University of Novi Sad
Faculty of Sciences
Department of Mathematics
and Informatics

Autonomous Ship Navigation Using Reinforcement Learning

Master Thesis

Tatjana Cucić

Supervisor: dr Dejan Vukobratović

Novi Sad, 2026

Acknowledgements

I would like to express deepest and sincere gratitude to my thesis supervisor dr Dejan Vukobratović for connecting me with the dr Sebastien Lefond from Abo Academy in Turku and enabling me to take part in a very exciting real-life research project. My gratitude also goes to dr Sebastien Lefond and dr Seponoud Azimi, who were my mentors during my time in Turku, Finland, and throughout the course of this research. I would also like to express my gratitude to Richard Nyberg, my colleague from Abo Academy with whom I cooperated during the research process of the project.

Finally, I must express my very profound gratitude to my mother Vesna, my husband Milan and my dog Balu for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you. However, the biggest thank you goes to my one-year-old daughter, Iskra, who was my ultimate motivation to finally finish my degree.

Contents

1 Abstract **6**

2 Introduction **7**

3 Related Work **9**

4 Data **12**

5 Methods **17**
5.1 Markov Decision Processes 17
5.2 Reinforcement Learning 18
5.3 Imitation Learning 19
5.4 Q-Learning 20
5.5 Metrics 22

6 Reinforcement Learning Model **23**
6.1 Model design 24
6.2 Model training 25
6.3 Post-processing 30
6.4 Results 30

7 Conclusion **32**

Bibliography **33**

Biography **36**

8 Appendix A **37**
8.1 GitHub Code 37

9 Appendix B **38**
9.1 **AIS parameter explanations** 38

10 Appendix C **41**
10.1 **NMEA documentation** 41
10.1.1 **NMEA (0183) Parameters** 41

11 Appendix D **46**
11.1 **AIS and NMEA comparison** 46
11.1.1 **AIS and NMEA compared** 46

List of Figures

4.1 Meaning of the different parts of an AIS message type 1. . . . 14
4.2 Meaning of the different parts of a NMEA sentence. 15

6.1 Plot of the list of rewards for all episodes 30

List of Tables

10.1 1.1 ZDA – Time & Date – UTC, day, month, year and local time zone 42
10.2 1.2 GLL – Geographic Position – Latitude/Longitude 42
10.3 1.3 GGA – Global Positioning System Fix Data 43
10.4 1.4 VTG – Track made good and Ground speed 44
10.5 1.5 RMC – Recommended Minimum Navigation Information . 44
10.6 2.1 VHW – Water speed and heading 45

Chapter 1

Abstract

This thesis focuses on efforts to create a dataset that is suitable for the application of machine learning algorithms for the purpose of autonomous navigation. More precisely, it is about creating a dataset that is varied and rich enough for the use of reinforcement learning. It also contains the documentation about all the parameters in the dataset, as well as the terminology that is essential for anyone who decides to use the dataset, regardless of professional background. In the thesis, two different methods are reviewed for obtaining the data from a ship simulator with documentation for the parameters, the code for transforming the data into a usable format and the related work that is relevant both for the development of autonomous navigation and data collection.

Chapter 2

Introduction

Recently, there has been a lot of research regarding autonomous systems, especially autonomous cars. Apart from cars being the most popular vehicle for automation, other types of vehicles have become a topic of interest, as well. The focus of my work and this document is to set the foundations for autonomous ship navigation.
Human error is the main cause of accidents at sea, which is one of the reasons autonomous systems are needed. Apart from improving safety, optimizing fuel consumption is another positive aspect of developing autonomous navigation systems. Of course, creating such a system is a challenging task. Although it may seem that the issue of autonomous cars is similar to the one of autonomous ships, the issue of making ship navigation autonomous is quite different. For instance, cars have roads and a limited number of directions to go, as well as paths, while ships can move in any direction. There are no roads at sea and traffic rules are much less strict than they are for car traffic, they often depend on human judgement and communication between ships.
One of the main issues with the development of autonomous ship navigation is a lack of data, as well as its availability. In order to successfully create an autonomous system, there should be a variety of navigation scenarios, as well as enough parameters to log all the elements that affect ship navigation. Large datasets are required to successfully use artificial intelligence in many cases and this one is no different. Since simulation-based learning has become very popular and successful, the goal of this project was to collect data that is suitable for implementation of machine learning. After that, the goal was to develop a reinforcement learning and imitation learning algorithm for

autonomous navigation and collision avoidance with multiple obstacles that includes environmental forces, ship manoeuvrability and more complex scenarios. The idea is that the work described in this paper with future work on the subject would provide a more complete development of autonomous ships that covers a wide variety of situations, conditions and ships. The ultimate goal is to make ships autonomously achieve optimal fuel consumption, travel time and safe navigation.

Given that a large portion of work was data collection, I determined what data is available and how to obtain it, created a documentation of parameters and provided understanding of data from different sensors, as well as calculated values. My source of information were different bridges of a ship simulator at Aboa Mare in Turku, Finland. Creating such a dataset using the data from real ships would be extremely difficult and impractical, hence, the opportunity I had to use the data from a very realistic ship simulator at Aboa Mare was extremely valuable. The datasets are meant to be public and open for other researchers to use which will contribute to a faster and more efficient development of autonomous ship navigation systems.

Chapter 3

Related Work

There have been a number of research papers on topics that are closely related to autonomous vehicle algorithms and data collection processes. Autonomous cars are more popular compared to other types of autonomous vehicles and they are more thoroughly researched, however, there are also papers that tackle the problem of designing autonomous ships. Most of them focus either on path following or collision avoidance. This literature review consists of two parts – data collection and autonomous navigation.

Gathering and processing data has been the topic of many research papers. In 1997, J. Llinas et al. [1] explained a process of data fusion from multiple sensors. The process applies to different types of sensors for different purposes. Optimization of data gathering for energy efficiency from wireless sensor network nodes using a mobile robot was the focus of the paper by O. Tekdas et al. [2] (2009). In 2015, M. Redmayne [3] recognized the problem of processing large amounts of data after a hydro graphic survey. As a solution, he described an autonomous method for collecting and processing of data during a survey in near real time. The method includes using some sensor data in order to create a model, as well as scaling down to a manageable size. A new model was introduced in 2015 by M. Menze et al. [4] for scene flow estimation in 3D. They used the KITTI dataset and improved segmentation of dynamic objects. In 2016, I. Cermakova et al. [5] collected data from an unmanned aerial vehicle (UAV) and classified the data in order to identify shorelines. A system with multiple wireless sensor networks (WSN), a ground control station (GCS) and two UAVs was set up by E. Vlasceanu et al. [6] (2019). The UAVs collect data from the WSNs and send it to the GCS along with their own position, speed and altitude.

The GCS calculates and sends new flight trajectories to the UAVs. In 2018, T. Kim et al. [7] designed an AUV with a wide range of motion and precise data collection ability which allows recognition of terrain and obstacles. A comparison of datasets for autonomous cars was done by J. Guo et al. [8] (2019) containing 54 datasets. They emphasize the importance of collecting all types of data including drivers' gaze and environmental data using all types of sensors. They also compare different ways of processing images used for autonomous cars. In 2019, J. Xue et al. [9] collected several environmental and control parameters from the Navi-Trainer Professional 5000 and analysed the key factors affecting the maneuverability of ships. A. C. Nica et al. [10] (2019) recognized the lack of descriptions of the data collection process in research papers. Therefore, they decided to create their own dataset and describe the steps of data collection and processing. They took measures to obtain quality data in terms of synchronization which is often equally important as quantity. Hardware and software tools descriptions are also included and the dataset was made public. In 2019, M. Meyer and G. Kuschk [11] created a dataset from a radar, lidar and camera for autonomous cars and used deep learning classification to identify objects. M. Novitzky et al. [12] (2019) created a competition of robots and humans in motorized kayaks. The goal of the experiment was to create as complete of a dataset as possible. The dataset includes questionnaires, performance data, video, audio, each vehicle's track data and the game state. The focus was human-robot communication and the dataset was made available to the public for the purpose of further research.

Recently, a lot of progress was made in the development of autonomous vehicles such as UAVs, AUVs, ships and cars. In 2013, S. Garg et al. [13] developed a prototype boat which follows a predefined path and avoids static obstacles using sensor values to determine the position of obstacles. A Q-learning based method for following a moving target underwater was developed by T. Sun et al. [14] (2015). The result was a successful simulation of the behaviour of the unmanned underwater vehicle. In 2017, R. Polvara et al. [15] introduced their own method for collision detection and path planning using artificial neural networks and a graph-search algorithm. They also present a kinematic model for ships which considers position, orientation and forces from current, wind and waves. The article also compares the capabilities of different types of sensors such as radar, lidar, sonar, visual and infrared sensors. A Q-learning approach was also used by C. Wang et al. [16] in 2019. They used the algorithm to plan a path in an unknown environment

and simulated the algorithm in pygame where after 1500 iterations a near optimal path is found and obstacles are avoided.

The research paper most similar to our goal was written by Zhao et al. [17] in 2019. They developed a deep reinforcement learning algorithm that includes both path following and collision avoidance. They designed a three-degree-of-freedom dynamic model for autonomous ships and used a line-of-sight (LOS) guidance system to enable the ship to follow a specific path. After that, they used a proximal policy optimization (PPO) algorithm for training their network. Simulations were done to test the algorithm using ROS. The environment was an open sea and they tested collision avoidance for 3 different types of scenarios with one other ship. Results showed that the autonomous ship is able to successfully avoid obstacles in compliance with the International Regulations for Preventing Collisions at Sea (COLREGs). However, no information was given about data collection.

Most recently, L. P. Perera [18] (2020) wrote a review of the possibility of autonomous ship navigation and collision avoidance while also considering scenarios which are not clearly defined in COLREGs. The paper also covers the topic of ship maneuverability considering a large inertia, strong environmental forces and under-actuated controls, i.e., the rudder and engines. In the paper there is also a discussion about decision support facilities including digital maps, lidar, sensors, AIS and vessel traffic information (VTI).

Chapter 4

Data

All the data that was used and accessible was derived from a ship simulator Navi-Trainer Professional 5000, made by Transas which was acquired by Wärtsilä in 2018 [19] (Transas information on Wärtsilä's website). This simulator is used by students at the Maritime Academy Aboa Mare in Turku for learning about ship navigation and take practical parts of exams. It includes realistic physical bridges, many realistic marine devices and an instructor's station for creating and monitoring simulations [20]. There were three types of files obtainable from the simulator - AIS, UHI and NMEA files.

Data Collection

Machine learning algorithms demand large datasets with many different scenarios for a computer to be able to learn a task. The aim of this research project was to collaborate with Aboa Mare, collect data from their ship simulators and learn about the maritime industry from them. The idea was to collect any data that might be relevant to ship navigation and to create a machine learning algorithm which would be able to perform some of the captain's duties. It was discovered that their simulator is able to generate such data but also much more. The simulator is also able to generate files containing weather data, emissions and anything that might be needed for navigation.

The data available at Aboa Mare's simulators is collectable with two different methods, the first one is by running a program which replays a simulation and simultaneously generates a comma separated values (CSV) file with the parameters that were selected. This method gives us the data in the format we want and there are several hundred parameters to select, depending on the ship model that was used. Using only this type of file it should be

feasible to use machine learning to handle navigation. The drawback is that each scenario takes some time to replay and to select parameters, the fastest replay is at 20 times the normal speed.

There is also an alternative method for collecting data from the simulators, a data logger was created by an employee at Aboa Mare which was for a few months collecting data from every simulation that was run. This program gave as output three files called UHI, AIS and NMEA. UHI files contain new values for changes in the simulation, AIS files contain encoded messages in the same format as a real ship would send out and NMEA files contain sentences with data from sensors and other equipment. For these files a script was created to convert the data in the files into CSV format. Unfortunately, the data logger had not worked correctly and failed to collect the NMEA files except for a few files and eventually the whole logger stopped running. We were sent their whole collection which contains AIS and UHI files for about two months or 700 simulations.

Dataset Description And Data Preprocessing

1. AIS files
   The automatic identification system [21] (AIS) is a system that sends and receives information about position, heading and speed of ships. AIS typically uses radio signals in the very high frequency (VHF) range, from 30 to 300 MHz, to transmit messages. The messages can be used by an electronic chart display and information system (ECDIS) to show other ships' position and heading on an electronic nautical chart. I have received files containing both sent and received AIS messages and made a decoder which extracts the useful data and saves it as a CSV file.
   The original AIS files are comprised of encoded data, checksum and five characters indicating whether the data is from the own ship or other ships. Own ship refers to a ship that is controlled by a person in the simulation, this can be multiple ships at once. The different own ships and other ships can be differentiated by their unique ship IDs. Once a file is loaded into a program, it takes each line of the file and separates the part of the line containing the data.

   If the line does not contain any data, the line is skipped. Based on the first five letters, a column is created indicating whether the data is about own ship or other ship. Decoding the encoded data is done by taking the whole sequence of ASCII characters and converting each character to its corresponding decimal number. From this number, 48

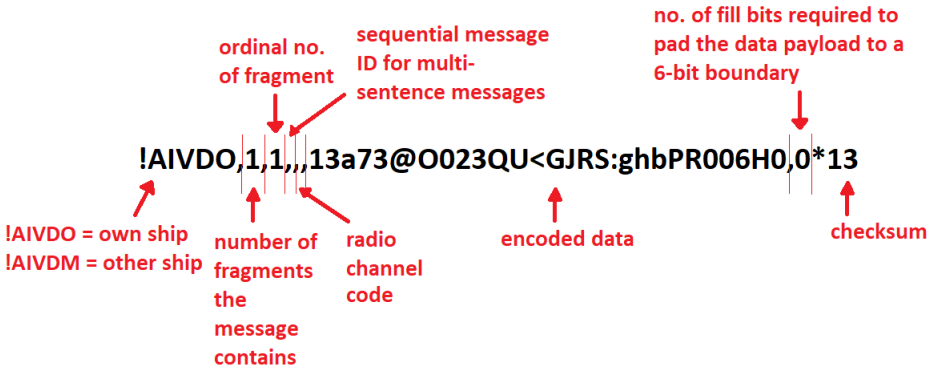

Figure 4.1: Meaning of the different parts of an AIS message type 1.

is subtracted and if the result is greater than 40 then another 8 is subtracted. Then each number is converted to six-bit binary which is then concatenated into a long binary sequence. According to the instructions in the documentation [22] (AIVDM/AIVDO protocol decoding) we take fields of different lengths and convert the binary into integer. Some fields are complete after the conversion, others need additional calculations. Finally, we create a CSV file containing these values and the parameter names. There are 27 possible formats for AIS messages, the programs and documentation were made only for the messages found in the data that was received (see appendix B).

2. UHI files
   Universal Hardware Interface (UHI) files are given in JSON format. The file consists of sections of attribute-value pairs. The attributes are parameter names. Each section represents an update of own ship's data. After determining the full set of parameters that are in the files, it was concluded that each section can contain a different subset of attributes. Unfortunately, there is no available documentation regarding this type of file. However, the data was not encoded and the meaning of parameter names was possible to infer. This enabled the parsing of the original files and turning them into CSV files.

3. NMEA files
   NMEA 0183 is a proprietary specification for communication between marine devices such as GPS, gyro compass, autopilot, and sonar. The

specification is defined by the National Marine Electronics Association. There are other, newer specifications such as NMEA 2000 which allow much higher bandwidth, however, this standard has not yet started to replace the earlier version. The data in the NMEA files that was received contain information about position, heading, speed, depth measurements and wind speed. The data is given as "sentences", which consist of two identifiers, some data and a checksum. The two identifiers indicate what type of device the data comes from and which unique "sentence" the values belong to.
An NMEA sentence could look like this:

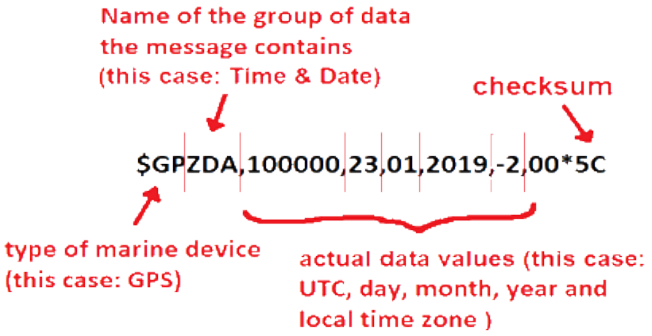

Figure 4.2: Meaning of the different parts of a NMEA sentence.

The program that was made during the project takes a whole folder as input and processes each file and converts it into a CSV that is comprehensible with parameter names and values in order. The resulting files consist of 81 columns each containing data for a unique parameter name throughout the simulation. In the beginning, we choose the directory and start processing all files in the folder, one by one. After a file is loaded, each line is read separately and put into a list where each element is separated by a comma. The commas are removed, and we are left with a list of strings and floats. In order to parse the data in the right manner, the program needs to recognize the type of sentence. If-else statements are used to identify the group our message belongs to. Each sentence contains a specific set of parameters. Once the program identifies the type of sentence, it places the values into corresponding

columns in the new CSV file. If some values are missing, they are marked as not a number (NaN). Once all message types are iterated through, the result is written into a CSV file containing headers and values for all the parameters. We made documentation about the data in the CSV files for the sentences in our data (see Appendix C).

4. CSV files from report tool
   In the end, the largest and most useful dataset we were granted is the data obtained from a kind of a report tool. It is a tool that could download a CSV file containing all sorts of relevant measurements regarding the movement of the ship, as well as environmental factors. Each file of the dataset consists of 55 features. The simulations that were selected for this project eventually were ones that were done in Rotterdam, 189 simulations from 2018 and 234 from 2019. The reason only one city was selected is the fact that we wanted to use the same map in order to train the algorithm as well as possible and this dataset was most complete of all.

Chapter 5

Methods

The next sections are about Markov decision processes, more precisely, reinforcement learning and imitation learning, as these are the approaches the dataset is created for. The two approaches can use similar datasets, however, imitation learning should have exclusively desirable behaviors to learn.

5.1 Markov Decision Processes

A Markov Decision Process (MDP) is a mathematical concept used for modeling decision-making in situations where outcomes are somewhat random and somewhat under the control of a decision-maker [23]. MDPs are often used in areas such as reinforcement learning, operations research, economics, and artificial intelligence.

Components of MDPs: A MDP is characterized by the following elements:

1. States (S): A set of possible states that the system can be in.
2. Actions (A): A set of actions the decision-maker can do in each state.
3. Transition Model (T): A probability distribution that defines the likelihood of moving from one state to another after taking a specific action.
4. Reward Function (R): A function that assigns a numerical reward for each state or action, indicating the immediate benefit or drawback of being in a certain state or taking a certain action.

5. Policy (π): A policy is a strategy that defines the action that a decision-maker will use from each state. A policy can be *deterministic* or *stochastic*.

Bellman Equation and Optimal Policy MDP has a goal of finding an optimal policy that maximizes the expected cumulative reward over time. The expected cumulative reward is often discounted using a discount factor γ. The Bellman equation plays a key role in MDPs, defining the value of a state under a given policy as the expected return starting from that state.

The Bellman equation is:

$$V^{\pi}(s) = \mathbb{E}\left[R(s, \pi(s)) + \gamma V^{\pi}(s')\right]$$

Where:

- $V^{\pi}(s)$ represents the value function under policy π - $\mathbb{E}$ denotes the expectation operator - γ is the discount factor, controlling the importance of future rewards - $\pi(s)$ represents the action taken in state s. - s' is the next state in the process

Solving MDPs There are several methods for solving MDPs and finding optimal policies:

- Value Iteration: Iteratively updates the value of each state based on the Bellman equation.
- Policy Iteration: Alternates between evaluating a policy and improving it.
- Q-Learning: A model-free approach, where the agent learns the value of action-state pairs directly.

5.2 Reinforcement Learning

The plan was to use reinforcement learning and imitation learning to train a network. Reinforcement learning falls under the category of machine learning which is neither supervised nor unsupervised. It is based on a state-action-reward system. In the beginning, all states, actions, and rewards are defined.

The reinforcement learning algorithm tries to discover the most optimal set of steps or actions to take in order to obtain the maximal reward. There are also negative rewards that are assigned for bad behavior. A reward is given for the successful completion of a set of steps rather than just one action. Otherwise, it would be very difficult to assign rewards since certain steps would sometimes contribute to achieving the desired goal, while at other times, they would be counter-productive. One of the main challenges of reinforcement learning is determining the ratio between exploration and exploitation. Exploitation is using steps that are known to be effective, while exploration means trying out new steps with unknown outcomes and the goal is to optimize the ratio between the two in order to get the best result.

All reinforcement learning is based on an interaction between an agent and the environment. An agent is a part of a system that interacts with the environment. The rest of the system is also a part of the environment, along with the surroundings of the system. Formally, the reinforcement learning problem is defined as a Markov decision process (MDP). R. S. Sutton and A. G. Barto wrote: "MDPs are a mathematically idealized form of reinforcement learning problem for which precise theoretical statements can be made." The main elements of MDP are state, action, and reward. A set of states represents all possible statuses of our system obtained from the information we have. The information can be measured, observed, or calculated. A set of actions contains all the choices an agent can make. Reward functions are used to reinforce the desired behavior of the agent. At each time step, an agent decides to transition from the current state to the next one by choosing an action based on a policy. A policy is a mapping from the set of states to the set of probabilities of taking an action given by a state. This set of probabilities is affected by value functions and it changes over time. We calculate value functions for each state based on rewards an agent gets during an episode.

5.3 Imitation Learning

Since reinforcement learning can sometimes be extremely challenging in terms of designing the reward functions in the right way to enable the agent to learn the best policy for achieving the end goal, there is another method that eases this issue. Contrary to reinforcement learning where an agent needs to learn to differentiate between good and bad behavior, as well as develop an optimal

policy for maximizing the reward function and achieving the ultimate goal, imitation learning offers an answer that surpasses these issues. It consists of experts running demonstrations that are perfectly correct and offer the best way to reach the goal. The only thing our network needs to do is to learn this exact behavior which is already optimal. This is also a Markov decision process, however, it is different from the one we have in RL. In this case, reward functions are unknown, while states, actions and policy are familiar. This is different from RL, where we are trying to find an optimal policy using reward functions that are familiar.

Both Reinforcement Learning and Imitation Learning have their advantages and disadvantages and they are chosen depending on the nature of the problem. The main question is whether it is more difficult to design reward functions or provide demonstrations done by experts to train the network.

5.4 Q-Learning

Q-learning is a model-free reinforcement learning algorithm used to train agents in various environments, where the agent learns which actions to take that maximize cumulative rewards. The main idea behind Q-learning is for the agent to learn an optimal policy by interacting with the environment and to improve over time based on rewards it receives for specific actions in particular states.

The main concepts of Q-learning include:

1. State (S): A representation of the agent's current position in the environment.

2. Action (A): A set of possible moves or decisions the agent can make.

3. Reward (R): The feedback the agent receives after taking an action in a given state. Positive rewards are given for good actions, while negative rewards (or penalties) for poor choices.

4. Q-Value (Q-function): This represents the expected utility (or cumulative future reward) of taking a particular action in a given state. Q-values are updated as the agent explores the environment, leading it towards an optimal policy.

The Q-Learning Algorithm Q-learning is an off-policy algorithm, meaning it learns from actions outside of its current policy and can update the Q-values even if the actions are not optimal. The key component of Q-learning is the Q-table, which stores the Q-values for each state-action pair.

The Q-values are updated using the following rule:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \cdot \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right] \tag{5.1}$$

- $Q(s_t, a_t)$: The current Q-value for the state s_t and action a_t.
- α: The learning rate (how much new information overrides the old).
- r_{t+1}: The reward received after taking action a_t from state s_t.
- γ: The discount factor (how much future rewards are valued over immediate rewards).
- $\max_{a'} Q(s_{t+1}, a')$: The maximum Q-value for the next state s_{t+1}, over all possible actions a' in that state.

Exploration vs. Exploitation:

- Exploration: The agent explores the environment by trying out new actions, even if they are not optimal, to learn more about the environment.
- Exploitation: The agent exploits its current knowledge by selecting the action that maximizes the expected reward based on the learned Q-values.

A popular method for managing the exploration-exploitation trade-off is the epsilon-greedy policy, where the agent occasionally selects a random action with a probability ϵ and chooses the best-known action with a probability $1 - \epsilon$.

Steps:

1. Initialize the Q-table: The Q-table is usually initialized with zero or random values for each state-action pair.

2. Agent explores the environment: The agent selects actions based on its policy (often using an epsilon-greedy strategy to balance exploration and exploitation).

3. Update Q-values: After each action, the agent receives a reward and uses the Q-learning update rule to adjust the Q-value for the corresponding state-action pair.

4. Convergence: Over time, as the agent repeatedly interacts with the environment, the Q-values converge toward the optimal values. This means that the agent has learned the best actions to take in each state to maximize its cumulative reward.

Q-learning is used in several fields such as robotics, games, finance and self-driving vehicles.

5.5 Metrics

It was difficult to find an appropriate metric for assessing the model. The output of the model in our case was a Q-table which is a table of probabilities of how likely it is that the agent will perform a certain step in each time step. We were able to test the resulting Q table in another team's simulator. It loads a Q table into a program and the program displays a map with the movement of a ship according to the Q table that was given as input. Common sense was used to asses whether the model performed well or not. If the model was successfully avoiding obstacles and using one of the shortest paths from point A to point B, it was considered a good model.

Chapter 6

Reinforcement Learning Model

This section chronologically describes the work that was done for the research project. Our first task was to visit Aboa Mare and get acquainted with the simulators and the bridges there. Ted Sjöblom was our associate during the project. He was in charge of showing us how simulators work, as well as providing the domain knowledge about ships. Aboa Mare has several rooms with bridges that are modeled after real ships. The bridges have controls, marine devices, and a view that resembles what is on a real ship. There is an instructor's room that is connected to all the bridges. From this room, you can set up simulations, monitor what is happening during the simulation, and even make changes in the simulation such as weather, traffic or introduce faults. The instructor's station is also where it is possible to extract the data from simulations [19].

In order to do the project, we needed to learn about ship terminology, as well as rules and standards regarding ship navigation. Therefore, we had a number of meetings with Ted who shared his knowledge and experience with us and answered our questions. He also provided us with literature on ships and collision avoidance rules (COLREGs). Moreover, we also needed to learn about previous research that had been done on similar topics. We had two focuses in our literature review. The first one was the process of data collection and the other one was autonomous navigation.

As for the actual data, the first sample we got was a CSV file that was collected by replaying a simulation. Once we were able to understand the parameters in the file, we started to design state, action, and rewards. Ideally, every possible parameter should be collected. However, this is not possible in practice. We discovered that different types of ships have different pa-

rameters to choose from. Also, the software that is used to extract the data can sometimes crash. This is the reason we started designing state, action, and rewards prior to collecting the data, so we could reduce the number of parameters to be extracted.

After that, we received a large sample of data collected using the Data Logger. The sample contained data from around 700 simulations including different bridges. For this type of file, we needed to create programs that would convert them into CSV files. Unfortunately, the details of the UHI files were considered confidential. For this reason, we were not able to get any documentation regarding this type of file, as well as the permission to use them. Another problem was that the Data Logger failed to log the NMEA files with the exception of just a few files. Moreover, we discovered that the Data Logger stopped working entirely about a year before our data collection (2020).

As a part of the data collection process, it is important for us to understand all the parameters and gain general domain knowledge. This is the reason parameter documentation was created for each type of file, as well as documentation about ship terminology including sketches and explanations. The documentation contains the names of all the parameters that are included in the data, their descriptions, and the units for the values.

The plan was to continue the project by finding an alternative way to collect data from the simulator and resolve legal issues regarding making the data public.

For the actual training, it was necessary to find the best and most complete data sample. Since the simulations were done in various settings and different locations, we chose around 80 simulations that happened in Rotterdam in 2018 and 2019. These datasets were most complete and most appropriate for the application of a reinforcement learning model.

6.1 Model design

Since no applicable existing model was found for this particular dataset and the problem at hand, there was a need for a creation of a brand new model which would implement a Q-learning algorithm adequately with respect to our data.

The created model implements a Q-learning algorithm where the agent interacts with the environment which is defined by location and states. The

model aims to learn the the most optimal policy, i.e. one that would maximize the reward by updating the values in the Q-table during the course of several episodes. Rewards were designed according to the location of the agent. There were restricted areas for which a negative reward was given, as well as a positive reward for when the agent arrives to the destination zone. In the end, the Q-table was updated with respect to the reward, actions of the agent and the future expected reward.

6.2 Model training

The training was done by initially loading the dataset into the model and setting the parameters. The parameters of the model are the following:

```
no_of_epochs = 10
discount = 0.6
learning_rate = 0.1
destination_reward = 10
restricted_area_penalty = -5
```

- **no_of_epochs**: This parameter defines how many iterations of training (episodes or epochs) will be performed in the training.
- **discount**: This is the discount factor (gamma, that controls the importance of future rewards. A lower value means the algorithm values immediate rewards more.
- **learning_rate**: This controls how much new information overrides the old information during Q-value updates.
- **destination_reward**: The reward given for reaching the destination. In this case, the destination is a whole area, not one exact location, it made more sense to design in this way since not all simulations end in the same location, but more of them are considered successful.
- **restricted_area_penalty**: This penalty is applied when the agent enters a restricted area.

The parameters are hard-coded in the beginning of the whole script, so they can be fine-tuned according to the needs and wishes of the person doing the training.

The actual training runs in a for loop, where the number of loops is actually the selected number of epochs we want to train.

```
for i in range(no_of_epochs):
    list_of_rewards = []
    for j in range(len(file_lengths)):
        reward_per_episode = 0
        for k in range(file_lengths[j]-1):
            index = sum(file\_lengths[:j]) + k
```

There are three for loops inside each other. As already said, the first loop is iterating through epochs, the second one iterates through simulations, while the third iterates through time steps. This way, we cover each simulation and each time step for as many epochs as we like to train.

In the beginning of the epoch loop, a list of rewards is initiated, it is there to store the cumulative rewards for each simulation. At the start of the simulation, we reset the reward stored in the `reward_per_episode` variable. Finally, the code iterates through each time step of the current simulation. Index is calculated to keep track of the state-action pairs within the large dataset.

```
state = [disc_lat[index]] + [disc_long[index]] +
↪  [disc_cog[index]]
action = [disc_rud[index]] + [disc_tel[index]]
```

The state is composed of latitude (`disc_lat[index]`), longitude (`disc_long[index]`) and course over ground (`disc_cog[index]`).

The action is defined by rudder position (`disc_rud[index]`) and telecommunication (`disc_tel[index]`).

```
if state[1] >= corners1[1][1] and state[1] <= corners1[2][1]:
↪   #penalty area 1
    reward += limit_line(corners1, state[0], state[1])

if state[1] >= corners2[1][1] and state[1] <= corners2[2][1]:
↪   #penalty area 2
```

```
    reward += limit_line(corners2, state[0], state[1])

if state[1] >= corners4[0][1] and state[1] <= corners4[2][1]
↪  and
state[0] >= corners4[1][0] and state[1] <= corners4[0][0]:
    reward += destination_reward/2    # half a reward for
    ↪  almost destination

if state[1] >= corners3[0][1] and state[1] <= corners3[2][1]
↪  and state[0] >= corners3[1][0] and state[1] <=
↪  corners3[0][0]:
    reward += destination_reward    #full reward for
    ↪  destination area
```

For each time step, we check whether our ship moved to a restricted area. There are two restricted areas, penalty area 1 and 2, in our map and entering either of them is equally penalized. As for the positive rewards, they are given either when the ship enters an area that is close enough to the destination area and for reaching the actual destination area. The logic behind rewarding the model for almost reaching the destination was to stimulate the model when it is 'on the right track' to achieving its goal. The reward for almost reaching the destination is half the reward for reaching the destination.

```
old_value = q_table[state[0], state[1], state[2], action[0],
↪  action[1]]
next_state = [disc_lat[index+1]] + [disc_long[index+1]] +
↪  [disc_cog[index+1]]
next_max = np.max(q_table[state[0], state[1], state[2]])
new_value = (1 - learning_rate) * old_value + learning_rate *
↪  (reward + discount * next_max)
q_table[state[0], state[1], state[2], action[0], action[1]] =
↪  new_value
```

Finally, the update of the Q table happens. In this piece of code, the Q-learning update is applied with these variables and formulae:

- `old_value`: The current Q-value for the specific state-action pair.

- `next_state`: The state resulting from taking the action. If the transition succeeds, the maximum Q-value for the next state is stored in `next_max`.

  The new Q-value is computed using the Q-learning update formula:

$$Q_{\text{new}}(s,a) = (1-\alpha)Q_{\text{old}}(s,a) + \alpha\left(R + \gamma \max_{a'} Q(s',a')\right)$$

  where:

  - alpha is the **learning rate**.
  - gamma is the **discount factor**.
  - R is the **reward**.

$$\max_{a'} Q(s',a')$$

    is the maximum Q-value for the next state.

  The Q-table is updated with the new value for the current state-action pair.

```
reward_per_episode += reward
list_of_rewards.append(reward_per_episode)
```

In the end, the reward for each simulation is accumulated over time steps and stored in `list_of_rewards` variable.

There were basically three models overall and their main differences were in the hyper parameters.

Model 1

```
no_of_epochs = 20
discount = 0.6
learning_rate = 0.1

destination_reward = 10
restricted_area_penalty = -5
```

Model 2

```
no_of_epochs = 10
discount = 0.6
learning_rate = 0.1

destination_reward = 10
restricted_area_penalty = -5
cpa_penalty = -5
```

Model 3

```
fail = 0
no_of_epochs = 20
discount = 0.6
learning_rate = 0.2

destination_reward = 10000
approaching_reward = 1
restricted_area_penalty = -5
cpa_penalty = -5
critical_cpa_penalty = -5
cpa_reward = 50
starboard_reward = 100
port_reward = 100
```

For the first 2 models, the state-action space was the same.

```
state_space = [31,91,73] #latitude, longitude, cog
action_space = [179,41] #one rudder and one telegraph
```

The third one was a bit different.

```
state_space = [5,5,3] #distance to destination,
    distance to closest ship in the front(lowest cpa),
    direction to previous state (left,front,right)
action_space = [5,3] #one rudder (-30,-15,0,15,30) and     one
↪  telegraph(50,0,speed+100)
```

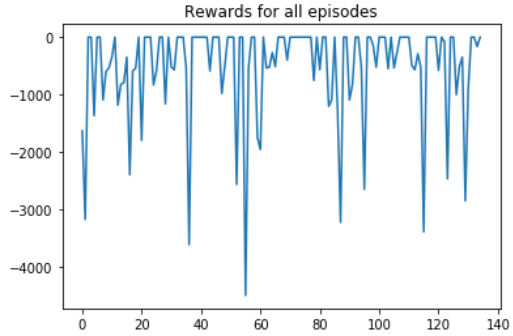

Figure 6.1: Plot of the list of rewards for all episodes

6.3 Post-processing

Unfortunately, the trained model did not perform well. There were attempts to experiments with hyper parameters, but they also proved to be unsuccessful.

Since the model is custom made, it was difficult to test it out. Luckily, another team was working on creating a program that would visualize movement of a ship given the map and the Q table. We were able to load our models into the program and see the behavior of the trained model.

Every time a model was trained, the results were unsatisfactory. The ship would still hit the restricted areas on the map and it would not follow the shortest path to the destination. There was one 'lucky' example where the visualization was satisfactory, but it is accidental, and not the result of a well-trained model.

This was the reason no particular post-processing happened as we would need a much large dataset in order to see some real results of the model training.

6.4 Results

The goal of this research project was to create a good dataset that is suitable for the application of machine learning. This dataset should contain a variety of parameters that enable a state, action, and reward design that includes many different collision avoidance scenarios. We managed to obtain a set of around 700 AIS, 700 UHI, and 5 NMEA files. We created programs that

convert these files into usable CSV files. In addition, we created documentation for parameters from the AIS and NMEA files, documentation about general ship knowledge that helps in understanding all the parameters in the data. Furthermore, we made documentation about the comparison of data in AIS and NMEA files. We observed the differences between the values of the same parameters in both file types (See Appendix D).

Chapter 7

Conclusion

In this thesis two methods are described for obtaining machine learning data for autonomous ship navigation and documentation is provided for understanding the process and the meaning of the data. This is a valuable scientific contribution to the field of autonomous ship navigation since such a dataset and documentation have not previously been made available. Furthermore, it is ensured that the collected data is usable in machine learning as there are similar parameters in our data as used by L. Zhao et al. [17] (2019) where a reinforcement learning algorithm for navigation was implemented in a simulation.

In the future, the plan is to create a reinforcement learning policy that predicts the best action, given the current state and reward function. It is unclear if the collected data allows for such a policy to be learned, however, trying may provide some insight into the problem and help to understand what might be missing in the data. As we learned more about the maritime industry, it has become clear that the reward function, i.e. the desired behavior is quite complex. In any case, the project has not ended, and there are many more aspects to consider in this challenging issue of autonomous ship navigation.

Bibliography

[1] J. Llinas and D. L. Hall. "An Introduction to Multisensor Data Fusion". In: *Proceedings of the IEEE* 85.1 (Feb. 1997), pp. 6–23.

[2] O. Tekdas, N. Karnad, and V. Isler. "Energy-Efficient Strategies for Collecting Data from Wireless Sensor Network Nodes using Mobile Robots". In: *Robotics Research: The 14th International Symposium ISRR*. 2009.

[3] M. Redmayne. "Autonomous Onboard Hydrographic Data Processing". In: *OCEANS 2015 – MTS/IEEE Washington*. Oct. 2015.

[4] M. Menze and A. Geiger. "Object Scene Flow for Autonomous Vehicles". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015.

[5] I. Cermakova and J. Komarkova. "Modelling a Process of UAV Data Collection and Processing". In: *International Conference on Information Society*. Oct. 2016.

[6] E. Vlasceanu, D. Popescu, and L. Ichim. "Multi-UAV Architecture for Ground Data Collection". In: *IEEE International Geoscience and Remote Sensing Symposium*. July 2019.

[7] T. Kim, J. Kim, and S. Yu. "Design of a Buoyancy Controllable AUV by Changing Volume for Data Collection in the Water Column". In: *IEEE/OES Autonomous Underwater Vehicle Workshop (AUV)*. Nov. 2018.

[8] J. Guo, U. Kurup, and M. Shah. "Is It Safe to Drive? An Overview of Factors, Metrics, and Datasets for Driveability Assessment in Autonomous Driving". In: *IEEE Transactions on Intelligent Transportation Systems* (2019). Early Access, pp. 1–17.

[9] J. Xue et al. "Grey Relational Analysis of Environmental Influencing Factors of Autonomous Ships' Maneuvering Decision-Making". In: *The 5th International Conference on Transportation Information and Safety*. July 2019.

[10] A. C. Nica et al. "Collecting and Processing a Self-Driving Dataset in the UPB Campus". In: *22nd International Conference on Control Systems and Computer Science (CSCS)*. 2019.

[11] M. Meyer and G. Kuschk. "Automotive Radar Dataset for Deep Learning Based 3D Object Detection". In: *16th European Radar Conference (EuRAD)*. Oct. 2019.

[12] M. Novitzky et al. "Aquaticus: Publicly Available Datasets from a Marine Human-Robot Teaming Testbed". In: *14th ACM/IEEE International Conference of Human-Robot Interaction (HRI)*. Mar. 2019.

[13] S. Garg, R. K. Singh, and R. Kapoor. "Autonomous Ship Navigation System". In: *Texas Instruments India Educators' Conference*. 2013.

[14] T. Sun et al. "Target Following for an Autonomous Underwater Vehicle Using Regularized ELM-based Reinforcement Learning". In: *OCEANS 2015 – MTS/IEEE Washington*. 2015.

[15] R. Polvara et al. "Obstacle Avoidance Approaches for Autonomous Navigation of Unmanned Surface Vehicles". In: *Journal of Navigation* 71.1 (2017), pp. 241–256.

[16] C. Wang et al. "Path Planning of Maritime Autonomous Surface Ships in Unknown Environment with Reinforcement Learning". In: *Cognitive Systems and Signal Processing*. 2019, pp. 127–137.

[17] L. Zhao, M. Roh, and S. J. Lee. "Control Method for Path Following and Collision Avoidance of Autonomous Ship Based on Deep Reinforcement Learning". In: *Journal of Marine Science and Technology* 27.4 (2019), pp. 293–310.

[18] L. P. Perera. "Deep Learning Toward Autonomous Ship Navigation and Possible COLREGs Failures". In: *Journal of Offshore Mechanics and Arctic Engineering* 142 (2020).

[19] *Transas Information on Wärtsilä's Website*. `https://www.wartsila.com/transas`. Retrieved 14 April 2020. 2020.

[20] *Simulation and Training Solutions PDF*. https://www.wartsila.com/docs/default-source/product-files/optimise/simulation-and-training/simulation-and-training-solutions-brochure.pdf. Retrieved 14 April 2020. 2020.

[21] E. S. Raymond. *AIVDM/AIVDO Protocol Decoding*. Version 1.54. 2019.

[22] *AIVDM/AIVDO Protocol Specification*. https://gpsd.gitlab.io/gpsd/AIVDM.html#_types_1_2_and_3_position_report_class_a. Accessed 2020. 2020.

[23] R. S. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 2018.

Biography

Tatjana Cucić was born on January 31, 1993, in Zrenjanin, Serbia. She completed her secondary education at the Gymnasium "Srednja škola" in Novi Bečej. In 2018, she graduated from the Faculty of Sciences, University of Novi Sad, where she obtained a degree in Theoretical Mathematics and the title of Bachelor Professor of Mathematics.

During her studies, she developed a strong interest in the application of mathematical methods and data analysis. In 2020, she participated in the Erasmus exchange program at the University of Turku, Finland. During this period, she worked on a research project at Åbo Akademi University focused on developing a reinforcement learning model for autonomous ship navigation.

Since March 2021, she has been employed at Levi9 Technology Services in Novi Sad, where she works as a Data Analyst/Engineer, focusing on data analysis and data-driven solutions.

She is particularly interested in data science, machine learning, and the practical application of mathematical models. She is known for being collaborative, communicative, and strongly team-oriented in her professional and academic work.

Chapter 8

Appendix A

8.1 GitHub Code

https://github.com/PurpleFlyingSheep/autonomous-ships

Chapter 9

Appendix B

9.1 AIS parameter explanations

Parameter	**Units**	**Description**
Message ID	1–3	Message type (1, 2, 3 are Position Report Class A).
Repeat indicator	0–3	Number of times a message was rebroadcast (3 = do not repeat).
User ID (MMSI)	9 decimal digits	Ship identifier according to Maritime Mobile Service Identity.
Navigational status	0–15	See Table 2.
Rate of turn (ROTAIS)	-127 to 128	0 = not turning 1–126 = turning right up to 708 degrees per minute -1—126 = turning left up to 708 degrees per minute 127 = turning right more than 5 degrees per 30 seconds -127 = turning left more than 5 degrees per 30 seconds 128 = no turn information available
Speed Over Ground (SOG)	0 to 102.3 knots	Speed over ground. 102.3 indicates speed not available, while 102.2 indicates 102.2 knots or higher.

Parameter	**Units**	**Description**
Position accuracy	0 or 1	0 = accuracy error greater than 10 meters 1 = accuracy error less than 10 meters
Longitude	-180 to 181 (degrees)	Distance measured east or west from the prime meridian. East = positive West = negative 181 = not available
Latitude	-90 to 91 (degrees)	Geographic coordinate specifying north–south position on Earth. North = positive South = negative 91 = not available
Course Over Ground (COG)	0 to 360 (degrees)	Course relative to true North (direction of ship movement). 360 = not available
True Heading	0 to 359 (degrees)	Direction the ship is pointing. 511 = not available
Time Stamp	0–59 (seconds)	Second of UTC timestamp.
Special maneuver indicator	0–2	0 = not available 1 = no special maneuver 2 = special maneuver
Spare	–	Not used
RAIM flag	0 or 1	0 = not used 1 = used
Sync state	0–3	0 = UTC direct 1 = UTC indirect 2 = synchronized to base station 3 = synchronized to another station
Slot Time-out	0–7	Specifies frames remaining until slot change 0 = last transmission in this slot 1–7 = frames remaining
UTC Hour	0–59	UTC hour
UTC Minute	0–59	UTC minute

Value	Description
0	Under way using engine
1	At anchor
2	Not under command
3	Restricted maneuverability
4	Constrained by her draught
5	Moored
6	Aground
7	Engaged in fishing
8	Under way sailing
9	Reserved for future amendment of Navigational Status for HSE
10	Reserved for future amendment of Navigational Status for WIG
11	Reserved for future use
12	Reserved for future use
13	Reserved for future use
14	AIS-SART is active
15	Not defined (default)

Chapter 10

Appendix C

10.1 NMEA documentation

10.1.1 NMEA (0183) Parameters

NMEA combines various marine devices such as GPS, gyrocompass, autopilot, sonars, etc. It is defined and controlled by the National Marine Electronics Association. The first two letters of an NMEA message indicate what type of device is used, while the next three refer to the type of standard sentence. This document contains interpretation of parameters found in certain types of NMEA sentences.

1. GP – Messages that start with "GP" refer to data collected from a GPS

Table 10.1: 1.1 ZDA – Time & Date – UTC, day, month, year and local time zone

Parameter	Units	Description
UTC Hour	0 to 23	UTC Hour
UTC Minute	0 to 59	(can be chosen in the simulator)
UTC Second	0 to 59	UTC Minute
Day	01 to 31	UTC Second
Month	01 to 12	Day
Year	4 digits	Month
Local zone description (Time zone hours)	00 to +-13	Year
Local zone minutes description (Time zone minutes)	00 to 59	Time zone (hours)

Table 10.2: 1.2 GLL – Geographic Position – Latitude/Longitude

Parameter	Units	Description
Latitude	-90 to 90 (degrees)	Geographic coordinate that specifies the North–South position of a point on the Earth's surface. North = Positive, South = Negative
Longitude	-180 to 180 (degrees)	Distance measured in degrees east or west from the prime meridian. East = Positive, West = Negative
Status	A or V	A - Data Valid, V - Data Invalid
FAA mode indicator	A, D, E, M, S, N	See FAA mode indicator table

Value	Description
A	Autonomous mode
D	Differential mode
E	Estimated (dead reckoning) mode
M	Manual input mode
S	Simulator mode
N	Data not valid

Table 10.3: 1.3 GGA – Global Positioning System Fix Data

Parameter	Units	Description
GPS Quality Indicator	0 to 8	0 – fix not available, 1 – GPS fix, 2 – Differential GPS fix, 3 – PPS fix, 4 – Real Time Kinematic, 5 – Float RTK, 6 – Estimated (dead reckoning), 7 – Manual input mode, 8 – Simulation mode
Number of satellites in use	00 to 12	Number of satellites in use for GPS
Horizontal Dilution of Precision	meters	Error in horizontal position
Antenna Altitude above/below mean-sea-level	meters	GPS antenna altitude above/below mean sea level
Units of antenna altitude	M = meters	Units of GPS antenna altitude
Geoidal separation	meters	Difference between WGS-84 ellipsoid and mean sea level
Units of geoidal separation		Units of geoidal separation
Age of differential GPS data	seconds	Time since last update
Differential reference station ID	0000-1023	DGPS reference station ID

Table 10.4: 1.4 VTG – Track made good and Ground speed

Parameter	Units	Description
Course over ground	0 to 360 (degrees)	Current real course (where the ship is actually going)
T = True	T	True heading is used
Course over ground, Magnetic	0 to 360 (degrees)	True heading (magnetic compass used)
M = Magnetic	M	Magnetic compass is used
Speed over ground, knots	knots	Speed relative to the ground
N = knots	N	N = knots
Speed over ground, km/h	km/h	Speed relative to the ground
K = Kilometers per hour	K	Kilometers per hour

Table 10.5: 1.5 RMC – Recommended Minimum Navigation Information

Parameter	Units	Description
Track made good	degrees	Course a ship must take to get back on track

2. VD – Messages that start with "VD" refer to data collected from the Velocity Sensor.

Table 10.6: 2.1 VHW – Water speed and heading

Parameter	Units	Description
Heading degrees, True	degrees	Direction the ship is pointing
T = True	T	True heading
Heading degrees, Magnetic	degrees	Direction the ship is pointing (magnetic compass)
M = Magnetic	M	Magnetic compass
Speed of vessel relative to water	knots	Ship speed relative to water
N = Knots	N	Knots
Speed of vessel relative to water	km/h	Ship speed relative to water
K = Kilometers	K	Kilometers per hour

Chapter 11

Appendix D

11.1 AIS and NMEA comparison

11.1.1 AIS and NMEA compared

The following conclusions were made by observing a small sample of files. We believe that the simulations from where the logs originate were not done for any real purpose other than logging data. Moreover, there could be more differences between the files as our observed sample was small.

-AIS

- Data from own ship and other ships
- Time is logged every two seconds (some errors/missing)
- COG and SOG are the same
- ROT float
- Longitude/Latitude different
- Heading integer

-NMEA

- Data from own ship only

- Time is logged each second

- COG and SOG are the same

- ROT integer

- Longitude/Latitude different

- Heading float

It is best to use ROT logs from the AIS, but Heading from the NMEA, because these values are floats. Latitude/Longitude left to check accuracy. Only actions and parameters that are not in AIS or NMEA should be taken from UHI due to the fact that UHI data is less consistent.