NTNU Trondheim
Norwegian University of Science and Technology
Department of Marine Technology

MASTER THESIS IN MARINE CYBERNETICS

SPRING 2020

FOR

STUD. TECHN. TONI KLAUSEN

Combining reinforcement learning and historical AIS-data for simulating realistic ship paths

Work description (short description)

Autonomous ships have the potential for safer, greener and smarter operation. For the development of autonomous systems, a realistic simulator is desirable. The work in this project concerns itself with the development of a simulator for the autonomous passenger ferry milliAmpère in the Open Simulation Platform. Particularly waypoint generation using a reinforcement learning approach combined with historical AIS-data is treated here. This can be used both for milliAmpère path-planning and to simulate realistic traffic scenarios for testing purposes as well. As part of the work, sensors on board milliAmpère will be modelled and implemented for use in the simulator.

Scope of work

1. Perform a literature review of HIL-testing, co-simulation, vessel- and sensor modelling, and reinforcement learning.
2. Develop functional mock-up units for the required sensors with different failure modes.
3. Collect, process and analyse historical AIS-data for a given area of relevance for milliAmpère.
4. Develop waypoint generation algorithms using reinforcement learning and the collected data
5. Test the algorithms on various cases
6. Suggest a framework for implementation of realistic traffic scenarios using the sensor FMUs, the collected data and results from the algorithms.

The report shall be written in English and edited as a research report including literature survey, description of mathematical models, description of control algorithms, simulation results, model test results, discussion and a conclusion including a proposal for further work. Source code should be provided. It is supposed that Department of Marine Technology, NTNU, can use the results freely in its research work, unless otherwise agreed upon, by referring to the student's work.

The thesis should be submitted within 23rd of August.

Advisors: Postdoc Børge Rokseth
PhD Tobias Valentin Rye Torben

Professor Asgeir J. Sørensen
Supervisor

Abstract

As ships grow increasingly more complex and autonomous, stringent measures to assure safety is required. Simulation-based testing and verification of autonomous ships is emerging as a promising approach. It revolves around creating comprehensive digital representations of all aspects affecting safety and performance of the system, including a digital twin of the vessel considered and a highly detailed virtual model of the environment. The environmental model should contain a module for traffic simulation, to assess behaviour in accordance with COLREG rules and other regulations. A simulator for the small autonomous passenger ferry milliAmpère has been developed, and the work here is in part a contribution to the simulator's traffic module. The simulator is made for the Open Simulation Platform, a co-simulation platform for the maritime industry, academia and other stakeholders.

In this thesis, historical AIS-data for the operating area of the small autonomous passenger ferry milliAmpère have been collected, processed and analysed in preparation for its use in machine learning algorithms. The dataset has been used as the basis for the reward function in reinforcement learning algorithms that generate vessel waypoints defining paths between arbitrary points A and B. The Q-learning algorithm was used as the starting point for development. Furthermore, an algorithm leveraging the AIS-data to compute a path from A to B was developed and used in the learning algorithm to provide a guess at the optimal path. This was shown to reduce runtime with no cost to results. The reinforcement learning algorithms work largely as intended, and are able to find paths for all the test cases. The obtained paths appear similar to those of a human navigator, and adhere to boating rules. Some undesirable fluctuation in assigned speed and heading is observed though, and improvements by way of penalizing this behaviour is suggested. While the algorithms achieve satisfactory results, large numbers of episodes are required to get these results. To improve on this, the created data set and proposed reward signal could be used with other reinforcement learning methods. Pleasure crafts make up the bulk of the traffic in the area considered. Hence, the applicability of the paths obtained from the generated waypoints are to vessels of similar type and dimensions. Statistics and knowledge learned from the dataset have been used with the developed algorithms to create a waypoint database defining vessel paths representative for the traffic in the operating area. This is intended for use in the environmental model of a simulator for milliAmpère, for selecting and simulating realistic traffic scenarios. A proposal for the simulator architecture needed to simulate ship traffic is provided.

Sammendrag

Siden skip i økende grad blir mer komplekse og autonome, behøver man gode verktøy for å ivareta sikkerheten. Simuleringsbasert testing og verifikasjon av autonome skip utpeker seg som en lovende tilnærming. Denne metoden baserer seg på å lage omfattende digitale representasjoner av alle aspekter som påvirker systemets sikkerhet og ytelse, inkludert en digital tvilling av skipet og en høydetaljert virtuell modell av miljøet. Miljømodellen bør inneholde en modul for trafikksimulering, for å sikre at skipets oppførsel i trafikk samsvarer med COLREG-regler og andre reguleringer. En simulator for den lille autonome passasjerferga milliAmpère har blitt utviklet, og arbeidet her er til dels et bidrag til simulatorens trafikkmodul. Simulatoren er laget for Open Simulation Platform; en kosimuleringsplattform for den maritime industrien, akademia og andre.

I denne oppgaven har historisk AIS-data for operasjonsområdet til milliAmpère blitt innsamlet, prosessert og analysert. Datasettet har blitt brukt som basis for utviklingen av maskinlæringsalgoritmer (spesifikt innen forsterkende læring) som genererer veipunkter for skipsbaner. Q-læring har blitt brukt som utgangspunkt. I tillegg har en algoritme som bruker AIS-data for å beregne baner mellom arbitrære punkter A og B blitt utviklet. Denne ble brukt i maskinlæringsalgoritmen til å gi et estimat av den optimale banen. Dette reduserte kjøretiden, uten å påvirke resultatene negativt. Algoritmene fungerer som tiltenkt, og finner baner for alle testcasene. Banene likner dem man ville forvente fra en menneskelig navigatør, og følger vanlige sjøvettregler. Noen uønskede variasjoner i hastighet og kurs var observert, noe som kan forbedres med å inkludere en straff for slik oppførsel i maskinlæringsalgoritmene. Selv om resultatene er akseptable, kreves et stort antall episoder for å oppnå disse. Datasettet og den foreslåtte belønningsfunksjonen kan brukes sammen med andre metoder innen forsterkende læring for å forbedre dette, f.eks DDQN. Majoriteten av skipstrafikken i det utpekte området består av mindre farkoster som fritidsbåter. Derfor er anvendelsen av veipunktene produsert fra algoritmene og det gitte datasettet begrenset til båter av lignende type og dimensjoner. Statistikk og annen lærdom hentet fra datasettet har blitt brukt sammen med de utviklete algoritmene til å lage en database med seilinger som representerer trafikkmønstre i området. Intensjonen er at resultatene skal brukes i miljømodellen til en simulator for milliAmpère, for å simulere realistisk trafikk i operasjonsområdet. Et eksempel på simulatorarkitektur som muliggjør dette er foreslått.

Preface

This thesis concludes my education in the master's degree programme in Engineering and ICT at NTNU, witha specialization in marine cybernetics. This semester has been unlike any other in modern times, in light of the COVID-19 pandemic. First- and second order effects of the lockdowns have impeded work progress. As a consequence, this thesis was submitted on the 23rd of August, rather than the 10th of June, as originally planned.

The work in this thesis is tied to research concerning autonomous ships at NTNU. The work is done independently, with supervision by Sørensen, Torben and Rokseth. In particular, it has been coordinated with PhD student Tobias Rye Torben, and is intended as a contribution to a simulator for the autonomous passenger ferry milliAmpère. Parts of this thesis makes use of material from the related project thesis from the preceding semester. The introduction in the project thesis, as well as the section on sensor models and vessel kinematics, have been adapted and expanded for this thesis. Karianne Skudal Tjøm co-authored the section on vessel kinematics in the project thesis, and thus is counted as a contributor here as well.

Acknowledgements

I would like to thank my family for love and support throughout these years of study, and make apologies to my daughter, who has had to put up with a father who has been preoccupied with thesis work this summer.

My supervisor Asgeir J. Sørensen, has my thanks and gratitude, not just for guidance and direction with the thesis work, but also for taking an interest in my life and career beyond the university halls.

I would also like to extend my thanks to my co-supervisors Tobias Rye Torben and Børge Rokseth, for valuable help and input throughout this final year at NTNU.

Table of Contents

List of Tables

List of Figures

Abbreviations

AIS	=	Automatic Identification System
ANS	=	Autonomous navigation system
ASV	=	Autonomous surface vehicles
AUV	=	Autonomous underwater vehicle
CPS	=	Cyber-physical system
CSE	=	Core simulation environment
DOF	=	Degrees of freedom
FMI	=	Functional Mock-up Interface
FMU	=	Functional Mock-up Unit
GPS	=	Global Positioning System
GLONASS	=	Global Navigation Satellite System
GNSS	=	Global Navigation Satellite Systems
International Regulations for Preventing Collisions at Sea 1972	=	COLREGs
LOS	=	Line-of-sight (guidance)
MEMS	=	Microelectromechanical systems
ML	=	Machine learning
IMO	=	International Maritime Organization
IMU	=	Inertial measurement unit
ICT	=	Information & communication technology
JIP	=	Joint Industry Project
OSP	=	Open Simulation Platform
RAM	=	Random access memory
RL	=	Reinforcement learning
ROV	=	Remotely operated vehicle
SOLAS	=	Safety of Life at Sea
VHF	=	Very High Frequency

Chapter 1

Introduction

The work in this thesis has mainly been directed towards developing machine learning (ML) waypoint generation algorithms that draws on historical AIS-data. Referring to Figure 1.1, the proposed method belongs in the mission layer of the control structure, specifically planning (and re-planning). In particular, the method has been applied to an area in Trondheim harbour to generate realistic ship trajectories. This work is a contribution to the environmental model of a simulator for the small autonomous passenger ferry milliAmpère, developed for the Open Simulation Platform (OSP). As part of the work, sensor models for the simulator have also been developed.

1.1 Background and motivation

1.1.1 Autonomous systems

An autonomous system can be defined as a system with the ability of integrated sensing, perceiving, analyzing, communicating, planning, decision-making, and acting, in order to achieve some specified goal [6]. This technology is enabled in part by advances in disciplines such as information- and communication technology (ICT), micro-, nano- and material technology and computer science, making autonomous systems a truly cross-disciplinary field of study. It has the potential to provide safer, more efficient and sustainable operations in applicable fields; the importance of which should be clear considering global challenges such as environmental concerns and resource scarcity.

In practice, the concept of autonomy will have different definitions across task and application domains, and separate levels of autonomy (LOA) are often defined. Various authors have proposed different taxonomies. Insaurralde and Lane defines 10 levels of autonomy for autonomous underwater vehicles (AUV) [7], SAE International proposes six levels of autonomy for cars [8] , while Utne, Sørensen and Schjølberg suggests four [2]. These typically progress from a level of high human involvement and low system complexity, to a low level of human involvement and high system complexity. As mentioned, how many

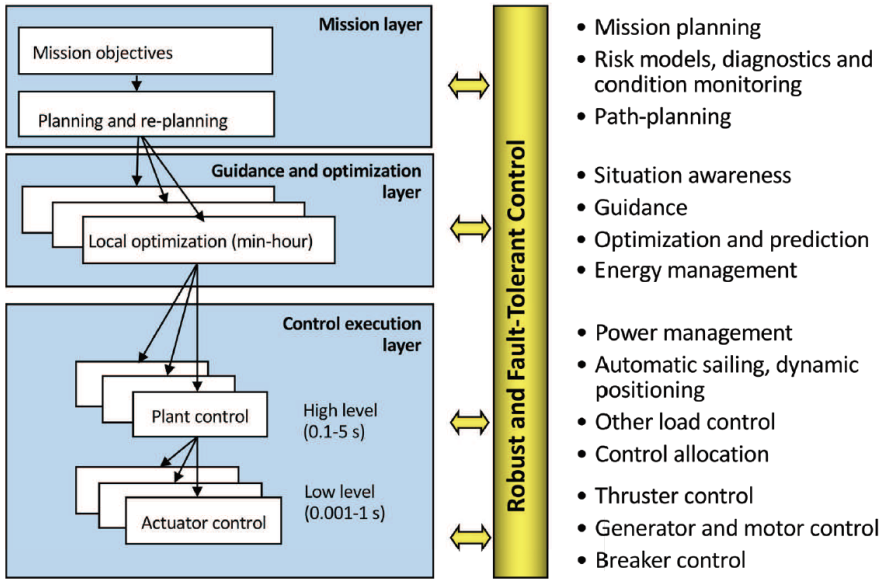


Figure 1.1: Overview of control structure. Source: [4].

LOA one operates with and their description varies according to industry and problem at hand. This can serve to give the engineer a clearer picture of the challenges ahead and what needs to be implemented in the system. From the usage of *levels* of autonomy, we see that autonomy is usually not considered a binary property, but may encompass the range from an automatic system to a fully autonomous one. Furthermore, an autonomous system may also be a subsystem in a larger and more complex (not necessarily autonomous) system.

Henceforth we restrict ourselves to autonomy in the marine domain. An example of a taxonomy for levels of autonomy in marine systems is presented in Table 1.1. It is based on the LOA defined in [1] and [2]. This taxonomy employs metrics such as operator dependency, human-machine interface, communication and mission planning & complexity to describe the system.

Instances where the use of marine autonomous systems are appealing abound. It has potential applications in shipping, ocean mapping, maritime surveillance and deep sea mining to name a few. Recent advances include passenger ferries and container ships; Rolls-Royce performed a successful testing of an autonomous ferry in 2018 in Finland [9], and the autonomous freight ship MV Yara Birkeland was set to launch in 2020, but development has been postponed due to the COVID-19 pandemic [10]. For subsea monitoring, maintenance and repair, the AUV Eelume is expected to be available to the market in 2021 [11].

Multiple research projects concerning autonomy are ongoing at NTNU. At the Applied Underwater Robotics lab, there is a fleet of robotics platforms encompassing autonomous

Table 1.1: Description of levels of autonomy, adapted from [1] and [2].

Level of autonomy	Description
1	Type of operation is automatic (remote control). This means that even though the system operates automatically, the human operator directs and controls all high-level mission-planning functions, often preprogrammed (human-in-the-loop/human operated).
2	Type of operation is management by consent (tele-operation). The system automatically makes recommendations for mission actions related to specific functions, and the system prompts the human operator at important points in time for information or decisions. At this level, the system may have limited communication bandwidth including time delay, due to i.e. distance. The system can perform many functions independently of human control when delegated to do so (human-delegated).
3	Type of operations is semi-autonomous or management by exception. The system automatically executes mission-related functions when response times are too short for human intervention. The human may override or change parameters and cancel or redirect actions within defined timelines. The operator's attention is only brought to exceptions for certain decisions (human-supervisory control).
4	Type of operation is highly autonomous, which means that the system automatically executes mission-related functions in an unstructured environment with ability to plan and re-plan the mission. The human may be informed about the progress. The system is independent and "intelligent" (human-out-of-the-loop).

surface vehicles (ASV), remotely operated vehicles (ROV) and autonomous underwater vehicles. Examples include the ASV Jetyak, the ROV 30K and the AUV REMUS. The Autoferry project has developed an autonomous, fully electric passenger ferry christened milliAmpère (seen in Figure 1.7). The ferry serves as a testbed for various technologies including anti-collision, dynamic positioning and risk management systems. The main goal of the project is to enable autonomous ferries to be used in urban water channel traffic.

1.1.2 Open Simulation Platform

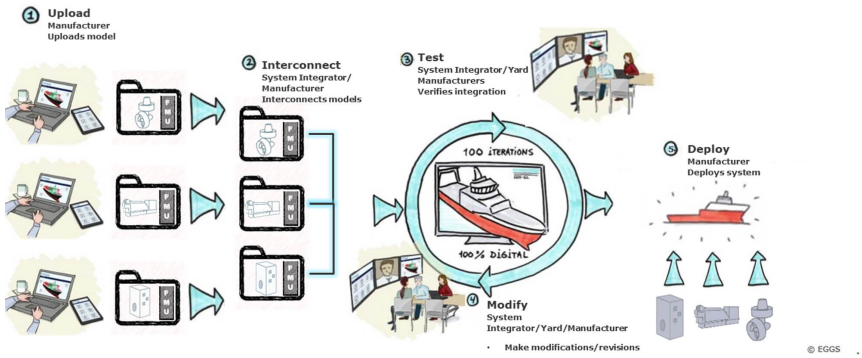


Figure 1.2: Open Simulation Platform, DNV GL.

The Open Simulation Platform is a joint industry project (JIP) initiated in 2018 by DNV GL, Kongsberg Maritime, SINTEF Ocean and NTNU. It endeavours to provide the maritime industry with an open-source core simulation environment (CSE) and a standard for co-simulation and model exchange. Co-simulation is an emerging technology in which the global simulation of complex system is accomplished through the distributed simulation of the sub-systems. Model- and simulation based approaches to systems development, implementation and maintenance are becoming increasingly important for cyber-physical systems (CPS). CPS are complex, multidisciplinary systems built on expertise in disparate fields like structural, mechanical, electrical, control and software engineering. This presents challenges, since the end product is the synthesis of the work of people with their own specialized domain of knowledge, terminology and tool sets [12]. One problem is the exchange and integration of subsystems created using different tools. Another deterrent to collaboration and free exchange of models is the protection of intellectual property. Co-simulation addresses this, and allows the engineer to independently develop models using their tools of choice, and the created models can then be exchanged and used in co-simulations of the entire subsystem. Subsystems can be run as a black-box, which protects intellectual property.

The OSP will make use of similar existing work, like the Functional Mock-up Interface (FMI). Functional Mock-up Units (FMU) are executables that implement the Functional Mock-up Interface (FMI). This is an open standard that enables the exchange and co-simulation of models. Using the FMI, parts comprising complex systems can be modelled in a given FMI-compatible tool and exported for later use in a different simulation environment. Thus FMUs can be used across simulation platforms; an attractive proposition as it means that models can be used, exchanged and integrated seamlessly by different parties for purposes of research and development. The source code is not available for those using the unit. Simulink supports FMI through the tool FMIKit, developed by Dassault Systèmes.

1.1.3 HIL-testing

Hardware-in-the-Loop (HIL) testing is a type of simulation that can be used to test and verify the performance of control systems. It essentially replaces the actual plant with a real-time simulation of it in the closed loop system, see Figure 1.3. Crucially, the target system to be tested does not experience a significant difference between being connected to the simulated and real world. It is based on interfacing an advanced, real-time simulation of the plant with a real version of the system to be tested. The plant model is highly realistic, and capable of simulating the vessel's dynamic response in various environmental conditions. It will include models of test-related equipment like sensors, thrusters and power systems for a complete description of the real vessel.

In HIL testing, the equipment intended for implementation on the real plant is used, but the loop is closed by replacing the actual plant with a highly realistic simulation of it. This is cost-effective, and allows one to test system performance in rare and hazardous scenarios, e.g extreme weather conditions and under various failure modes. In this way, one can uncover errors in the proposed systems without posing safety risk to people or equipment. In [13] the following possible errors are listed:

- Erroneous configuration parameters
- Design flaws in the software, including building an operational philosophy
- Sleeping and hidden software errors
- Missing functionality and "gaps" across interfaces and system integration
- Errors in documentation

This form of testing has proven to be effective in the maritime industry (see e.g [14], [13], [15], with DNV GL providing a standard for HIL-testing in [16].

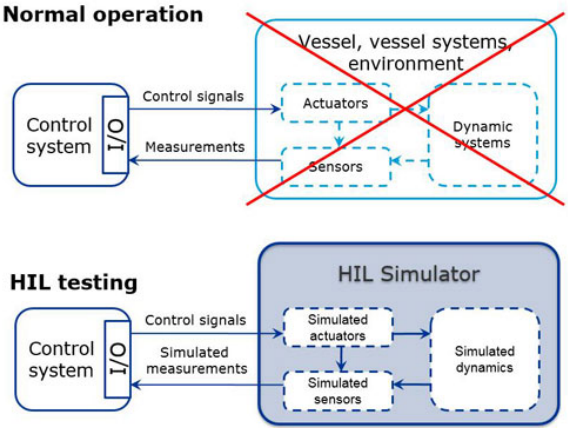


Figure 1.3: HIL testing

1.1.4 Automatic Identification System

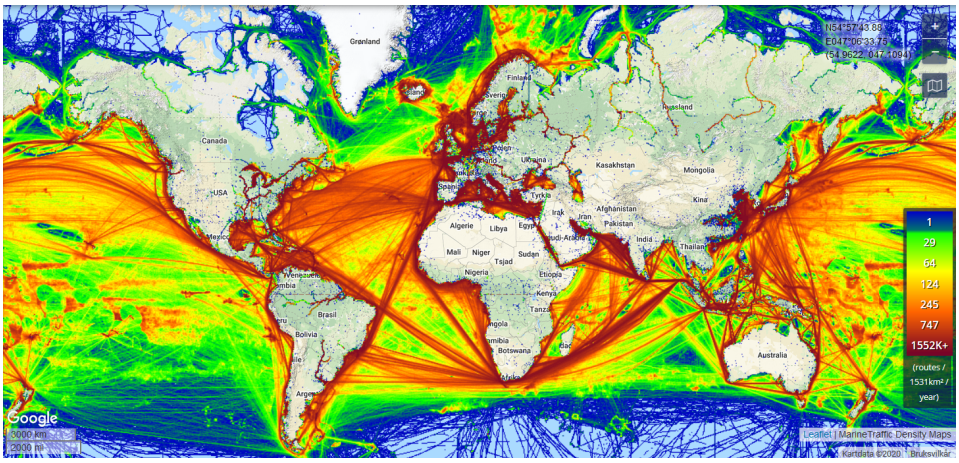


Figure 1.4: Density plot of global historical AIS-data for 2016-2017. Traffic density is indicated by color; the mapping of color to number of ships in a region is seen to the right. Plot from MarineTraffic.com.

The Automatic Identification System (AIS) is a maritime safety system that provides real-time and historical tracking of ships. Developed by the International Maritime Organization (IMO), its primary application was in collision avoidance, as it provided vessels with information about the identity, location and other kinematic data for other vessels in the vicinity. Since then it has been used for traffic surveillance and control, monitoring of fishing vessels, accident investigation and more. AIS data is stored by national authorities such as the Norwegian Coastal Administration for several years. This provides an informational basis for data analysis, with conceivable applications in route planning, identifying sailing patterns & trends with regards to weather, seasonal variation and more. Figure 1.4 provides an overview of saling patterns for 2016-2017 as determined from AIS-data. In Figure 1.5 and 1.6 we get a closer look of the patterns near the Norwegian coast and outside the county of Trøndelag, respectively.

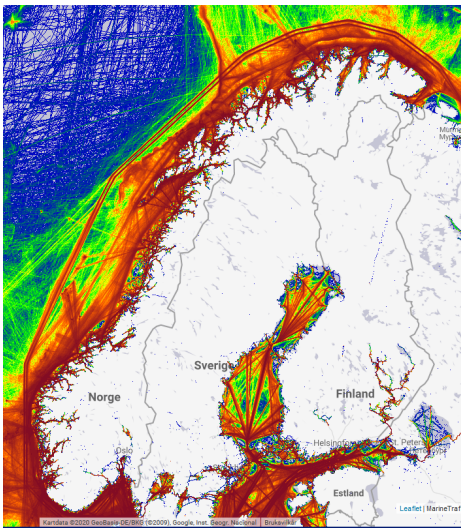


Figure 1.5: Density plot of AIS-data for 2016-2017 near the Norwegian coast.

In essence, an on-board AIS transceiver functions by automatically transmitting the re-

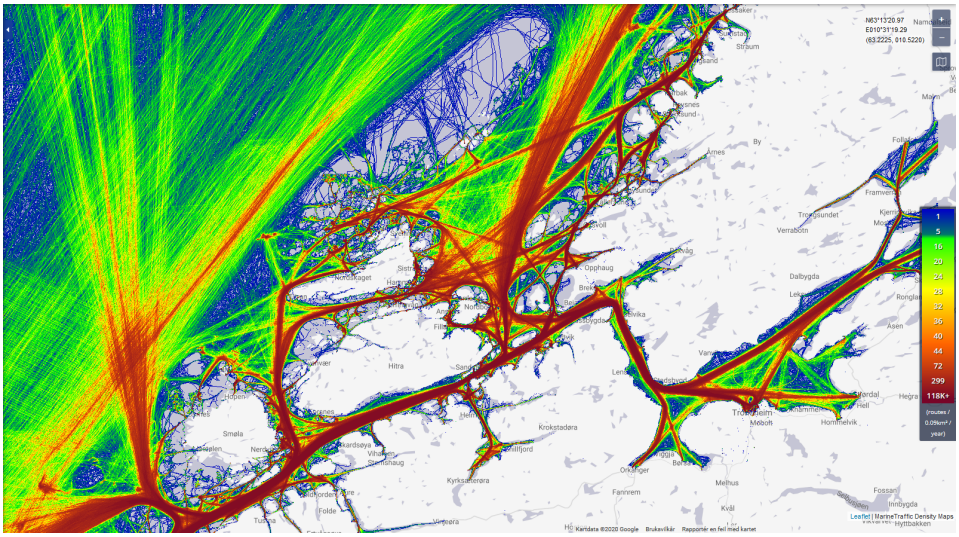


Figure 1.6: The density plot of Figure 1.5 zoomed in on Trøndelag. Trondheim seen in the lower right of the figure.

quired vessel information to shore- or satellite base stations via Very High Frequency (VHF) communication. The base stations then broadcast the information. Usually the AIS interfaces with other shipboard sensors to obtain kinematic data like position and timing data, which is commonly provided by a Global Navigation Satellite System (GNSS).

IMO's International Convention for the Safety of Life at Sea (SOLAS) requires all ships above 300 gross tons to be outfitted with AIS. Norwegian regulations[1] provide stronger stipulations on ships entering Norwegian harbours. For example, all fishing vessels over 15 m, and passenger ships exceeding 150 gross tons that are capable of reaching speeds of 20 knots must have AIS on board. Furthermore, the AIS should always be switched on in normal operating conditions.

Two classes of AIS exist; Class A and Class B, the latter of which is divided into Class B-SO and Class B-CS. In general, vessels required by laws and regulations to be outfitted with AIS equipment use Class A equipment, due to its accuracy and reliability. The less expensive Class B systems are mostly used by smaller vessels like pleasure crafts, to enhance safety on board. Comparing the two classes[2], a few relevant differences are tabulated in Table 1.2. Note that Class A has an internal GNSS fallback, interfacing with the vessels GNSS by default. Class B equipment optionally interfaces to an external GNSS – by default a GNSS internal to the AIS transponder is used.

[1] See
https://lovdata.no/forskrift/2000-06-13-660/§10-4a
https://lovdata.no/forskrift/2014-09-05-1157/§19

[2] See https://www.navcen.uscg.gov/pdf/AIS_Comparison_By_Class.pdf

Table 1.2: Key differences between Class A and Class B AIS equipment.

	Class A	Class B-SO	Class B-CS
Transmitter power	12.5 W	5 W (2 W low power)	2 W
Update frequency at 2-14 knots	10 s	30 s	30 ± 4 s
Primary positioning source	Vessel GNSS	GNSS internal to the AIS	

1.2 Research questions and objectives

Figure 1.7: milliAmpère in Trondheim Harbour.

The main research question is to evaluate if historical AIS-data can be combined with a machine learning approach to obtain vessel paths. Secondary to this is to figure out if the paths obtained from the aggregated data comply with traffic rules and other best practices. The primary intended application of this is to generate waypoints defining realistic vessel paths for use in an OSP simulator. The simulator is created for the autonomous passenger ferry milliAmpère, seen in Figure 1.7. The objectives of this project thesis are summarised below.

1. Create sensor FMUs for use in an OSP simulator
2. Collect AIS-data for a region in Trondheim harbour

3. Prepare the data for use in a waypoint generation algorithm
4. Develop machine learning algorithms producing vessel path waypoints from the data
5. Validate the algorithms in the OSP simulator

1.3 Main contributions

The first main contributions of the work herein are a dataset of historical ship AIS information for a region in Trondheim harbour. This data has been made ready for use in machine learning algorithms. The second contribution are the development of reinforcement learning algorithms providing a means of generating realistic traffic for the milliAmpère simulator. Results from the work here have been used to create a database containing waypoints defining sailings that reflect real traffic patterns in the region. Additionally, a framework for setting up traffic scenarios in the simulator have been proposed.

1.4 Organization of the thesis

- Chapter 2 presents background information relating to vessel kinematics, sensor modelling and reinforcement learning.
- Chapter 3 describes IMU and GNSS sensor implementation in Simulink/FMU.
- Chapter 4 presents the collection, processing and analysis of AIS-data from Trondheim harbour. This chapter includes a results and discussion section concerning observations and statistics from analysis of the data set.
- Chapter 5 describes the developed reinforcement learning algorithms. These leverage the collected data to generate path waypoints for arbitrary start- and stop points in the area considered. A results and discussion section comprising a few algorithm test cases is included in this chapter.
- Chapter 6 proposes an inclusion of the thesis work in a simulation based test method.
- Chapter 7 rounds off the thesis with conclusions and recommendations for future work.

Chapter 2

Background material

This chapter presents theory necessary for the succeeding chapters. It contains background information on vessel kinematics, sensor modelling and reinforcement learning relevant for this thesis. Both sections 2.1 and 2.2 are based in part on material from the project thesis, and the former section has been co-authored with Karianne Skudal Tjøm.

2.1 Vessel kinematics

To be able to describe a vessels movements accurately and conveniently, several different frames of reference may be defined. We will make use of the inertial North-East-Down (NED) geographical reference frame and a body-fixed reference frame. The NED reference frame is a local tangent plane coordinate frame, with origin fixed at a point on the surface of the Earth. The x-axis points towards true North, the y-axis towards east, and the z-axis towards the center of the Earth. Since this reference frame approximates the Earth locally as a plane, it is only valid for small areas. It is used for navigation since it gives a reference on how the vessel moves relative to earth. The body-fixed reference frame has its center at the vessel, and it moves with it. In this thesis, variables in terms of the body-frame may be super-scripted with b, while the superscript n is used for variables expressed in the NED-frame. The NED-frame and body-frame is connected, and forces can be expressed in terms of the different frames using a rotation matrix that transforms coordinates in one frame to the other. The 6-degrees-of-freedom (DOF) movements and forces in body-frame is shown in Figure 2.1 and the notation is described in Table 2.1, using the Society of Naval Architects and Marine Engineers (SNAME) nomenclature.

The kinematic 6-DOF equation of motion for marine craft is given in matrix-vector form as

$$\dot{\boldsymbol{\eta}} = \mathbf{R}_b^n(\boldsymbol{\Theta}_{nb})\boldsymbol{\nu} \tag{2.1}$$

where

- $\boldsymbol{\eta} \in \mathbf{R}^{6\times 1}$ is the generalized position, given in the NED-frame.

Table 2.1: The 6-DOF ship nomenclature as adopted from SNAME [3].

DOF		Forces and moments	Translational and angular velocities	Position and Euler angles
1	Surge	X	u	x
2	Sway	Y	v	y
3	Heave	Z	w	z
4	Roll	K	p	ϕ
5	Pitch	M	q	θ
6	Yaw	N	r	ψ

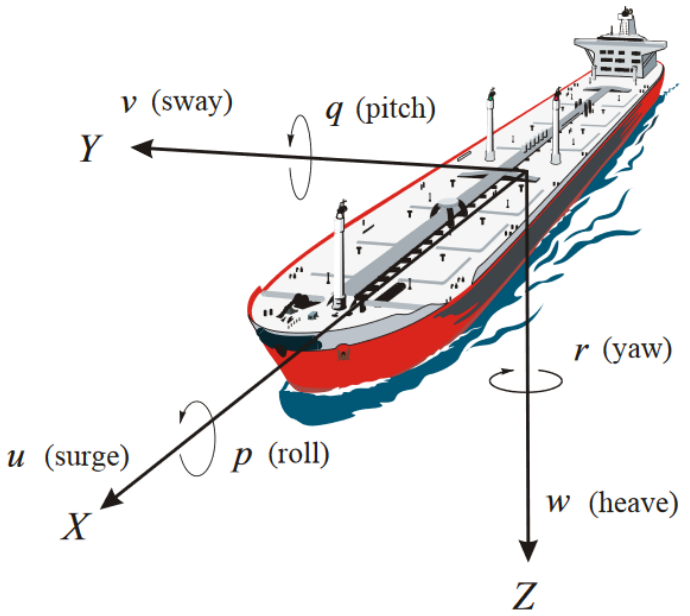


Figure 2.1: Displacement vessel with body-frame axes and velocities. Source: [4].

- $\boldsymbol{\nu} \in \mathbf{R}^{6\times 1}$ is the velocity given in the body-frame.
- $\boldsymbol{\Theta}_{nb} = \begin{bmatrix}\psi & \theta & \phi\end{bmatrix}^\top$ are the Euler angles from body-frame to NED.
- $\mathbf{R}_b^n(\boldsymbol{\Theta}_{nb}) \in \mathbb{R}^{3\times 3}$ is the Euler angle transformation matrix, that transforms a body-frame vector to the NED-frame. This is done by three sequential principal rotations, in the order ψ-θ-ϕ (the zyx-convention). $\mathbf{R}_b^n$ is a member of the special orthogonal group SO(3). We have that

$$\begin{aligned}\mathbf{R} \in SO(3) \implies \mathbf{R}^\top\mathbf{R} = \mathbf{R}\mathbf{R}^\top = \mathbf{I}_3 \quad \wedge \quad \det\mathbf{R} = 1 \\ \implies \mathbf{R}^\top = \mathbf{R}^{-1},\end{aligned}$$

Note also that $(\mathbf{R}_b^n)^{-1} = \mathbf{R}_n^b$. That is, the inverse of the rotation matrix transforming vectors from b to n is the rotation matrix that transforms vectors from n to b

A comment on $\mathbf{R}_b^n(\mathbf{\Theta}_{nb})$ is warranted. In the zyx convention common with Euler angle representations, we have the undesirable property that a singularity exists when the pitch angle θ is $\pm 90^\circ$. For surface vessels this bears little concern since they do not operate in proximity to the singularity. Also, as we shall soon see, pitch is often wholly neglected when modelling surface vessels. For the general case, an alternative to the Euler angle representation is the quaternion. This representation avoids the aforementioned mathematical singularity, at the expense of not being as intuitively understandable. Furthermore, quaternions are computationally easier, which makes them preferable in computer applications. An algorithm from [17] can be used to calculate quaternions from known Euler angles, and a rotation matrix using quaternions can be defined.

In the general case, there is coupling in all DOFs. However, by exploiting symmetry in the ship longitudinal plane, one can decouple surge from sway and yaw. Moreover, if the ship possesses longitudinal and lateral metacentric stability, and if we assume that heave, pitch and roll motions are small, one can neglect these modes, and thus obtain a surge-decoupled, 3-DOF model applicable to surface vessels. In this case, only horizontal motion in surge, sway and yaw are considered. Since we are concerned with surface vessels, only 3-DOF models are considered henceforth. Under these assumptions the variables in the kinematic equation reduces to

$$\boldsymbol{\eta} \equiv \begin{bmatrix} x^n & y^n & \psi \end{bmatrix}^\top \tag{2.2}$$

$$\boldsymbol{\nu} \equiv \begin{bmatrix} u & v & r \end{bmatrix}^\top \tag{2.3}$$

$$\mathbf{R}_b^n(\mathbf{\Theta}) \equiv \mathbf{R}_b^n(\psi). \tag{2.4}$$

For details see e.g [18]. In the following, we write $\mathbf{R}$ instead of $\mathbf{R}_b^n(\psi)$ and $\mathbf{R}^\top$ instead of $\mathbf{R}_n^b(\psi)$ for readability.

2.2 Sensor models

The positioning systems of marine craft require measurements from various sensors. These typically include (but are not limited to) inertial measurement units (IMUs), with gyro- or magnetic compasses, and GNSS. These sensors are briefly presented below, based on material from [18], Ch. 11, and [19], Ch. 7. As mentioned, this is a modified and extended chapter from the project thesis. The presented models have been used to create sensor FMUs, with selected failure modes.

2.2.1 IMU

IMUs typically contain accelerometers and rate gyros, and are often based on micro-electromechanical systems (MEMS) technology. Accelerometers measure specific force, which is used to provide the acceleration in surge, sway and yaw. The rate gyros are

used to measure the angular rate in roll, pitch and yaw. Measurements from IMUs can be modeled such that the output is the sum of the true value, the sensor bias vector $\boldsymbol{b}$ and the additive noise vector $\boldsymbol{w}$ [20]. Thus, for the accelerometer and rate gyro we have the respective models

$$\boldsymbol{a}^b_{\text{imu}} = \mathbf{R}^\top(\dot{\boldsymbol{\nu}}^n + \boldsymbol{g}^n_0) + \boldsymbol{b}^b_{acc} + \boldsymbol{w}^b_{acc} \tag{2.5}$$

$$\boldsymbol{\omega}^b_{\text{imu}} = \boldsymbol{\omega}^b + \boldsymbol{b}^b_{\text{gyro}} + \boldsymbol{w}^b_{\text{gyro}}, \tag{2.6}$$

where $\dot{\boldsymbol{\nu}}^n$ is the linear acceleration in NED-frame, $\boldsymbol{g}^n_0$ the gravity vector in NED and $\boldsymbol{\omega}^b$ the angular rate vector expressed in body-frame. The the $\boldsymbol{w}$-vector is additive zero-mean noise, used to model the rapidly fluctuating part of the measurement. The cause of the noise in MEMS accelerometers comes from molecular agitation of the material, which affects the moving parts of the accelerometer. The $\boldsymbol{b}$-vector represents sensor bias that are due to the intrinsical physical properties of the sensor: MEMS architecture is sensitive to changes in its physical parameters, which may be brought about from varying temperature. This introduces bias in the sensor output. The bias is typically modeled as a slowly varying process, which we can mathematically represent by a Wiener process, namely

$$\dot{\boldsymbol{b}}_i = \sigma \boldsymbol{w}_i \tag{2.7}$$

$$\boldsymbol{w}_i \sim \mathcal{N}(0, 1), \tag{2.8}$$

where σ is the strength of the noise (noise density).

IMUs also sometimes include compass functionality, by providing orientation data through use of gyroscopes or magnetometers.

2.2.2 Compass

Compasses are used to provide a measurement of the heading of the craft. A magnetic compass measures the strength of the magnetic field along three orthogonal axes, which can be used to calculate heading. The model is

$$\boldsymbol{m}^b_{\text{imu}} = \mathbf{R}^\top \boldsymbol{m}^n + \boldsymbol{b}^b_{mag} + \boldsymbol{w}^b_{mag}, \tag{2.9}$$

which is similar in structure to the above models. $\boldsymbol{m}^b_{imu}$ is the vector of the measured strength of the magnetic field. The vector $\boldsymbol{m}^n$ is the true strength of the magnetic field, $\boldsymbol{b}^b_{mag}$ is the bias vector, also modelled as a Wiener process, and $\boldsymbol{w}^b_{mag}$ the noise. The $\boldsymbol{b}^b_{mag}$ vector does not model bias per se, but the local magnetic disturbance. This disturbance can be quite significant, as it is influenced by local magnetic fields from e.g electro-motors. Assuming $\boldsymbol{m}^b_{\text{imu}}$ has been filtered to remove noise and bias, and that $\phi \approx 0, \theta \approx 0$, we can obtain the measured heading ψ_m as

$$\psi_m = -\text{atan2}(m_y, m_x) \tag{2.10}$$

where atan2 restricts ψ_m to $[-\pi, \pi]$ by taking into account the sign of m_x and m_y. If the roll and pitch angle are significant, $\boldsymbol{m}$ must first be transformed to the horizontal plane.

Since the Earth's magnetic pole differs from the location of true north, the declination angle d must be added to ψ_m to get the heading ψ;

$$\psi = \psi_m + d. \tag{2.11}$$

Moreover, since the Earth's magnetic field varies over time, d will change for a given location. Values for d should therefore be retrieved from updated sources.

2.2.3 GNSS

Global Navigation Satellite Systems (GNSS) provides position information based on measurements from satellites. Examples of such systems include the American Global Positioning System (GPS), the Russian Global Navigation Satellite System (GLONASS) and the European Galileo system; the former of which is most common in marine craft. The crucial element of satellite navigation lies in measuring the time-of-flight of a signal between a GNSS receiver and a satellite. This gives an estimate of the distance to the satellite – termed the pseudo-range – because synchronization errors exist between satellite and receiver clock. Thus, to determine the three-dimensional position of a GNSS receiver, we need one measurement for each of longitude, latitude and height above the Earth, and finally a fourth to account for clock offset. This results in a set of 4 non-linear algebraic equations in four unknowns. The 24 satellites connected to the GPS are arrayed in a constellation designed to provide coverage of every point on Earth by at least 4 satellites. Hence this set of equations is usually solvable. Note that for surface vessels, only three measurements are required since height above sea-level is known. GNSS satellites transmit continuous signals with operating frequency of 1-2 GHz (denoted the L-band). These signals are modulated by a pseudo-random noise (PRN) code that carry information used by the receiver to determine travel time. The signal is further modulated by a data message containing satellite data. The receiver will attempt to lock on to GNSS signals, and process these to extract user position, velocity and clock time.

The GNSS sensor model is chosen to be represented as a true position vector $\boldsymbol{p}$ corrupted by a slowly varying bias vector $\boldsymbol{b}_{gnss}$

$$\boldsymbol{p}_{gnss} = \boldsymbol{p} + \boldsymbol{b}_{gnss} \tag{2.12}$$

$$\dot{\boldsymbol{b}}_{gnss} = \sigma_{gnss}\boldsymbol{w}_{gnss}, \tag{2.13}$$

with the bias vector modelled as a Wiener process. In the modelled receiver, the bias was chosen to give drift on the order of centimetres, as this is the precision of a typical GNSS Real-time Kinematic (RTK) receiver. By approximating the Earth as a sphere, rather than an oblate spheroid, with circumference 40075 km, we get that one degree of latitude corresponds to a meridian length of 111319 m. The meridian length of one degree of longitude depends on the latitude θ such that the length l is

$$l = \frac{40075 \cdot 10^3 \cdot \cos\theta}{360^\circ}\text{m}. \tag{2.14}$$

Thus, at Trondheim's latitude of about 10.4° we get a meridian arc of 49844 m. Another way of looking at it is that a 1 m change in longitudinal/latitudinal position corresponds to

a change in longitude/latitude of about $0.00002°/0.00001°$, which we can use to give the desired drift.

2.2.4 Sensor failure modes

Sensors may fail to work properly under certain failure modes. This kind of failure is known from industrial experience to be the most frequent source of system failure [4]. Below, a selection of common sensor failure modes are presented.

- Signal range failure: A situation may occur where sensor output is outside the defined range of the signal.
$$x[k] \in \{x_{min}, x_{max}\}, \tag{2.15}$$
- Wildpoint: A wildpoint measurement is when the sensor signal deviates significantly from previous measurements. This is quantified by the signal acceptance criteria
$$x[k] \in \{\bar{x}_k - a\sigma, \bar{x}_k + a\sigma\}, \tag{2.16}$$
where $\bar{x}_k$ is the signal mean estimated from the previous measurements, σ is the standard deviation, and the constant a is chosen to lie in the interval $[3, 9]$.
- Variance: High or low variance may signify a sensor failure mode. High variance can be caused by a large amount of measurement noise, while low variance can indicate a frozen signal.
- Blackout: Complete failure of the sensor results in blackout - no sensor measurements are available.

A number of causes for GNSS errors have been identified [21]. These causes include satellite clock and drift, intentional and unintentional interference, iono- and tropospheric errors and multipath errors. Intentional interference may take the form of jamming to force loss of lock, or injection of a spurious GNSS signal (spoofing). Error consequences can be the complete loss of signal or introduction of large errors in calculated position. Large errors can be modeled by increasing the sensor variance. A highly realistic (co-)simulation of a GNSS receiver in the chosen platform is interesting. This would allow one to easily generate spurious GNSS signals, which could be used in tests of the cyber-security of the autonomus system. Implementing such a model would quickly become out of the scope of this thesis, however.

2.3 Reinforcement learning

Reinforcement learning (RL) is the third of the most commonly considered machine learning paradigms, along with supervised and unsupervised learning. In RL, a software agent learns how to behave in an environment through a scalar feedback reward signal. The agent receives rewards when it performs actions, and the overarching goal is to find the set of actions that maximize rewards in the long run. Figure 2.2 illustrates an agent interacting with the environment through action a, which causes it to transition between states

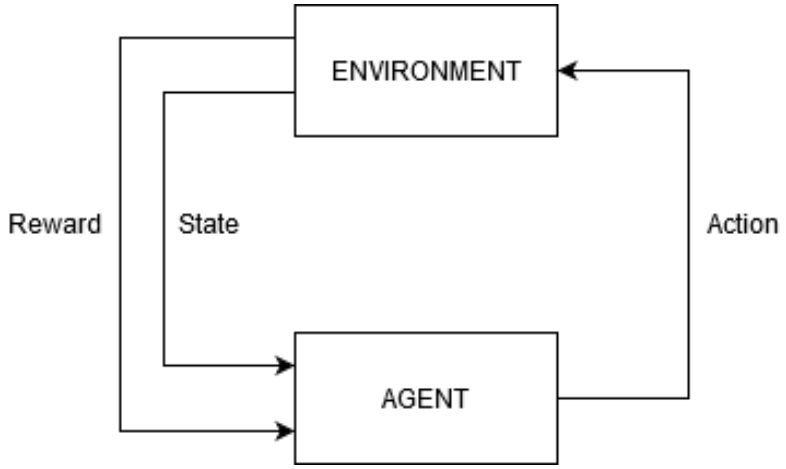


Figure 2.2: A simplified RL scenario.

and obtain a reward. Learning usually happens across episodes, which are sequences of agent-environment interactions starting in state s_0 and ending in a terminal state s_T. The next episode restarts in s_0, with the knowledge gained from previous episodes. The terms ”agent”, ”environment” and ”actions” used in ML are analogous to the respective terms ”controller”, ”plant” and ”control signal” commonly used in control theory. Unless otherwise stated, the theory in this subsection is based on [22] and lecture notes from the specialization module TTK23 – Introduction to Autonomous Robots for Industry 4.0, taught at NTNU in the fall semester of 2019.

2.3.1 Markov Decision Processes

The environment inhabited by the agent is usually described as a Markov Decision Process (MDP). MDPs are defined by the 4-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$, where

- $\mathcal{S}$ is the state space, which uniquely characterize important aspects of the modeled problem. A distinguishment is made between legal and illegal states, referring to whether the agent can exist in that state or not.
- $\mathcal{A}$ is the action space, which define how the agent can interact with and control the environment. The set of allowable actions may be a function of the state. We denote the set of actions allowable in state s as $\mathcal{A}(s)$.
- $\mathcal{T}$ denotes the state-transition probabilities, given by

$$p(s', r \mid s, a) \doteq \Pr\{S_t = s', R_t = r \quad | \quad S_{t-1} = s, A_{t-1} = a\}, \tag{2.17}$$

  which describe the probability of transitioning from state s to the next state s’ by taking action a (in state s). Transition functions are regular probability distributions;

we have

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r \mid s, a) = 1, \quad \forall s \in \mathcal{S}, a \in \mathcal{A}(s). \tag{2.18}$$

Note that the transition function only cares about the current state; p defines the dynamics of the environment. In other words, the current state – as opposed to the sequence of previous states and actions – holds all information relevant for future states. This is called the Markov property.

- $\mathcal{R}$ are the rewards found from the reward function,

$$r(s, a, s') \doteq \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a, S_t = s'] \tag{2.19}$$

which gives a scalar reward for moving from state s to state s' through action a. It serves as an implicit specification of the goal of learning. It may also include smaller rewards for sub-goals of the learning task.

Note that no prior knowledge of the environment is required. RL algorithms are tasked with interacting with the environment to uncover the best course of action, being guided by reward feedback. These algorithms need to sample and explore the environment, because they operate from an initial state of ignorance. At the same time, one also wishes to use the information gleaned from previous interactions. This illustrates the trade-off in exploration (of the environment) vs. exploitation (of current knowledge).

2.3.2 Value functions and policies

Central to reinforcement learning lies the notion of value functions and policies. Value functions give an estimate of the expected reward when starting from state s and subsequently following some policy π. Policies are mappings from states to action probabilities; $\pi(a|s)$ is the probability of choosing action a in state s. Since π is a probability distribution, $\sum_{a \in \mathcal{A}(s)} \pi(a|s) = 1$. We denote the state-value function as $V^\pi(s)$, which is the value function when starting in state s and then following π. A similar action-value function is denoted $Q^\pi(s, a)$. It differs from V^π in that one takes action a in state s, and *thereafter* follows π. Formally we can define the value functions as the expectation of the return G_t,

$$V^\pi(s) \doteq \mathbb{E}\left[G_t \mid s_t = s\right] = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\right] \tag{2.20}$$

$$Q^\pi(s, a) \doteq \mathbb{E}\left[G_t \mid s_t = s, a_t = a\right] = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a\right]. \tag{2.21}$$

Here we have specified the return as the infinite horizon, time-discounted sum of all future rewards,

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}. \tag{2.22}$$

It is implied that all future rewards proceeding from state s are obtained by following π. The parameter $\gamma \in [0, 1]$ is the discount factor, used to discount the value of future rewards.

For $\gamma = 0$ the agent is completely myopic and is only concerned with immediate rewards. If $\gamma = 1$ then the long term reward becomes important. A recursive relationship is easily obtained for the return G_t, namely

$$\begin{aligned} G_t &= r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... \\ &= r_{t+1} + \gamma \left(r_{t+2} + \gamma r_{t+3} + ... \right) \\ &= r_{t+1} + \gamma G_{t+1}. \end{aligned} \tag{2.23}$$

By inserting Equation 2.23 in the given equations for the value function, we can get a similar recursive relationship. For the state-value function, this becomes

$$\begin{aligned} V^\pi(s) &= \mathbb{E}\left[r_{t+1} + \gamma G_{t+1} \mid s_t = s \right] \\ &= \sum_{a \in \mathcal{A}(s)} \pi(a|s) \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r \mid s, a) \left\{ r + \gamma \mathbb{E}\left[G_{t+1} \mid S_t = s \right] \right\} \\ &= \sum_{a \in \mathcal{A}(s)} \pi(a|s) \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r \mid s, a) \left[r + \gamma V^\pi(s') \right]. \end{aligned} \tag{2.24}$$

Now, when following an optimal policy π^*, the corresponding optimal state-value function V^* satisfies

$$V^*(s) \geq V^\pi(s), \quad \forall s \in \mathcal{S}. \tag{2.25}$$

Moreover, the optimal state-value function is

$$V^*(s) \doteq \max_\pi V^\pi, \quad \forall s \in \mathcal{S} \tag{2.26}$$

$$= \max_{a \in \mathcal{A}(s)} \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r \mid s, a) \big[r + \gamma V^*(s') \big], \quad \forall s \in \mathcal{S} \tag{2.27}$$

Equation 2.27 is the Bellman optimality equation for V^π. Given the optimal value function, one may obtain the optimal action by

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}(s)} \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r \mid s, a) \left[r + \gamma V^*(s') \right]. \tag{2.28}$$

This is because the optimal value function already considers future choices, hence making the locally optimal choice (termed the greedy choice) is also long-term optimal. Note that finding the optimal action from the value function requires looking ahead one step, by taking the sum over the next states s'.

Similar results exist for the action-value function. In particular.

$$Q^*(s, a) = \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r \mid s, a) \left[r + \gamma \max_{a'} Q^*(s') \right] \tag{2.29}$$

and

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}(s)} Q^*(s, a). \tag{2.30}$$

Equation 2.30 shows us that action-value functions allows us to compute the optimal policy by finding the actions that maximizes Q locally. No one-step-ahead search is needed here. This is a nice property, because it allows one to look up optimal action choices without knowing anything about the environment.

2.3.3 Temporal credit asssignment

The utility of being in a given state is difficult to assess if its reward is only apparent much later. A possible solution is to adjust the estimated state value using the immediate reward and the estimated discounted value of the next state. Thus new estimates are formed from older estimates, a concept called bootstrapping. This solution strategy is called temporal difference learning. An example implementation is the update rule of the simple temporal difference algorithm TD(0):

$$V(s) \leftarrow V(s) + \alpha\big[r + \gamma V(s') - V(s)\big], \tag{2.31}$$

where $\alpha \in (0, 1]$ is the learning rate parameter, which quantifies how heavily we wish to weight new information. Note that the term $r + \gamma V(s')$ is the new information, while $V(s)$ is the old information. Thus α determines the degree to which the new information overwrites the old.

2.3.4 Reward signals

For the RL agent to learn how to behave in accordance with the designer's intention, it is crucial that the reward signal is properly constructed. Since the agent optimizes some defined return which depends on the rewards, the optimized behaviour is with respect to the reward signals. Path-finding in simple environments like a gridworld is an example in which construction of this signal is fairly straightforward. A large reward whenever the goal is reached is adequate. To avoid meandering on the path towards goal, a small negative reward for each visited state can also be given. In more complex environments with harder tasks, the construction of rewards is not trivial. For the path-finding example, in environments with many possible states a single large reward for reaching goal is not a good option. If the goal is far removed from the initial state, the agent has no guiding principles directing it towards the goal, and relies solely on random experimentation to eventually reach goal. This is the problem with so-called sparse rewards. Another possibility is to give smaller rewards for accomplishing sub-goals related to the main goal, effectively guiding the agent towards goal. This runs the risk of introducing unintended behaviour in the agent, as it may discover unexpected ways to receive rewards. The notion of guiding the agent may be better accomplished by providing it with an approximation of the optimal value function, before running the learning algorithm.

2.3.5 Q-learning

The Q-learning algorithm was first introduced by Chris Watkins in [23]. It is a temporal-difference, off-policy reinforcement learning method, that incrementally updates the eponymous Q-table, which approximates the optimal action-value function Q^*. The invention of the algorithm was a small revolution in RL as it possessed characteristics that simplified analysis and construction of convergence proofs for the algorithm. Furthermore, it converges to the optimum in the limit, provided a state-action pair can be visited infinitely many times and that the learning rate is decreased appropriately. The algorithm is given in

pseudocode below.

Algorithm 1: Q-learning

Data: Learning parameter α, discount rate γ, starting state s_0, arbitrarily initialized Q-table.

1 **foreach** episode **do**
2 $s \leftarrow s_0$
3 **repeat**
4 Choose action $a \in \mathcal{A}(s)$ using a policy derived from Q
5 Perform action a, observe the next state s' and reward r
6 $Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a' \in \mathcal{A}(s')} Q(s',a') - Q(s,a) \right]$
7 $s \leftarrow s'$
8 **until** s is terminal;
9 **end**

The parameters α and γ are the already encountered parameters for the learning rate and the discount factor.

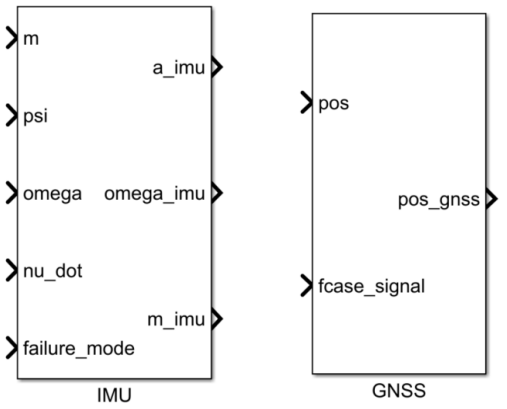
We recognize the similarity of the update rule in line 6 with the one in Equation 2.31. Note however that this update does not depend on the actual action taken – the locally optimal action is used instead. This is what is meant by an off-policy method. The action taken matters in that it determines the states visited, however.

In line 4, a policy derived from the Q-table is required. A common choice is the ϵ-greedy policy, in which the greedy (locally optimal) action is chosen with probability ϵ, and a random action is chosen with probability $(1-\epsilon)$. ϵ is determined according to how one wishes to balance exploration and exploitation. Note that some authors define the ϵ-parameter differently, taking $(1-\epsilon)$ as the probability of choosing the greedy action. This parameter may also be adjusted along the way - it is possible to start the algorithm with an ϵ that favours exploration, and gradually increasing it as the agent becomes better at performing its task.

Chapter 3

Implementation of sensor FMUs

The presented IMU and GNSS sensor models were implemented in Simulink and exported as FMUs with the freely available Simulink library FMIKit, courtesy of copyright holders Dassault Systèmes. In general the sensor models used consist of a true value corrupted by noise and bias. In the FMU implementation, it is assumed that discretely sampled true values are available, and used as input to the FMU block. The black-box sensor FMUs are seen below in Figure 3.1. Inputs are the true measurements and a signal representing which failure mode, if any, is triggered. For each failure mode a separate system with that failure mode is created. A switch case block determines which of these systems are executed. An overview of how this looks for the GNSS sensor in Simulink is seen in Figure 3.12. Details on the Simulink implementation for the IMU and GNSS follow.



(a) The IMU FMU block. **(b)** The GNSS FMU block.

Figure 3.1: The sensor FMU blocks.

3.1 IMU

The IMU installed on milliAmpère is the Xsens MTi-10. Some relevant specifications are given in Table 3.1. The given noise density parameters for each sensor was applied for each of the sensor's three axes. To implement a discrete sensor model, the bias and noise must be discretized. The bias process becomes

$$\boldsymbol{b}_d[k] = \boldsymbol{b}_d[k-1] + \sigma_b d\boldsymbol{w}[k] \tag{3.1}$$

$$\sigma_{bd} = \sigma_{bc}\sqrt{T_s} \tag{3.2}$$

$$\boldsymbol{w}[k] \sim \mathcal{N}(0,1), \tag{3.3}$$

while the discretized noise process is

$$\boldsymbol{w}_d[k] = \sigma_d \boldsymbol{w}[k] \tag{3.4}$$

$$\sigma_d = \sigma_c \frac{1}{\sqrt{T_s}} \tag{3.5}$$

$$\boldsymbol{w}[k] \sim \mathcal{N}(0,1) \tag{3.6}$$

The parameter σ_c is the typical noise densities given in Table 3.1. No data was given for σ_{bd}, and was set to $0.01\frac{\text{m}}{s^3\sqrt{\text{Hz}}}$ for the accelerometer, $0.05\frac{\text{rad}}{\text{s}^2\sqrt{\text{Hz}}}$ for the gyro and 0 for the magnetometer. The sampling frequency for the IMU sensor installed on milliAmpere is 1000 Hz, thus sampling time T_s was set to 0.001 s.

Table 3.1: Xsens MTi-10 specifications

Sensor specifications						
	Accelerometer		Gyroscope		Magnetometer	
	Typical	Max	Typical	Max	Typical	Max
Standard full range	$450°/\text{s}$	-	$50\text{m}/\text{s}^2$	-	-	+/- 80μT
Noise density	$\frac{0.03°/\text{s}}{\sqrt{\text{Hz}}}$	$\frac{0.05°\text{s}}{\sqrt{\text{Hz}}}$	$\frac{80\mu g}{\sqrt{\text{Hz}}}$	$\frac{80\mu g}{\sqrt{\text{Hz}}}$	$\frac{200\mu\text{G}}{\sqrt{\text{Hz}}}$	-
In-run bias stability	$18°/\text{h}$	-	$40\mu\text{g}$	-	-	-

System specifications	
Sampling frequency	10 kHz/channel
Max output frequency	2 kHz

The IMU was implemented with the fault cases i) wildpoint, ii) high signal variance, iii) frozen signal and iv) drifting bias. For each failure mode a separate system with that failure mode is created. A switch case block determines which of these systems are executed.

i) Wildpoints were created by keeping a running score of the signal mean and variance, and adding a value to the measurement before outputting it, such that the output lies a standard deviations away from the mean.

ii) High signal variance was implemented by simply adjusting the parameters governing the noise, indicated in the block diagram.

iii) A frozen signal was implemented by outputting the last measured value for all timesteps after the fault is triggered.

iv) Drifting bias was implemented by multiplying the output from a digital clock and a step function, creating a staircase function with equally spaced jumps of equal length. This is then added to the measurement. If the fault is triggered after $t = 0$, the time elapsed until sensor fault is subtracted from the clock measurement before multiplication.

Figures 3.2-3.5 show an overview of the default (meaning without failure modes) IMU system and subsystems in Simulink. In Figure 3.3, the upper part corresponds to the noise in the sensor model, i.e Equation 3.1. The three blocks to the left generate white noise, while the triangular (gain) block multiplies the noise by σ_d-term. The middle blocks represent the first term in Equation 2.5, where the yellow block performs the rotation operation. The lower part makes up the bias term, and is similar to the noise part, with the exception of the discrete integrator block. Figures 3.4 and 3.5 have the same general structure, and correspond to Equations 2.6 and 2.9. For that reason there is no rotation block in the rate gyro subsystem, and no gain blcoks in the magnetometer subsystem.

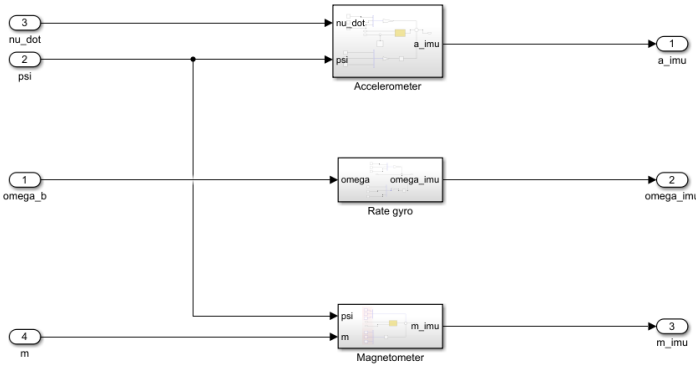


Figure 3.2: Overview of IMU system in Simulink.

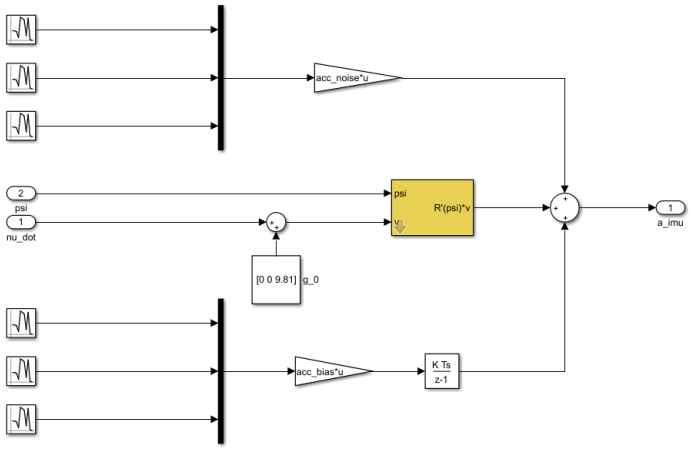


Figure 3.3: Accelerometer subsystem.

Accelerometer readings for with faults are shown in Figures 3.6-3.9, the failure modes being wildpoint, high variance, frozen signal and drift, respectively. Only one selected DOF is shown.

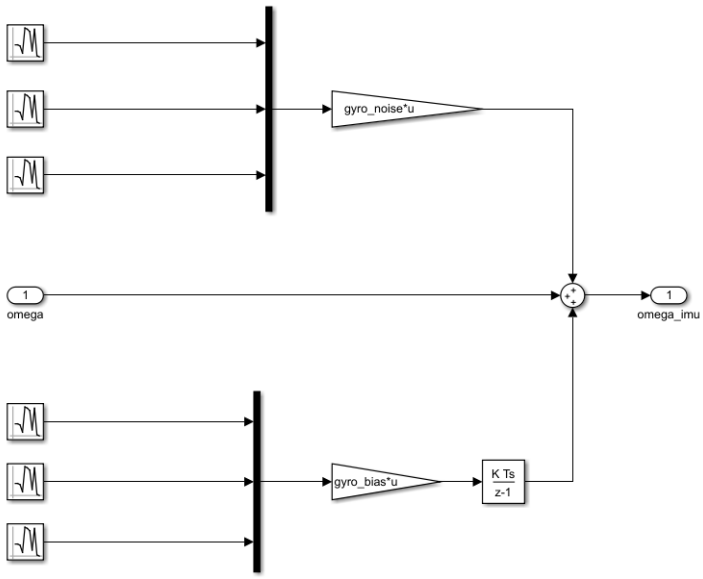


Figure 3.4: Rate gyro subsystem.

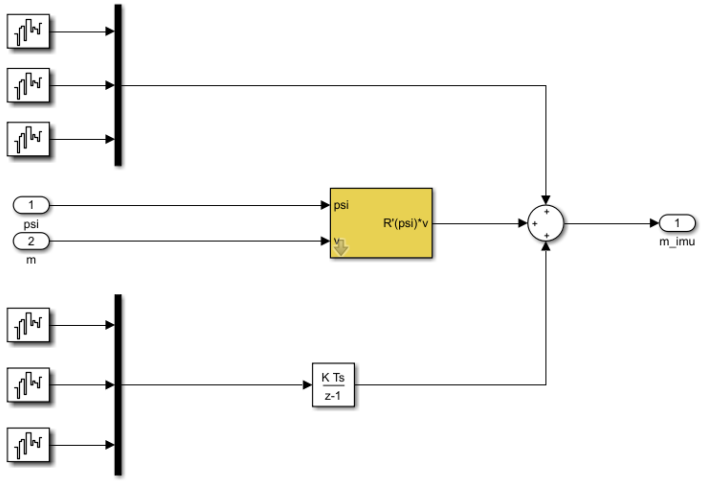


Figure 3.5: Magnetometer subsystem.

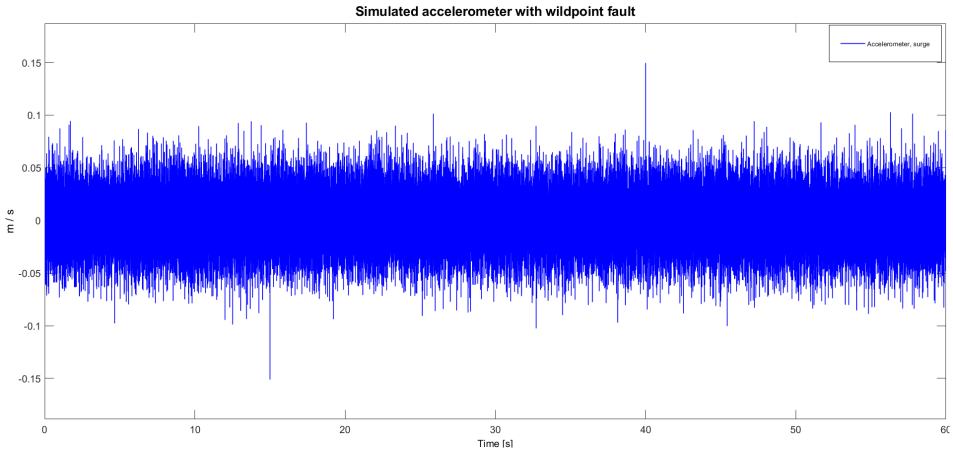


Figure 3.6: Simulated accelerometer with wildpoints at $t = 15$s and $t = 40$s. Surge DOF shown.

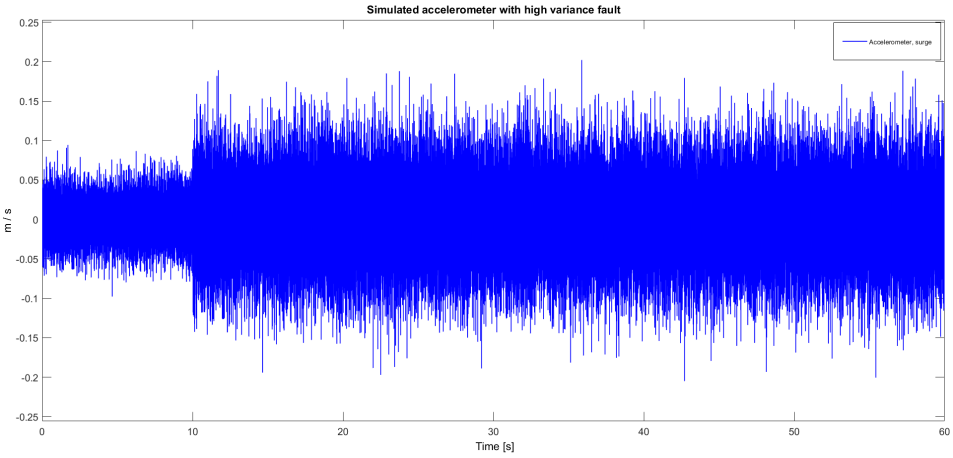


Figure 3.7: Simulated accelerometer with high variance fault triggered at $t = 10s$. Surge DOF shown.

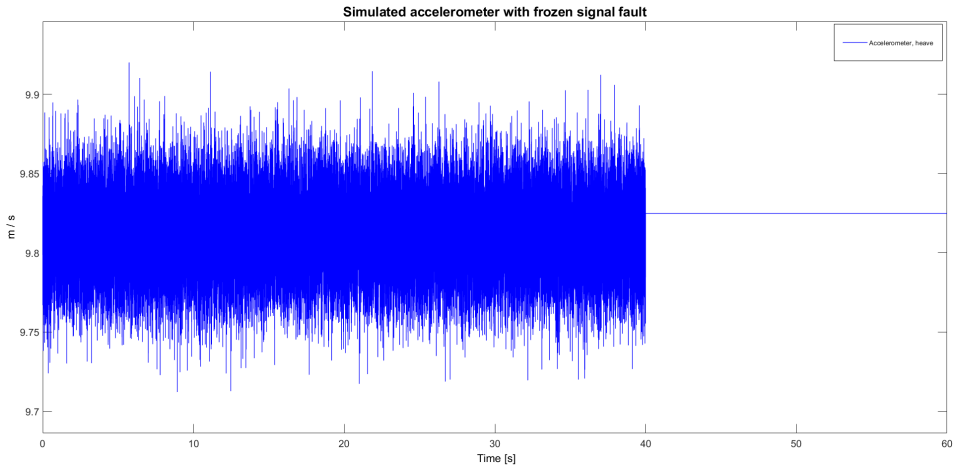


Figure 3.8: Simulated accelerometer with frozen signal fault triggered at $t = 40s$. Heave DOF shown.

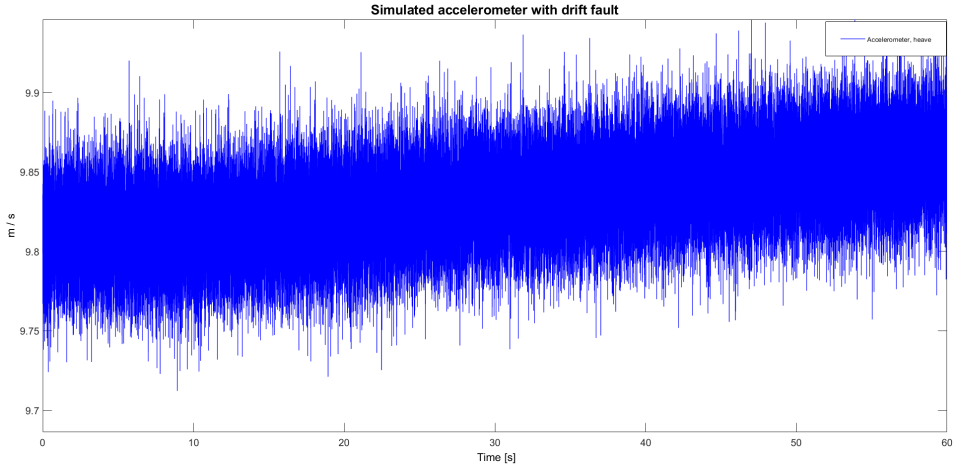


Figure 3.9: Simulated accelerometer with drift triggered at $t = 10$s. Heave DOF shown.

3.2 GNSS

The GNSS sensor was implemented with the discrete model

$$\boldsymbol{p}_{gnss}[k] = \boldsymbol{p}[k] + \boldsymbol{b}_{gnss}[k] \tag{3.7}$$
$$\boldsymbol{b}_{gnss}[k] = \boldsymbol{b}_{gnss}[k-1] + \sigma_{gnss}\boldsymbol{w}[k] \tag{3.8}$$
$$\boldsymbol{w}[k] \sim \mathcal{N}(0,1), \tag{3.9}$$

with the corresponding block diagram in Simulink seen in Figure 3.10. To obtain the desired magnitude of drift, σ_{gnss} was through trial and error set to

$$\sigma_{gnss} = diag\{1.25 \cdot 10^{-7}, 3 \cdot 10^{-8}, 0\}. \tag{3.10}$$

Figure 3.11 shows example GNSS measurements for a vessel stationary at coordinates $63.434°$ N, $10.393°$ E. The measurements have been transformed to a local NED-system with center at the given coordinates to plot the change in measured position due to random drift in meters.

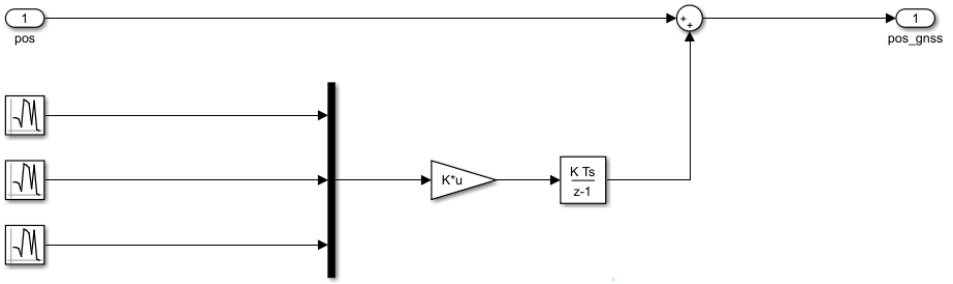


Figure 3.10: Simulink block diagram of implemented GNSS sensor.

An overview of the system is given in Figure 3.12. As the failure modes are essentially the same as those shown for the IMU, further details are omitted for brevity.

Figure 3.11: GNSS measurements for a stationary sensor.

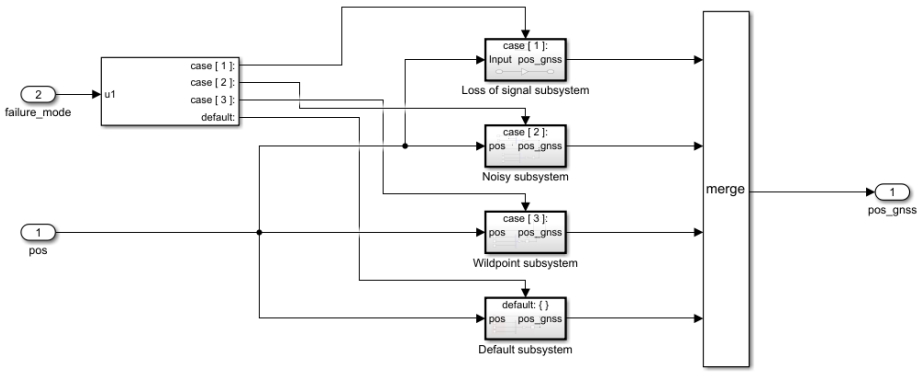


Figure 3.12: Overview of the GNSS sensor in Simulink, with implemented failure modes.

Chapter 4

AIS-Data: Collection, processing and analysis

4.1 Data collection

The data was collected in GeoJSON format from kystdatahuset.no, by specifying the area of interest as the blue region seen in Figure 5.1 and the desired time-frame, which was set to the years 2016-2020 (up to and including May 2020). Due to host server limitations, the data query had to be made for each year separately. Table 4.1 shows the file sizes for each year, as well as the number of geographical features (described below) in each file.

Table 4.1: Table showing GeoJSON file sizes and number of features for each year.

Year	File size	Number of features
2016	9418 KB	22327
2017	10013 KB	25642
2018	9713 KB	24924
2019	20544 KB	53300
2020 (01.20 - 05.20)	1301 KB	3321
Sum	50.99 MB	129514

For generating realistic simulated trajectories of ships in the channel, the data should be representative for ship traffic in the region. Norwegian regulations[1] stipulate that only ships of a certain size are required to have AIS transponders. Thus one may assume that only AIS data does not give a full picture of the ship traffic in the area, since most of it consists of smaller boats with no requirements on AIS transponders. Analysis of the gathered data shows that 125 different ships sailed in the temporal and spatial region of interest, as

[1] See
https://lovdata.no/forskrift/2000-06-13-660/§10-4a
https://lovdata.no/forskrift/2014-09-05-1157/§19

identified from their unique Maritime Mobile Service Identity (MMSI) codes. Of these, only 18 (14.4%) are required by law to have AIS equipment. The rest are predominantly identified as sailing vessels and pleasure crafts. Furthermore, the canal harbour is narrow, with limited room for maneuvering. For these reasons, it is argued that the collected data forms a representative sample of the sailings in the region.

4.1.1 Description of data format

The GeoJSON format describes simple geographical features like points, lines and polygons, and may also store other non-spatial properties. The top level data object is the FeatureCollection, which contains any number of features described by its geometry type and an optional list of properties. The geometry type primitives are Points, LineStrings and Polygons, which are defined by a set of coordinates, or a set whose elements are sets of coordinates. Additionally, there are the geometry types MultiPoints, MultiLineStrings and MultiPolygons defined by a collection of the corresponding geometry primitives. Lastly there is the GeometryCollection type, containing any combination of the other mentioned types. The code example in Figure 4.1 contains GeoJSON code that should be fairly self-explanatory based on the preceding text.

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "LineString",
        "coordinates": [
          [1.0, 7.0], [3, 8.0], [5.0, 8.0], [7.0, 9.0]
        ]
      },
      "properties": {
        "starttime": "0.0",
        "endtime": "1.0"
        "mmsi": "0"
      }
    },
    {
      "type": "Feature",
      "geometry": {
        "type": "MultiLineString",
        "coordinates": [
          [
            [1.0, 4.0], [3.0, 3.0], [5.0, 3.0],
          ]
          [
            [5.0, 4.0], [7.0, 4.0]
          ]
        ]
      },
      "properties": {
        "starttime": "0.0",
        "endtime": "1.0"
        "mmsi": "1"
      }
    }
  ]
}
```

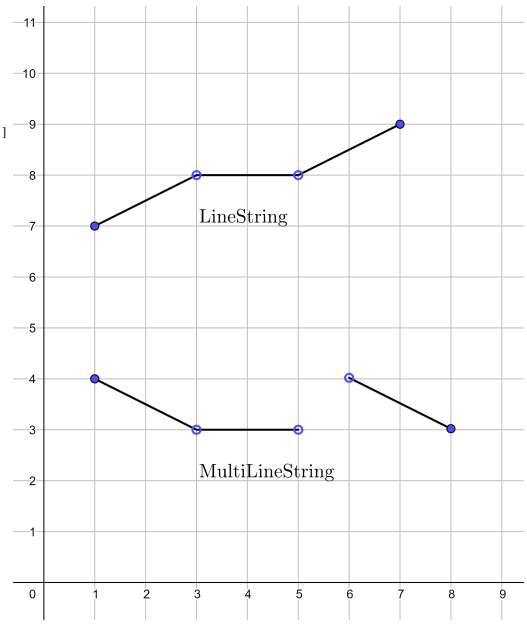


Figure 4.1: Example GeoJSON code in Cartesian coordinates, producing the geometries seen to the right.

4.1.2 Data description

Initially, the GeoJSON data offered from kystdatahuset did not have recorded speeds. This did become available as the semester went on, but not before some work had gone into processing the first dataset. Furthermore, the data was represented somewhat differently, necessitating some changes to the processing code.

The first set of sailing data collected were given as FeatureCollections of LineStrings and MultiLineStrings, with coordinates of each LineString given in chronological order. However, only the start- and endtime of the feature was given, not times for each coordinate. Another aspect was that in many cases the endtime and end coordinate of one (Multi)LineString was identical to the start time and start coordinate of another (Multi)-LineString for the same vessel. Clearly, these LineStrings form a part of the same sailing. For that reason, there is not a one-to-one correspondence between (Multi)LineStrings and a sailing, which would be preferable when analysing the data. Therefore, code for identifying and linking together the LineStrings belonging to the same sailing was written.

The dataset with speeds was essentially the same as the one before, with some differences in data representation. As before, the data was represented in GEOjson format. Each sailing was separated into several LineStrings defined by two points, with the average speed s between these points recorded. Hence all sailings consist of a set of LineStrings, each defined by two points. These LineStrings were not explicitly connected to one another, and had to be linked. Code for processing the dataset from the previous section was rewritten for this purpose. Since this data set contains more information, the first was discarded.

4.2 Data processing

Before further processing, the collected data needed to be cleaned due to spurious and low quality samples. A small number of the collected coordinates placed the boat on land. This is likely caused by inaccuracy in position data, which may be due to signal noise or a fault on the GPS system. By using a visualisation and editing tool that displayed each sailing, these were removed manually. Starting in late October 2019 and continuing for the rest of the year, a massive number of sailings was recorded for one particular vessel. The coordinates appeared to be clustered around a position on the western part of the canal. The registered positions of this vessel from late October 2019 to the end of the year is seen in Figure 4.2. The vessel was identified by MMSI number 257465900 as the DS Hansteen, a museum ship owned and operated by Trondhjem Sjøfartsmuseum. While the ship did indeed sail in the harbour during the summer of 2019, as confirmed from the data set and information from the museum, Figure 4.2 seems to indicate a mostly stationary vessel with faults on position or AIS equipment. As a consequence, all data from DS Hansteen was removed for this period. This also explains the disparity in file sizes seen in 4.1, with the file for 2019 being about twice as large as the others. Removal of the data from DS Hansteen effectively halved the file size. The sailing vessel Anne Margrethe presented similar faulty data as DS Hansteen, having thousands of recorded positions at

essentially the same coordinates and recorded speeds of up to 102.3 knots. This was also removed. Some of the erroneous data was not readily apparent, and was only discovered and removed after running the processing program followed by troubleshooting.

Shuttle boats (MS Nidarholm) depart from Ravnkloa to Munkholmen with tourists and locals very frequently from May to September. Just for 2019, the number of sailings between Ravnkloa and Munkholmen number to at least 1256, according to planned departures[2]. Checking this by counting the number of occurences of each MMSI number confirms this suspicion, as the MMSI of MS Nidarholm has by far the highest count in the data set. Because these scheduled sailings would dominate the traffic density and hence the performance of the algorithms presented later, this data has been omitted from the set.

After processing, the complete dataset for 2016-2020 is visualised in Figure 4.3. This figure represents each sailing as a solid gray line, and as such does not give an impression of the traffic density in the area.

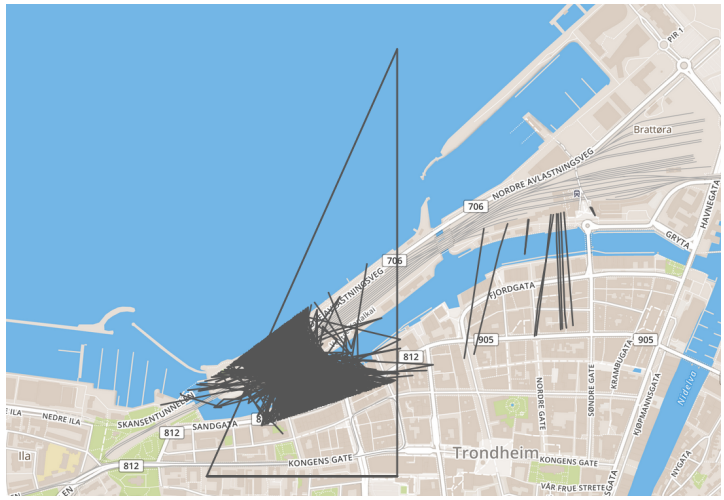


Figure 4.2: Recorded sailings of DS Hansteen for October and November 2019.

[2]https://www.munkholmen.no/velkommen-om-bord

Figure 4.3: Visualisation of entire 2016-2020 dataset post-processing. Each sailing is represented as a solid gray line.

To infer the traffic density, the idea was to overlay a grid on the area of interest, and use the data to record how often ships sailed through a given cell in the grid. For convenience, this work was done by first transforming coordinates from latitude/longitude to a local North-East-Down (NED) coordinate system, with origin defined at 63.43095° N, 10.37723° E. Each sailing is defined by a set of coordinates, and by using the straight line interpolation between pairwise points to estimate the sailing route, one can register the square cells each line crosses as indicated in Figure 4.4. The cell size used here was 1 m x 1 m, providing a fairly high resolution. An overview of the area with the grid overlay is given in Figure 4.6. A zoomed in view with grid spacing of 5 m is seen in Figure 4.7. An additionally zoomed in view of the 1 m x 1 m cells would remove much of the spatial context, and is therefore not shown.

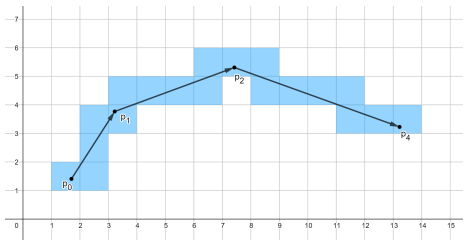


Figure 4.4: A sailing defined by the Cartesian point coordinates $\{p_0, p_1, p_2, p_3\}$, with the interpolated cell positions shown in blue.

A simple map to differentiate water and land areas was created by roughly approximating the land-water boundaries with lines, and assigning a value of -1 and 1 to positions on land and water, respectively. This was used as a filter to remove interpolated lines crossing land. The map superimposed on the real area is seen in Figure 4.5.

Figure 4.5: Map differentiating land and water areas. Land shaded in red, water in blue.

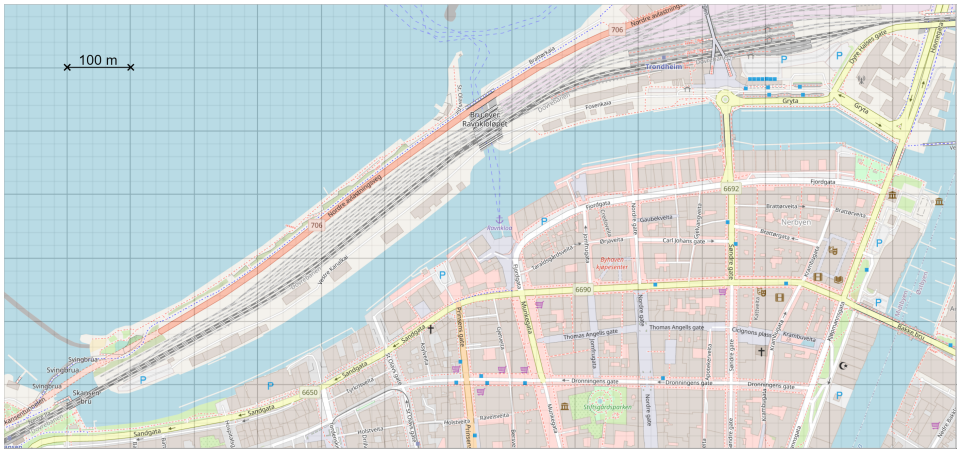


Figure 4.6: 1500 m x 700 m rectangle covering the canal harbour, with major and minor gridlines spaced 100 m and 20 m apart, respectively. The lower left corner located at 63.43095° N, 10.37723° E is the origin of the local NED coordinate system.

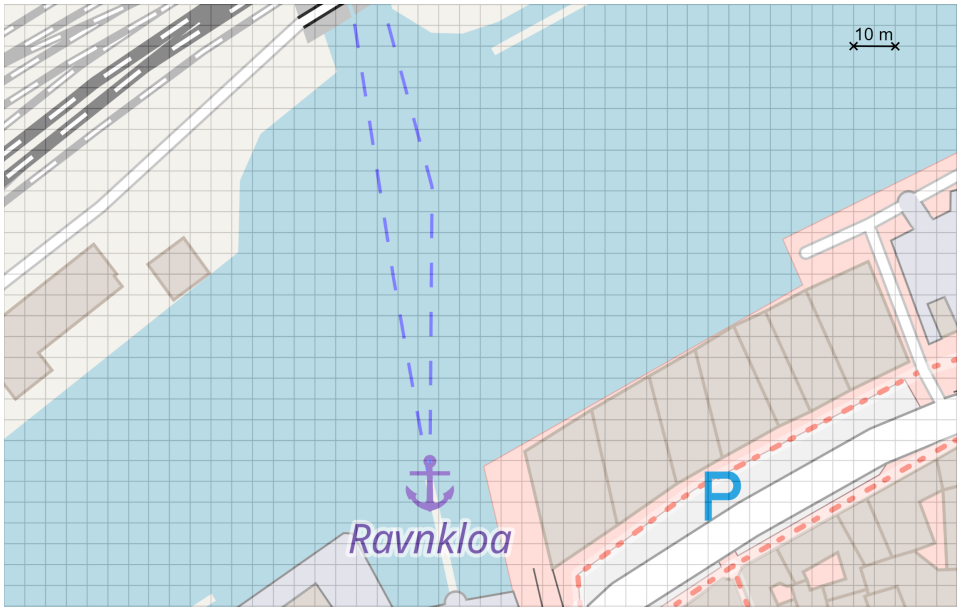


Figure 4.7: Zoomed in view with 5 m x 5 m cells.

4.2.1 Discretization of features

When processing the data, the recorded heading h and speed s were discretized. Speeds were given in the collected dataset as the average speed between the two points in the LineString. The recorded speeds s were mapped to a discrete set of values according to Equation 4.1. Here v_n is the discretized value of the highest recorded speed in knots. For our case, $n = 180$, corresponding to a maximum measured speed of about 45 knots. The minimum speed was 0.6 knots. The recorded heading h was calculated from the starting and ending coordinates of the LineString, and mapped to 32 discrete values by Equation 4.2. The values for the discrete heading ψ_i matches the 32 point compass rose (seen in Figure 4.8). Note that a zero degree heading corresponds to due north.

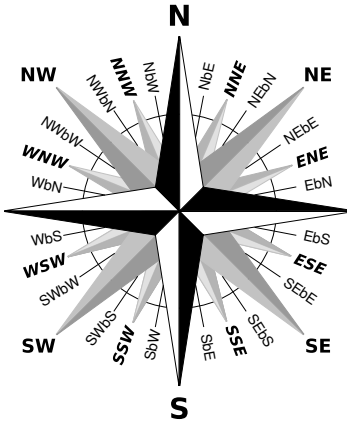


Figure 4.8: The 32 point compass rose with the eight principal winds, eight half-winds and 16 quarter-winds indicated.

$$v(s) = \begin{cases} v_1 = 0.25, \quad 0 < s \leq 0.25 \\ v_2 = 0.5, \quad 0.25 < s \leq 0.75 \\ \vdots \\ v_n = n \cdot 0.25, \quad (n-1) \cdot 0.25 < s \leq (n+1) \cdot 0.25 \end{cases} \tag{4.1}$$

$$\psi(h) = \begin{cases} \psi_1 = 0°, \quad -5.625° < h \leq 5.625° \\ \psi_2 = 11.25°, \quad 5.625° < h \leq 16.875° \\ \vdots \\ \psi_{32} = 348.75°, \quad 343.125° < h \leq 354.375° \end{cases} \tag{4.2}$$

4.3 Results and discussion

This section presents results from the data processing and analysis, including traffic density plots, observations from the data set, selected statistics and possible improvements.

4.3.1 Traffic density plots

In Figure 4.9 we see the traffic density plot from the collected and processed AIS-data. Furthermore, the data set has been subdivided according to start and stop region. For the area around Ravnkloa there are three entrances/exits: west, east and directly north of Ravnkloa. Four subsets were created: Northbound (exiting in the north), southbound (entering from north and stopping near Ravnkloa), westbound (stopping in or exiting the region west of Ravnkloa) and eastbound (stopping in or exiting the region east of Ravnkloa). These subsets were created with a view to generating new routes based on the collected data and desired start and stop region. The density plots for these cases are shown in Figures 4.10-4.13. From these figures it is evident that sailings from west to east, and vice-versa, dominate the traffic.

In Figure 4.11 there appears to be traffic starting and stopping in the middle of the canal. This could be from a number of reasons relating to faults in the AIS-equipment or the logic in the processing program. In the former case, a temporary shut-down of the AIS-system could cause the seemingly sudden appearance of a vessel in the middle of the canal. For the latter case, recall that two separate LineStrings were linked if the start time, coordinates and MMSI numbers of one matched the the stop times, coordinates and MMSI numbers another. Offsets in the times or coordinates would cause them to not be linked. While this is particularly apparent in Figure 4.11 due to the smaller amount of visualised data, it is reasonable to think that this also exists in the other subsets. This is not, however, likely to affect subsequent analysis because the analysis does not depend explicitly on all parts of a sailing being linked. The reason it appears in the figure is an artifact of the east and west boundaries used to demarcate sailings denoted as southbound: all LineStrings within these boundaries and where the start north position is greater than the stop north position is counted in this subset.

Figure 4.12 shows a marked reduction in density to the right, because boats dock to the west and east of the second bridge seen from the right. Additionally, it seems like vessels traveling under the bridge do so mostly along a fairly narrow line. A similar tendency is seen in Figure 4.13, but here the traffic under the bridge is more spread out compared to the previous figure. This is likely due to the increase in traffic density here – perhaps due to more ships with AIS docked here – as seen from the colorbars in the plots. Satellite imagery might explain the travel along the narrow line. Figure 4.14 shows a satellite image of the canal near this bridge, where we see that there are markers guiding passage underneath the bridge.

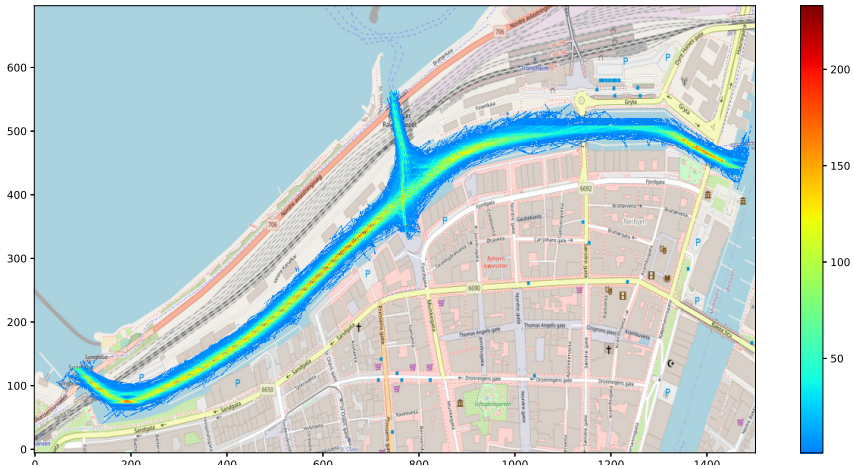


Figure 4.9: Density plot derived from the processed AIS-data in the indicated region from Jan. 2016 - May 2020. The sidebar indicates the number of recorded ships in each cell.

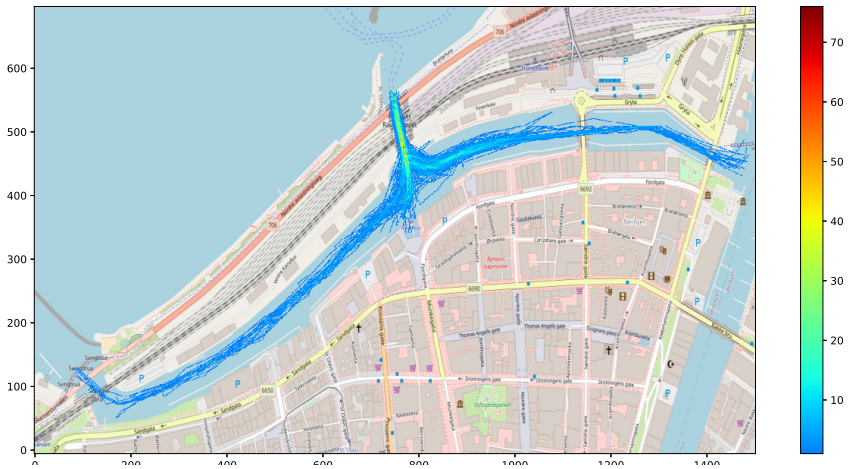


Figure 4.10: Density plot of northbound traffic.

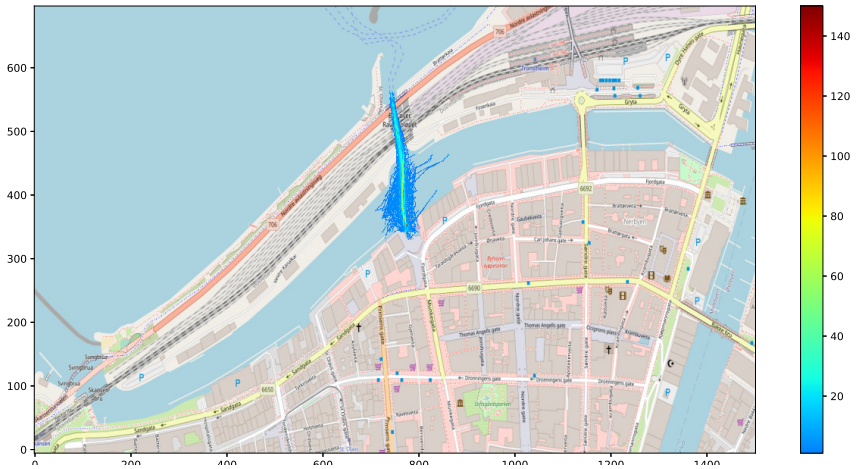


Figure 4.11: Density plot of southbound traffic.

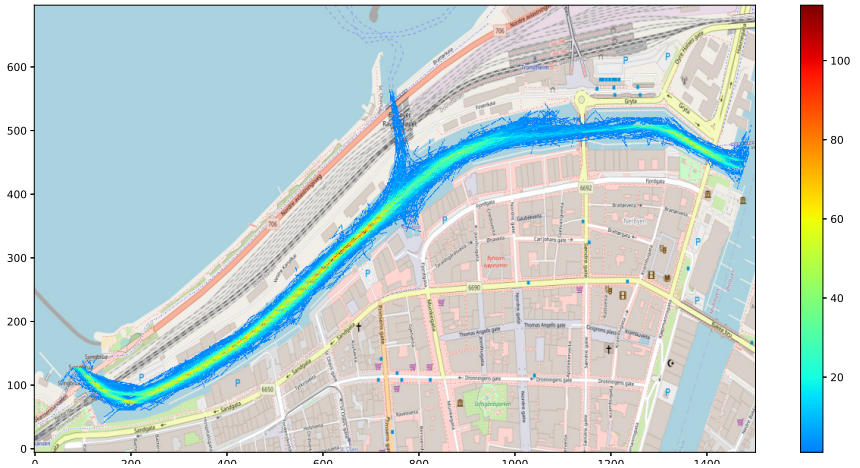


Figure 4.12: Density plot of westbound traffic.

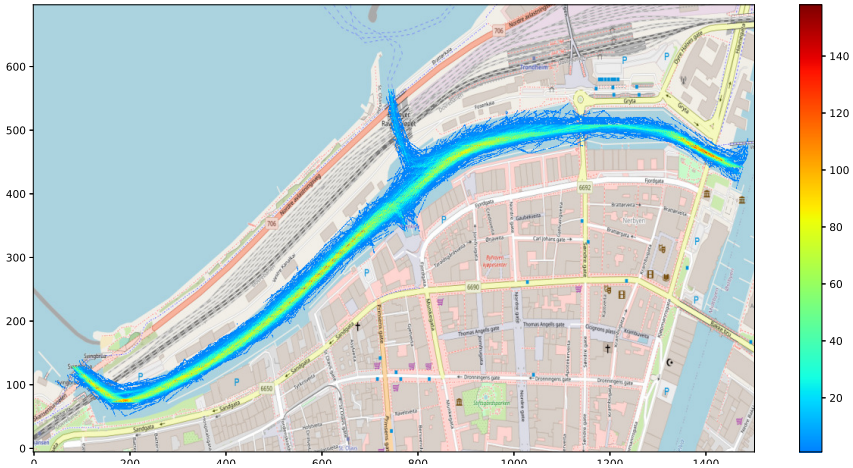


Figure 4.13: Density plot of eastbound traffic.

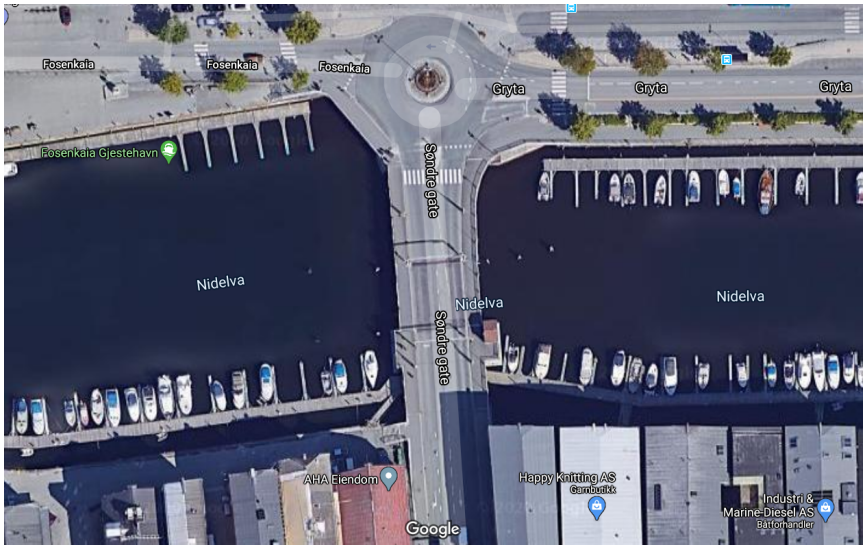


Figure 4.14: Satellite image of the canal near the second bridge.

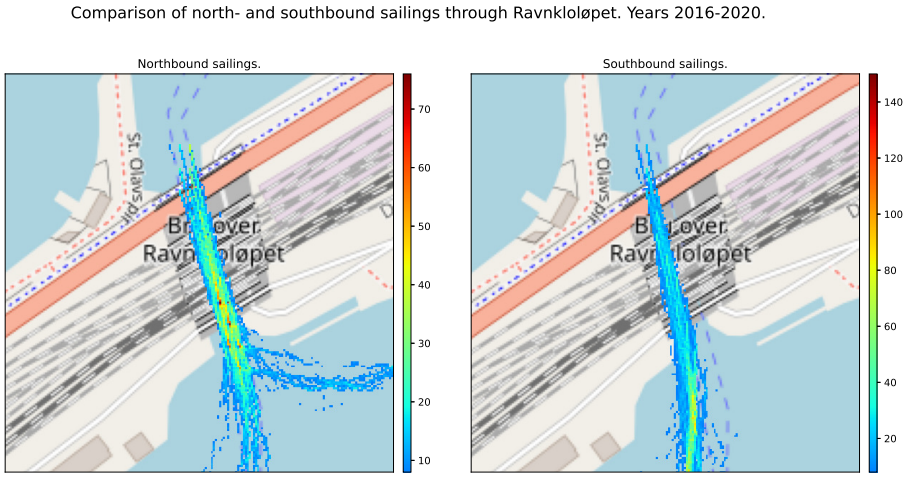


Figure 4.15: Manoeuvring differences through Ravnkloløpet according to direction of travel.

4.3.2 Observations from data set

From the collected data, a few observations were made that are of relevance when intending to use the data for path generation for autonomous ships. These are presented in the following.

Figure 4.15 shows movement in accordance with "Sjøveisreglene"[3], stating that vessels shall pass each other on the port side. Holding to starboard through narrow passages, as we see in the figure, is a corollary to this rule. By inferring from past data, it is possible for autonomous vessels to learn traffic rules, or at least act in accordance with such rules. This is potentially useful since it would not require hardcoding of such rules.

Figure 4.16 shows a marked difference in sailing patterns between 2016 and 2018. Years after 2016 show similar patterns to 2018 but have been omitted for brevity. The reason for this dissimilarity may be an increase in vessels outfitted with AIS equipment, but referring back to Table 4.1 we see that there is little difference between the years (2019 notwithstanding, for reasons discussed). Other causes may be due to pure chance - e.g that fewer vessels happen to be docked in the region in 2016 than later, or that there are constraints on the traffic here. The bridges indicated on the map does not allow for vessels of a certain height to pass without the bridges being raised. Thus infrequent bridge raising may cause little traffic through here. The reasons for these differences are less important than their existence in the first place. It shows that there is *something* belaying passage, which must be considered in path planning based on previous data. In other words, when leveraging historical data for path planning, steps to ensure its validity within the current operational time-frame should be made. This notion can be extended to include other aspects for

[3] https://lovdata.no/dokument/SF/forskrift/1975-12-01-5

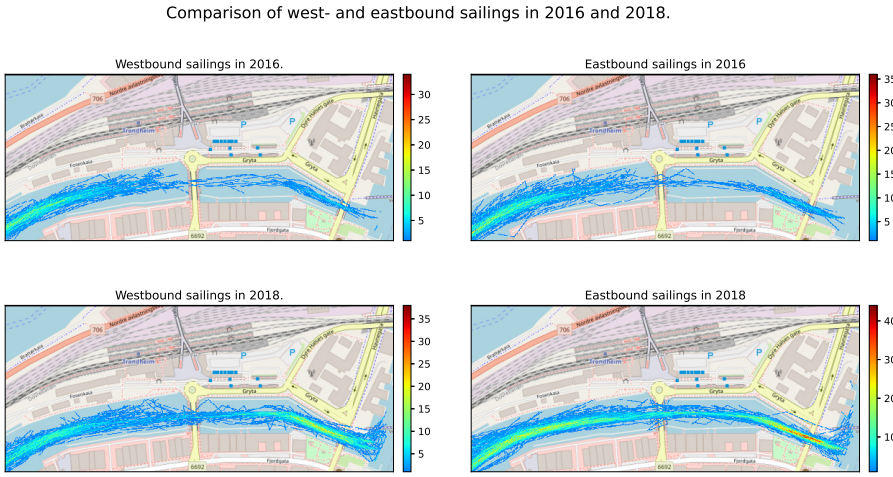


Figure 4.16: Temporal differences in sailing patterns. Little traffic through region in 2016, compared to 2018.

which sailing patterns are likely to change. For example, AIS-data could be collated with weather data, which can be used to plan a route tailored to the weather forecast for the mission. One can also restrict data to those belonging to vessels of similar dimensions, load conditions and so on. For our case in the canal, the data largely belongs to small pleasure crafts, restricting data applicability primarily to other vessels of similar type and dimensions.

4.3.3 Statistics on recorded traffic

Some selected statistics for the traffic in the area around Ravnkloa is presented in this section. These provide a basis for selecting scenarios for traffic simulation. Since the collected data is not an exhaustive collection of the traffic, as mentioned earlier, relative rather than absolute numbers are presented.

Monthly ship traffic for 2019 is seen in Figure 4.17. As one would perhaps expect from knowing that most of the boats in the area are pleasure crafts, traffic increases in the warmer months and peaks in July. In fact, as can be seen from the Figure, nearly 25% of all AIS-recorded traffic in 2019 occurred in July. One can break the numbers further down by days and hours to find the one-hour interval with the greatest number of ships. The maximum is found to be 15 different ships, on the 18th of July from 11:00-12:00. Again, this is the maximum as given by the AIS-data; at best this is a lower bound on the true maximum for the time period considered. This can be used to simulate scenarios with a reasonable estimate of the maximum number of ships sailing simultaneously through the area.

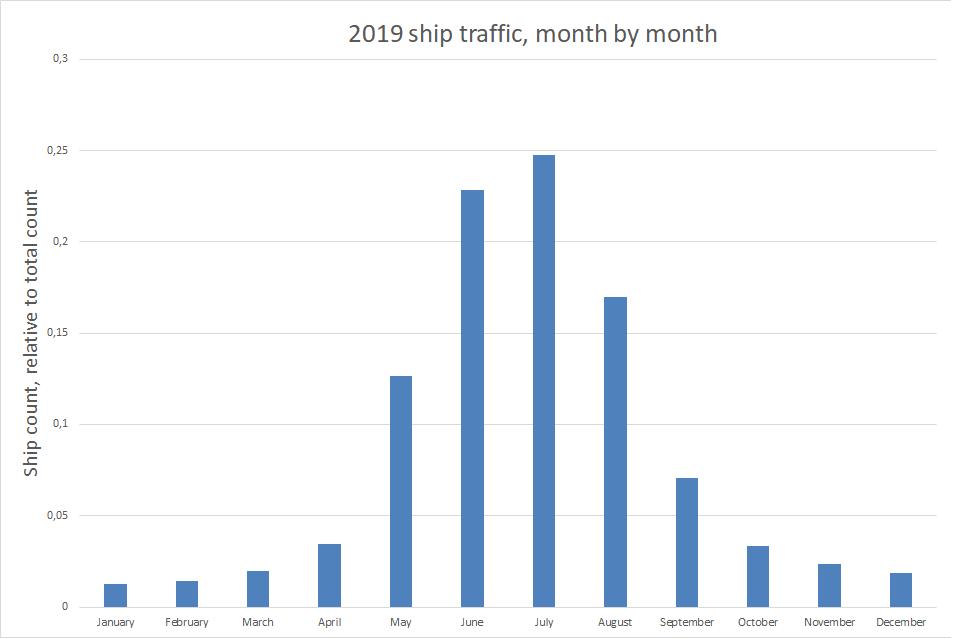


Figure 4.17: Histogram of ship traffic in 2019.

A histogram of the recorded speeds is seen in Figure 4.18. The number of speed samples were 75481. Note that since recorded speeds are averaged over a pair of coordinates, samples have been weighted by the Euclidean distance between these coordinates. Since histograms of the same data set with variations in bin sizes and locations can have large qualitative differences, kernel density estimation techniques were used to provide an estimate of the probability density function (pdf) of the speeds. The resulting pdf is seen in Figure 4.19, and the pdf superimposed on the histogram is seen in Figure 4.20. The main takeaway from the distribution is its high and narrow peak, showing that almost all recorded speeds are tightly clustered around the sample mean of 4.11 knots. Standard deviation of the sampled speeds is 2.72 knots. However, as can be seen from Figure 4.20, the tail of the distribution is non-exponential and does not taper off rapidly towards zero. This means that while most observed speeds will be clustered around the peak, the probability of observing a very high speed is greater than one would intuitively expect. For that reason, an autonomous passenger ferry operating in this area may experience edge cases where other vessels reach high speeds of, say, around 30 knots. This is a potentially hazardous situation, and systems like the collision avoidance system of the ferry should be verified for these scenarios.

4.3.4 Possible improvements

The collected data contains inherent inaccuracies in position, speed and heading. This may manifest itself in sailings that appear to cross or travel on land. As reasoned before,

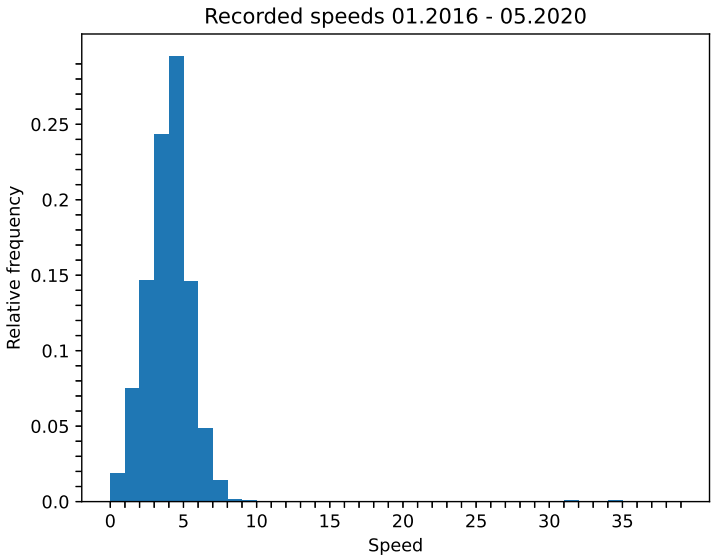


Figure 4.18: Histogram of recorded speeds.

the AIS equipment is likely to be mostly of Class B, which has lower accuracy and update frequency. Speeds are averaged over two timestamped, measured positions - if these positions are far apart in time, the speed becomes more uncertain. The same applies to the heading. Furthermore, in the data processing, positions are linearly interpolated. If the two measured positions are far apart, the uncertainty in the interpolations grow larger. In particular, this is a potential problem in areas where straight line movement between two measured points is impossible. This could for example be the case when moving around a corner, with the measurements being made before and after the turn. For these reasons, sailings with measurements far apart could be disqualified from the interpolation procedure. Another possible improvement is adding functionality to the data processing script, e.g by accessing an API like onwater.io (paid service) that checks whether a lat/lon position (interpolated or measured) is on water. If it is not, the entire sailing is removed from the set. As it stands, some of these positions have been removed by using a simple approximation.

While using data from Class A equipment is preferable, this may not always exist. In regions where mostly smaller vessels sail, AIS data will probably be from Class B until costs of Class A are reduced. Thus, as long as this is the only data source, these inaccuracies are not something one is easily rid of. Direct access to the raw data, and not the data provided by Kystdatahuset, would be better since the analysis would not depend (to some degree) on choices and assumptions made by the data scientists at Kystdatahuset. In any case, more stringent measures like the ones mentioned should be introduced to improve the accuracy of the data set, if it is used for learning algorithms with real-world applications.

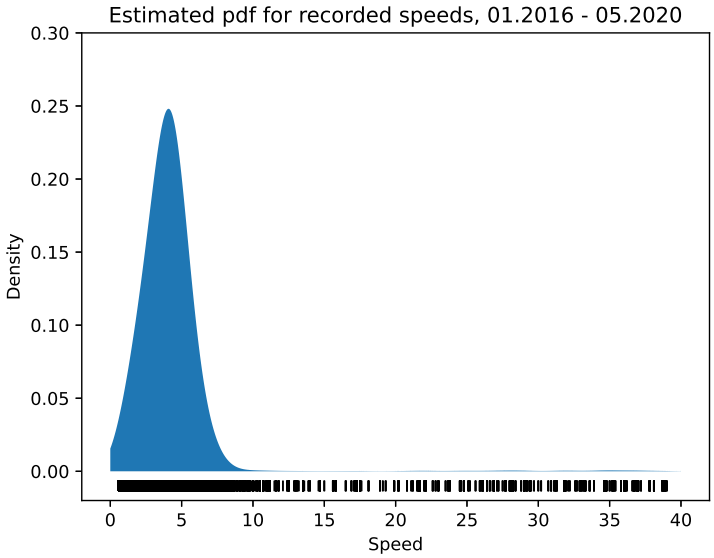


Figure 4.19: Kernel density estimate of recorded speeds.

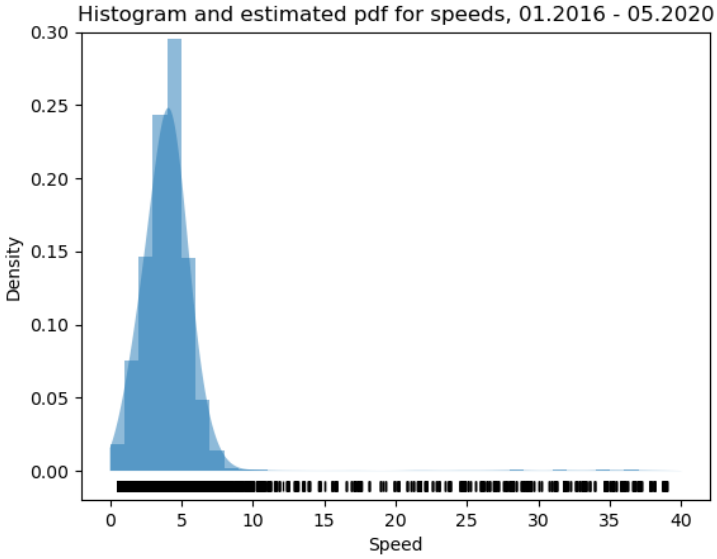


Figure 4.20: Histogram and pdf of recorded speeds.

Chapter 5

Methods for waypoint generation based on AIS-data

AIS-data for a region of the proposed operating area of milliAmpere was gathered and processed to obtain data for generating route waypoints. This can be used in a simulator to simulate realistic ship traffic in the region(s) of interest, or to provide routes for use by autonomous vessels. The main operating area of milliAmpere in Trondheim Harbour

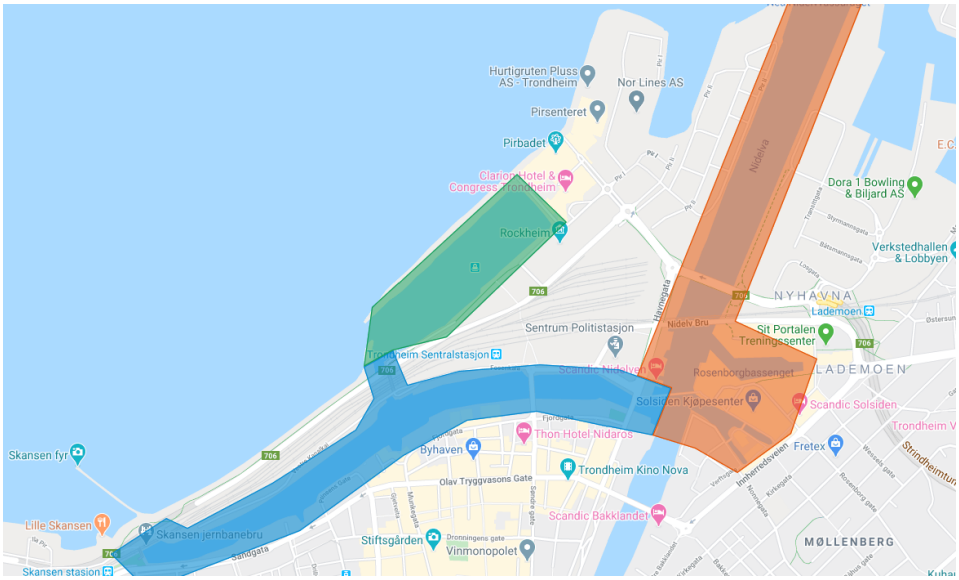


Figure 5.1: Possible operating areas of milliAmpere. Core area around Ravnkloa shaded in blue, with the area out towards Hurtigbåtterminalen shaded in green and the area towards Pir II in orange.

is the area around Ravnkloa, with other proposed operating areas from Ravnkloa to Hurtigbåtterminalen, and from Ravnkloa to Pir II, see Figure 5.1. The area of interest for this thesis was set to the area around Ravnkloa.

We want to be able to generate new sailing routes defined by a set of waypoints, based on the collected data. Each waypoint will be given by latitude, longitude, speed and heading. One possibility is to simply reproduce recorded sailings. There are many caveats with this, chiefly an inability to define arbitrary start- and endpoints, or desired speeds – one is completely dependent on the properties of previous sailings. Two approaches have been considered here. Both approaches apply reinforcement learning – specifically a modified Q-learning algorithm – wherein the agent is tasked with learning the best route from a reward function based on our data. The main difference of the algorithms lie in the initialization of the Q-table. In the first case, the 4-dimensional Q-table is simply initialized to zeros. In the second case, the Q-table is initialized by an algorithm that performs a one-shot task of computing a path from start to stop. This algorithm is run for n_p episodes, during which the Q-table is updated. The idea is that the aggregate of the paths obtained from this are not too far from optimal, and thus can serve as a guide to the learning algorithm, as mentioned in Section 2.3.4. This method is a (semi)probabilistic method that takes in start- and end position as well as speed, and simulates vessel movement by selecting actions probabilistically based on the sampled data, followed by a position update. After the initialization run, the learning algorithm from the first case is used.

5.1 Reinforcement learning approach

Reinforcement learning was applied to the problem, by using the collected data as the basis for a reward function. Using a reinforcement learning algorithm is attractive for several reasons: It reduces the amount of explicit programming needed, and by specifying the starting and terminal state the algorithm learns the best route, given some reward function. The Q-learning algorithm was used as a starting point.

Before implementing the algorithm, some considerations with regards to data representation and dimensionality had to be made. States were represented by 2D position, and actions were described by heading and speed. With the given discretizations, this would result in a large number of state-actions: If we reduce the number of discrete speeds to 10 we would have $1500 \cdot 700 \cdot 32 \cdot 10 = 3.36 \cdot 10^8$. Assuming each value in the Q-table is represented as a standard double-precision floating point number, the memory requirement would be 8 bytes $\cdot\ 3.36 \cdot 10^8 \approx 2.8$GB. While not a huge number in terms of the random-access memory (RAM) capabilities of modern computers, run-time would be slow. A few things were done to mitigate this "curse of dimensionality"[1]. Firstly, cell size was increased to 5 m x 5 m (necessitating a re-processing of the AIS-data to account for this), decreasing the grid size to 300 x 140. The cells in the grid could also be restricted to only cells covering ocean area, which significantly reduces the number of represented state-actions. However, these positions will not be searched in the algorithm since they are

[1] The "curse of dimensionality" was coined in [24], referring to the large computational demands incurred by introducing many or large dimensions to a problem.

represented as terminal states. The only consideration is that they must be represented in computer memory, which, with the given decrease in grid size, is not a problem. Secondly, the set of possible headings in the Q-table was reduced to 17 by disallowing movement backwards (relative to the intended direction of travel). Changing the discretization to correspond to a 16 point compass rose was also attempted, but the resulting lack of resolution in headings gave poor performance. This was in part due to the geometry of the canal to the west of Ravnkloa. This runs at an angle of about 55 degrees clockwise from the north. Thus, when using a similar discretization as before but to a 16 point compass rose, this angle would be mapped to 45 degrees. With the original mapping, it would be mapped to 56.25 degress. Since most traffic runs essentially parallel to the canal walls here, this is not an acceptable discrepancy. Other, more *ad hoc* mappings for reducing the number of headings are certainly possible, but the original mapping with its reasonably high resolution was kept. Speeds were discretized by a new mapping:

$$v(s) = \begin{cases} v_1 = 1, & 0 < s \leq 2 \\ v_2 = 3, & 2 < s \leq 4 \\ v_3 = 5, & 4 < s \leq 6 \\ v_4 = 7, & 6 < s \leq 8 \\ v_5 = 9, & 8 < s \leq 10 \end{cases} \tag{5.1}$$

Speeds above 10 knots – which we know there are comparatively few of from Figure 4.20 – were not considered. The size of the Q-table thus becomes $140 \cdot 300 \cdot 17 \cdot 5 = 3.57 \cdot 10^6$ which is considerably more tractable.

The considered state space can be expressed as the Cartesian product of the sets containing the cell indices, and the action space is the Cartesian product of the discretized headings and speeds:

$\mathcal{I} \doteq \{1, 2, ..., 300\}$	Cell column indices
$\mathcal{J} \doteq \{1, 2, ..., 140\}$	Cell row indices
$\mathcal{H} \doteq \{\psi_1, \psi_2, ..., \psi_{32}\}$	Headings
$\mathcal{V} \doteq \{1, 3, 5, 7, 9\}$	Speeds
$\mathcal{S} \doteq \mathcal{I} \times \mathcal{J}$	State space
$\mathcal{A} \doteq \mathcal{H} \times \mathcal{V}$	Action space

Moreover, $\mathcal{A}(s) \subseteq \mathcal{A}$.

Rewards for each state-action pair was formulated as a function of the empirically observed frequency f of that state-action pair. The key idea here is to let the collected data from the previous sailings form a kind of aggregated heuristic that teaches the agent how to behave. With this in mind, the reward function obtained in this case was the result of prior analysis of needed requirements followed by some iteration and trial and error. Three requirements were specified:

i) Minimize number of states visited. This is used to obtain an efficient path to goal. Letting all rewards in non-goal states be negative forces a reward-maximizing agent

to minimize the number of visited states. With this assignment, it follows that "higher rewards" are equivalent to "less negative" rewards

ii) Place higher rewards on state-actions with higher frequency. This requirement stands at the crux of the proposed method because it is the requirement that allows the agent to learn best practices from human navigators. Thus it is the requirement that should be considered most important.

iii) In a given cell, state-actions with a higher frequency relative to the total number of observed frequencies *in that cell*, should also have higher rewards. This is perhaps a less intuitive requirement. It is included to reduce the disparity in assigned reward between cells with significant differences in data count.

A fourth requirement stating that state-actions closer to the goal should have higher rewards was considered and also implemented at first, but its inclusion gave no benefit to performance and was discarded.

The rewards were defined as

$$\mathcal{R} = -\frac{1}{f^2} \cdot \exp\left(-\frac{f}{f_{max}}\right), \quad \forall s \in \mathcal{S}, \forall a \in \mathcal{A}(s) \tag{5.2}$$

where

- f is the frequency of action a in state s
- f_{max} is $\max_f \mathcal{A}(s)$, i.e the action with the highest observed frequency in state sv

In the case that no data is recorded for a given state-action, the reward is set to -1. Terminal states representing walls, quays, docks and so on have a reward of -10. The goal terminal state has a reward of 100. For the state-actions with data we have

$$-\frac{1}{f^2} \in [-1, 0) \tag{5.3}$$

$$\exp\left(-\frac{f}{f_{max}}\right) \in [e^{-1}, 1) \tag{5.4}$$

$$\implies -\frac{1}{f^2} \exp\left(-\frac{f}{f_{max}}\right) \in [-e^{-1}, 0) \tag{5.5}$$

We have $\mathcal{S}$, $\mathcal{A}$ and $\mathcal{R}$ for defining the MDP (see Section 2.3.1, and require only the transition probabilities $\mathcal{T}$. We treat the environment as deterministic, so that there is only one possible next state s' when taking action a. In other words, Equation 2.17 evaluates to 1 for exactly one state $s' \in \mathcal{S}$ and is 0 for all others. As such, an explicit definition for $\mathcal{T}$ is not needed.
In each state, actions were chosen by a partly random policy informed by the collected data, using the epsilon-greedy strategy described previously: For a fraction ϵ of the time, the best action a as inferred from the Q-table was chosen: $a = \arg\max_a Q(s, a)$. For the remaining fraction $(1 - \epsilon)$ of the time, a random action was selected. Restrictions on

	Symbol	Value
Learning rate	α	0.1
Discount factor	γ	1.0
Greedy probability	ϵ	0.9

Table 5.1: Parameters used for Q-learning.

allowed actions were applied (see Equation 5.10), and are described in the next section. Aside from these restrictions, the algorithm proceeds as described in Algorithm 1. The values for the parameters used are provided in Table 5.1. These were obtained by trial and error to balance performance and runtime.

After running the learning algorithm for the predefined number of episodes, the optimal policy is extracted from the Q-table by taking $Q \approx Q^*$ and running Algorithm 2, which uses Equation 2.30.

Algorithm 2: Optimal policy from Q-table

Data: Q from Q-learning algorithm. Empty optimal policy π^*

$s \leftarrow s_0$
repeat
  $a \leftarrow \text{argmax}_{a \in \mathcal{A}(s)} Q^*(s, a)$
  Perform action a, observe the next state s'
  $s \leftarrow s'$
  $\pi^*(s) \leftarrow a$
until s is the goal state;
return π^*

With the mentioned reductions to the state-action space and given rewards, the Q-learning algorithm works quite well, but requires much time to converge to a solution. This motivates the use of a method that provides a notion of what the best route should like to the RL agent. In doing so, the optimum should be reached faster. A probabilistic method used to achieve this is described next.

5.2 Probabilistic method

This algorithm was implemented by combining stochastic action selection with a LOS guidance scheme. It does so by essentially simulating a vessel's movement from start to stop, using the historical AIS data. Actions in a given cell were chosen using an epsilon-greedy strategy combined with a probabilistic policy: For a proportion ϵ of the time, the most frequently observed action was chosen. For the remaining proportion $(1 - \epsilon)$ of the time, the action was chosen probabilistically, with the probability of each action taken to be its empirical frequency. The reason for not just always choosing the greedy action is to introduce a degree of randomness to the action selection, which functions as a form of naïve exploration (naïve because the exploration is independent of previous algorithm executions). Once an action was chosen, the position at the next time step, $\boldsymbol{p}_{i+1}^n$, was

simulated, and the process repeated. The equation for $\boldsymbol{p}_{i+1}^n$ is

$$\boldsymbol{p}_{i+1}^n = \begin{bmatrix} x_{i+1}^n & y_{i+1}^n \end{bmatrix}^\top = \boldsymbol{p}_i^n + \boldsymbol{R}_b^n(\psi) \cdot s_i \cdot 0.514 \cdot h, \tag{5.6}$$

which in component form reads

$$x_{i+1}^n = x_i^n + \cos(\psi) \cdot s_i \cdot 0.514 \cdot h \tag{5.7}$$
$$y_{i+1}^n = y_i^n + \sin(\psi) \cdot s_i \cdot 0.514 \cdot h. \tag{5.8}$$

The factor 0.514 is present to convert speeds in knots to m/s, while h is the time step.

The set of allowable actions were restricted using the speed s_c and the heading ψ_c at the current timestep. This was done to keep the speed close to the desired speed, and to avoid large jumps in heading. In state $\mathcal{S}_i$ the set of actions is $\mathcal{A}_i$, and each action is a tuple of speed s_i and heading ψ_i. The restricted set of allowable actions $\mathcal{A}_i^R$ can be defined as

$$\{a = (s, \psi) \in \mathcal{A}_i^R \mid \mathcal{A}_i^R \subseteq \mathcal{A}_i \quad \wedge \quad \Delta_s \in \{\Delta_{s_{min}}, \quad \Delta_{s_{min}} + \delta_s\} \tag{5.9}$$
$$\wedge \quad \Delta_\psi \leq \delta_\psi\}, \tag{5.10}$$

where

$$\Delta_s = |s_c - s|$$
$$\Delta_\psi = |\psi_c - \psi|$$
$$\Delta_{s_{min}} = \min_{s \in \mathcal{A}_i} |s_c - s|,$$

δ_s is a parameter that determines how large Δ_s can be, δ_ψ is the largest allowed change in heading. The speed restriction lets the allowable speeds be as close to the desired speed as possible ($\pm\delta_s$), while forcing an increase/decrease in assigned speed if the desired speed deviates from the observed speeds in the cell.

Since not all cells have empirical data, a fallback method was also needed. A few approaches were tried. When initiating sailing from a starting point with no empirical data, lookahead-based LOS guidance was used by scanning the surrounding area in a circular arc, identifying cells with sufficient data, and choosing a reference and target from these by considering the direction of intended travel. Figure 5.2 shows the geometry of the LOS scheme with two waypoints $\boldsymbol{p}_n$ and $\boldsymbol{p}_{n+1}$. The desired course χ_d is assigned as the sum of the path-tangential angle χ_p and the velocity-path relative angle χ_r, see equations 5.11-5.12. If no data was available in cells when en-route to the goal, then LOS guidance was also attempted. However, the resulting behaviour was erratic in regions with sparse data. An alternative was to use actions from the previous cell, which worked better but had the same drawbacks when working with little data.

Ultimately, the solution was to create a single curve serving as a reference path and doing path following on this with LOS guidance. This was done by fitting a piecewise parametric curve (spline) to each data subset. This reference serves two purposes: Firstly, if there is no recorded data to select actions from, the reference can be followed. Secondly, since

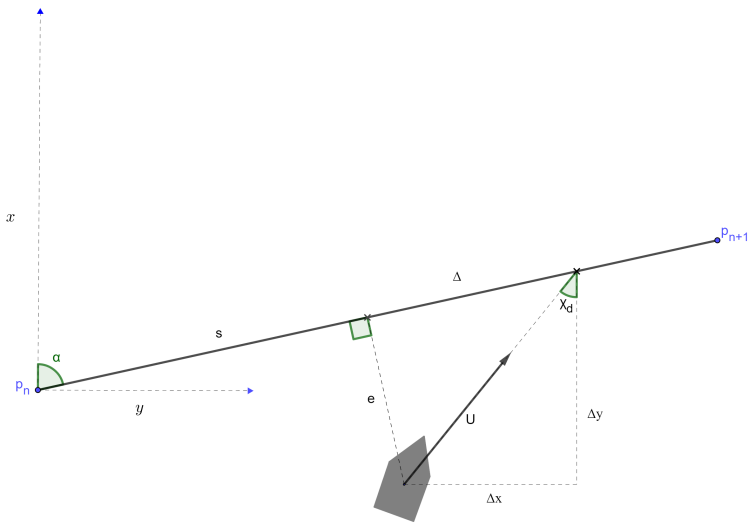


Figure 5.2: Geometry of the look-ahead based LOS guidance. p_n is the reference; p_{n+1} the target.

the reference is a curve fitted to the historical data, moving in towards this curve ensures that one is in a region with dense data. With a reference path defined over the possible sailing region, a search for it is conducted when no empirical data have been available for two timesteps. For the preceding step, the allowable actions from the previous cell are used. The search starts by scanning around the present position in a radius R_s, increasing scan radius progressively until two points sufficiently far apart on the reference have been identified. The search procedure is illustrated in Figure 5.3. The reference path is seen in blue. The procedure locates the intersection points of a circle centered at the vessel and the reference path. These points will serve as reference and target in the LOS guidance. The algorithm looks for the reference by first scanning in a circle of radius r_0, and successively increases the search radius by a factor of β until it is found. If the reference is found, but the Euclidean distance between the two intersection points are smaller than d_{min}, the search is repeated with increased radius. A straight line is created with the two points assigned as the reference and target for LOS guidance. Which point is the target/reference depends on the intended direction of travel. This is done assuming the reference path can be approximated by a linear curve between these two points. Subsequently, straight line path following by LOS guidance proceeds until the distance to the target is small enough, and gives control back to the stochastic action selector. Once within a radius R_g of the goal, and provided there is a clear line of sight to goal, the algorithm switches to LOS guidance for the remainder of the sailing. The equations used for this part are Equations

	Symbol	Value
Timestep	h	1.0 s
Lookahead distance	Δ	7.5 m
Search radius	R_s	20 m
Search radius factor	β	1.2
Goal distance	R_g	40 m
Speed assignment parameter	κ	500
Speed offset	δ_s	2
Angle offset	δ_ψ	0.4

Table 5.2: Parameters used in the probabilistic algorithm.

5.11-5.15, given below.

$$\chi_d = \chi_p + \chi_r \tag{5.11}$$

$$= \alpha + \arctan\left(\frac{-e}{\Delta}\right), \quad \Delta \neq 0. \tag{5.12}$$

The along-track distance a and the cross track distance e are given as

$$\begin{bmatrix} a \\ e \end{bmatrix} = \mathbf{R}(\alpha)(\boldsymbol{p}^n - \boldsymbol{p}_t^n), \tag{5.13}$$

where

$$\mathbf{R}(\alpha) = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix} \tag{5.14}$$

is a member of SO(2), the rotation group for two-dimensional space. $\boldsymbol{p}^n$ is the current vessel position vector, and $\boldsymbol{p}_t^n$ the position vector of the target. The speed assignment if using the LOS path following to initiate the sailing is

$$s = s_d - s_d \cdot \frac{|a|}{\sqrt{a^2 + \kappa^2}}, \tag{5.15}$$

which drives the speed from 0 to s_d as one approaches the target. The speed assignment is however bounded by the average speed in the present cell. This avoids jumps in speed assignment when exiting the LOS procedure, if the desired speed is much higher than the average. The parameter κ is used to tune how smoothly the assigned speed goes to s_d.

Note that both the assigned speeds and angles had to be discretized to be represented in the Q-table.

To reiterate: This algorithm finds a set of waypoints from start to goal, by selecting actions and simulating vessel movement by simple position updates. Actions are selected from the AIS-data when it exists. When it does not, a line-of-sight guidance scheme is applied. The algorithm performs a one-shot task, as it is run once with no information about the run itself being stored, aside from the waypoints. Due to its probabilistic nature, the algorithm

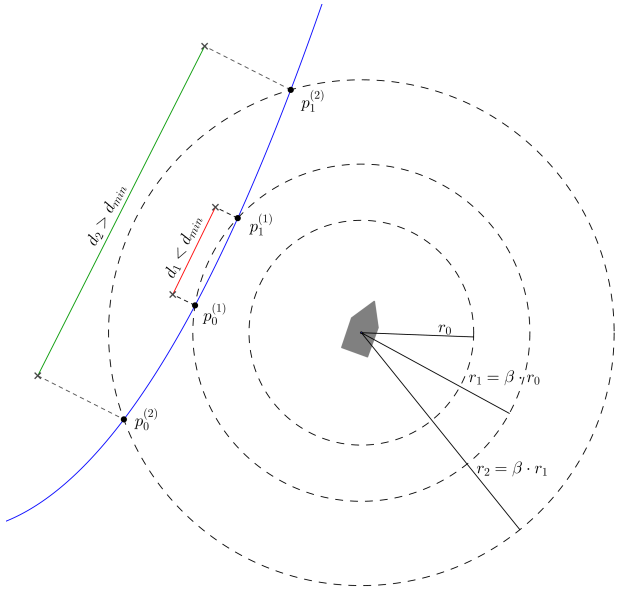


Figure 5.3: Search procedure in the LOS algorithm.

may fail to reach goal. This may happen due to randomly choosing poor actions or because a combination of the vessel pose and LOS tuning parameters introduces edge cases not accounted for that results in a crash. For example, if the initial heading is opposite to the intended direction of travel, and the initial actions are selected from the LOS procedure, a U-turn can result in a crash. Since this algorithm is meant to be used as a means of providing the RL agent with an idea of where to begin exploration, these drawbacks are not terribly inauspicious. Runtime is quite fast, requiring only at most a few seconds in the worst case scenarios when the start and end state are at opposite ends of the considered area. Additionally, with sensible input the procedure usually terminates in the goal state. A flowchart of the algorithm is given in Figure 5.4. The loop indicated in blue should be considered the main loop as it is most often executed. At each simulation step, the Q-table is updated (not pictured).

5.3 Hybrid approach

The first, probabilistic algorithm was used to provide an estimate of the optimal Q-table. This was done by instantiating the Q-table to 0, and running the first algorithm for n_P episodes while updating the table concurrently with rewards from $\mathcal{R}$. The resulting Q-table was then used in the second algorithm. In this case, the agent considered in the Q-learning algorithm does not start from a position of ignorance, having been provided with a ”running start” in the form of the non-zero Q-table. Q-learning is subsequently run for n_Q episodes. There is also a technicality to be considered in how to treat states not

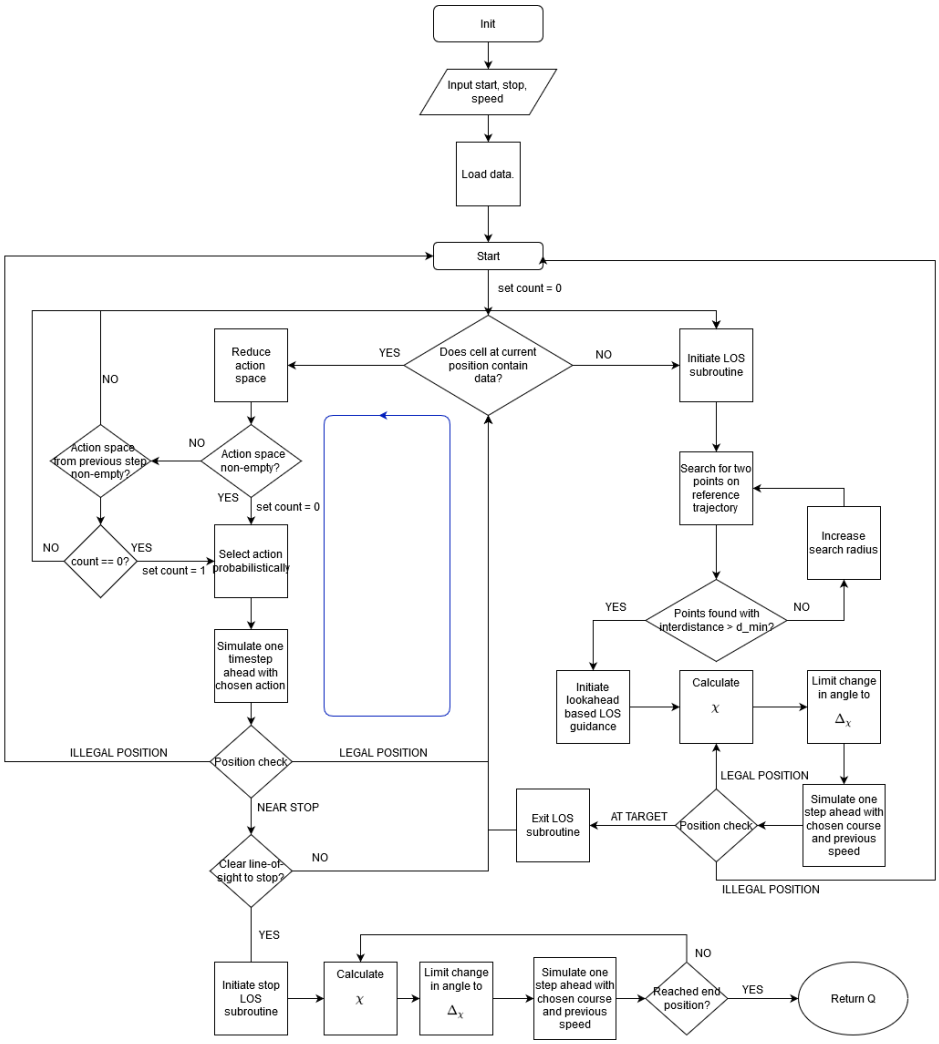
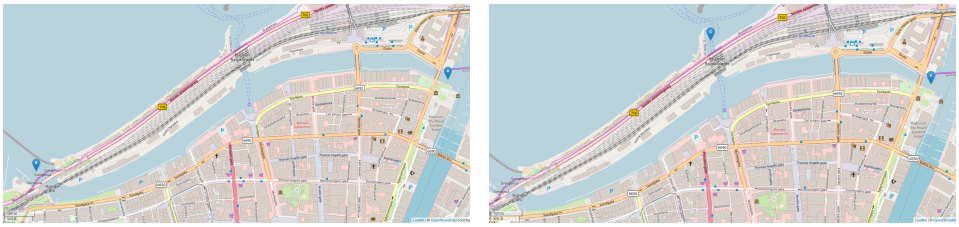


Figure 5.4: Flowchart presentation of the probabilistic algorithm.

visited by the probabilistic algorithm. The purpose of this initialization run is to update the Q-table with rewards from trajectories considered to be close to the optimal. In other words, on the one hand we do not want to consider paths significantly different from those found from the probabilistic algorithm as good candidates for the optimum. On the other hand, we do not want to discard all paths consisting of unvisited states. Consequently, the Q-values for these unvisited states were set to a small negative number that depends on the values in the Q-table after the first initialization run. The goal of this is to discourage exploration of the unvisited states while not expressly forbidding it. Exploration in general is achieved with the epsilon-greedy strategy. Parameters were the same as in Tables 5.1 and 5.2.

5.4 Results and discussion



(a) Case I. Start in A, stop in B.

(b) Case II. Start in A, stop in C.

Figure 5.5: The cases considered.

Two cases were considered in testing the RL algorithms, with start and stop points seen in Figures 5.5a-5.5b. The first case places the start and stop position far from each other. Furthermore, in this case we have a lot of data to learn from, as seen in Figure 4.12. The second case is shorter, but passes through the narrow northern passage. Additionally, from Figure 4.10 we see that there is less data to learn from, allowing us to see how the algorithms handle that. Also, the stop position in this case is in a region with no data, which should pose a challenge. For both cases high desired speeds (relative to the area average of 4.11 knots) are used. The algorithms are run on each case with different numbers of episodes, and also with the start and stop positions reversed. With these positions reversed, the underlying data subset is different. For both Case I and II with reversed start and stop positions, the data used is seen in Figure 4.13. Thus, for Case II reversed, we have access to more data than in the regular Case II. However, the start position is in a region with no data.

Because of the discrete action set and time steps, jumps will be apparent in the obtained waypoint position, headings and speeds. For that reason, the results of the algorithms do not in itself represent a path for vessels to follow. Rather, a subset of the generated waypoints are to be provided to the vessel guidance system, which computes a trajectory or path based on the waypoints. After extracting the optimal policy with Algorithm 2, every 10th waypoint is shown in the plots. Each waypoint is represented by a vessel-like symbol to indicate heading, while the color displays the speed. Vessel size have been exaggerated for plot visibility; in the figures seen the size is about 15 m x 10 m.

5.4.1 Case I

Figure 5.6 and Figure 5.7 compare the results of running the Q-learning algorithm and the hybrid algorithm on Case 1. We see that the obtained paths are virtually identical, with little differences in speed as well. The assigned speeds – indicated by the color bar in the figures – are also much lower than the desired speed. Notably the speed decreases more for the Q-learning algorithms across the junction near Ravnkloa. This is at a place where crossing traffic might occur. We also know from Figure 4.17 that the majority of

Figure 5.6: Learned waypoints from the hybrid algorithm. The colorbar indicates speed at each waypoint. Case I, 5000 episodes.

Figure 5.7: Learned waypoints from the Q-learning algorithm. The colorbar indicates speed at each waypoint. Case I, 20000 episodes.

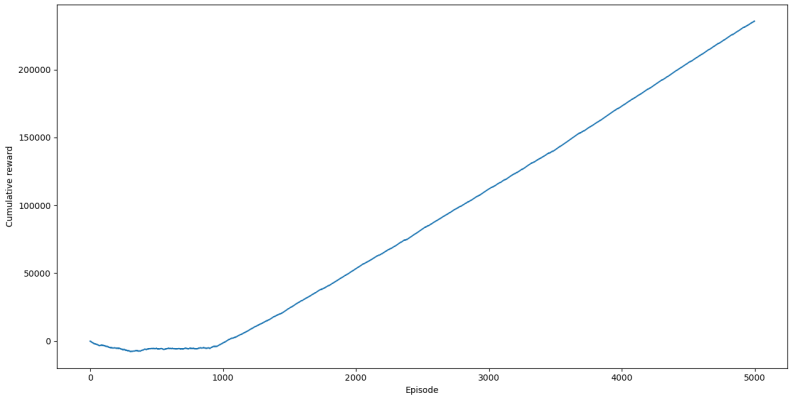
the learning data is from the summer months, when a significant amount of shuttle boat departures/arrivals in just this region happens. From this it makes sense that the speed should decrease to reduce maneuvering risk here. A speed reduction is also seen before the turn on the left hand of the figures. The hybrid algorithm achieves nearly the same result as the Q-learning algorithm using only a quarter of the number of episodes.

Cumulative and average rewards are plotted against episode in Figures 5.8-5.9. The cumulative reward at episode n is the sum of the rewards obtained at each episode up to and including episode n. Figure 5.8a shows that the hybrid algorithm begins stabilizing after about 1000 episodes, as evidenced by the plot approximating a straight line. This happens because the agent has discovered a path returning near optimal rewards, and is exploiting this. This is reflected in the plot of average rewards, where we see the average rewards asymptotically approaching the best reward. If α is decreased appropriately, with a concomitant increase in ϵ, this asymptote would be reached faster. Recall that a small α means we are favouring previously learned knowledge – which is appropriate if we have discovered a good policy – and that a large ϵ means we are almost always selecting the locally optimal action. Thus an improvement would be to implement such a scheme.

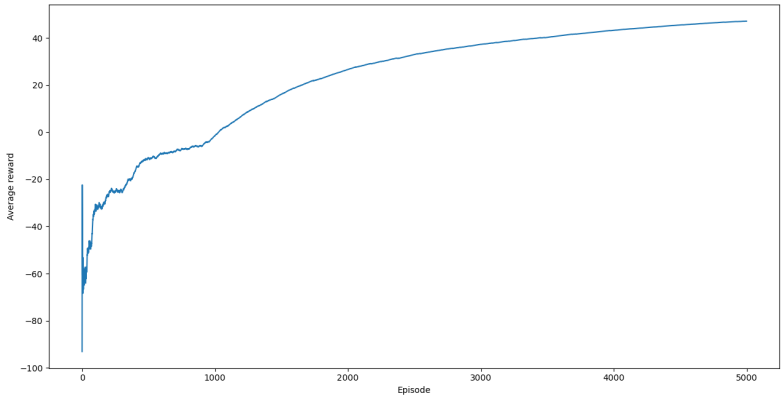
As we see in Figure 5.9a, Q-learning uses over 7000 episodes before stabilization begins. Of note is the inflection point in both Figures 5.9a and 5.9b close to the 12500 episode mark. To the left of this point it looks like we are approaching an asymptote, with the average rewards stabilizing. However, through exploration the agent must have discovered a better path. Comparing the slopes to the right and left of the inflection point in Figure 5.9a shows that it is steeper to the right, which means the policy is better. This exemplifies the importance of continued exploration and willingness to learn from new information, even in the face of having found a seemingly good policy. To an extent this serves as a contrast to the above discussion of changing α and ϵ to facilitate fast convergence – if this is done prematurely, one risks not discovering the true optimum. Hence, manipulating these parameters is not a straightforward matter.

While the hybrid algorithm stabilizes faster, it is not allowed to run for as long as the Q-learning algorithm does after stabilization. This affects the results due to the learning parameter α, which determines to what degree new data overwrites old data. Therefore, one expects the policy found by Q-learning to be closer to optimal. To closer match the episode count after stabilization, the hybrid algorithm was run for 13500 episodes, with results seen in 5.10. From this we see that the obtained path is quite similar to those before, the variation being largely in the assigned speeds. It is hard to say whether this extended run provides an improvement; compared to the result from the Q-algorithm there are still more frequent speed changes. Since we would like avoid many acceleration and deceleration phases, adding a penalty for changing speeds in the learning algorithm is another suggested improvement.

For the reversed case, the results are seen in Figures 5.11-5.12. The hybrid algorithm assigns slightly higher speeds than the Q-learning, and actually speeds up near Ravnkloa. There is also a difference in the path chosen near the second bridge, where the hybrid algorithm takes a more northerly route. We see that the heading and speed takes a larger jump here too. Looking at the reward plots in Figures 5.13-5.14 is illuminating in that regard. From Figure 5.13b we see that stabilization has just occurred before algorithm termination. As such, the resulting policy is likely to be erratic. The Q-learning algorithm also requires more time to stabilize, compared to the prior case. This might seem counter-intuitive considering we have merely reversed the start- and stop positions, and that the dataset used is



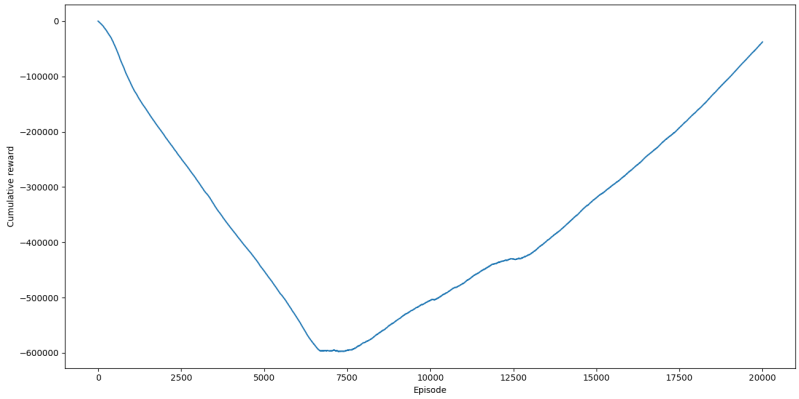
(a) Cumulative rewards.



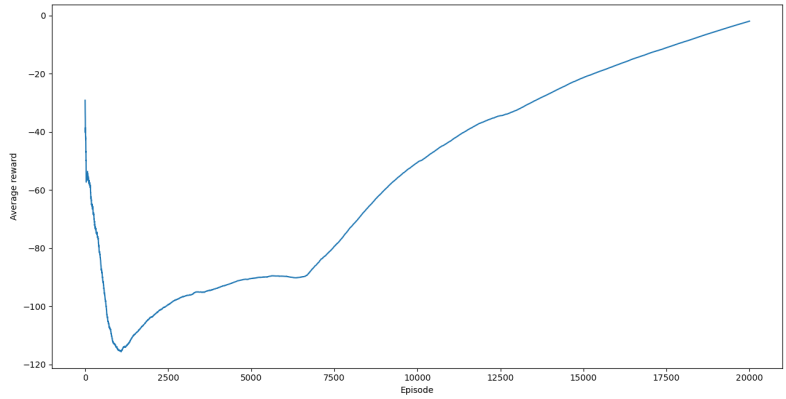
(b) Average rewards.

Figure 5.8: Rewards from the hybrid algorithm, Case I. Episode number along the horizontal axis. The cumulative reward is the sum of all rewards across episodes, while the average reward is the cumulative reward divided by the current episode.

fairly dense, as seen in Figure 4.13. However, note the initial heading. Since we in this case have limited the set of possible headings to $\psi_1 - \psi_1 7$ (see Equation 4.2), the initial heading must be south-facing to some degree. When the exploration policy calls for a random action, the action chosen is more likely to have an easterly inclined heading. Looking at the map, we see that this is probable to result in grounding. Therefore, more exploration is needed initially to find the better path. Still, we see that the hybrid algorithm is faster to stabilize. The Q-learning algorithm begins stabilizing after nearly 10000 episodes, and has another 10000 episodes to update the Q-table. This has an effect on the policy compared to the hybrid algorithm case; in particular we do not have large jumps in the heading in this case. Also, the speed is lower and the path chosen lies more to starboard. The latter point is more in line with traffic rules, as described before.



(a) Cumulative rewards.



(b) Average rewards.

Figure 5.9: Rewards from the Q-learning algorithm, Case I. Episode number along the horizontal axis. The cumulative reward is the sum of all rewards across episodes, while the average reward is the cumulative reward divided by the current episode.

In Figure 5.11, it also appears as if the vessel is heading towards the dock seen in the figure. This can be seen as an example of a consideration relevant for route planning via learning algorithms. Namely the selection of the data used to learn from. In the data set used, sailings that terminate (dock) are included. Since we are not interested in docking, but passing through the canal, the presence of this data affects the learning algorithm negatively. This is overcome with by running through more episodes, as shown with the results from Q-learning for this case, but to avoid movements towards docks in the interior area, it is possible to prune sailings that terminate here before applying the learning algorithm.

Figure 5.10: Learned waypoints from the hybrid algorithm Case I, 13500 episodes.

Figure 5.11: Learned waypoints from hybrid algorithm. The colorbar indicates speed at each waypoint. Case I, with start and stop reversed. 5000 episodes

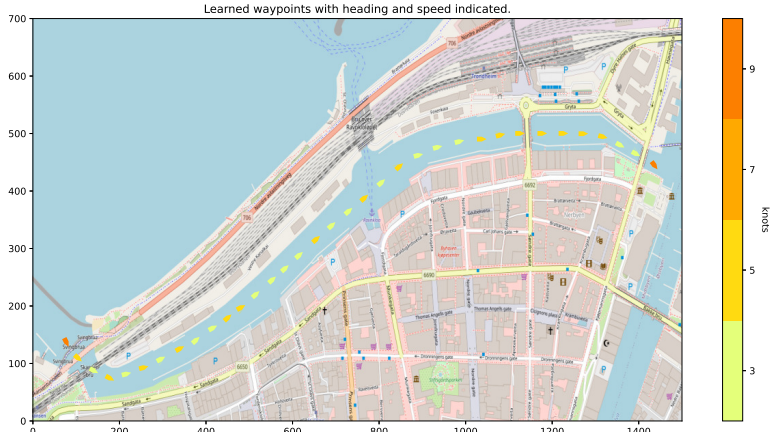
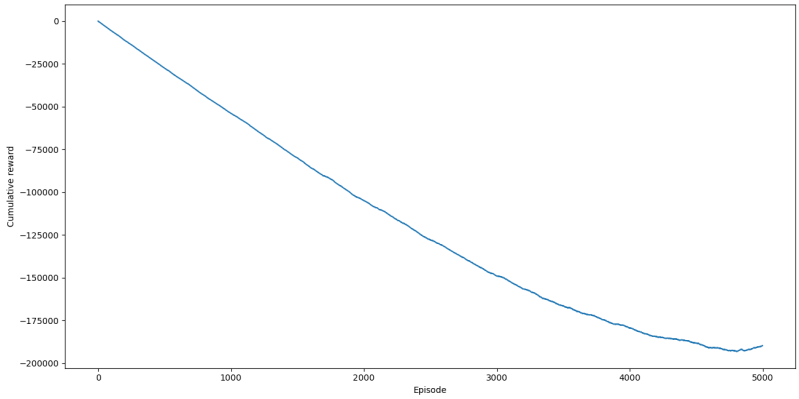
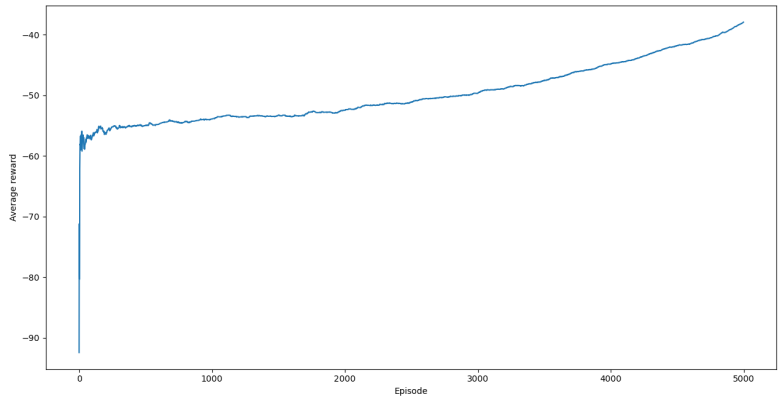


Figure 5.12: Learned waypoints from the Q-learning algorithm. The colorbar indicates speed at each waypoint. Case I, with start and stop reversed. 20000 episodes.

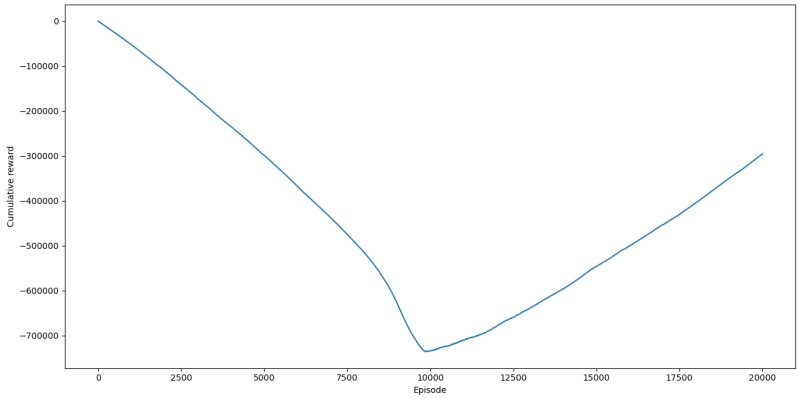


(a) Cumulative rewards.

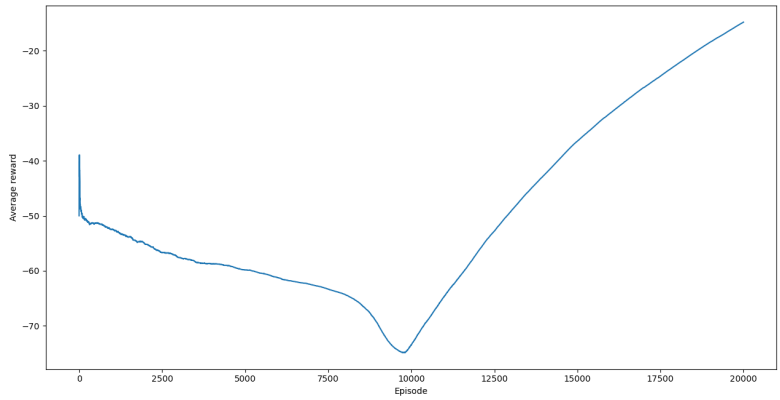


(b) Average rewards.

Figure 5.13: Rewards from the hybrid algorithm, Case I, reversed. Episode number along the horizontal axis. The cumulative reward is the sum of all rewards across episodes, while the average reward is the cumulative reward divided by the current episode.



(a) Cumulative rewards.



(b) Average rewards.

Figure 5.14: Rewards from the Q-learning algorithm, Case I, reversed. Episode number along the horizontal axis. The cumulative reward is the sum of all rewards across episodes, while the average reward is the cumulative reward divided by the current episode.

Figure 5.15: Learned waypoints from hybrid algorithm. The colorbar indicates speed at each waypoint. Case II, 5000 episodes.

5.4.2 Case II

In Figure 5.15 we see that the performance of the hybrid algorithm on a 5000 episode run is not as good as before, despite occurring over a shorter distance. In particular, near start there are large changes in speed and direction. From Figure 4.10 we note that this happens where the data is most sparse. Where the amount of data increases, the obtained route is quite similar to the one from a Q-learning 25000 episode run, seen in Figure 5.16. However, running the Q-learning algorithm on this case with 5000 episodes ends with not a single episode terminating in the goal state. The cumulative rewards for the 25000 episode run is seen in Figure 5.17. Over 17500 episodes are required before stabilization. This is likely due to the sparsity of the data used here, as it results in much exploration. This necessarily increases the number of episodes required for good results. However, provided that we are willing to run a larger number of episodes, the algorithms still achieve a reasonable policy. From Figures 5.15-5.16 we can also note the compliance with boating rules through Ravnkloløpet. For Figure 5.16 we also note a fairly smooth turn-in when approaching Ravnkloløpet.

Results from Case II with reversed start and stop positions are seen in Figures 5.18-5.19. Note that the episode number used for the hybrid algorithm had to be increased in comparison with prior cases. This is in large part due to the lack of data near the start position, necessitating more exploration here. Furthermore, the difference between the hybrid and Q-learning algorithm in episode number is smaller, because exploration near start is required regardless of algorithm. The action-value function can be visualised by plotting

$$\max_{a \in \mathcal{A}(s)} Q(s,a) \quad \forall s \in \mathcal{S},$$

which is seen in Figure 5.20. Unvisited state-actions, i.e entries in the Q-table with value 0, are not shown to create a better plot. The exploration process is seen to the right of the start position; the agent moves randomly until a terminal state is reached (compare with

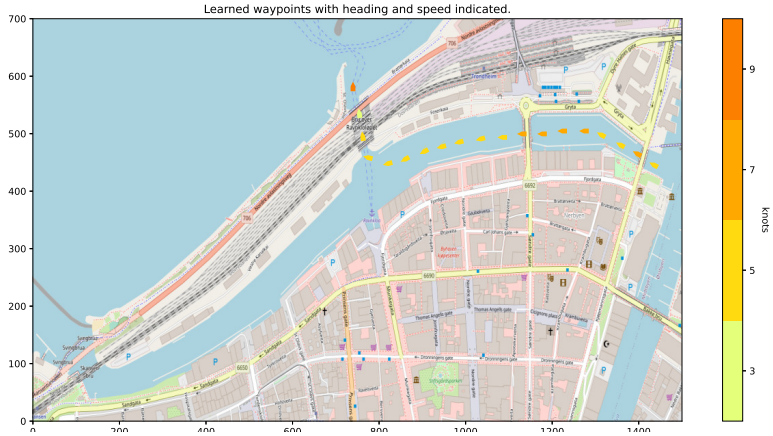


Figure 5.16: Learned waypoints from the Q-learning algorithm. The colorbar indicates speed at each waypoint. Case II, 25000 episodes.

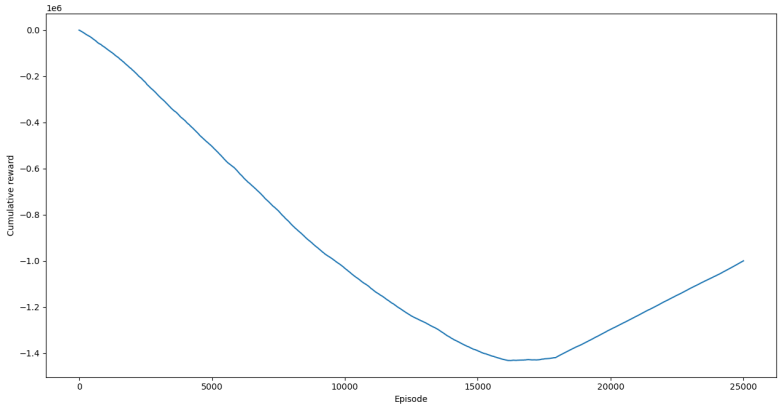


Figure 5.17: Cumulative rewards from Q-learning, case II. Episode number along the horizontal axis. The cumulative reward is the sum of all rewards across episodes.

Figure 5.18: Learned waypoints from the hybrid algorithm. The colorbar indicates speed at each waypoint. Case II, reversed. 10000 episodes

the map separating land from water in Figure 4.5). A more accurate land-water map would improve performance here because the agent would encounter a terminal state sooner.

Plotting the rewards in Figures 5.21-5.22 shows the effect the lack of data near start has on performance. The hybrid algorithm does not stabilize after the 10000 episode run, leading to a worse policy compared to the Q-algorithm. We see this in the generally higher speeds, in particular through Ravnkloløpet and a more aggressive turn. Looking at previous cases, the hybrid algorithm does not offer the same benefit in runtime reduction here. As mentioned, this will be due to the exploration need in the start.

The cases investigated have covered different scenarios challenging the machine learning algorithms in various ways. The policies obtained (after sufficiently large numbers of training episodes) are in accordance with what one would expect when applying human intuition and knowledge of traffic rules. In general, the inclusion of the probabilistic method reduces the number of episodes required for stabilization. However, the benefit is not as great when the underlying dataset demands much exploration to uncover the best policy.

While the learning algorithms using the AIS-data work as intended, the runtime does leave something to be desired. Algorithm time complexity has not been a priority in this work. The collected data and the proposed reward function work well in a reinforcement learning context, and could be applied in a more sophisiticated RL method like Double Deep Q-learning (DDQN), for improved performance [25]. The data set would benefit from a removal of sailings crossing land. Moreover, priming the data for the learning algorithms by discarding sailings that do not start and stop in the desired regions should be considered. Note that it is not possible to generate a route where the behaviour near the goal position is that of a docking procedure – it is assumed that docking will by achieved by other means.

Figure 5.19: Learned waypoints from the Q-learning algorithm. The colorbar indicates speed at each waypoint. Case II, reversed. 20000 episodes.

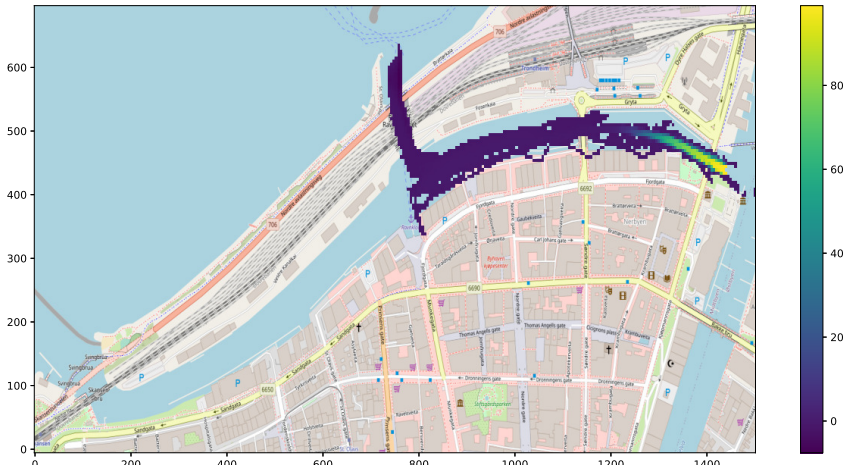
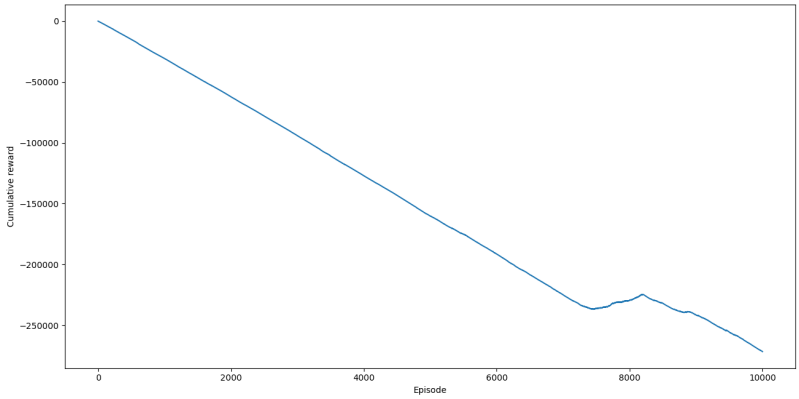
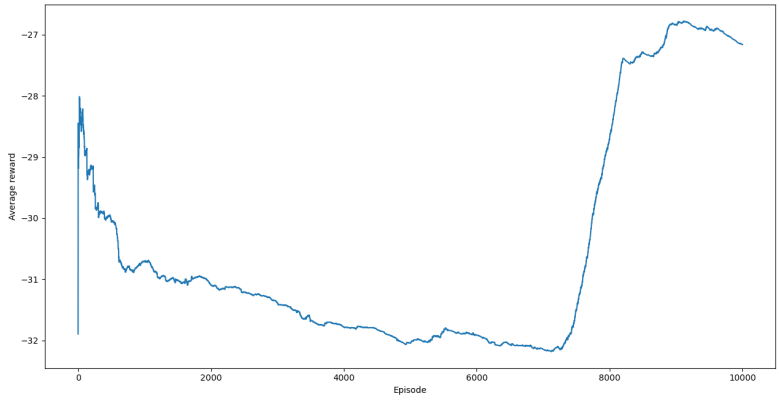


Figure 5.20: Visualisation of the action-value function obtained by the hybrid algorithm in Case II, reversed, over 10000 episodes. The colorbar indicates the maximum value of the action-value function at that position on the map.



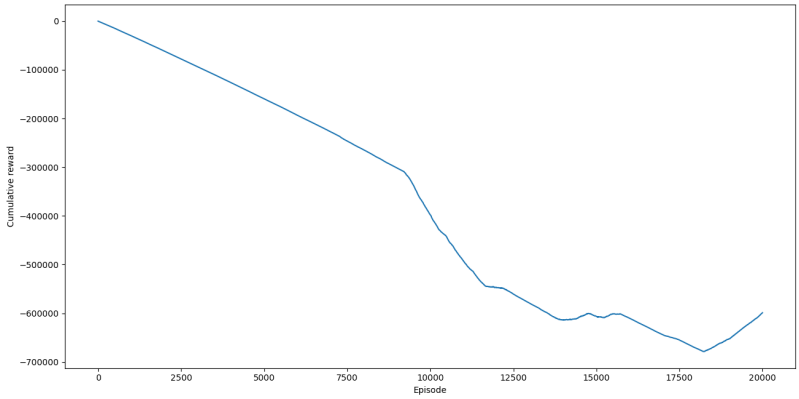
(a) Cumulative rewards.



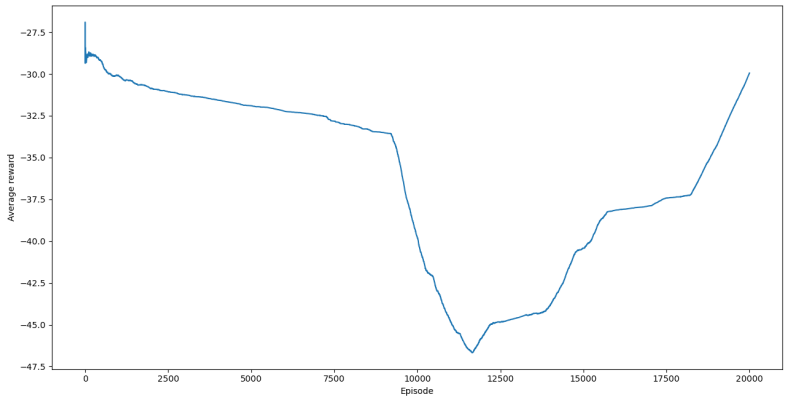
(b) Average rewards.

Figure 5.21: Rewards from the Q-learning algorithm, Case II, reversed. Episode number along the horizontal axis. The cumulative reward is the sum of all rewards across episodes, while the average reward is the cumulative reward divided by the current episode.

For simulating traffic in the selected region, a database of sailings defined by waypoints has been created by running the Q-learning algorithm for 20000 episodes on a number of cases. These cases cover many of the traffic patterns observable from the density plots in Section 4.3.1. The application of this database is described closer in the next chapter.



(a) Cumulative rewards.



(b) Average rewards.

Figure 5.22: Rewards from the Q-learning algorithm, Case II, reversed. pisode number along the horizontal axis. The cumulative reward is the sum of all rewards across episodes, while the average reward is the cumulative reward divided by the current episode.

Chapter 6

Applications for testing and verification

Systems testing and verification refers to various techniques employed to ensure that the system comports with the specified performance requirements and safety standards. An overview of analysis techniques for these purposes is shown in Figure 6.1, which groups techniques according to how scalable and exhaustive they are [5]. The scalability of a technique refers to the level of model detail and size that the technique can address. The exhaustiveness describes how well the technique accounts for all possible system behaviours. Note that techniques with lower levels of exhaustiveness fall into the testing group, as they account only for a finite subset of possible behaviours. More formal techniques are of the verification group, and can account for all possible behaviours.

Simulation based testing is emerging as a promising method of testing and verification of increasingly complex cyber-physical systems because of its great scalability. In short, this method proceeds by representing the system by a digital twin, and performing tests on the twin by simulating scenarios. The similarity to HIL testing – which tests the performance of real controllers by interfacing them to a simulated plant – should be noted. While highly scalable, this form of testing has limitations in terms of (formal) exhaustiveness – the systems are tested for a discrete set of cases, thus the system behaviour for cases outside the test scope must be considered unknown. This entails that one must use a great number of test cases for verification of the system. In that regard, the selection of cases becomes important. Various techniques like [26] and [27] exist where scenarios are selected intelligently. This is also currently being addressed by PhD candidate Tobias Rye Torben at NTNU, whose work intends to bridge the gap between simulation based testing and more exhaustive formal techniques.

In [28], TestIT, a method for the simulation based testing of autonomous navigation systems (ANS) is proposed. This includes a module for the operating environment, comprising weather, waves, current, location and interactive traffic situation. The interactivity

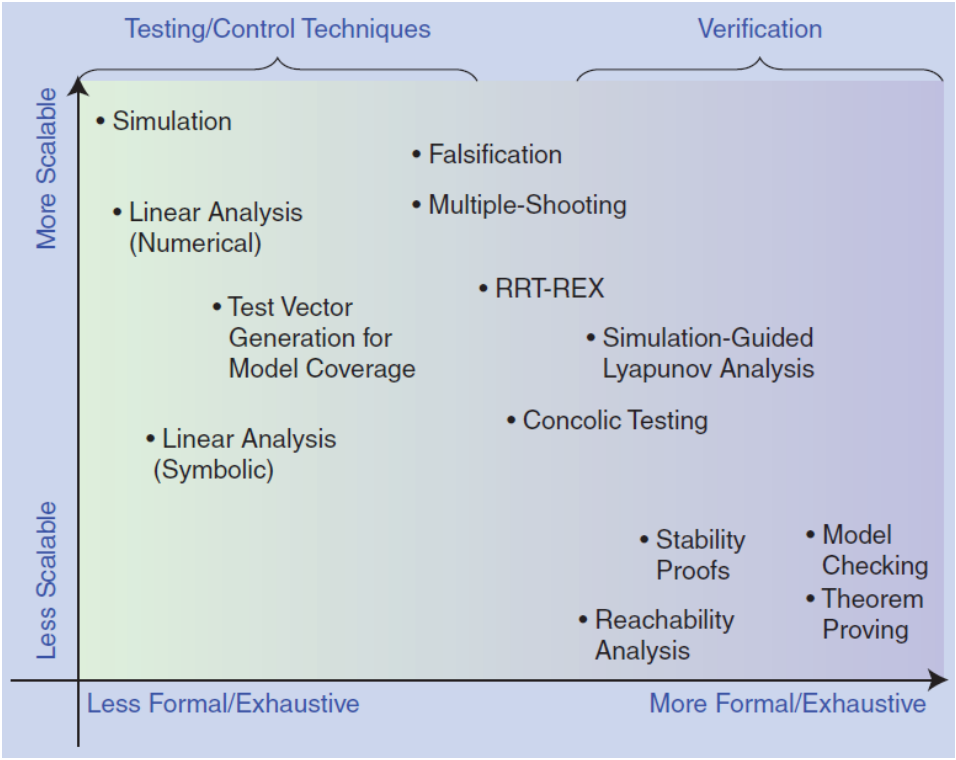
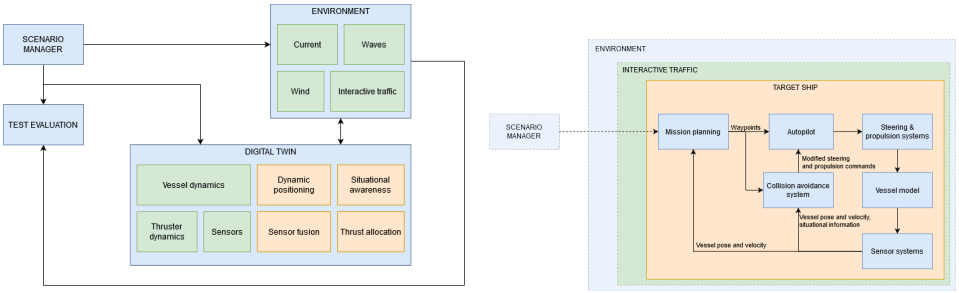


Figure 6.1: Techniques for testing and verification. Source: [5]

of the traffic is emphasised, as the simulated vessels (denoted ”target ships”) will interact with each other as well as the vessel model being tested (denoted ”test ship”). Another part of the method is a scenario manager, which is responsible for setting up various scenarios where traffic is considered. An overview of an example of the TestIT architecture is seen in Figure 6.2a, with submodules of the digital twin being examples of systems relevant for milliAmpère.

A simulator for the small autonomous passenger ferry milliAmpère utilizing the Open Simulation Platform has been developed. It is similar in structure to TestIT, and includes a digital twin of the vessel itself (vessel- and thruster dynamics, sensors etc.) and an environmental model. Since real ships will behave differently in proximity to other ships, the target ship will need to be navigated by some control system to achieve the desired interactivity. A COLREG-compliant collision avoidance system (CAS), as described in [29], is attractive since it follows the traffic rules used by humans. Figure 6.2b illustrates how the system in [29] can be utilized in conjunction with the milliAmpère simulator to simulate target ships. The vessel model, sensor systems etc. are not required to be as complex as those of the digital twin, since the target ships are not the systems being tested. For example, there is not necessarily any need to simulate sensors. If simulating a sce-

nario with COLREGs-compliant behaviour, target ships could simply have direct access to the data needed for the CAS algorith, i.e its own position and velocity, and those of the test ship and other target ships. However, simulating paths that are not in accordance with COLREGs is also desirable since human navigators are not perfect. Plausible causes for these behaviours could be due to intoxication or negligence of the navigator or ship faults on propulsion or sensor equipment. In the latter case, the developed sensor FMUs may be used in the ”Sensor systems” block in Figure 6.2b. Then, various failure modes can be triggered, to simulate scenarios where the target ships experience a sensor failure, with following unpredictability in behaviour. Some example models in the OSP have been released for free use, including vessel models of the RV Gunnerus, an offshore DP ship and a construction vessel. These are however quite a bit larger than the vessels making up the traffic where milliAmpère will operate. A preliminary solution is to use the vessel model for milliAmpère but with a different thruster setup to simulate ships with a rudder and propeller (milliAmpère has two azimuth thrusters).



(a) Overview of TestIT system presented in [28].

(b) Proposed schematic for target ship using the system in [29].

Figure 6.2: Block diagrams for proposed test system.

The work in this thesis can be used to obtain waypoints for realistic paths in the milliAmpère OSP simulator. Furthermore, the traffic statistics provides an idea of likely speed ranges and traffic density of vessels in the operating area, which can be used for scenario selection. In the context of a scenario generation module in the simulator, one can define a number of different traffic scenarios for simulation. For example a head-on situation might be simulated, where the target ship will approach along a realistic path. A traffic scenario for a target ship may be parameterized by a start and stop position and a start time. A corresponding set of waypoints outputted from the proposed algorithm can then be used for path following by the target ships. A waypoint database populated with sailings covering likely traffic patterns has been created using the proposed algorithm. Thus, by defining a procedure with the desired start and stop position as input, it is a matter of a database lookup to retrieve the set of waypoints most closely matching the input. Schematically, this belongs in the mission planning block in Figure 6.2b. By setting up the simulation of multiple target ships with different paths and start times, one can create complex traffic scenarios and evaluate test ship performance in these events. Due to the lack of variance in measured speeds in the data set, most of the sailings in the database have speeds near

average. Since it is desirable to be able to select scenarios with high target ship speeds, such paths can be drawn directly from the historical data set. As long as the target vessel is being actively controlled, this path will not be merely reproduced. If the operating area is changed, or more data with high speeds is collected, it is possible to restrict the dataset to only contain samples with speeds close to the desired one. Subsequently running the RL algorithm on this data set will yield realistic high speed paths between two given points.

Chapter 7

Conclusions and recommendations for future work

7.1 Conclusions

The main research question of this thesis was whether we could use machine learning to learn vessel paths, ideally while upholding boating rules. AIS-data for the main proposed operating area of the autonomous passenger ferry milliAmpère was collected, processed and analysed. Subsequently, this data was used to create reward signals for reinforcement learning algorithms that obtained paths between locations, defined by sets of waypoints. The data set revealed different sailing patterns, according to entry and exit points. Some variance in patterns was also observed across spatial and temporal dimensions, with implications for machine learning on this data. Testing the algorithms on a selected number of cases shows that the learned paths converge towards paths compliant with traffic rules, as episode number increases. This is also the case when the underlying data set is less dense, but a greater number of episodes is needed for good results. A method for improving algorithm performance by telling the reinforcement learning agent where to direct its search was also developed. This decreased run time by causing faster stabilization of the algorithm. The results show that the main research question has been answered in the affirmative.

A suggestion for the inclusion of the present work in an OSP simulator for milliAmpère have been provided. This suggestion leverages previous work, with the aim of simulating realistic traffic scenarios. This is intended to be a part of the environmental module of a simulation-based test method for milliAmpère. The environmental module describes relevant aspects of the operating environment, including traffic. Developed sensor FMUs with various failure modes can be used in the simulation of target ships: By triggering failure modes, unexpected behaviour is caused, which is relevant for testing systems like collision avoidance.

7.2 Recommendations for future work

Observations on differences in sailing patterns have been noted. This has implications for methods derived from historical data. For path planning, the validity of the data in the context of the current time-frame must be ensured. Improvements to the developed algorithms have been proposed in the above. A scheme for reducing the learning rate, and increasing the parameter ϵ for more frequent selection of the locally optimal action is suggested. Additionally, penalties for unwanted behaviour like frequent speed changes can be administered. However, a more attractive proposition is to use the created data set and the proposed reward function in the implementation of a more modern reinforcement learning algorithm. A good candidate is DDQN.

Bibliography

[1] M. Ludvigsen and A. J. Sørensen. Towards integrated autonomous underwater operations for ocean mapping and monitoring. *Annual Reviews in Control*, 42:145–157, 2016.

[2] I. B. Utne, A. J. Sørensen, and I. Schjølberg. Risk management of autonomous marine systems and operations. In *International Conference on Offshore Mechanics and Arctic Engineering*, volume 57663, page V03BT02A020. American Society of Mechanical Engineers, 2017.

[3] Society of Naval Architects, Marine Engineers (U.S.). Technical, and Research Committee. Hydrodynamics Subcommittee. *Nomenclature for Treating the Motion of a Submerged Body Through a Fluid: Report of the American Towing Tank Conference*. Technical and research bulletin. Society of Naval Architects and Marine Engineers, 1950.

[4] A. J. Sørensen. Marine cybernetics : Towards autonomous marine operations and systems : Lecture notes.

[5] J. Kapinski, J. V. Deshmukh, X. Jin, H. Ito, and K. Butts. Simulation-based approaches for verification of embedded control systems: An overview of traditional and advanced modeling, testing, and verification techniques. *IEEE Control Systems Magazine*, 36(6):45–64, 2016.

[6] H. Huang. Autonomy levels for unmanned systems (alfus) framework: Safety and application issues. In *Proceedings of the 2007 Workshop on Performance Metrics for Intelligent Systems*, PerMIS '07, pages 48–53, New York, NY, USA, 2007. ACM.

[7] C. C. Insaurralde and D. M. Lane. Autonomy-assessment criteria for underwater vehicles. In *2012 IEEE/OES Autonomous Underwater Vehicles (AUV)*, pages 1–8, Sep. 2012.

[8] *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*, jun 2018.

[9] Rolls-Royce. Rolls-royce and finferries demonstrate world's first fully autonomous ferry. https://www.rolls-royce.com/media/press-releases/2018/03-12-2018-rr-and-finferries-demonstrate-worlds-first-fully-autonomous-ferry.aspx. Last visited: 06.08.2020.

[10] K. Nordal. Yara birkeland press kit. https://www.yara.com/news-and-media/press-kits/yara-birkeland-press-kit/. Last visited: 06.08.2020.

[11] P. Liljebäck and R. Mills. Eelume: A flexible and subsea resident imr vehicle. https://www.kongsberg.com/maritime/about-us/news-and-media/our-stories/eelume-a-flexible-and-subsea-resident-imr-vehicle/?OpenDocument. Last visited: 06.08.2020.

[12] C. Steinbrink, S. Lehnhoff, S. Rohjans, T. I. Strasser, E. Widl, C. Moyo, G. Lauss, F. Lehfuss, M. Faschang, P. Palensky, et al. Simulation-based validation of smart grids–status quo and future research trends. In *International Conference on Industrial Applications of Holonic and Multi-Agent Systems*, pages 171–185. Springer, 2017.

[13] T. A. Johansen and A. J. Sørensen. Experiences with hil simulator testing of power management systems. In *Marine Technology Society Dynamic Positioning Conf*, 2009.

[14] T. A. Johansen, T. I. Fossen, and B. Vik. Hardware-in-the-loop testing of dp systems. In *Dynamic positioning conference*, page 33. Citeseer, 2005.

[15] Ø. Smogeli. Experiences from five years of dp software testing. 2010.

[16] DNV GL. Standard for hardware in the loop (HIL) testing, 2016.

[17] S. W. Shepperd. Quaternion from rotation matrix. *Journal of Guidance and Control*, 1(3):223–224, 1978.

[18] T. I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley Sons, Ltd, Chichester, UK, 2011.

[19] R. W. Beard. Small unmanned aircraft : theory and practice, 2012.

[20] R. Mahony, T. Hamel, and J. Pflimlin. Nonlinear complementary filters on the special orthogonal group. *IEEE Transactions on Automatic Control*, 53(5):1203–1218, June 2008.

[21] U. Bhatti and W. Ochieng. Failure modes and models for integrated gps/ins systems. *Journal of Navigation*, 60:327 – 348, 05 2007.

[22] R. S. Sutton. Reinforcement learning : an introduction, 2018.

[23] C. J. C. H. Watkins. Learning from delayed rewards. 1989.

[24] R. E. Bellman. *Adaptive control processes: a guided tour*, volume 2045. Princeton university press, 2015.

[25] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *arXiv preprint arXiv:1509.06461*, 2015.

[26] A. Donzé. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *International Conference on Computer Aided Verification*, pages 167–170. Springer, 2010.

[27] Y. Annpureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan. S-taliro: A tool for temporal logic falsification for hybrid systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 254–257. Springer, 2011.

[28] T. Pedersen, J. Glomsrud, E-L. Ruud, A. Simonsen, J. Sandrib, and B-O. Eriksen. Towards simulation-based verification of autonomous navigation systems. *Safety Science*, 129:104799, 09 2020.

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