to undergo a pre-processing step, a pre-processing function that all messages must pass through before extraction can be defined. For example, to apply the speed constraint from Section 5.5.2, the instantiation can be performed as described in Code Listing J.1. After creating the instance, the `extract_all()` method can be used to construct and, if desired, also split the trajectories according to the split-point procedure from Section 5.6.2. The method returns a Python dictionary with every unique MMSI as keys and

their corresponding `TargetShip` objects as values. These objects contain information about the ship type and length and all trajectories identified as belonging to this MMSI. The method has an integrated switch to turn the split-point procedure on or off. If turned off, every returned `TargetShip` only contains a single trajectory, consisting of all AIS messages sent from its MMSI sorted by time.

We look at the code used to produce Figure 5.12 to demonstrate this approach's capabilities. We also utilize the visualization module shipped with *PyTSA* for this approach.

```
#
# This listing continues Code Listing J.1
#
# To perform only trajectory construction without
# using the split-point procedure, we use the
# `skip_tsplit=True` option.
ships = search_agent.extract_all(njobs=16,skip_tsplit=True)

# Define the area around Aabenraa
AABENRAA = pytsa.BoundingBox(
    LATMIN = 54.9, # [°N]
    LATMAX = 55.5, # [°N]
    LONMIN = 9.2,  # [°E]
    LONMAX = 10.0,  # [°E]
)

# Plot trajectories on the map for the area
# around AABENRAA
plot_trajectories_on_map(
    ships = ships,
    extent = AABENRAA
)

# With split-point procedure applied
ships = search_agent.extract_all(njobs=4,skip_tsplit=False)
plot_trajectories_on_map(
    ships = ships,
    extent = AABENRAA
)
```

Assessing spatial properties of extracted trajectories With the help of the `SearchAgent` class, we could easily construct and split trajectories from raw AIS messages. The spatial assessment from Section 5.9 is implemented into *PyTSA* via the `Inspector` class, which filters the extracted trajectories based on a set of user-defined `Rules`. All `Rules` must have a function signature of `def my_rule(track: Track) -> bool`, where `Track = list[AISMessage]` is a list of AIS message objects defined in `pytsa.structs`. `Rules` must be defined such that they take in a single trajectory and return `True` if the trajectory is to be *rejected*. `Rules` must be combined to create a recipe given to the inspector for trajectory assessment. To showcase the

functionality of the `Inspector` class and its usage with pre-defined `Rules`, we reconstruct the heatmaps from Figures H.16 and H.17 in the following code example.

```
#
# This listing continues Code Listing J.1
#
# Construct and split trajectories in parallel
ships = search_agent.extract_all(njobs=4)

# Construct a Recipe object to assess every trajectory based on the rule functions
# given to it. In this case, we use the partial class to coerce the rules to only take a single argument,
# which is the trajectory given to it.
assessment = Recipe(
    partial(too_few_obs, n = 50),
    partial(convex_hull_area, area = 3e5)
)

# Instantiate the Inspector and split the trajectories into accepted and rejected ones.
# In this example, the rejected trajectories are discarded.
accepted, rejected = pytsa.Inspector(data = ships, recipe = assessment).inspect()

# Save a 500x500 px heatmap for every ship type
types = [t for t in pytsa.ShipType]
names = [t.name for t in types]
expanded = []
for t in types:
    if isinstance(t.value, int):
        expanded.append([t.value])
    else:
        expanded.append(list(t.value))
for i,t in enumerate(expanded):
    a = {mmsi:s for mmsi,s in accepted.items() if any(st in t for st in s.ship_type)}
    binned_heatmap(targets = a, bb = frame, npixels = 500, title = f"Heatmap for {names[i]}")

```

Part III

Conclusion

Chapter 6

Conclusion

This thesis investigates the potential of deep reinforcement learning (DRL), a rapidly advancing branch of machine learning, and large-scale data processing techniques to guide and control inland vessels. By fusing the function-approximation power of neural networks with the trial-and-error learning paradigm of reinforcement learning, DRL excels at solving complex sequential decision problems. Through rigorous, real-world-data testing, we demonstrate that DRL offers a robust framework for tackling autonomous control challenges in the open, two-dimensional maritime environment.

The first two papers in this thesis lay the groundwork for safely navigating vessels through the complex topology and fluid dynamics of inland waterways. In the first paper, we tackle precise path-following, a task often oversimplified in the literature by ignoring shallow-water effects and strong directional currents. We achieve substantially lower path-tracking errors by deploying a state-of-the-art bootstrapped DRL algorithm than conventional PID controllers, even when currents and depth variations significantly distort vessel motion. The controller was evaluated on real bathymetry data from both the lower and middle Rhine to demonstrate robustness and geographic generality. Building on these results, the second paper elevates the challenge to a combined planning-and-following problem. We introduce a hierarchical DRL framework that dynamically replans waypoints to avoid obstacles under maritime traffic regulations, while a spatial-temporal recurrent policy outputs continuous rudder commands. This two-level system remains effective under high traffic densities and strict rule compliance. Looking ahead, we identify two key extensions: (1) incorporating inter-vessel communication via a multi-agent DRL formulation mirroring real-world practices where pilots share intentions to coordinate maneuvers, and (2) closing the "sim-to-real" gap by deploying these algorithms on physical vessels, starting in controlled testing basins and ultimately scaling to full-size ships. This progression will expose practical challenges and accelerate the translation of DRL-based ship control into operational waterways.

The third paper examines how trajectory extraction from large-scale maritime AIS streams can support algorithm training and surveillance. Prior work made strides in filtering erroneous data, but typically relied on expert-chosen threshold metrics that ignore vessel size and maneuverability differences. We introduce a decision algorithm incorporating vessel dimensions as a proxy for maneuverability, yielding a fully data-driven trajectory extraction pipeline in-

dependent of hand-tuned thresholds. Future extensions could fuse this pipeline with advanced classical or ML-based interpolation that likewise accounts for maneuverability, creating a more complete maritime surveillance system resilient to missing data.

In conclusion, this thesis makes two key contributions: first, it demonstrates how deep reinforcement learning can be employed for low-level control and guidance of inland vessels, showing that DRL agents can safely, efficiently, and comfortably maneuver through complex traffic conditions. Second, it presents a streamlined approach for preprocessing extensive maritime AIS datasets, proving that vessel-informed cleaning yields clear, interpretable data ideal for surveillance and algorithm testing. By validating DRL in real-world waterway scenarios and enhancing the quality of AIS data, this work points to a future in which intelligent, data-driven strategies reshape maritime transportation.

Bibliography

Akdağ, M., Solnør, P., and Johansen, T. A. (2022). Collaborative collision avoidance for maritime autonomous surface ships: A review. *Ocean Engineering*, 250:110920.

Al Enezy, O., van Hassel, E., Sys, C., and Vanelslander, T. (2017). Developing a cost calculation model for inland navigation. *Research in Transportation Business & Management*, 23:64–74.

Amendola, J., Miura, L. S., Costa, A. H. R., Cozman, F. G., and Tannuri, E. A. (2020). Navigation in restricted channels under environmental conditions: Fast-time simulation by asynchronous deep reinforcement learning. *IEEE Access*, 8:149199–149213.

Amendola, J., Tannuri, E. A., Cozman, F. G., and Reali Costa, A. H. (2019). Port channel navigation subjected to environmental conditions using reinforcement learning. In *International Conference on Offshore Mechanics and Arctic Engineering*, volume 58844, page V07AT06A042. American Society of Mechanical Engineers.

Amigo Herrero, D., Sanchez Pedroche, D., Garcia Herrero, J., and Molina Lopez, J. M. (2019). Ais trajectory classification based on imm data. In *2019 22ND INTERNATIONAL CONFERENCE ON INFORMATION FUSION (FUSION 2019)*. Int Soc Informat Fus; Thales Grp; Syst & Technol Res; Univ Windsor. 22nd International Conference on Information Fusion (FUSION), Ottawa, CANADA, JUL 02-05, 2019.

Amin, O. M. and Hasegawa, K. (2010). Generalised mathematical model for ship manoeuvrability considering shallow water effect. In *Conference Proc*, volume 10, pages 531–534. of Japan Society of Naval Architects and Ocean Engineers.

Ankudinov, V., Miller, E., Jakobsen, B., and Daggett, L. (1990). Manoeuvring performance of tug/barge assemblies in restricted waterways. *Proceedings MARSIM & ICMS*, 90:515–525.

Annamalai, A. S., Sutton, R., Yang, C., Culverhouse, P., and Sharma, S. (2015). Robust adaptive control of an uninhabited surface vehicle. *Journal of Intelligent & Robotic Systems*, 78:319–338.

Bačkalov, I., Vidić, M., and Rudaković, S. (2023). Lessons learned from accidents on some major european inland waterways. *Ocean Engineering*, 273:113918.

Bai, Y., Jones, A., Ndousse, K., Askell, A., Chen, A., DasSarma, N., Drain, D., Fort, S., Ganguli, D., Henighan, T., et al. (2022). Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Bailey, N. J. (2005). Training, technology and ais: looking beyond the box. *Proceedings of the Seafarers International Research Centre's Fourth International Symposium.*

Barber, C. B., Dobkin, D. P., and Huhdanpaa, H. (1996). The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, 22(4):469–483.

Baroud, H., Barker, K., Ramirez-Marquez, J. E., et al. (2014). Importance measures for inland waterway network resilience. *Transportation research part E: logistics and transportation review*, 62:55–67.

Bedoya-Maya, F., Shobayo, P., Beckers, J., and van Hassel, E. (2024). The impact of critical water levels on container inland waterway transport. *Transportation Research Part D: Transport and Environment*, 131:104190.

Bellemare, M. G., Candido, S., Castro, P. S., Gong, J., Machado, M. C., Moitra, S., Ponda, S. S., and Wang, Z. (2020). Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588(7836):77–82.

Bellman, R. (1954). The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–515.

Bellman, R. (1957). Dynamic programming, princeton univ. *Press Princeton, New Jersey.*

Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., et al. (2019). Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680.*

BIS Research (2018). Global autonomous ship and ocean surface robot market: Analysis and forecast, 2018-2028. `https://www.whatech.com/markets-research/it/archive/521945-global-ocean-surface-robot-market-anticipated-to-reach-2-90-billion-by-2028-at-a-cagr-of-16-8-and-global-autonomous-ship-market-expected-to-generate-a-cumulative-revenue-of-3-48-billion-by-2035-bis-research-report` [Accessed: 2022-04-21].

Blindheim, S. and Johansen, T. A. (2021). Electronic navigational charts for visualization, simulation, and autonomous ship control. *IEEE Access*, 10:3716–3737.

Botev, Z., Grotowski, J., and Kroese, D. (2010). Kernel density estimation via diffusion. *The Annals of Statistics*, pages 2916–2957.

Breivik, M. and Fossen, T. I. (2004). Path following of straight lines and circles for marine surface vessels. *IFAC Proceedings Volumes*, 37(10):65–70.

Breivik, M. and Fossen, T. I. (2009). Guidance laws for autonomous underwater vehicles. *Underwater vehicles*, 4:51–76.

Breu, D. A. and Fossen, T. I. (2011). L1 adaptive and extremum seeking control applied to roll parametric resonance in ships. In *2011 9th IEEE International Conference on Control and Automation (ICCA)*, pages 871–876. IEEE.

Bundesministerium für Digitales und Verkehr (1998). *Seeschiffahrtsstraßen-Ordnung*.

Calderón-Rivera, N., Bartusevičienė, I., and Ballini, F. (2024). Sustainable development of inland waterways transport: a review. *Journal of Shipping and Trade*, 9(1):3.

Camargo-Díaz, C. P., Paipa-Sanabria, E., Zapata-Cortes, J. A., Aguirre-Restrepo, Y., and Quiñones-Bolaños, E. E. (2022). A review of economic incentives to promote decarbonization alternatives in maritime and inland waterway transport modes. *Sustainability*, 14(21):14405.

Cao, S., Fan, P., Yan, T., Xie, C., Deng, J., Xu, F., and Shu, Y. (2022). Inland waterway ship path planning based on improved RRT algorithm. *Journal of Marine Science and Engineering*, 10(10):1460.

Capobianco, S., Millefiori, L. M., Forti, N., Braca, P., and Willett, P. (2021). Deep learning methods for vessel trajectory prediction based on recurrent neural networks. *IEEE Transactions on Aerospace and Electronic Systems*, 57(6):4329–4346.

Castañeda, H., Rodriguez, J., and Gordillo, J. L. (2021). Continuous and smooth differentiator based on adaptive sliding mode control for a quad-rotor mav. *Asian Journal of Control*, 23(2):661–672.

Chen, C., Chen, X.-Q., Ma, F., Zeng, X.-J., and Wang, J. (2019). A knowledge-free path planning approach for smart ships based on reinforcement learning. *Ocean Engineering*, 189:106299.

Chen, L., Negenborn, R. R., and Lodewijks, G. (2016). Path planning for autonomous inland vessels using A* BG. In *International Conference on Computational Logistics*, pages 65–79. Springer.

Chen, P., Shi, G., Liu, S., and Gao, M. (2018). Pattern knowledge discovery of ship collision avoidance based on ais data analysis. *International Journal of Performability Engineering*, 14(10):2449.

Chen, X., Ling, J., Yang, Y., Zheng, H., Xiong, P., Postolache, O., and Xiong, Y. (2020). Ship trajectory reconstruction from ais sensory data via data quality control and prediction. *Mathematical Problems in Engineering*, 2020:1–9.

Cheng, Y. and Zhang, W. (2018). Concise deep reinforcement learning obstacle avoidance for underactuated unmanned marine vessels. *Neurocomputing*, 272:63–73.

Chu, T., Wang, J., Codecà, L., and Li, Z. (2019). Multi-agent deep reinforcement learning for large-scale traffic signal control. *IEEE transactions on intelligent transportation systems*, 21(3):1086–1095.

Chun, D.-H., Roh, M.-I., Lee, H.-W., Ha, J., and Yu, D. (2021). Deep reinforcement learning-based collision avoidance for an autonomous ship. *Ocean Engineering*, 234:109216.

Dang, X.-K., Tran, T.-D., Tran, M.-H., and Pham, T. D.-A. (2022). Inland waterway transport in vietnam: Strategies to improve transportation efficiency during covid-19 pandemic. In *IOP Conference Series: Earth and Environmental Science*, volume 1072, page 012006. IOP Publishing.

de Barros, B. R. C., de Carvalho, E. B., and Junior, A. C. P. B. (2022). Inland waterway transport and the 2030 agenda: Taxonomy of sustainability issues. *Cleaner Engineering and Technology*, page 100462.

Defryn, C., Golak, J. A. P., Grigoriev, A., and Timmermans, V. (2021). Inland waterway efficiency through skipper collaboration and joint speed optimization. *European journal of operational research*, 292(1):276–285.

Degrave, J., Felici, F., Buchli, J., Neunert, M., Tracey, B., Carpanese, F., Ewalds, T., Hafner, R., Abdolmaleki, A., de Las Casas, D., et al. (2022). Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419.

Ding, F., Zhang, Z., Fu, M., Wang, Y., and Wang, C. (2018). Energy-efficient path planning and control approach of USV based on particle swarm optimization. In *OCEANS 2018 MTS/IEEE Charleston*, pages 1–6.

Directorate-General for Mobility and Transport (2023). Inland waterways.

Donha, D. C., Desanj, D., Katebi, M., and Grimble, M. (1998). H_∞ adaptive controllers for auto-pilot applications. *International Journal of Adaptive Control and Signal Processing*, 12(8):623–648.

Drożdż, W., Miśkiewicz, R., and Pomianowski, A. (2023). Modern types of propulsion for inland waterway transportas a response to contemporary challenges in the logistics chain across polish seaports. *Sustainability*, 15(21):15254.

Duan, H., Ma, F., Miao, L., and Zhang, C. (2022). A semi-supervised deep learning approach for vessel trajectory classification based on ais data. *OCEAN & COASTAL MANAGEMENT*, 218.

Duan, S., Yu, G., Xing, H., and Wu, Z. (2010). Inland waterway transport in china: Situation and problems. In *ICLEM 2010: Logistics For Sustained Economic Development: Infrastructure, Information, Integration*, pages 379–385.

Eberhart, R. C. and Shi, Y. (2000). Comparing inertia weights and constriction factors in particle swarm optimization. In *Proceedings of the 2000 congress on evolutionary computation. CEC00 (Cat. No. 00TH8512)*, volume 1, pages 84–88. IEEE.

E.U. Copernicus Marine Service (2023a). Global ocean 1/12° physics analysis and forecast updated daily. `https://doi.org/10.48670/moi-00016`. Accessed: 2023-04-17.

E.U. Copernicus Marine Service (2023b). Global ocean daily gridded sea surface winds from scatterometer. `https://doi.org/10.48670/moi-00182`. Accessed: 2023-04-17.

E.U. Copernicus Marine Service (2023c). Global ocean waves analysis and forecast. `https://doi.org/10.48670/moi-00017`. Accessed: 2023-04-17.

Fan, Y., Sun, Z., and Wang, G. (2022). A novel reinforcement learning collision avoidance algorithm for USVs based on maneuvering characteristics and COLREGs. *Sensors*, 22(6):2099.

Fossen, T. I. (2021). *Handbook of Marine Craft Hydrodynamics and Motion Control, 2nd Edition*. John Wiley & Sons.

Fossen, T. I., Breivik, M., and Skjetne, R. (2003). Line-of-sight path following of underactuated marine craft. *IFAC proceedings volumes*, 36(21):211–216.

Fossen, T. I. and Lekkas, A. M. (2017). Direct and indirect adaptive integral line-of-sight path-following controllers for marine craft exposed to ocean currents. *International journal of adaptive control and signal processing*, 31(4):445–463.

Fossen, T. I. and Pettersen, K. Y. (2014). On uniform semiglobal exponential stability (USGES) of proportional line-of-sight guidance laws. *Automatica*, 50(11):2912–2917.

Fujimoto, S., Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR.

Gan, L., Yan, Z., Zhang, L., Liu, K., Zheng, Y., Zhou, C., and Shu, Y. (2022). Ship path planning based on safety potential field in inland rivers. *Ocean Engineering*, 260:111928.

Gonzalez-Garcia, A., Castañeda, H., and Garrido, L. (2020). Usv path-following control based on deep reinforcement learning and adaptive control. In *Global Oceans 2020: Singapore–US Gulf Coast*, pages 1–7. IEEE.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Gu, Q., Zhen, R., Liu, J., and Li, C. (2023). An improved rrt algorithm based on prior ais information and dp compression for ship path planning. *Ocean Engineering*, 279:114595.

Guo, S., Mou, J., Chen, L., and Chen, P. (2021a). An anomaly detection method for ais trajectory based on kinematic interpolation. *Journal of Marine Science and Engineering*, 9(6):609.

Guo, S., Mou, J., Chen, L., and Chen, P. (2021b). Improved kinematic interpolation for ais trajectory reconstruction. *Ocean Engineering*, 234:109256.

Guo, S., Zhang, X., Zheng, Y., and Du, Y. (2020). An autonomous path planning model for unmanned ships based on deep reinforcement learning. *Sensors*, 20(2):426.

Ha, J., Roh, M.-I., and Lee, H.-W. (2021). Quantitative calculation method of the collision risk for collision avoidance in ship navigation using the cpa and ship domain. *Journal of Computational Design and Engineering*, 8(3):894–909.

Han, Q., Yang, X., Song, H., and Du, W. (2022). Multi-objective ship path planning using non-dominant relationship-based WOA in marine meteorological environment. *Ocean Engineering*, 266:112862.

Hansen, P. N., Enevoldsen, T. T., Papageorgiou, D., and Blanke, M. (2022). Autonomous navigation in confined waters-a colregs rule 9 compliant framework. *IFAC-PapersOnLine*, 55(31):222–228.

Harati-Mokhtari, A., Wall, A., Brooks, P., and Wang, J. (2007). Automatic identification system (ais): data reliability and human error implications. *the Journal of Navigation*, 60(3):373–389.

Hart, F. and Okhrin, O. (2023). Enhanced method for reinforcement learning based dynamic obstacle avoidance by assessment of collision risk. *Neurocomputing*, page 127097.

Hart, F. and Okhrin, O. (2024). Enhanced method for reinforcement learning based dynamic obstacle avoidance by assessment of collision risk. *Neurocomputing*, 568:127097.

Hart, F., Okhrin, O., and Treiber, M. (2023a). Vessel-following model for inland waterways based on deep reinforcement learning. *Ocean Engineering*, 281:114679.

Hart, F., Waltz, M., and Okhrin, O. (2023b). Two-step dynamic obstacle avoidance. *arXiv preprint arXiv:2311.16841*.

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107.

He, Y. K., Zhang, D., Zhang, J., Zhang, M., and Li, T. (2019). Ship route planning using historical trajectories derived from ais data. *TransNav, International Journal on Marine Navigation and Safety od Sea Transportation*, 13(1):69–76.

Heiberg, A., Larsen, T. N., Meyer, E., Rasheed, A., San, O., and Varagnolo, D. (2022). Risk-based implementation of COLREGs for autonomous surface vehicles using deep reinforcement learning. *Neural Networks*, 152:17–33.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hofbauer, F. and Putz, L.-M. (2020). External costs in inland waterway transport: An analysis of external cost categories and calculation methods. *Sustainability*, 12(14):5874.

Howard, R. A. (1960). *Dynamic programming and markov processes.* John Wiley.

Huang, Y., Chen, L., Chen, P., Negenborn, R. R., and Van Gelder, P. (2020). Ship collision avoidance methods: State-of-the-art. *Safety science*, 121:451–473.

Hunt, J. D., Pokhrel, Y., Chaudhari, S., Mesquita, A. L. A., Nascimento, A., Leal Filho, W., Biato, M. F., Schneider, P. S., and Lopes, M. A. (2022). Challenges and opportunities for a south america waterway system. *Cleaner Engineering and Technology*, 11:100575.

Ibarz, J., Tan, J., Finn, C., Kalakrishnan, M., Pastor, P., and Levine, S. (2021). How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, 40(4-5):698–721.

IMO (2003). Regulations for carriage of ais. `https://www.imo.org/en/OurWork/Safety/Pages/AIS.aspx`. Accessed: March 14, 2024.

IMO (2010). Guidelines for the presentation of navigational-related symbols, terms and abbreviations. Circular SN.1/Circ.289, International Maritime Organization.

IMO (2019). *Ships' Routeing*. International Maritime Organization.

International Maritime Organization (1972). *COLREG: Convention on the International Regulations for Preventing Collisions at Sea*.

International Maritime Organization (2023). AIS transponders. `https://www.imo.org/en/OurWork/Safety/Pages/AIS.aspx`. Accessed June 2, 2023.

International Telecommunication Union (2014). Technical characteristics for an automatic identification system using time-division multiple access in the vhf maritime mobile band. *Recommendation ITU: Geneva, Switzerland*.

Ionescu, R.-V. (2016). Inland waterways' importance for the european economy. case study: Romanian inland waterways transport. *Journal of Danubian Studies and Research*, 6(2).

ITU (2001). Technical characteristics for an automatic identification system using time-division multiple access in the vhf maritime mobile frequency band. Technical Report ITU-R M.1371-1, ITU-R.

Jadhav, A. K., Pandi, A. R., and Somayajula, A. (2023). Collision avoidance for autonomous surface vessels using novel artificial potential fields. *Ocean Engineering*, 288:116011.

Jankowski, D., Lamm, A., and Hahn, A. (2021). Determination of ais position accuracy and evaluation of reconstruction methods for maritime observation data. *IFAC-PapersOnLine*, 54(16):97–104.

Jiang, Y., Li, X., Luo, H., Yin, S., and Kaynak, O. (2022). Quo vadis artificial intelligence? *Discover Artificial Intelligence*, 2(1):4.

Johansen, T. A., Perez, T., and Cristofaro, A. (2016). Ship collision avoidance and COLREGS compliance using simulation-based control behavior selection with predictive hazard assessment. *IEEE Transactions on Intelligent Transportation Systems*, 17(12):3407–3422.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. (2021). Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134.

Kang, K., Belkhale, S., Kahn, G., Abbeel, P., and Levine, S. (2019). Generalization through simulation: Integrating simulated and real data into deep reinforcement learning for vision-based autonomous flight. In *International Conference on Robotics and Automation*, pages 6008–6014. IEEE.

Karney, C. F. (2011). Transverse mercator with an accuracy of a few nanometers. *Journal of Geodesy*, 85:475–485.

Kaufmann, E., Bauersfeld, L., Loquercio, A., Müller, M., Koltun, V., and Scaramuzza, D. (2023). Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90–98.

Kijima, K. and Nakiri, Y. (1990). Prediction method of ship maneuverability in deep and shallow waters. *Proceedings of MARSIM and ICSM 90*.

Kim, T.-e., Perera, L. P., Sollid, M.-P., Batalden, B.-M., and Sydnes, A. K. (2022). Safety challenges related to autonomous ships in mixed navigational environments. *WMU Journal of Maritime Affairs*, 21(2):141–159.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kitchat, K., Lin, M.-H., Chen, H.-S., Sun, M.-T., Sakai, K., Ku, W.-S., and Surasak, T. (2024). A deep reinforcement learning system for the allocation of epidemic prevention materials based on DDPG. *Expert Systems with Applications*, 242:122763.

Krüger, L. (1912). *Konforme Abbildung des Erdellipsoids in der Ebene*. Number 52. BG Teubner.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. (2020). Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191.

Kuwata, Y., Wolf, M. T., Zarzhitsky, D., and Huntsberger, T. L. (2013). Safe maritime autonomous navigation with COLREGS, using velocity obstacles. *IEEE Journal of Oceanic Engineering*, 39(1):110–119.

Last, P., Bahlke, C., Hering-Bertram, M., and Linsen, L. (2014). Comprehensive analysis of automatic identification system (AIS) data in regard to vessel movement prediction. *The Journal of Navigation*, 67(5):791–809.

Lefeber, E., Pettersen, K. Y., and Nijmeijer, H. (2003). Tracking control of an underactuated ship. *IEEE Transactions on Control Systems Technology*, 11(1):52–61.

Lenart, A. S. (1983). Collision threat parameters for a new radar display and plot technique. *The Journal of Navigation*, 36(3):404–410.

Li, L., Wu, D., Huang, Y., and Yuan, Z.-M. (2021a). A path planning strategy unified with a COLREGs collision avoidance function based on deep reinforcement learning and artificial potential field. *Applied Ocean Research*, 113:102759.

Li, Z., Yu, H., Zhang, G., Dong, S., and Xu, C.-Z. (2021b). Network-wide traffic signal control optimization using a multi-agent deep reinforcement learning. *Transportation Research Part C: Emerging Technologies*, 125:103059.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Liu, W., Qiu, K., Yang, X., Wang, R., Xiang, Z., Wang, Y., and Xu, W. (2023). COLREGS-based collision avoidance algorithm for unmanned surface vehicles using modified artificial potential fields. *Physical Communication*, 57:101980.

Liu, Y., Bu, R., and Gao, X. (2018). Ship trajectory tracking control system design based on sliding mode control algorithm. *Polish Maritime Research*, (3):26–34.

Liu, Y. and Bucknall, R. (2015). Path planning algorithm for unmanned surface vehicle formations in a practical maritime environment. *Ocean Engineering*, 97:126–144.

Liu, Z., Zhang, Y., Yu, X., and Yuan, C. (2016). Unmanned surface vehicles: An overview of developments and challenges. *Annual Reviews in Control*, 41:71–93.

Luo, D., Chen, P., Yang, J., Li, X., and Zhao, Y. (2023). A new classification method for ship trajectories based on ais data. *JOURNAL OF MARINE SCIENCE AND ENGINEERING*, 11(9).

Lyu, H. and Yin, Y. (2018). Fast path planning for autonomous ships in restricted waters. *Applied Sciences*, 8(12):2592.

Lyu, H. and Yin, Y. (2019). COLREGS-constrained real-time path planning for autonomous ships using modified artificial potential fields. *The Journal of Navigation*, 72(3):588–608.

Ma, J., Jia, C., Shu, Y., Liu, K., Zhang, Y., and Hu, Y. (2021). Intent prediction of vessels in intersection waterway based on learning vessel motion patterns with early observations. *Ocean Engineering*, 232:109154.

Mahesh, B. et al. (2020). Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, 9(1):381–386.

Mahmoud, H. and Akkari, N. (2016). Shortest path calculation: a comparative study for location-based recommender system. In *2016 world symposium on computer applications & research (WSCAR)*, pages 1–5. IEEE.

Mao, S., Tu, E., Zhang, G., Rachmawati, L., Rajabally, E., and Huang, G.-B. (2018). An automatic identification system (ais) database for maritime trajectory prediction and data mining. In *Proceedings of ELM-2016*, pages 241–257. Springer.

Martinsen, A. B. and Lekkas, A. M. (2018a). Curved path following with deep reinforcement learning: Results from three vessel models. In *OCEANS 2018 MTS/IEEE Charleston*, pages 1–8. IEEE.

Martinsen, A. B. and Lekkas, A. M. (2018b). Straight-path following for underactuated marine vessels using deep reinforcement learning. *IFAC-PapersOnLine*, 51(29):329–334.

Martinsen, A. B., Lekkas, A. M., Gros, S., Glomsrud, J. A., and Pedersen, T. A. (2020). Reinforcement learning-based tracking control of usvs in varying operational conditions. *Frontiers in Robotics and AI*, 7:32.

Matsuo, Y., LeCun, Y., Sahani, M., Precup, D., Silver, D., Sugiyama, M., Uchibe, E., and Morimoto, J. (2022). Deep learning, reinforcement learning, and world models. *Neural Networks*.

Meng, L., Gorbet, R., and Kulić, D. (2021). Memory-based deep reinforcement learning for POMDPS. In *International Conference on Intelligent Robots and Systems*, pages 5619–5626. IEEE.

Meyer, E., Heiberg, A., Rasheed, A., and San, O. (2020). COLREG-compliant collision avoidance for unmanned surface vehicle using deep reinforcement learning. *IEEE Access*, 8:165344–165364.

Mircetic, D., Nikolicic, S., Bojic, S., and Maslaric, M. (2017). Identifying the barriers for development of inland waterway transport: A case study. In *MATEC Web of Conferences*, volume 134, page 00039. EDP Sciences.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

Mohanty, S., Nygren, E., Laurent, F., Schneider, M., Scheller, C., Bhattacharya, N., Watson, J., Egli, A., Eichenberger, C., Baumberger, C., et al. (2020). Flatland-rl: Multi-agent reinforcement learning on trains. *arXiv preprint arXiv:2012.05893*.

Moreira, L., Fossen, T. I., and Soares, C. G. (2007). Path following control system for a tanker ship model. *Ocean Engineering*, 34(14-15):2074–2085.

Morien, L. (2023). pyais: Ais message decoding and encoding in python (aivdm/aivdo). `https://github.com/M0r13n/pyais`.

Motwani, A., Sharma, S., Sutton, R., and Culverhouse, P. (2013). Interval kalman filtering in navigation system design for an uninhabited surface vehicle. *The Journal of Navigation*, 66(5):639–652.

Mou, J. M., Van Der Tak, C., and Ligteringen, H. (2010). Study on collision avoidance in busy waterways by using AIS data. *Ocean Engineering*, 37(5-6):483–490.

Munim, Z. H., Dushenko, M., Jimenez, V. J., Shakil, M. H., and Imset, M. (2020). Big data and artificial intelligence in the maritime industry: a bibliometric review and future research directions. *Maritime Policy & Management*, 47(5):577–597.

Negenborn, R. R., Goerlandt, F., Johansen, T. A., Slaets, P., Valdez Banda, O. A., Vanelslander, T., and Ventikos, N. P. (2023). Autonomous ships are on the horizon: here's what we need to know. *Nature*, 615(7950):30–33.

Nelson, D. R., Barber, D. B., McLain, T. W., and Beard, R. W. (2007a). Vector field path following for miniature air vehicles. *IEEE Transactions on Robotics*, 23(3):519–529.

Nelson, D. R., Barber, D. B., McLain, T. W., and Beard, R. W. (2007b). Vector field path following for miniature air vehicles. *IEEE Transactions on Robotics*, 23(3):519–529.

Nguyen, T. V. and Nguyen, H. P. (2020). Legal, institutional and financial solutions for the sustainable development strategy of inland waterway transport in vietnam. *Res World Econ*, 11(3):151–170.

Notteboom, T. and Cariou, P. (2009). Fuel surcharge practices of container shipping lines: Is it about cost recovery or revenue making. In *Proceedings of the 2009 international association of maritime economists (IAME) conference*, pages 24–26. IAME Copenhagen, Denmark.

Oh, S.-R. and Sun, J. (2010). Path following of underactuated marine surface vessels using line-of-sight based model predictive control. *Ocean Engineering*, 37(2-3):289–295.

Orzechowski, S. C., Verheyen, W., and Sys, C. (2024). A systematic literature review of factors influencing the regulation of autonomous inland shipping in europe. *European Transport Research Review*, 16(1):54.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. (2016). Deep exploration via bootstrapped dqn. *Advances in neural information processing systems*, 29.

Öztürk, Ü., Akdağ, M., and Ayabakan, T. (2022). A review of path planning algorithms in maritime autonomous surface ships: Navigation safety perspective. *Ocean Engineering*, 251:111010.

Öztürk, Ü. and Cicek, K. (2019). Individual collision risk assessment in ship navigation: A systematic literature review. *Ocean Engineering*, 180:130–143.

Pallotta, G., Vespe, M., and Bryan, K. (2013). Vessel pattern knowledge discovery from AIS data: A framework for anomaly detection and route prediction. *Entropy*, 15(6):2218–2245.

Pandey, V. and Boyles, S. D. (2018). Multiagent reinforcement learning algorithm for distributed dynamic pricing of managed lanes. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2346–2351. IEEE.

Pandey, V., Wang, E., and Boyles, S. D. (2020). Deep reinforcement learning algorithm for dynamic pricing of express lanes with multiple access locations. *Transportation Research Part C: Emerging Technologies*, 119:102715.

Paramesh, S. and Rajendran, S. (2021). A unified seakeeping and manoeuvring model with a pid controller for path following of a kvlcc2 tanker in regular waves. *Applied Ocean Research*, 116:102860.

Park, S., Cap, M., Alonso-Mora, J., Ratti, C., and Rus, D. (2020). Social trajectory planning for urban autonomous surface vessels. *IEEE Transactions on Robotics*, 37(2):452–465.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32:8026–8037.

Paulig, N. (2022). Mmg standard model for ship maneuvering. `https://github.com/nikpau/mmgdynamics`.

Paulig, N. (2023). pytsa: trajectories interpolation from AIS records. `https://github.com/nikpau/pytsa`.

Paulig, N. (2024). PyTSA: Python trajectory splitting and assessment agent for ais data. `https://github.com/nikpau/pytsa`.

Paulig, N. and Okhrin, O. (2024). Robust path following on rivers using bootstrapped reinforcement learning. *Ocean Engineering*, 298:117207.

Pauwelyn, A.-S. and Turf, S. (2022). Smart shipping on inland waterways. In *Smart Rivers*, pages 951–958. Springer.

Peng, Z., Liu, E., Pan, C., Wang, H., Wang, D., and Liu, L. (2023). Model-based deep reinforcement learning for data-driven motion control of an under-actuated unmanned surface vehicle: Path following and trajectory tracking. *Journal of the Franklin Institute*, 360(6):4399–4426.

Perera, L. P., Ferrari, V., Santos, F. P., Hinostroza, M. A., and Soares, C. G. (2014). Experimental evaluations on ship autonomous navigation and collision avoidance by intelligent guidance. *IEEE Journal of Oceanic Engineering*, 40(2):374–387.

Pfoser, S., Jung, E., and Putz, L.-M. (2018). Same river same rules?-administrative barriers in the danube countries. *Journal of Sustainable Development of Transport and Logistics*, 3(3 (6)):27–37.

Plotnikova, E., Vienažindienė, M., and Slavinskas, S. (2022). Development of inland waterway transport as a key to ensure sustainability: A case study of lithuania. *Sustainability*, 14(17):10532.

Porathe, T. and Rødseth, Ø. J. (2019). Simplifying interactions between autonomous and conventional ships with e-navigation. In *Journal of Physics: Conference Series*, volume 1357, page 012041. IOP Publishing.

Puterman, M. L. (1994). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons Inc.

Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Restrepo-Arias, J. F., Branch-Bedoya, J. W., Zapata-Cortes, J. A., Paipa-Sanabria, E. G., and Garnica-López, M. A. (2022). Industry 4.0 technologies applied to inland waterway transport: Systematic literature review. *Sensors*, 22(10):3708.

Ringbom, H. (2019). Regulating autonomous ships—concepts, challenges and precedents. *Ocean Development & International Law*, 50(2-3):141–169.

Riquelme-Solar, M., Van Slobbe, E., and Werners, S. (2015). Adaptation turning points on inland waterway transport in the rhine river. *Journal of Water and Climate Change*, 6(4):670–682.

Rødseth, Ø. J., Wennersberg, L. A. L., and Nordahl, H. (2023). Improving safety of interactions between conventional and autonomous ships. *Ocean Engineering*, 284:115206.

Rogerson, S., Santén, V., Svanberg, M., Williamsson, J., and Woxenius, J. (2020). Modal shift to inland waterways: dealing with barriers in two swedish cases. *International Journal of Logistics Research and Applications*, 23(2):195–210.

Rohács, J. and Simongati, G. (2007). The role of inland waterway navigation in a sustainable transport system. *Transport*, 22(3):148–153.

Rong, H., Teixeira, A., and Soares, C. G. (2020). Data mining approach to shipping route characterization and anomaly detection based on ais data. *Ocean Engineering*, 198:106936.

Rong, H., Teixeira, A., and Soares, C. G. (2022). Ship collision avoidance behaviour recognition and analysis based on AIS data. *Ocean Engineering*, 245:110479.

Sakamoto, T. and Baba, E. (1986). Minimisation of resistance of slowly moving full hull forms in short waves. In *Proceedings of Sixteenth Symposium on Naval Hydrodynamics*, pages 598–613.

Sallab, A. E., Abdou, M., Perot, E., and Yogamani, S. (2017). Deep reinforcement learning framework for autonomous driving. *arXiv preprint arXiv:1704.02532*.

Sanchez Pedroche, D., Amigo, D., Garcia, J., and Manuel Molina, J. (2020). Architecture for trajectory-based fishing ship classification with ais data. *SENSORS*, 20(13).

Sandeepkumar, R., Rajendran, S., Mohan, R., and Pascoal, A. (2022). A unified ship manoeuvring model with a nonlinear model predictive controller for path following in regular waves. *Ocean Engineering*, 243:110165.

Sang, L.-z., Wall, A., Mao, Z., Yan, X.-p., and Wang, J. (2015). A novel method for restoring the trajectory of the inland waterway ship by using ais data. *Ocean Engineering*, 110:183–194.

Sawada, R., Sato, K., and Majima, T. (2021). Automatic ship collision avoidance using deep reinforcement learning with LSTM in continuous action spaces. *Journal of Marine Science and Technology*, 26(2):509–524.

Scheepers, H., Wang, J., Gan, T., and Kuo, C. (2018). The impact of climate change on inland waterway transport: Effects of low water levels on the mackenzie river. *Journal of Hydrology*, 566:285–298.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Šemrov, D., Marsetič, R., Žura, M., Todorovski, L., and Srdic, A. (2016). Reinforcement learning approach for train rescheduling on a single-track railway. *Transportation Research Part B: Methodological*, 86:250–267.

Serigstad, E., Eriksen, B.-O. H., and Breivik, M. (2018). Hybrid collision avoidance for autonomous surface vehicles. *IFAC-PapersOnLine*, 51(29):1–7.

Seyde, T., Gilitschenski, I., Schwarting, W., Stellato, B., Riedmiller, M., Wulfmeier, M., and Rus, D. (2021). Is bang-bang control all you need? solving continuous control with bernoulli policies. *Advances in Neural Information Processing Systems*, 34:27209–27221.

Sharma, S., Naeem, W., and Sutton, R. (2012). An autopilot based on a local control network design for an unmanned surface vehicle. *The Journal of Navigation*, 65(2):281–301.

Shen, H. and Guo, C. (2016). Path-following control of underactuated ships using actor-critic reinforcement learning with mlp neural networks. In *2016 Sixth International Conference on Information Science and Technology (ICIST)*, pages 317–321. IEEE.

Siciliano, B., Khatib, O., and Kröger, T. (2008). *Springer handbook of robotics*, volume 200. Springer.

Siegwart, R., Nourbakhsh, I. R., and Scaramuzza, D. (2011). *Introduction to autonomous mobile robots*. MIT press.

Silveira, P., Teixeira, A., and Soares, C. G. (2013). Use of ais data to characterise marine traffic patterns and ship collision risk off the coast of portugal. *The Journal of Navigation*, 66(6):879–898.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, pages 387–395. Pmlr.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *nature*, 550(7676):354–359.

Singh, Y., Sharma, S., Sutton, R., Hatton, D., and Khan, A. (2018). A constrained A* approach towards optimal path planning for an unmanned surface vehicle in a maritime environment containing dynamic obstacles and ocean currents. *Ocean Engineering*, 169:187–201.

Skinner, B. (1938). *The behavior of organisms: an experimental analysis*. Appleton-Century.

Soler, D., Mariño, O., Huergo, D., de Frutos, M., and Ferrer, E. (2024). Reinforcement learning to maximize wind turbine energy generation. *Expert Systems with Applications*, 249:123502.

Solomon, B., Otoo, E., Boateng, A., and Koomson, D. A. (2021). Inland waterway transportation (iwt) in ghana: A case study of volta lake transport. *International Journal of Transportation Science and Technology*, 10(1):20–33.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Szepesvári, C. (2010). Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1):1–103.

Szlapczynski, R. and Szlapczynska, J. (2017). Review of ship safety domains: Models and applications. *Ocean Engineering*, 145:277–289.

Taimuri, G., Matusiak, J., Mikkola, T., Kujala, P., and Hirdaris, S. (2020). A 6-dof maneuvering model for the rapid estimation of hydrodynamic actions in deep and shallow waters. *Ocean Engineering*, 218:108103.

Tam, C. and Bucknall, R. (2010). Path-planning algorithm for ships in close-range encounters. *Journal of Marine Science and Technology*, 15:395–407.

Tam, C., Bucknall, R., and Greig, A. (2009). Review of collision avoidance and path planning methods for ships in close range encounters. *The Journal of Navigation*, 62(3):455–476.

Tang, X., Tong, S., Huang, G., and Xu, G. (2020). Numerical investigation of the maneuverability of ships advancing in the non-uniform flow and shallow water areas. *Ocean Engineering*, 195:106679.

Thrun, S. and Schwartz, A. (1993). Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*, volume 6, pages 1–9.

Toyoda, S. and Fujii, Y. (1971). Marine traffic engineering. *The Journal of Navigation*, 24(1):24–34.

Tran, H. A., Johansen, T. A., and Negenborn, R. R. (2023). Collision avoidance of autonomous ships in inland waterways–a survey and open research problems. In *Journal of Physics: Conference Series*, volume 2618, page 012004. IOP Publishing.

Treiber, M. and Kanagaraj, V. (2015). Comparing numerical integration schemes for time-continuous car-following models. *Physica A: Statistical Mechanics and its Applications*, 419:183–195.

Trivedi, A., Jakhar, S. K., and Sinha, D. (2021). Analyzing barriers to inland waterways as a sustainable transportation mode in india: A dematel-ism based approach. *Journal of cleaner production*, 295:126301.

Tu, E., Zhang, G., Rachmawati, L., Rajabally, E., and Huang, G.-B. (2017). Exploiting AIS data for intelligent maritime navigation: A comprehensive survey from data to methodology. *IEEE Transactions on Intelligent Transportation Systems*, 19(5):1559–1582.

Tutsoy, O., Asadi, D., Ahmadi, K., Nabavi-Chashmi, S. Y., and Iqbal, J. (2024). Minimum distance and minimum time optimal path planning with bioinspired machine learning algorithms for faulty unmanned air vehicles. *IEEE Transactions on Intelligent Transportation Systems*.

Vagale, A., Oucheikh, R., Bye, R. T., Osen, O. L., and Fossen, T. I. (2021). Path planning and collision avoidance for autonomous surface vehicles I: a review. *Journal of Marine Science and Technology*, 26:1292–1306.

Van Hasselt, H. (2010). Double q-learning. *Advances in neural information processing systems*, 23.

Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Van Meir, N., Rashed, Y., Storms, K., Sys, C., Vanelslander, T., and van Hassel, E. (2022). Forecasting the future demand for containerized inland shipping on the traditional rhine. *Available at SSRN 4043354*.

Van Rossum, G. and Drake, F. L. (2009). *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA.

Vanneste, A., Vanneste, S., Vasseur, O., Janssens, R., Billast, M., Anwar, A., Mets, K., De Schepper, T., Mercelis, S., and Hellinckx, P. (2022). Safety aware autonomous path planning using model predictive reinforcement learning for inland waterways. In *Annual Conference of the IEEE Industrial Electronics Society*, pages 1–6.

Vilarinho, A., Liboni, L. B., Cezarino, L. O., Micco, J. D., Mommens, K., and Macharis, C. (2024). Challenges and opportunities for the development of inland waterway transport in brazil. *Sustainability*, 16(5):2136.

Vincenty, T. (1975). Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Survey review*, 23(176):88–93.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. (2019). Grandmaster level in starcraft ii using multi-agent reinforcement learning. *nature*, 575(7782):350–354.

Voskresenskaya, E., Vorona-Slivinskaya, L., and Ponomareva, T. (2018). Application of public-private partnership mechanisms for competitive growth of inland water transport in russia. In *MATEC Web of Conferences*, volume 239, page 08009. EDP Sciences.

Waltz, M. and Okhrin, O. (2022). Two-sample testing in reinforcement learning. *arXiv preprint arXiv:2201.08078*.

Waltz, M. and Okhrin, O. (2023). Spatial–temporal recurrent reinforcement learning for autonomous ships. *Neural Networks*, 165:634–653.

Waltz, M. and Paulig, N. (2022). Rl dresden algorithm suite. `https://github.com/MarWaltz/TUD_RL`.

Waltz, M., Paulig, N., and Okhrin, O. (2023). 2-level reinforcement learning for ships on inland waterways. *arXiv preprint arXiv:2307.16769*.

Wan, L., Su, Y., Zhang, H., Shi, B., and AbouOmar, M. S. (2020). An improved integral light-of-sight guidance law for path following of unmanned surface vehicles. *Ocean Engineering*, 205:107302.

Wang, P., Gao, S., Li, L., Sun, B., and Cheng, S. (2019). Obstacle avoidance path planning design for autonomous driving vehicles based on an improved artificial potential field algorithm. *Energies*, 12(12):2342.

Wang, X., Liu, Z., and Cai, Y. (2017). The ship maneuverability based collision avoidance dynamic support system in close-quarters situation. *Ocean Engineering*, 146:486–497.

Wang, Y., Cao, J., Sun, J., Zou, X., and Sun, C. (2023). Path following control for unmanned surface vehicles: A reinforcement learning-based method with experimental validation. *IEEE Transactions on Neural Networks and Learning Systems*.

Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3):279–292.

Wilcox, R. R., Erceg-Hurn, D. M., Clark, F., and Carlson, M. (2014). Comparing two independent groups via the lower and upper quantiles. *Journal of Statistical Computation and Simulation*, 84(7):1543–1551.

Wolsing, K., Roepert, L., Bauer, J., and Wehrle, K. (2022). Anomaly detection in maritime ais tracks: A review of recent approaches. *Journal of Marine Science and Engineering*, 10(1):112.

Woo, J., Yu, C., and Kim, N. (2019). Deep reinforcement learning-based controller for path following of an unmanned surface vehicle. *Ocean Engineering*, 183:155–166.

Wu, L., Xu, Y., Wang, Q., Wang, F., and Xu, Z. (2017). Mapping global shipping density from ais data. *The Journal of Navigation*, 70(1):67–81.

Wurman, P. R., Barrett, S., Kawamoto, K., MacGlashan, J., Subramanian, K., Walsh, T. J., Capobianco, R., Devlic, A., Eckert, F., Fuchs, F., et al. (2022). Outracing champion gran turismo drivers with deep reinforcement learning. *Nature*, 602(7896):223–228.

Xia, G., Liu, J., and Wu, H. (2013). Neural network based nonlinear model predictive control for ship path following. In *2013 Ninth International Conference on Natural Computation (ICNC)*, pages 210–215. IEEE.

Xu, H. and Guedes Soares, C. (2023). Review of path-following control systems for maritime autonomous surface ships. *Journal of Marine Science and Application*, 22(2):153–171.

Xu, H., Rong, H., and Soares, C. G. (2019). Use of ais data for guidance and control of path-following autonomous vessels. *Ocean Engineering*, 194:106635.

Xu, X., Cai, P., Ahmed, Z., Yellapu, V. S., and Zhang, W. (2022a). Path planning and dynamic collision avoidance algorithm under COLREGs via deep reinforcement learning. *Neurocomputing*, 468:181–197.

Xu, X., Lu, Y., Liu, G., Cai, P., and Zhang, W. (2022b). COLREGs-abiding hybrid collision avoidance algorithm based on deep reinforcement learning for USVs. *Ocean Engineering*, 247:110749.

Yan, Z., Xiao, Y., Cheng, L., Chen, S., Zhou, X., Ruan, X., Li, M., He, R., and Ran, B. (2020a). Analysis of global marine oil trade based on automatic identification system (ais) data. *Journal of Transport Geography*, 83:102637.

Yan, Z., Xiao, Y., Cheng, L., He, R., Ruan, X., Zhou, X., Li, M., and Bin, R. (2020b). Exploring ais data for intelligent maritime routes extraction. *Applied Ocean Research*, 101:102271.

Yasukawa, H. and Yoshimura, Y. (2015). Introduction of mmg standard method for ship maneuvering predictions. *Journal of Marine Science and Technology*, 20(1):37–52.

You, X., Li, S., Liu, J., and Yan, X. (2023). Experimental research of the pid tune method for ship path following control. In *ISOPE International Ocean and Polar Engineering Conference*, pages ISOPE–I. ISOPE.

Yu, H., Meng, Q., Fang, Z., Liu, J., and Xu, L. (2023). A review of ship collision risk assessment, hotspot detection and path planning for maritime traffic control in restricted waters. *The Journal of Navigation*, pages 1–27.

Yuan, Z., Liu, J., Liu, Y., and Li, Z. (2019). A novel approach for vessel trajectory reconstruction using ais data. In *ISOPE International Ocean and Polar Engineering Conference*, pages ISOPE–I. ISOPE.

Zhai, P., Zhang, Y., and Shaobo, W. (2022). Intelligent ship collision avoidance algorithm based on DDQN with prioritized experience replay under COLREGs. *Journal of Marine Science and Engineering*, 10(5):585.

Zhang, J., Liu, J., Hirdaris, S., Zhang, M., and Tian, W. (2023a). An interpretable knowledge-based decision support method for ship collision avoidance using ais data. *Reliability Engineering & System Safety*, 230:108919.

Zhang, L., Meng, Q., Xiao, Z., and Fu, X. (2018). A novel ship trajectory reconstruction approach using ais data. *Ocean Engineering*, 159:165–174.

Zhang, W., Goerlandt, F., Kujala, P., and Wang, Y. (2016). An advanced method for detecting possible near miss ship collisions from ais data. *Ocean Engineering*, 124:141–156.

Zhang, X., Yang, G., Zhang, Q., Zhang, G., and Zhang, Y. (2017). Improved concise backstepping control of course keeping for ships using nonlinear feedback technique. *The Journal of Navigation*, 70(6):1401–1414.

Zhang, Y., Chen, P., Chen, L., and Mou, J. (2023b). A path planning method for the autonomous ship in restricted bridge area based on anisotropic fast marching algorithm. *Ocean Engineering*, 269:113546.

Zhang, Z., Wu, D., Gu, J., and Li, F. (2019). A path-planning strategy for unmanned surface vehicles based on an adaptive hybrid dynamic stepsize and target attractive force-RRT algorithm. *Journal of Marine Science and Engineering*, 7(5):132.

Zhao, L., Roh, M.-I., and Lee, S.-J. (2019). Control method for path following and collision avoidance of autonomous ship based on deep reinforcement learning. *Journal of Marine Science and Technology*, 27(4):1.

Zhao, L., Shi, G., and Yang, J. (2018). Ship trajectories pre-processing based on ais data. *The Journal of Navigation*, 71(5):1210–1230.

Zhao, W., Queralta, J. P., and Westerlund, T. (2020a). Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *IEEE Symposium Series on Computational Intelligence*, pages 737–744. IEEE.

Zhao, Y., Ma, Y., and Hu, S. (2021). Usv formation and path-following control via deep reinforcement learning with random braking. *IEEE Transactions on Neural Networks and Learning Systems*, 32(12):5468–5478.

Zhao, Y., Qi, X., Ma, Y., Li, Z., Malekian, R., and Sotelo, M. A. (2020b). Path following optimization for an underactuated usv using smoothly-convergent deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 22(10):6208–6220.

Zhen, R., Jin, Y., Hu, Q., Shao, Z., and Nikitakos, N. (2017). Maritime anomaly detection within coastal waters based on vessel trajectory clustering and naïve bayes classifier. *The Journal of Navigation*, 70(3):648–670.

Zhou, Y., Daamen, W., Vellinga, T., and Hoogendoorn, S. (2019). Review of maritime traffic models from vessel behavior modeling perspective. *Transportation Research Part C: Emerging Technologies*, 105:323–345.

Zhu, M., Wang, Y., Pu, Z., Hu, J., Wang, X., and Ke, R. (2020). Safe, efficient, and comfortable velocity control based on reinforcement learning for autonomous driving. *Transportation Research Part C: Emerging Technologies*, 117:102662.

Zhuge, D., Wang, S., Zhen, L., and Psaraftis, H. N. (2023). Data-driven modeling of maritime transportation: Key issues, challenges, and solutions. *Engineering*. DOI: 10.1016/j.eng.2022.12.009.