Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Marine Technology

Karianne Skudal Tjøm

Guidance and Decision Making using Machine Learning for Small Autonomous Ships

Master's thesis in Marine Technology

Supervisor: Asgeir J. Sørensen (Main), Tobias V. R. Torben (Co) and Børge Rokseth (Co)

June 2020

NTNU
Norwegian University of
Science and Technology

Guidance and Decision Making using Machine Learning for Small Autonomous Ships

Author:
Karianne Skudal TJØM

Supervisor:
Asgeir J. SØRENSEN

Co-supervisors:
Tobias V. R. TORBEN
Børge ROKSETH

NTNU

Department of Marine Technology
Norwegian University of Science and Technology

June 8, 2020

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

MASTER THESIS IN MARINE CYBERNETICS

SPRING 2020

FOR

STUD. TECHN. KARIANNE SKUDAL TJØM

Guidance and decision making using machine learning for small autonomous ships

Work description (short description)
Autonomous ships are gaining increased attention in the maritime industry. They can potentially reduce costs, environmental impact and improve safety at sea. The levels of autonomy may be described in steps where both manned and un-manned ships supported with remote control are considered. The need for autonomy is increasing when the operator is limited in ability to interact with the ship control systems. In order to take advantage of the benefit of autonomy, proper understanding of risk and risk management is required. In particular, decision making taking place on the ship itself embedded in the autonomy control system must be addressed.

The scope of this thesis is to evaluate two machine learning methods, to be used in supervisory control, where the algorithm decides the waypoints. In order to apply machine learning in this matter, a new guidance system has to be developed to fit the demands of flexible waypoint navigation. Mainly the risk associated with navigation for the small passenger ferry MilliAmpère is considered. The corresponding risk factors can be collision and grounding subject to varying environmental and operational conditions.

Scope of work

1. Provide an overview of vessel modelling and control for marine surface vessels
2. Review necessary literature within machine learning and study two algorithms applicable for safe navigation of ships
3. Develop a flexible guidance system for waypoint navigation, and conduct a simulation study to verify its performance
4. Evaluate the results of the guidance system by conducting a full-scale test
5. Propose a method for using machine learning to generate waypoints
6. Train and test the chosen machine learning model through simulations

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 10th of June.

Advisors: Postdoc Børge Rokseth
PhD Tobias Valentin Rye Torben

Professor Asgeir J. Sørensen
Supervisor

Abstract

Risk-based decision making is important for autonomous ships to avoid collisions. For highly autonomous ships, these decisions are taken by the system and this thesis investigate the use of machine learning as the decision maker. More specifically, deep reinforcement learning is studied to set the waypoints for the autonomous ship. For this to be doable, the vessel needs to have a customised guidance, navigation and control system. Therefore this thesis presents a novel guidance system, in addition to investigating the reliability and use of deep reinforcement learning as decision maker.

The first contribution in this thesis is a novel guidance system. For this case an over-actuated vessel with a DP system is considered, and hence the guidance system should generate position, velocity and acceleration references in 3 DOFs. The proposed method combines LOS and reference filters to ensure the system posses four key properties: The first is that the desired velocity is maintained through the waypoints, and can vary. The second is that the waypoints are needed only once the previous is reached. The third property is that it is compatible with a DP system, and the last is that the heading is not included in the waypoint as LOS calculates the desired heading. Through simulations it was found that the guidance system gives the desired behaviour and that the key properties are obtained. However, with the physical limitations of the ship the vessel turns and change velocity quite slow with the chosen reference filter. It is also seen that with a varying velocity, both the look-ahead distance and the acceptance value for when a waypoint is reached should be adapted. As a final test of the guidance system, a sea trial were performed, where the goal was to verify the results in a real-life situation, and investigate how the system copes with environmental forces. The sea trial showed promising results, and proves that the guidance system should be further developed to ensure it is as flexible as desired.

The second contribution is an evaluation and future work for a method to use machine learning to generate waypoints. DDQN and Q-learning were implemented and tested in two different environments. Q-learning provided good results for the simple environment, while the number of states where to many for it to converge to a stable behaviour in the more advanced environment. DDQN did not perform as desired for any of the environments for it to be trustworthy, and measures to improve it should be considered. Two suggestions are to use the method as a part of a larger system with online risk management, or try approach with a different algorithm and training regime. Hence, more research is needed to conclude if the two machine learning methods are beneficial for autonomous ships.

Sammendrag

Risikobasert beslutningstaking er viktig for at autonome skip skal unngå kollisjoner. For skip med høy grad av autonomi blir disse beslutningene tatt av systemet, og denne masteroppgaven undersøker bruken av maskinlæring som beslutningstager. Mer presist benyttes dyp forsterket læring for å sette veipunktene for det autonome skipet. For at dette skal være gjennomførbart trenger skipet et tilpasset veiednings-, navigasjons- og kontrollsystem. Derfor presenterer denne oppgaven et nytt veiledningssystem og undersøker påliteligheten ved bruk av dyp forsterket læring som beslutningstager.

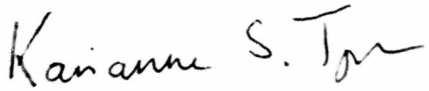
Det første bidraget i denne oppgaven er et nytt veiledningssystem. Fokuset ligger på overaktuerte skip med DP system, og derfor må veiledningssystemet generere referanser for posisjon, hastighet og akselerasjon i tre frihetsgrader. Den nye metoden kombinerer LOS og referansefiltre for å oppnå fire egenskaper: den første er å vedlikeholde ønsket hastighet gjennom veipunktene, og at ønsket hastighet kan variere. Den andre er at neste veipunkt ikke trengs før det forrige er nådd. Den tredje egenskapen er at systemet er kompatibelt med et DP system og den siste er at ønsket heading ikke trenger å presiseres av operator, men heller bestemmes av LOS. Gjennom simuleringer så man av veiledningssystemet ga ønsket oppførsel og oppfylte de fire egenskapene. Det ble også tydelig at de fysiske begrensningene til skipet førte til trege svinger og endring av hastighet med det valgte referansefilteret. I tillegg fører den varierende hastigheten til at både look-ahead lengden og akseptanseverdien bør variere. Til slutt ble det gjennomført en fullskalatesting for å verifisere resultatene i en reell situasjon og undersøke hvordan systemet taklet å bli påvirket av miljøkrefter. Testingen ga lovende resultater og understreker at veiledningssystemet bør bli videre utviklet så det blir så fleksibelt som ønskelig.

Det andre bidraget er å foreslå og evaluere en metode som benytter maskinlæring for å generere veipunkter. DDQN og Q-læring ble implementert og testet i to ulike miljøer. Q-læring ga gode resultater i det enkle miljøet, men klarte ikke å konvergere til en stabil oppførsel i det avanserte miljøet grunnet for mange mulige tilstander. DDQN oppnådde ikke tilstrekkelig godt resultat i noen av miljøene til at man kan stole på algoritmen uten at den først blir forbedret. To forslag til endringer er å enten benytte metoden som en del av et større system med online risikostyring, eller prøve med andre algoritmer og nye treningsregimer. Det trengs mer forskning for å konkludere om de foreslåtte metodene for maskinlæring er egnet for autonome skip.

Preface

This master thesis concludes my studies at The Norwegian University of Science and Technology (NTNU) in Marine Technology, with specialisation in Marine Cybernetics. It is a continuation of the project thesis conducted fall 2019. Chapter 2 and Section 3.2-3.4.1 are based on preliminary work done in project thesis, but is included here to present a complete, stand-alone thesis. All content is written by me, except where stated otherwise.

Since there are many interesting topics, and one (hopefully) only writes one master thesis during a lifetime, this thesis combines two topics: Guidance and Machine Learning. The thesis is structured as a monologue, with one of the main contributions resulting in a paper which is attached in Appendix D.



Acknowledgements

Writing a master thesis while a pandemic breaks out is not without complications. Meetings and scheduled lab were suddenly difficult, which required new working methods. I am therefore grateful for the advise and motivation received, as well as the shared company this spring. Firstly, I would like to thank my main supervisor Asgeir J. Sørensen for having faith in me and the project throughout the process. In addition i would like to give my thanks to my two co-supervisors Tobias V. R. Torben and Børge Rokseth for always being available for discussions and providing new insight. You have all been more engaged in and enthusiastic about the project than I could have ever dreamed of.

List of Abbreviations

API	Application Programming Interaface
CNN	Convolutional Neural Network
COLREGs	International Regulations for Preventing Collisions at Sea COLREGs
DQN	Deep Q-Network
DDQN	Double Deep Q-Network
DRQN	Deep Recurrent Q-learning
DOF	Degree of Freedom
DP	Dynamic Positioning
GNSS	Global Navigation Satellite Systems
GRU	Gated Recurrent Unit
HIL	Hardware-In-the-Loop
HMI	Human Machine Interface
ILOS	Integral Line-of-Sight
IMO	International Maritime Industry
IMU	Inertial Measurement Units
JONSWAP	Joint North Sea Wave Project
LOA	Levels of Autonomy
LOS	Line-of-Sight
LSTM	Long-Short-Term Memory
MASS	Autonomous Surface Ships
NED	North-East-Down
NIST	National Institute of Standards and Technology
NN	Neural Network
ReLu	Rectified Linear Unit
RNN	Recurrent Neural Network
ROS	Robot Operating System
SNAME	Society of Naval Architects and Marine Engineers

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The goal with this thesis is to investigate a new method to apply machine learning for vessel guidance and navigation, and evaluate the risk factors it is subject to. The approach to be inspected will use machine learning as a decision maker for an autonomous ship, which is enabled by a customised guidance, navigation and control system.

For autonomous ships to be an integrated part of the shipping industry, or the public transport in a city, the systems have to be trustworthy. Both autonomy and machine learning bring different risk factors. Therefore this chapter will give an introduction to the development in autonomous ships recent years, what risk factors they are subject to, some earlier work on using machine learning for navigational purposes and describe the scope of this thesis.

1.1 Background and Motivation

Autonomous ships have been mentioned as the next step, and an important piece of the puzzle for maritime industry moving forward for some years now. As an example the International Maritime Industry (IMO) decided in 2018 to investigate how Maritime Autonomous Surface Ships (MASS), can be assessed using IMO instruments (IMO, 2018). Factors like safety, security and how environmental friendly the operation is will be evaluated. Getting a framework for securing the operations and hence increase the industry and the public thrust in autonomous ships is important.

NIST (2008) has defined autonomy as an unmanned vehicles ability to achieve goals set by a human operator using its own ability to plan the mission, sense and analyse the surroundings to make the correct decisions. ISO (2019) has another definition stating that at least on process has to be executed automatically without a human operator involved during a whole operation or part of it. Hence, vessels can be autonomous at many different levels. This is referred to as level of autonomy (LOA), where a high level indicates a low level of human dependency. In order to achieve high LOA, the vehicle has to be able to do this as well as handle exceptions, and make quick decisions in critical situations.

Since these are very strict requirements, not all systems classify as highly autonomous.

In order to separate different systems there are developed different taxonomies on how to classify the systems. They range from coarse with few levels, to very detailed and many levels. Rødseth & Nordahl (2017) and Utne et al. (2017) both use four different levels, while Insaurralde & Lane (2012) have defined ten different levels. Utne et al. (2017) also includes risk aspects according to the respective levels. The below definitions are based on Utne et al. (2017) and Rødseth & Nordahl (2017).

- **Remote Control** - The human is in direct control of the ship, and is presented all systems states, sensor data, and environmental conditions. Based on the received information the human takes a decision. The system may give decision support. The risk lies at the operator and whether he is experienced enough and how well the procedures are.
- **Management by Consent** - The system is more advanced and it is expected to maintain position by itself using dynamic positioning (DP). It continuously makes recommendations to the human operator, but does not execute actions unless specifically instructed to. At this level the human machine interface (HMI) is an important tool for the operator to handle risk. Also the system has to be secure and equipped with anti-collision sensors, since the system can be delegated some tasks.
- **Management by Exception** - The vessel operates by itself in most situations, and has predefined missions that it executes. A human operator can at any time override and change the decision done by the system. The system asks for human intervention when faced with a situation that cannot be solved within its defined work environment. In this case the need of a capable and competent operator is reduced, while the system demands are increasing. There is also a risk of the human operator getting bored, when the system mainly handles itself, making the operator unable to make a correct decision in a difficult situation.
- **Highly Autonomous** - The system can plan and re-plan missions by itself and executes all mission related activities automatically. This indicates no use for human operator, but the human may still be informed of the progress and situation. The risk is now only associated with the system. Therefore the system is required to be robust, intelligent and with an online risk management system that can be trusted.

The range is from remote control where the system automatically does what the human operator wants, to the system recommending beneficial actions, and all the way to highly autonomous where the system itself takes all the decisions and only informs the operator. Hence, a highly autonomous ship has to gather, and efficiently use large amount of data compared to a manually operated vessel. An alternative approach to defining the autonomy of a vessel is done by Rødseth (2018). Instead of focusing on different levels of human Independence, it suggest to define LOA based on characteristic factors.

There are many reasons motivating a more autonomous marine industry. Increased safety by removing human errors and increased earnings by reducing the manning and accommodation are two motives. The lack of crew on-board that wants to come home can be exploited by operating at a lower speed on shipping routes which will reduce emissions. New and better technology can open a door to missions previously considered impossible when executed by an operator, and the machine can work without breaks around the clock unlike personnel. In a larger perspective the vessel movements may be even more predictable, considering that a machine will make

consistent choices.

Some of these gains are further into the future than others. Even though the IMO is on board, more research is needed before the autonomous ships can overtake the maritime industry. To test methods in a safe environment, ferries is a good starting point. This is because they require navigation and often operate in mild environmental conditions. There are research going on both in the industry and in academia.

1.2 Risk Factors

There are many risk factors associated with autonomous ships. In order to discuss relevant risk factors and their impact, the concept risk has to be defined. Kaplan & Garrick (1981) first define risk as something that is uncertain, and has a negative impact. A risk in this thesis, will always refer to undesired events. After this distinction, one must think what the possible undesired events are. How high the chance that the scenario will occur is, and what consequences the scenario has when it happens. This is summarised as the triplet in Equation (1.1).

$$\langle s_i,\ p_i,\ c_i \rangle \tag{1.1}$$

Here s_i is the scenario or event, p_i is the probability that this scenario will occur, and c_i is the consequence if this scenario happens. The index i refer to the relevant events. Risk is then defined as the complete set of these triplets as in Equation (1.2).

$$R = \{\langle s_i,\ p_i,\ c_i \rangle\}, \quad i = 1,\ 2,\ 3, ...,\ N \tag{1.2}$$

Some scenarios with high consequence could be collisions with other vessels or grounding. Another scenario could be lack of power for an electric vessel which could lead to a collision.

If all scenarios are well known and can be described, R is well defined. However, it is easier to plan for known risk, while black swans are difficult to handle. A black swan is an scenario that is impossible to predict with existing knowledge, and has huge consequences when they occur (Taleb, 2007). Aven (2013) characterise both unknown scenarios and known scenarios not having the expected behaviour as black swans. It is difficult to prepare an autonomous ship for unknown events, as they are in fact unknown. This make them dangerous, but the known scenarios that have a higher consequence or probability than expected can be equally challenging. Consider a vessel in open sea, approaching an iceberg. If the vessel then willingly navigates into the iceberg thinking it will do little resistance, but in fact it is much larger than expected. This could lead to hull breaches, which is a severe consequence.

It is an easy mistake to assume that an event is characterised correctly. To include the uncertainty that is present in risk assessments, another risk model is proposed (Aven, 2012). It is expressed in Equation (1.3).

$$\langle s_i,\ c_i,\ q_i \rangle \mid k \tag{1.3}$$

s_i and c_i are as explained earlier, and q_i is the uncertainty factor. k_i represent the knowledge that the values are based on. This definition combines the existing knowledge, surprises that may arise and probability based thinking.

What risk factors are relevant, depend upon what mission is executed. There are many factors that count in when assessing the complexity of a mission, and the environment it operates in. Depending on how difficult the operation is to conduct, appropriate actions have to be in place in order to achieve a satisfying risk level. There are mainly three factors to take into account when assessing overall complexity: mission complexity, human independence, and environmental difficulty (NIST, 2008).

Human independence is thoroughly discussed trough the levels of autonomy, and will therefore not be further discussed here.

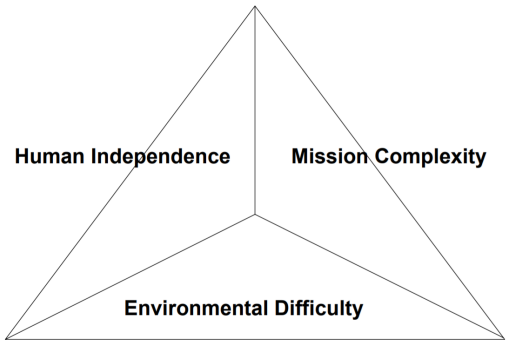


Figure 1.1: Characteristics of an autonomous system, source: (Huang, 2006)

Environmental complexity can vary strongly from location to location as well as day to day. If the area the mission takes place in is well known and explored, the environmental complexity decreases. Oppositely the complexity increases if the mission takes place in an unknown area, or possibly under water where the sight and communication range are bad. The environmental complexity also varies with the environmental conditions as it is more difficult to operate in heavy weather and strong currents. Other factors that increase environmental complexity can be the number of reefs in the area and how dense the traffic is as there is a higher probability for collision.

It is much easier for an autonomous system to execute a simple straight forward mission, than one with many subtasks where the requirements to the system are many (ISO, 2019). The Operational Design Domain decides the mission complexity, which is defined as:

"The specific conditions and scenarios under which a given autonomous ship system is designed to function, including, but not limited to, its different operational modes as well as all anticipated failures." in (ISO, 2019).

To take an autonomous ferry as an example, the conditions it operates in is the path between its docks. It has different operational modes as lift-off, transit and berthing. There are also expected scenarios where it has to avoid colliding with stationary and dynamic obstacles such as other vessels, kayaks or reefs . If something unexpected happens and it does not know how to handle the situation, the ferry has to go to a safe condition.

In addition to the risk posed by an autonomous ship, increased risk may be added by using black-box machine learning methods as the decision maker. Figure 1.2 shows some relevant scenarios for an autonomous ship. The grey are associated with the surroundings of the vessel, the green are relevant for the DP system, and the yellow factors are due to the machine learning algorithm.

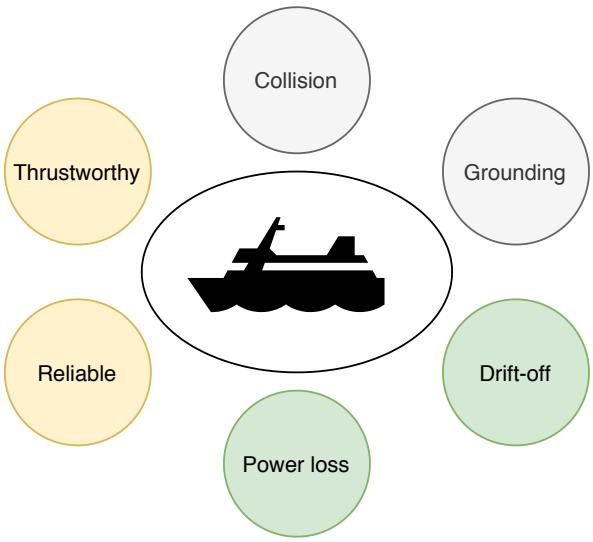


Figure 1.2: Relevant scenarios for autonomous ships

Machine learning can be used at both a high and low control levels, as sensor interpreter and decision maker. The possibilities are endless, and not fully explored yet. A high level decision maker depend upon a customised vessel control plant, while low level control can be adjusted to the vessel model, characteristics and level of autonomy. However, no matter how it is done the system is a liability that has to be verified for its use.

1.3 Previous Work

There are different methods to both control and set up decision making for autonomous ships.

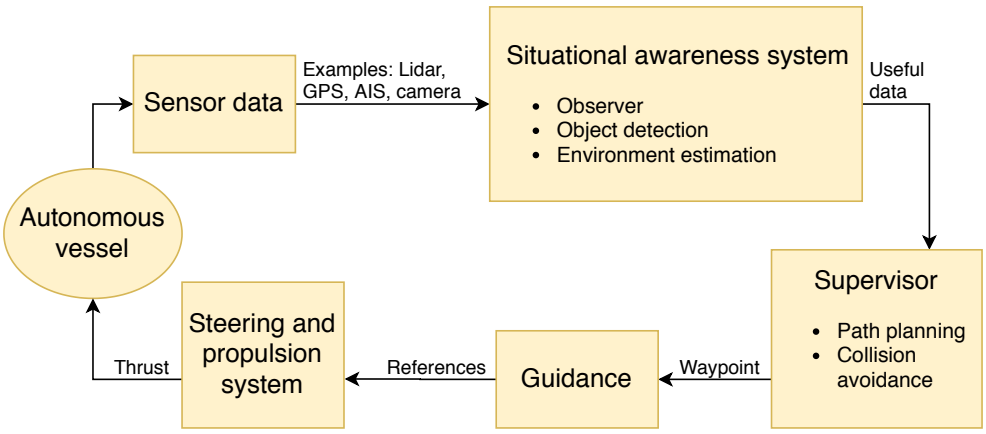


Figure 1.3: Block diagram of autonomous system

Figure 1.3 shows how a block diagram of how an autonomous system may look. In addition to type of sensor data used, the supervisor function is an important part of

the autonomous system. The decision making is done in the supervisor. Since the supervisors role often is to lead the vessel towards a desired location without colliding, it is connected to path planning and collision avoidance.

Figure 1.4: Autonomous ferry, MilliAmpère from NTNU

At NTNU a small autonomous ferry called "MilliAmpère" serve as a test vessel for autonomous ships (NTNU, 2019). Previous work have been done on implementing a collision avoidance system on MilliAmpère. Thyri (2019) implemented a collision avoidance system which acts different depending on if the path can be executed as planned or if it necessary to change plan due to interference by dynamic obstacles. For an autonomous ship to successfully navigate and avoid other vessels, it is necessary to be in compliance with Convention on the International Regulations for Preventing Collisions at Sea (COLREGs). COLREGs state how vessels should behave when an encounter occurs (IMO, 1972). When the vessel is autonomous, the responsibility of following the conventions are transferred from the captain, to the system. Therefore research is also done on collision avoidance systems compliant to COLREGs. Kufoalor et al. (2019), and Johansen et al. (2016) showed promising results with a model predictive control approach.

However, in this thesis a new approach is investigated where the supervisor is a machine learning agent. We will study how reinforcement learning methods will work and the main idea is that the algorithm learns the desired behaviour by itself. The focus will be to implement and test if the method works on simple guidance and navigation cases. Implementation of COLREGs will be disregarded here and is subject for further work.

Earlier work where reinforcement learning is applied to vehicle navigation have often controlled the vessel directly through the rudder angle, used an image of the area, or processed lidar sensor data itself to decide the action (Zhang et al., 2019), (Khan et al., 2017), (Lei et al., 2018), (Le Pham et al., 2017), (Figueiredo & Rejaili, 2018). Here it is proposed to use the algorithm to set the waypoints based on processed sensor data, leaving the low level control to the DP system. This require a functioning situational awareness system that processes the sensor data, and feeds the machine learning algorithm with relevant information in a predefined format.

Both Q-learning and Double Deep Q-learning (DDQN) will be applied as machine learning algorithms. Q-learning is used mostly to establish a baseline and investigate how it copes with a large amount of data.

1.4 Research Questions

The following research questions are investigated in this thesis:

- How to develop a guidance system with configurable velocity for waypoint tracking, preparing for a machine learning decision maker?
- How to use machine learning to define waypoints as input for guidance and navigational systems?
- Is the proposed application of machine learning reliable for autonomous ships?

1.5 Main Contributions

The aim of this thesis is to investigate the use of a different approach with machine learning for safe guidance and navigation, where machine learning is used to generate waypoints. The main contributions in this thesis are the guidance system developed to make this approach feasible, and concrete suggestions for further work to make waypoint generation by machine learning reliable. A paper has been produced on the guidance system, which is attached in Appendix D.

1.6 Outline

This report has two main contributions which are presented in separate chapters for a comprehensive presentation of the two parts. Chapter 2 and 3 are background information. Chapter 4 presents the method and results of the flexible guidance system, and then the method and results for using machine learning as decision maker are described in Chapter 5. This master's thesis is a continuation of a project report written fall 2019.

The chapters in this report are structured as follows:

Chapter 2 gives a brief introduction to vessel model and control.

Chapter 3 presents basic machine learning theory as well as deep learning and some selected machine learning methods.

Chapter 4 describes the development of a new flexible guidance system for waypoint-following. Results from simulation and full-scale testing is discussed.

Chapter 5 explain how to use machine learning as decision maker for an autonomous ship, and investigates how the method performs.

Chapter 6 concludes and suggests further work for both contributions.

Appendix A contains further description of the research approach used for developing both the guidance system described in Chapter 4 and the implementation and testing of machine learning as decision maker in Chapter 5.

Appendix B describes the experimental test setup for the full-scale testing of the guidance system, and includes additional results.

Appendix C shows an alternative reward function for the the machine learning algorithm in Chapter 5.

Appendix D contains the appended paper on the flexible guidance system for waypoint following.

Chapter 2

Background on Vessel Model and Control

To control or interact with an existing system, there is a need to understand how it functions. This section will cover relevant notation and specifications of how vessels are mathematically modelled. In addition to that, the environment it operates in has to be modelled as well as the sensor systems in order to do realistic simulations. This is also described in this section. This chapter has been co-authored with Toni Klausen.

2.1 Mathematical Modelling

This section presents how to model a vessel mathematically. Figure 2.1 presents a simplified structure of a DP system based on Sørensen (2011) to get an overview. From the vessel, the system receives measurements, which are processed to ensure proper data quality before the vessel observer use the data to do wave filtering and estimate states.

The operator sets the waypoints, and the guidance system generates trajectories from one waypoint to another.The controller commands the desired thrust in the available DOFs, and the thrust allocation decides the setpoints for thrusters to achieve the desired thrust.

2.1.1 Frames of Reference

To be able to describe a vessels movements accurately, it is necessary to use two different frames of reference. The inertial North-East-Down (NED) 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, as described in Table 2.1. 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, it translates and rotates with the vessel. The movements and forces in body-frame are visualised in Figure 2.2.

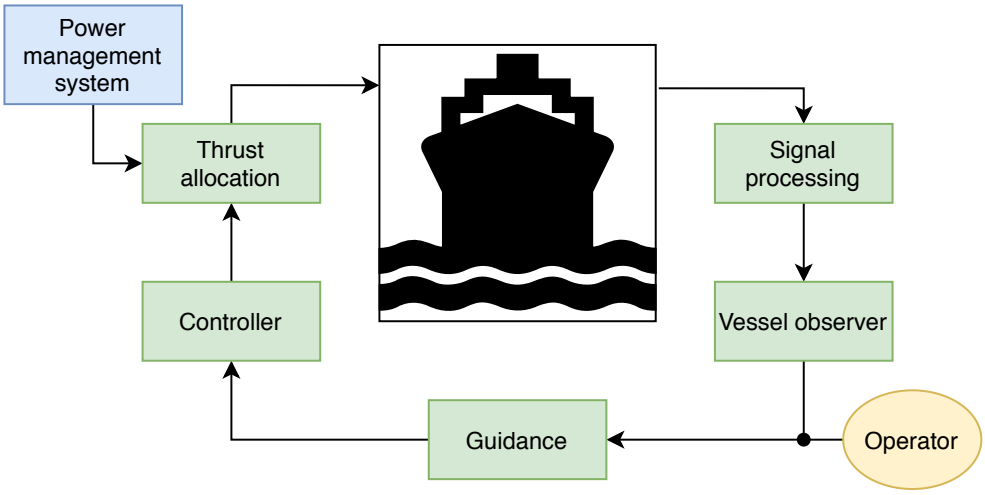


Figure 2.1: Simplified representation of a DP system

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

Table 2.1: Naming conventions adopted from the Society of Naval Architects and Marine Engineers (SNAME, 1950).

The NED-frame and body-frame are connected. Forces cannot be shifted between but expressed in terms of the different reference frames using a rotation matrix.

In this thesis, variables in terms of body-frame may be super-scripted with b, while the superscript n is used for variables in NED-frame.

2.1.2 Vessel Kinetics and Kinematics

The general 6-DOF equations of motion for marine craft are given in matrix-vector form as

$$\dot{\boldsymbol{\eta}} = \mathbf{R}(\boldsymbol{\eta})\boldsymbol{\nu} \tag{2.1}$$

$$\underbrace{\mathbf{M}_{RB}\dot{\boldsymbol{\nu}} + \mathbf{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu}}_{\text{rigid-body forces}} + \underbrace{\mathbf{M}_A\dot{\boldsymbol{\nu}}_r + \mathbf{C}_A(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \mathbf{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r}_{\text{hydrodynamic forces}} + \underbrace{\boldsymbol{g}(\boldsymbol{\eta}) + \boldsymbol{g}_0}_{\text{hydrostatic forces}} = \boldsymbol{\tau} + \boldsymbol{\tau}_{wind} + \boldsymbol{\tau}_{waves} \tag{2.2}$$

where

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

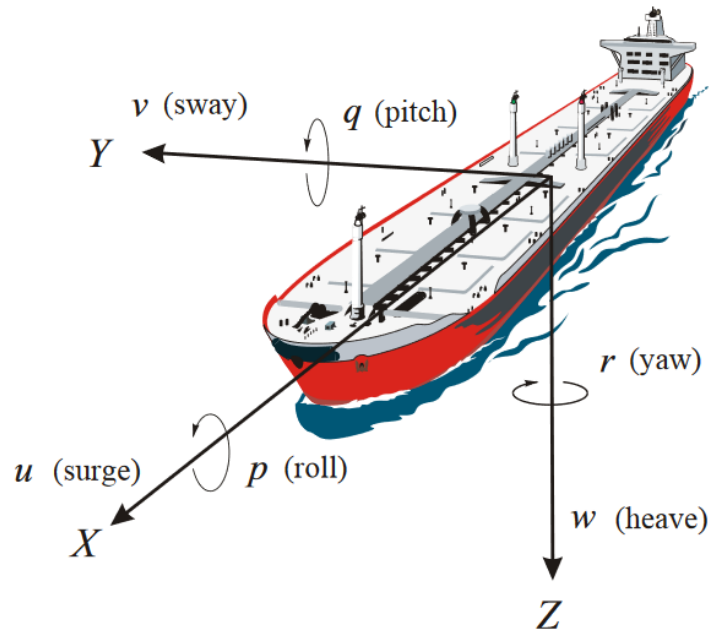


Figure 2.2: Displacement vessel with body-frame axes and velocities, source: (Sørensen, 2018).

- $\boldsymbol{\nu} \in \mathbb{R}^{6\times1}$ is the velocity given in the body-frame.
- $\boldsymbol{\nu}_r \in \mathbb{R}^{6\times1}$ is the relative velocity vector; namely $\boldsymbol{\nu} - \boldsymbol{\nu}_c$ - where $\boldsymbol{\nu}_c$ is the velocity of a (constant and irrotational) current.
- $\mathbf{R}(\eta) \in \mathbb{R}^{6\times6}$ 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).
- $\mathbf{M}_{RB} \in \mathbb{R}^{6\times6}$ is the rigid body inertia matrix.
- $\mathbf{M}_A \in \mathbb{R}^{6\times6}$ is the inertia matrix of the added mass.
- $\mathbf{C}_{RB}(\boldsymbol{\nu}) \in \mathbb{R}^{6\times6}$ is the Coriolis and centripetal matrix of thr rigid body
- $\mathbf{C}_A(\boldsymbol{\nu_r}) \in \mathbb{R}^{6\times6}$ is the Coriolis and centripetal matrix of the added mass.
- $\mathbf{D}(\boldsymbol{\nu_r}) \in \mathbb{R}^{6\times6}$ is the damping matrix.
- $\boldsymbol{\tau} \in \mathbb{R}^{6\times1}$ is the vector of generalised propulsion forces.
- $\boldsymbol{\tau}_{wind} \in \mathbb{R}^{6\times1}$ is the vector of generalised wind force.
- $\boldsymbol{\tau}_{waves} \in \mathbb{R}^{6\times1}$ is the vector of generalised wave-induced forces.
- $\boldsymbol{g}(\boldsymbol{\eta}) + \boldsymbol{g}_0 \in \mathbb{R}^{6\times1}$ are the generalised hydrostatic forces from gravity and buoyancy.

For the derivation of these equations and a more detailed treatment of the variables, the interested reader is referred to Fossen (2011). Generalised forces from the water current are not explicit in Equation (2.2), but they are actually included in the left hand side by the use of the relative velocity vector $\boldsymbol{\nu}_c$.

A comment on the rotation matrix $\mathbf{R}$ 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 $\pm 90°$. For surface vessels this is not a problem since they do not operate in proximity to the singularity. Also, as we shall see later, pitch is often 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 Shepperd (1978) can be used to calculate quaternions from known Euler angles, and a rotation matrix using quaternions can be defined.

2.2 Vessel Control

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 (Fossen, 2011). In this case, only horizontal motion in surge, sway and yaw are considered. This model is often referred to as a manoeuvring model.

2.2.1 Low Frequency Model

Using a low frequency model and wave filtering, the control action will compensate only slowly varying forces. Slowly varying forces could be wind, current or wave drift. The 3-DOF equations of motions is given by Equations (2.3) - (2.5).

$$\dot{\boldsymbol{\eta}} = \mathbf{R}(\psi)\boldsymbol{\nu} \tag{2.3}$$

$\mathbf{R}$ is the rotation matrix given by Equation (2.4) and $\boldsymbol{\eta}$ and $\boldsymbol{\nu}$ consist of components from surge, sway and pitch. ψ is the actual heading of the vessel.

$$\mathbf{R} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

Disregarding 1. order waves the low frequency model becomes:

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{D}\boldsymbol{\nu} + \mathbf{R}^T(\psi)\mathbf{G}\boldsymbol{\eta} = \boldsymbol{\tau} + \boldsymbol{\tau}_{env} + \mathbf{R}^T\boldsymbol{b} \tag{2.5}$$

$$\dot{\boldsymbol{b}} = 0 \tag{2.6}$$

In Equation (2.5) $\mathbf{M}, \mathbf{D}, \mathbf{G} \in \mathbb{R}^{3\times 3}$ with only surge, sway and yaw terms. $\mathbf{M}$ also contains the added mass, which for this linearised model consists of asymptotic values. These values correspond to the linearised drag forces. $\boldsymbol{b}$ is the bias term and the centripetal and coriolis forces are assumed negligible, which is valid for low speed applications.

2.2.2 Dynamic Positioning

To obtain and maintain a vessel's desired position and heading automatically, DP is often used. It is a computer-controlled system, which calculates the necessary force needed in each direction generated by the vessel's actuators.

The low frequency model is applied and a nonlinear PID is used as the control law with reference feedforward (Fossen, 2011) as expressed in Equation (2.7).

$$\boldsymbol{\tau} = -\mathbf{R}^\top(\psi)\mathbf{K_p}(\hat{\boldsymbol{\eta}} - \boldsymbol{\eta}_d) - \mathbf{K_d}(\hat{\boldsymbol{\nu}} - \boldsymbol{\nu}_d) - \mathbf{R}^\top(\psi)\mathbf{K_i}\int(\hat{\boldsymbol{\eta}} - \boldsymbol{\eta}_d)dt + \mathbf{M}\boldsymbol{a}_d + \mathbf{D}\boldsymbol{\nu}_d \quad (2.7)$$

The proportional, derivative and integral gains are given by the non-negative matrices $\mathbf{K_p}, \mathbf{K_d}, \mathbf{K_i} \in \mathbb{R}^{3\times3}$. $\boldsymbol{\eta}_d, \boldsymbol{\nu}_d$ and $\boldsymbol{a}_d$ are the reference signals for position, velocity and acceleration, respectively. The "ˆ" indicates that the states are estimated using an observer as e.g. Kalman-filter or nonlinear passive observer. For details see Fossen (2011) on state estimation for DP.

2.3 Environmental Forces

This section presents models for the generalised forces from the environmental disturbances wave, wind and current. These models are due to (Fossen, 2011), and are needed for realistic simulations in a marine environment.

2.3.1 Waves

There are several methods used for estimating wave loads, depending on what environment the vessel operates in. There are different wave spectra for some areas such as the Joint North Sea Wave Project (JONSWAP) spectrum which is used for the North Sea.

2.3.2 Wind

The wind forces are often divided into three components: mean, slowly-varying and rapidly-varying. As the experienced wind speed for the vessel is coupled with the vessels own speed, relative velocities are used. For a vessel in motion the wind load is defined in Equation (2.8) for 3-DOF in body-frame.

$$\boldsymbol{\tau}_{wind} = \frac{1}{2}\rho_a V_{rw}^2 \begin{bmatrix} C_X(\gamma_{rw})A_{Fw} \\ C_Y(\gamma_{rw})A_{Lw} \\ C_N(\gamma_{rw})A_{Fw}L_{oa} \end{bmatrix} \quad (2.8)$$

Here A_{Fw} and A_{Fw} are frontal and lateral projected areas above the water line, and L_{oa} the total length of the vessel. ρ_a is the air density. C_X, C_Y and C_N are non-dimensional wind coefficients, which can be found model testing. For a symmetrical ship they can be estimated by Equation (2.9) with values for c_x, c_y and c_n found through experiments (Isherwood, 1972).

$$C_X \approx -c_x\cos(\gamma_{rw}), \quad C_Y \approx c_y\sin(\gamma_{rw}), \quad C_N \approx c_n\sin(2\gamma_{rw}) \quad (2.9)$$

The ranges in Equation (2.10) are suggested by Fossen (2011).

$$c_x \in \{0.50, 0.90\}, \qquad c_y \in \{0.70, 0.95\}, \qquad c_n \in \{0.05, 0.20\} \tag{2.10}$$

The relative wind velocity V_{rw} is given by Equation (2.11), where u_{rw} and v_{rw} are the relative wind velocities in respectively surge and sway.

$$V_{rw} = \sqrt{u_{rw}^2 + v_{rw}^2} \tag{2.11}$$

γ_{rw} is the relative wind angle and is found using Equation (2.12).

$$\gamma_{rw} = \text{atan2}(-v_{rw}, -u_{rw}) \tag{2.12}$$

2.3.3 Current

Current arise due to gravity and that the water density and the wind friction have variations. The current forces are included in Equation (2.2) using the relative velocity between the vessel velocity and the current velocity. The current velocity is defined in Equation (2.13) for 3-DOF.

$$\boldsymbol{\nu}_c = [V_c \cos(\psi_c),\ V_c \sin(\psi_c),\ 0]^T \tag{2.13}$$

Where V_c is the absolute value of the current velocity and ψ_c is the direction of the current. To model the absolute value of the current velocity, a first order Gauss-Markov process can be used as in Equation (2.14).

$$\dot{V}_c + \mu V_c = w \tag{2.14}$$

$\mu \geq 0$ is a constant and w is Gaussian white noise. If μ is zero, then the expression models a random walk instead. To avoid the current from being unrealistically high or low, it is often implemented with maximum and min values yielding Equation (2.15).

$$V_{min} \leq V_c(t) \leq V_{max} \tag{2.15}$$

If the current direction varies as well, the direction can be modelled as a Gauss-Markov process too.

2.4 Thrust Allocation

Thrust allocation maps the desired forces and moments to control inputs for the actuators. For a low-speed surface vessel, the forces in heave and moments in roll and pitch are neglected. Consider a symmetric vessel equipped with with two actuators, one in the front and one in the back. Both are placed 1,8 meters away from the geometric center of the vessel giving $L_x = \pm 1.8$. This control allocation method was developed by Torben (2019). Figure 2.3 illustrates the setup.

This configuration gives the thrust load vector in expression (2.16), hence including the forces in surge and sway and moment in yaw.

$$\boldsymbol{\tau} = [X \ \ Y \ \ N]^T \tag{2.16}$$

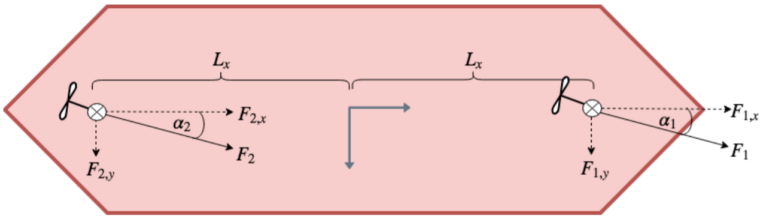


Figure 2.3: Configuration of thrusters for double ended ferries, source: Torben (2019)

The problem that will be evaluated is expression (2.17), where $\mathbf{B}(\boldsymbol{\alpha})$ is the thrust allocation matrix, which depends on the azimuth angles $\boldsymbol{\alpha}$, and $\boldsymbol{u}$ is the control input. The allocation problem is often solved using Pseudo Inverse or Linearised QP techniques. However, it is possible to exploit the symmetry of the vessel (Torben, 2019). That makes it possible to solve the problem fast using nonlinear scalar control allocation.

$$\boldsymbol{\tau} = \mathbf{B}(\boldsymbol{\alpha})\boldsymbol{u} \tag{2.17}$$

Using the extended thrust configuration matrix for rotatable actuators, the thrust configuration matrix no longer depends on azimuth angles. This is done by splitting the two thrusters into two virtual thrusters. The two virtual thrusters are fixed in either surge or sway direction and the extended thrust configuration matrix is given by matrix (2.18).

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ -L_{1,y} & L_{1,x} & -L_{2,y} & L_{2,x} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & L_x & 0 & -L_x \end{bmatrix} \tag{2.18}$$

The values in the final thrust allocation matrix are found stating that $L_{i,y}$ is zero for both of the actuators and using the symmetry of the vessel. Both actuators are L_x away from center in opposite directions. The resulting force and angle of the thrusters are then easily calculated using Equation (2.19).

$$F_i = \sqrt{F_{i,x}^2 + F_{i,y}^2}, \qquad \alpha_i = \arctan\left(\frac{F_{i,y}}{F_{i,x}}\right) \tag{2.19}$$

By visual inspection it can be noticed that matrix (2.18) has rank 3, while there are four forces to be determined. Therefore there is only one free variable to optimise. The exploitation of this fact is done by a reformulation of the optimisation problem, see Torben (2019) for more details.

2.5 Guidance

A guidance system is needed to ensure smooth control actions between waypoints. It computes the desired reference position, velocity and acceleration based on the waypoints. There are two commonly used guidance laws, reference filter and line-of-sight (LOS) which will be described in this section.

2.5.1 Position Reference Filter

Changing the setpoint can lead to abrupt movements, and to avoid this a reference filter is implemented. To ensure a smooth transition, a third order reference model is used. It is obtained by cascading a first-order low-pass filter with a mass-damper-spring system (Fossen, 2011).

$$\boldsymbol{\eta}_d^{(3)} + (2\boldsymbol{\Delta} + \mathbf{I})\boldsymbol{\Omega}\ddot{\boldsymbol{\eta}}_{\boldsymbol{d}} + (2\boldsymbol{\Delta} + \mathbf{I})\boldsymbol{\Omega}^2\dot{\boldsymbol{\eta}}_{\boldsymbol{d}} + \boldsymbol{\Omega}^3\boldsymbol{\eta}_{\boldsymbol{d}} = \boldsymbol{\Omega}^3\boldsymbol{r}^n \tag{2.20}$$

The reference filter is given in Equation (2.20) where $\boldsymbol{\eta_d}$ is the desired position vector, $\boldsymbol{\Delta}$ is the relative damping ratio matrix, $\mathbf{I}$ is the identity matrix, $\boldsymbol{\Omega}$ is the natural frequency matrix, and $\boldsymbol{r}^n$ is the position setpoint vector.

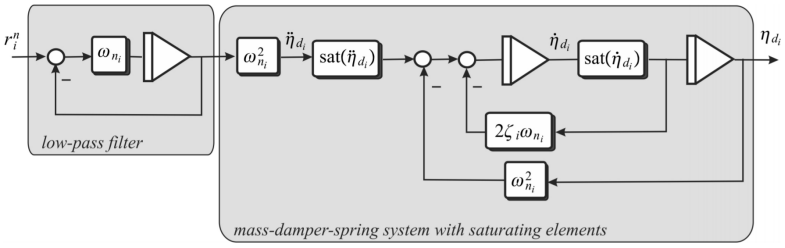


Figure 2.4: Reference model, source: (Fossen, 2011)

Since the reference model is linear, the response may vary depending on what the operating point is. Also the references must have feasible values. To circumvent these problems, saturation elements should be added for velocity and acceleration. The final block diagram of the reference model is given in Figure 2.4.

2.5.2 Velocity Reference Filter

For references that only need to be one time differentiable a velocity reference filter can be used instead. Fossen (2011) describes a velocity reference filter that is constructed as a second order low-pass filter to avoid steps in the velocity and acceleration references and is described by Equation (2.21).

$$\ddot{\boldsymbol{\nu}}_{\boldsymbol{d}} + 2\boldsymbol{\Delta}\boldsymbol{\Omega}\dot{\boldsymbol{\nu}}_{\boldsymbol{d}} + \boldsymbol{\Omega}^2\boldsymbol{\nu}_{\boldsymbol{d}} = \boldsymbol{\Omega}^2\boldsymbol{r}^b \tag{2.21}$$

Here $\boldsymbol{\nu_d}$ is referenced velocity vector, and $\boldsymbol{r}^b$ is the velocity setpoint vector. For the same reasons as for the reference filter, saturation elements should be implemented for this filter.

2.5.3 Line-of-Sight

When a ferry is transitioning between docks it usually follows a path from dock A to dock B consisting of lines from waypoint to waypoint. Between the docks there are no temporal constraints, and the ferry can follow a predefined path. This is known as path following and a LOS guidance law can be used to solve the path following problem (Fossen, 2011). LOS guidance is often used for under-actuated vehicles, since it only demand control action in surge and yaw as demonstrated in Fossen et al. (2003), Borhaug et al. (2008) and Caharija et al. (2012).

LOS guidance navigates towards a point on the path, which is at a constant distance Δ away from the vessel. Δ is referred to as the look-ahead distance. The path $\mathcal{P}$ has to be linear and is given between the starting point WP_k and end point WP_{k+1}. The parameters are defined in Figure 2.5.

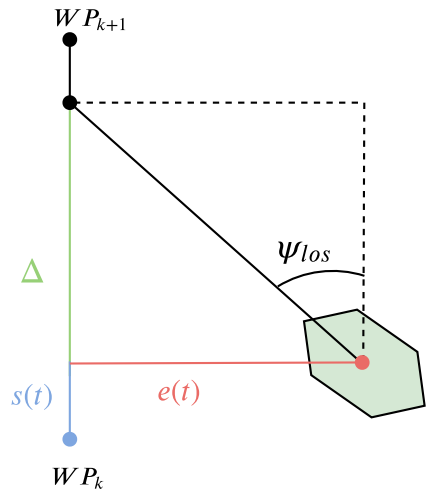


Figure 2.5: Line-of-sight principles

$e(t)$ is the cross-track error, $s(t)$ is the along-track distance and ψ_{los} is the LOS-angle, which often is the desired course angle. The control objective in path following is to make the cross-track error to go to zero as expressed in Equation (2.22).

$$\lim_{t \to +\infty} e(t) = 0 \tag{2.22}$$

The cross-track error is given by Equation (2.23).

$$e(t) = -(x(t) - x_k) \cdot sin(\alpha) + (y(t) - y_k) \cdot cos(\alpha) \tag{2.23}$$

$x(t)$ and $y(t)$ are the current position in north and east direction while x_k and y_k are the previous waypoint. α is the path-tangential angle, and is calculated by Equation (2.24).

$$\alpha = atan2\left(\frac{y_{k+1} - y_k}{x_{k+1} - x_k}\right) \tag{2.24}$$

y_{k+1} and x_{k+1} are the north and east coordinates of the current waypoint in NED-frame while y_k and x_k are the coordinates of the previous waypoint. $atan2(\,)$ is used instead of $arctan(\,)$ to enable waypoints in all quadrants, and ensure that the correct direction is obtained.

The desired course angle for the vessel also depends on the velocity-path relative angle. The vessel has a point on the desired path at a look-ahead distance, which the vessel movement is directed towards due to the velocity-path relative angle. The velocity-path relative angle is given by Equation (2.25).

$$\chi_r = atan2\left(\frac{-e}{\Delta}\right) \tag{2.25}$$

All components of the desired course angle χ_{LOS} is calculated, and is given by Equation 2.26.

$$\chi_{LOS} = \chi_r + \alpha \tag{2.26}$$

In addition to the cross-track error, the along-track distance $s(t)$ is often calculated to measure how far along the course from the previous waypoint the vessel is. In Equation (2.27) the along-track distance is calculated.

$$s(t) = (x(t) - x_k) \cdot cos(\alpha) + (y(t) - y_k) \cdot sin(\alpha) \tag{2.27}$$

For the vessel to be able to follow the path satisfactorily when subject to environmental forces, integral effect is added. LOS with integral effect is known as integral line-of-sight (ILOS) and the guidance law is given by Equation (2.28) and (2.29) (Caharija et al., 2016).

$$\chi_{ILOS} = -\arctan\left(\frac{e + \sigma e_{int}}{\Delta}\right) \tag{2.28}$$

$$\dot{e}_{int} = \frac{\Delta e}{(e + \sigma e_{int})^2 + \Delta^2} \tag{2.29}$$

2.6 Sensor Models

The positioning systems of marine craft require measurements from various sensors. These typically include (but are not limited to) Global Navigation Satellite Systems (GNSS), inertial measurement units (IMUs) and gyro- or magnetic compasses. These sensors are briefly presented below, based on material from Fossen (2011) and Beard & McLain (2012). The lidar theory is based on NOAA (2012).

- GNSS provides position information based on measurements from satellites. Examples of such systems include the American GPS, the Russian 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 GPS receiver and a satellite. This gives an *estimate* of the distance to the satellite - termed the pseudo-range - because synchronisation errors exist between satellite and receiver clock. Thus, to determine the three-dimensional position of a GPS 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.

- IMUs typically contain accelerometers and rate gyros. Accelerometers measure specific force, and 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. In

general, measurements from IMUs can be modelled such that the output is the sum of the measured quantity, together with sensor bias $\boldsymbol{b}$ and additive noise $\boldsymbol{w}$ (Mahony et al., 2008). Thus, for the accelerometer and rate gyro we have the respective models

$$\boldsymbol{a}^b_{\text{imu}} = \mathbf{R}^\top(\dot{\boldsymbol{\nu}}^n + \boldsymbol{g}^n_0) + \boldsymbol{b}^b_{acc} + \boldsymbol{w}^b_{acc} \tag{2.30}$$

$$\boldsymbol{\omega}^b_{\text{imu}} = \boldsymbol{\omega}^b + \boldsymbol{b}^b_{\text{gyro}} + \boldsymbol{w}^b_{\text{gyro}}, \tag{2.31}$$

where $\dot{\boldsymbol{\nu}}^n$ is the linear acceleration in NED-frame. The $\boldsymbol{b}$-variables represent sensor bias, while the $\boldsymbol{w}$-variables are additive zero-mean measurement noise. IMUs also sometimes provide orientation data through use of gyroscopes or magnetometers.

- 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

$$\boldsymbol{m}^b_{\text{imu}} = \mathbf{R}^\top \boldsymbol{m}^n + \boldsymbol{b}^b_{mag} + \boldsymbol{w}^b_{mag}, \tag{2.32}$$

  which is similar in structure to the above models. However, $\boldsymbol{b}^b_{mag}$ 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 electromotors. 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

$$\psi_m = -\text{atan2}(m_y, m_x) \tag{2.33}$$

  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, 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;

$$\psi = \psi_m + d. \tag{2.34}$$

- Lidar (a portmanteau of "light" and "radar") is a sensor used for generating spatial information. By emitting intense laser pulses and measuring the time it takes for the pulse to be reflected and subsequently received by the sensor, the range to the reflecting object can be determined. Considering the magnitude of the speed of light, and the fact that most Lidar platforms are moving, it should be clear that it is crucial to know the instantaneous position of the sensor to get a good range estimate. Thus Lidar sensors are enabled in part by accurate measurements from GNSS and IMUs.

Chapter 3

Background on Machine Learning

Machine learning has been used for decades, and deep learning surfaced in 1940s (Goodfellow et al., 2016). Different names and procedures have been popular, while the goal of machine learning to use past experience to solve problems for new situations (Alpaydın, 2010) has been more or less the same. The machine learning algorithm learns patterns, and connects solutions to a problem when similar patterns are seen later. This can be valuable for solving problems previously unanswered, or handle situation one did not realise could occur.

3.1 Taxonomy and Definitions

There are quite a few notations and definitions used in this chapter and in the case study in Chapter 5. To make it easier to read the rest of the thesis, important terms and definitions used regarding machine learning in this thesis is summarised in Table 3.1. Most of these terms are described more in depth in this chapter.

There are many different problems that can be solved using machine learning. The two most common problems are the classification and regression problems (Goodfellow et al., 2016). In the classification problem the algorithm is asked to decide what category some inputs belong to. Mathematically this is denoted as in Equation (3.1), where y is the output, x is the input and the function f maps the input to a category.

$$y = f(\boldsymbol{x}), \quad f : \mathbb{R}^n \rightarrow \{1, ..., k\} \tag{3.1}$$

This corresponds to a discrete, final number of possible outputs. It could be the task of deciding weather a ball is blue or red. Then the input would be some sensor data of the ball, probably a picture. Then red could be category one and blue category two.

For the regression problem the output is a numerical value, and there are infinite amount of possible outputs. In Equation (3.2), it can be seen that the mathematical expression is quite similar to that of the classification problem. The only difference is that the function maps to a continuous space instead of an integer.

Term	**Definition**
Action	The agent choose an action to interact with the environment and transfer from one state to another
Agent	Decides the next action based on current state
Classification	Machine learning problem where the output is a category
Deep learning	Machine learning that use multiple layered neural networks
Deep Q-learning (DQN)	Q-learning with neural network
Deep reinforcement learning	Reinforcement learning with the use of deep neural networks
Discount rate	Determines the value of future rewards compared to now
Environment	Everything the agent cannot control, but has to consider when taking decisions
Episode	Consists of all steps from the initial state to a terminal state
Learning rate	Determines how fast the algorithm should converge to a solution
Neural network (NN)	Numerous algorithms in series that try to discover patterns and relationships in data sets by imitating the behaviour of the human brain
Observation	Data received about the vessel condition and environment
Q-learning	Reinforcement learning method
Recurrent neural network (RNN)	Neural network with recurrent layer, that is especially good at processing sequences due to its ability to mimic memory
Regression	Machine learning problem where the output is continuous
Reinforcement learning	Machine learning that does not require test data, but learns directly by trial and error
Reward	The feedback the agent receives after a step
Reward function	Calculates the reward such that the reinforcement learning algorithm behaves desirable
Supervised learning	Machine learning method that maps input to output based on labelled training data.
State	Processed observation, which the agent base its decision upon
State space	A set of all possible states
Step	Transit to new state by executing one action and receive the reward
Terminal state	A state that ends the episode
Unsupervised learning	Machine learning method that looks for patterns in unlabelled data

Table 3.1: Important machine learning terms

$$y = f(\boldsymbol{x}), \quad f : \mathbb{R}^n \to \mathbb{R} \tag{3.2}$$

An example could be to predict how many millimetres it will rain the next day. Often a problem can be described as either a regression or a classification problem,

depending on how the problem is defined.

Both the regression, and the classification problem can be solved by different approaches depending on the specific problem:

- **Supervised machine learning:** using training data with a known correct answer is required. The machine learning algorithm learns a pattern connecting the input to the correct answers on labelled training data, and then utilise those learned patterns on unlabelled test data.
- **Unsupervised machine learning:** the correct answer is not known, and the goal is to discover trends or clusters in the data set. Regression and clustering techniques are commonly applied.
- **Reinforcement learning:** learn a behaviour in an environment based on trial and error.

The relationship between a problem solver and a problem type is given in Figure 3.1.

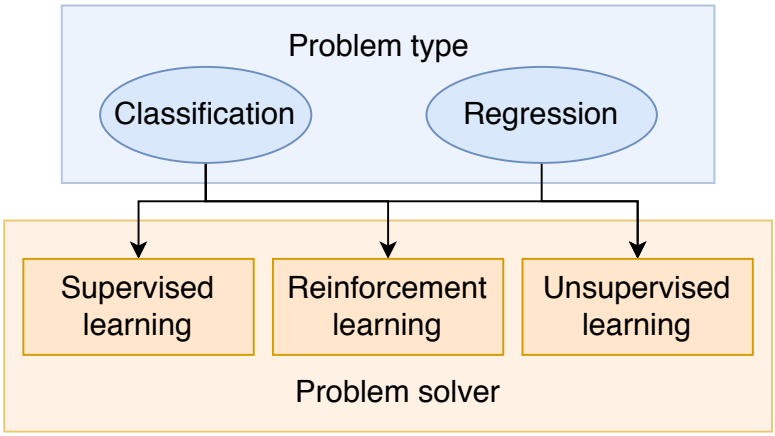


Figure 3.1: Machine learning taxonomy

Reinforcement learning will be used in this study, and is described more thoroughly in the next section.

3.2 Reinforcement Learning

Between unsupervised and supervised machine learning, one finds reinforcement learning (Otterlo & Wiering, 2012). Reinforcement learning is used when several actions are needed to reach an answer and there is limited feedback. It uses trial and error to learn what the best actions are, and therefor no "truth" about the process is required beforehand (Alpaydın, 2010). In order to decide what results are better than others, policies are developed.

For an autonomous ship two examples of accidents to be avoided are collision and grounding. This can also be expanded to include travel length, or the number of changes in direction. The voyage is successful if there are no collisions and groundings, and the travel length is reasonably short. The point here is that it is the complete voyage from start to finish that is evaluated, and not a single turn or decision. This does not mean that a turn cannot be bad, but instead that the chosen route is judged

based on the whole voyage. In this example, the voyage for the vessel is what is normally described as an episode in reinforcement learning. An episode is the whole sequence of actions (Alpaydın, 2010).

3.2.1 Markov Decision Process

Markov decision process is a framework used to describe reinforcement learning problems mathematically. The most important elements in a Markov decision process, are the agent and the environment. The agent takes decisions and can be seen as the algorithm that is trained. Surrounding the agent, is the environment. It contains everything the agent cannot control, but has to consider when choosing an action to execute. This includes what state the agent is in, what possible actions can be taken and how large the reward each action gives (Alpaydın, 2010). This relationship is visualised in Figure 3.2.

The assumption that all the information that is needed to take the next action is stored in the current state is essential for a Markov decision process. This means that the past states and actions are unnecessary to move forward. This is called the Markov property (Sutton & Barto, 2015).

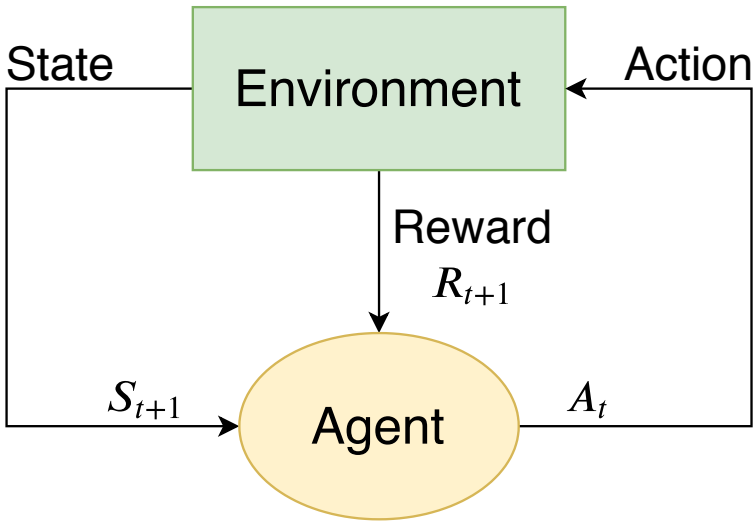


Figure 3.2: Relationship between the agent and the environment

There are three main properties in a Markov decision process as defined in Sutton & Barto (2015):

1. $\mathcal{S}$ is a set that contain all possible states. $S_t \in \mathcal{S}$ is the state the agent receives from the environment at time t.
2. At each state, there is a set of possible actions $\mathcal{A}(S_t)$, where the agent chooses an action $A_t \in \mathcal{A}(S_t)$.
3. After the action is chosen, the agent receive "feedback" from the environment: a numerical reward R_{t+1} and a new state S_{t+1}

Considering the agent as a vessel operating at sea, the environment would consists of everything that effects the vessel. This could be environmental forces like wind, waves and current, or stationary and dynamic obstacles. In addition to this, the environment

consists of all possible states the vessel can have, and the rewards corresponding to each state.

At the beginning of the run, the vessel may be in state one, and it can choose between e.g. ten different actions. If this is the first simulation, it guesses on an action and sees how large reward it earns. This action takes the vessel to state five, and then it has six actions to choose between. The goal is to find a policy, so that the algorithm learns an optimal action for every state (Otterlo & Wiering, 2012). In this example, the states are discretely defined, while they in an episode may be more vague, and there could be continuous transitions between states.

3.2.2 Reward System

The reward system, or credit the agent receives after an action, has to be developed to achieve the desired behaviour. Since the agent's goal is to gather as high reward as possible, the way the reward system is built up is crucial for the learning (Sutton & Barto, 2015). It's important to note that the rewards are built up to achieve some end goal, and not how the goal should be reached. This gives the agent flexibility to find the best solutions.

The total rewards earned is the sum of each reward earned after an action (3.3).

$$G_t = R_1 + R_2 + R_3 + ... + R_T \tag{3.3}$$

Where G_t is the total reward gained in an episode, and R_T is the reward received at a terminal state. An optimal solution is the one gaining the highest reward G_t. For a vessel, the agent could be rewarded for ending up at the terminal state after an episode without any collisions, while the optimal solution is the path achieving this in as few steps as possible.

3.2.3 Exploration versus Exploitation

At each step the model has a number of different steps it can take. If it takes the step it has already learned to be the best, it is exploitation of the current model. Doing something else, more or less random to try to discover even better steps is exploration of the space. This is commonly known as the exploration/exploitation trade off (March, 1991).

Exploitation sticks to the best known solution and explores only a limited part of the region to reach a global solution. Exploration utilises the whole state space in a search for the global best solution. On one hand a model with high degree of exploration usually takes longer for the algorithm to converge since it tries a large amount of actions and states than with exploration. On the other hand it has a higher probability of finding a global optimal solution since it explores the whole space.

Considering this relationship in a safety perspective for vessels, it could involve too much risk having a high degree of exploration. For vessel navigation it is important to stay away from areas that are known to be shallow or avoid moving forward if it involves danger of collision. Even though the model could take an exploration step every 100 steps, it could be fatal if the vessel collides with an object. This relationship has to be decided carefully and needs to be considered in a risk index guiding the control system.

3.2.4 Dynamic Environment

Since a vessel operates in an environment subject to changes, also the reward function and transition function can change. Imagine another vessel entering the environment, making a no-go zone with low rewards where it previously was safe to go. Also the transition function can change with different weather conditions, as the vessel can end up at a different state than before due to heave current. These are some examples of situations the method has to cope with in order to get realistic simulations and therefore develop a good policy for a real-life situation.

There are many different algorithms and ways to implement reinforcement learning on a process. Two different models will be presented in the following sections.

3.3 Q-Learning

Q-learning is a basic reinforcement learning method built on temporal-differences learning. It learns after every action, instead of after the whole episode in contrary to Monte-Carlo methods (Sutton & Barto, 2015). This section will cover the basics of Q-learning, but for more in-depth reading Watkins (1989) can be explored.

Central for Q-learning is a controlled Markov process, where the agent works as the controller. At the beginning the agent is in a random or an initialising state, and has possible actions depending on the state. The agent's task is to maximise reward and find a policy that achieves that (Watkins & Dayan, 1992).

The agent tries different actions at different states, and the resulting discounted rewards make the basis for a matrix. For each action and received discounted reward, the value in the reward matrix is updated. This value is known as the Q-value. For this learning process to converge towards an optimal policy, it requires that the amount of episodes the agent can learn from are infinite. To train the model on an infinite amount of episodes is unrealistic. However, with a large amount of episodes it can converge to a local optimal solution, just not necessarily to the global optimal solution (Watkins & Dayan, 1992).

In Table 3.2 there is an example of a reward matrix. There are 6 possible states and 5 actions. The values are the average values received doing the different actions while at the respective states.

	state					
action	1	2	3	4	5	6
1	0	0.1	4	2	0	0
2	2	0	0	0.1	4	2
3	0	0	0	1	1	3
4	3	1	1	0	0	1
5	2	0	0	0.1	4	2

Table 3.2: Example of a reward matrix with Q-values

If the vessel is at state one, the agent knows that in can receive a reward value of zero for action one and three,two for action two and five and three for executing action four. Assuming that the agent chooses the action that seems to give the largest reward, action four is taken. Transitioning from state one taking action four, the vessel may end up at state six and a new action is choosen.

In Algorithm 1 the basic concept of Q-learning is described. It is based on Figure 6.12 in (Sutton & Barto, 2015), but is adjusted according to the MilliAmpère case. The outer loops run through all the episodes available for learning. The inner loop represents each episode which terminates when the terminal state is reached. Q is the reward matrix, S the states, and A are the actions. When Q(S, A) is initialised, all the elements in the reward matrix receive an initial Q-value which could be zero or arbitrary.

```
Initialise Q(S,A);
repeat
    Initialise S ;
    repeat
        Choose A from S using policy derived from Q ;
        Take action A, observe R, S'
         Q(S,A) ← Q(S,A) + α[R + γmax_A Q(S',A) − Q(S,A)] ;
        S ← S' ;
    until S is terminal;
until No more episodes;
```

Algorithm 1: Pseudo-code for Q-learning

γ is the discount rate, deciding how fast the value of future rewards decreases. The highest reward obtainable for the action is given as max_A. α is the learning rate and influence to which degree the Q-values are updated. $\alpha = 0$ gives no update, while 1 gives quickly updates.

The policy that chooses the action is often a ϵ-greedy algorithm choosing the most beneficial action according to the reward matrix, with the possibility of doing a random action. The probability of choosing a random action ϵ is what gives exploration, while utilising an already known best move exploits the developed model.

The reward matrix is updated by Equation (3.4).

$$Q(S,A) = Q(S,A)_{old} + \alpha[R + \gamma \cdot max_A Q(S',A) - Q(S,A)] \tag{3.4}$$

It is dependant of both the current Q-value $Q(S,A)_{old}$, and the potential received reward in the next state. S' is the next state. $max_A Q(S',A)$ is the maximum reward that can be received by taking an action, and is found by comparing the values of a line in the reward matrix.

3.4 To Deep Learning: Neural Networks

A common problem for reinforcement learning algorithms, is the need to store large amount of data. Creating huge tables to store data can lead to new problems often referred to as *The curse of dimensionality*. To be able to store as much information as needed, while avoiding these problems, neural networks are used. This is known as deep learning.

Neural networks used in machine learning applications have been explored for a long time, and as early as in 1960 for an adaptive classifier (Widrow & Hoff, 1960). Neural networks have their inspiration from animal brains, and therefore use terminology from biological concepts. The main idea is to train the processing ability of a network by learning from experiences or previously seen patterns (Gurney, 1997).

3.4.1 Network Structure

A neural network consist of several neurons that are grouped in different layers, where the outer layers are input and output of the network as seen in Figure 3.3.

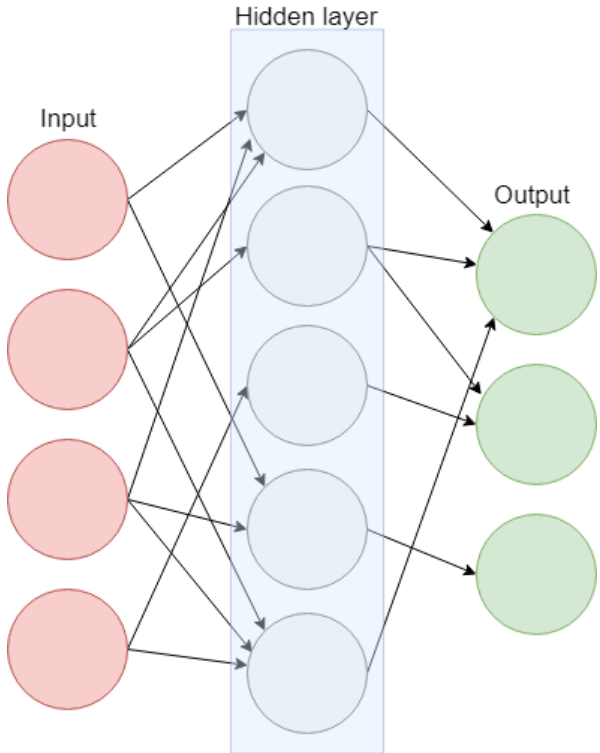


Figure 3.3: Neural network

The neurons in the layers next to each other have weighted connections. The result of this is the input to the different neurons being multiplied by the weight in the connection. This gives the basis for the value of the next neuron which is found by inserting it into a bias function. The values propagate like this through all the layers until the output is reached (Gurney, 1997). In order for the neural network to function properly, the weights have to be trained or tuned. Tuning can be done using machine learning as well.

A standard neural network is a feedforward neural network (Goodfellow et al., 2016). To compare a neural network mapping to the classifier defined in Section 3.1 Equation (3.5) is examined.

$$y = f(\boldsymbol{x}; \boldsymbol{\theta}), \quad f : \mathbb{R}^n \rightarrow \{1, ..., k\} \tag{3.5}$$

The function still maps to a category, but it is done by learning $\boldsymbol{\Theta}$ values. When $\boldsymbol{\theta}$ values are learned, it should produce a good function approximation. Being a feedforward neural network it implies that the neurons only have connections to the next layer as opposed to to the previous. The number of layers decide the depth of the network, and the structure is given by Equation (3.6).

$$f(\boldsymbol{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\boldsymbol{x}))) \tag{3.6}$$

$f^{(3)}$, $f^{(2)}$, and $f^{(1)}$ are the different layers, and are connected in a chain. Hence, this example structure has a depth of three.

Considering that there could be thousands of neurons and millions of connections, it is difficult to comprehend why an input leads to a certain output. This is why a neural network is often considered to be a "black box". You apply a known input and receive an output that could be good, but without knowing why or how it worked.

3.4.2 Activation Functions

Every hidden layer has an activation function to decide the values in the layers. Each neuron in the layer is dependant on the activation function, which decides if the data the neuron has is necessary for the mapping. The relationship is shown in Figure 3.4, where $g(\boldsymbol{x})$ is the activation function. Often the activation function is non-linear to avoid the network being simple linear relations.

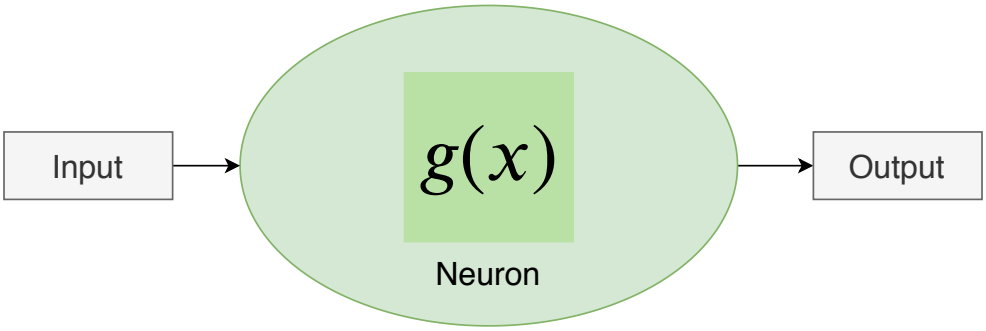


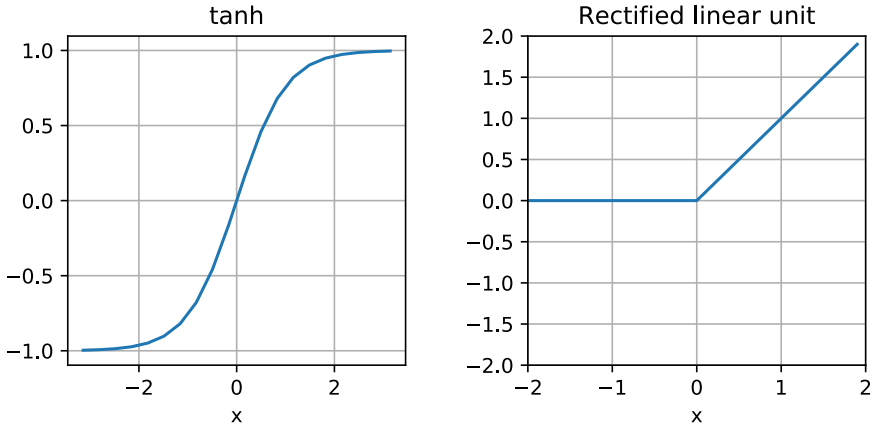
Figure 3.4: Relationship between neuron and activation function

There are many activation functions to choose between. Tanh, sigmoid, softmax and the rectified linear unit (ReLU) are some examples. ReLu has shown promising results for quick convergence, and therefore the network is trained faster than with other activation functions (Krizhevsky et al., 2017). Often the training is slowed down due to a saturation in the activation functions, but this is avoided by the non-saturated gradient ReLu use. For the output layer it is important that the activation function is adapted to the problem. Two activation functions are shown in Figure 3.5.

For a regression network, one typically use a linear activation function, while for the classification network a softmax or sigmoid function can be applied. Softmax is typically preferred over sigmoid unless the classification problem is binary. If it is a multiclass classification, you would use softmax which returns an array with probability or q-values for each category. To get the desired category, the softmax function is often combined with an arg-max function.

3.4.3 Convolutional Neural Networks

Convolutional neural networks (CNN) are often used to process large input that have a grid structure (Goodfellow et al., 2016). Hence, it is often used for image classification. It is the convolutional operator that makes the network convolutional. In at



(a) Tanh (b) ReLu

Figure 3.5: Two activation functions

least one of the layers, the matrix multiplication is substituted by convolution. The convolutional operation is given in Equation (3.7)

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{3.7}$$

More in depth information about CNN can be found in Peng (2018) and Khan et al. (2019).

3.4.4 Recurrent Neural Networks

Recurrent neural network (RNN) is a network used to process sequential input (Goodfellow et al., 2016). This is often time series or word sentences. RNN is often used to introduce artificial memory to the agent. Hence, input previously received can be used to calculate the present output. Hyperbolic tangent is often used as the activation function, and there are several different ways of implementing a recurrent network.

SimpleRNN is a straight forward recurrent layer, where the output is reused, and fed back to the input for the next step. Other more sophisticated layers are long-short-term memory (LSTM) and gated recurrent unit (GRU). Both these layers are gated, which have weights that can change with each time step, and construct time dependant paths that depend upon derivatives that does not go to zero or infinity. More on LSTM and GRU can be found in Hochreiter & Schmidhuber (1997) and Dey & Salem (2017) respectively.

3.5 Deep Q-learning

The biggest limitation for Q-learning is the capability to handle large data sets. The reward matrix has to be recomputed if the scenario changes, and if the table is to large, the behaviour becomes unstable. Therefor neural networks have been used instead of a table, making it deep Q-learning (DQN). Previously it has been used widely to learn to play Atari 2600 games, where Mnih et al. (2013) is one example.

However, DQN showed to overestimate actions values. To cope with this problem Hasselt (2010) introduced Double Q-learning. Lei et al. (2018) developed the theory further and proposed an approach inspired by Hasselt et al. (2015). The major breakthrough of this came with Lei et al. (2018) which combined double Q-learning with deep reinforcement learning. This approach called DDQN yielded good results for agents in dynamic environments. That the network is double refer to the use of a target network which stays constant for some time period. The main network is used to choose actions, while the second network, often called target network, evaluate the action.

DDQN can be used with different types of neural networks, but previously it has been used mostly with convolutional networks. When convolutional networks are used, some graphical representation of a game or the environment is the input.

Pseudocode for DDQN is given in Algorithm 2.

```
Initialise Q^A, Q^B, s;
repeat
    Choose a, based on Q^A(s,·) and Q^B (s,·), observe r, s';
    Choose (e.g. random) either UPDATE(A) or UPDATE(B);
    if UPDATE(A) then
        Define a* = arg max_a Q^A(s',a);
        Q^A(s,a) ← Q^A(s,a) + α(s,a)(r+γ Q^B(s',a*)-Q^A(s,a);
    else if UPDATE(B) then
        Define b* = arg max_a Q^B(s',a);
        Q^B(s,a) ← Q^A(s,a) + α(s,a)(r+γ Q^A(s',a*)-Q^B(s,a);
    end
    s ← s';
until end;
```

Algorithm 2: Pseudo-code for DDQN

3.6 Transfer Learning

Transfer learning is the concept of reusing already learned behaviour in a new setting (Goodfellow et al., 2016). It is beneficial when the algorithm is to perform a new task that is heavily linked with previous experiences. It is often used for supervised learning where CNN are trained, due to the need for massive amount of training data. An example is image recognition which require large amount of training data to perform well. By using transfer learning Kendall et al. (2015) managed to train a deep CNN to act as a robust camera pose regressor, by utilising transfer learning on a network pre-trained on large datasets as ImageNet (Fei-Fei et al., 2016) and Places (MIT, 2015).

For reinforcement learning, transfer learning has proved more difficult. Gamrian & Goldberg (2018) showed that if the visuals change, the performance drops significantly and that the learned policies cannot be reused. However, Hafner et al. (2019) dynamics were learned from pixels and that the learned dynamics transferred well to new tasks. Both these examples for deep reinforcement learning with transfer learning utilise CNN. It is easier as an image always has pixels as and the colours are the same. For RNN it is crucial that the networks learns from other networks that have similar challenges. As an example one cannot use transfer learning on a RNN used to

classify text in English to classify text in Norwegian. There are some examples that have proved promising results as Gupta et al. (2018) and Giel & Diaz (2015).

A relevant example for this thesis, is vessel guidance and navigation. If the vessel has already learned how the different actions interact with the environment, it may easier learn to handle obstacles and kayaks when they are introduced. This scenario will be tested in the case study in Chapter 5.

3.7 Challenges

Even though deep learning has shown promising results recent years, there are some challenges with NNs.

3.7.1 Underfitting and Overfitting

The goal of training a machine learning algorithm is to achieve a small training error, and then test the algorithm on test data and receive performance as close to the training performance as possible. This would lead to the algorithm generalising learned lessons well to new situations, and is the ideal behaviour. When reaching for this result, there are some pitfalls that are not so easy to avoid according to Goodfellow et al. (2016):

- **Underfitting:** Is when the the error during training is not low enough to achieve reliable enough results.
- **Overfitting:** Is when the training error is acceptable low, but the test error is much larger. The algorithm does not generalise learned behaviour to new situations.

Neither of these situations are acceptable, and there are many methods to try to reduce the risk of them occurring. However, for many situations it is necessary to train in simulators or synthetic data, which have a tendency to overfit and struggle to adjust to a real situation.

In addition to the expected performance of the algorithm, deep learning has thrust issues. For crucial decisions, it is required that the humans around can thrust the system. This thrust is not so easily obtained, as long as the algorithm is a black box. There are so many parameters and operations that it is practically impossible to interpret why the algorithm decide as it does. Is this is acceptable or not for the tasks the algorithm performs is debatable.

3.7.2 Testing and Verification

As the industry search for new ways to utilise the computational capacity of deep networks and machine learning in general, a need for verification of the systems occurs. DNV GL has written a technical note, where they point out characteristics that a trustworthy artificial intelligence system must possess (DNV GL, 2019). This is important as artificial intelligence gives new opportunities, new risks are also introduced at the same time.

DNV GL defines an artificial intelligence system as trustworthy if it displays the following properties:

- **Legitimate:** The risk of using the algorithm has to be approved by all stakeholders. This is independent of how well the system can perform. For the system to be legitimate it has to be designed for the correct use, and focus on risk management.
- **Ability to perform and capacity to verify delegated tasks:** The same requirements that leaders and co-workers are evaluated on, the AI system should be required to fulfil. The system should be knowledgeable enough and able to do the tasks it is delegated. Hence, it has to be robust and explainable enough to be able to verify that the system is trustworthy.
- **Appropriate human-machine interaction:** As a higher level of autonomy is reached, it is important to understand the different parts the machine learning agent and human plays in the system. To know how the system should be maintained, and how the system influence external stakeholders is also necessary. To succeed in reaching a trustworthy system, all these roles must communicate in an open and recognisable way.
- **Clearly defined purpose:** The need and usage of the AI system have to be clarified, in addition to identify the benefits and potential risks the system may pose.
- **Transparent impact on relevant stakeholders:** How the AI system may affect the decision making is also relevant. There have been examples of AI systems being discriminatory, or biased. These potential effects have to be transparent, and evaluated up against the benefits for using the system. It is also important to monitor how the system impacts stakeholder through the lifecycle to be able to catch up on eventual changes or unacceptable behaviour.

A recurring point is the need for transparency and identification of risks. In addition the importance of clarifying what the system is supposed to do and why it is beneficial is central.

DNV GL splits AI system into three different groups: Knowledge-based, experience-based systems and reinforcement learning. Knowledge-based is transparent in the way that the logic is clear, but often difficult to interpret in practice. Experience-based is data-driven and hence difficult to understand, which amplifies the need for sufficient testing and verification. Reinforcement learning is like experienced-based AI, in the way that it is not possible to verify the model beforehand.

There are several pitfalls when using Reinforcement learning. How the optimisation object is defined is critical for how the AI system solves its tasks. Amodei et al. (2016) have identified and investigated several issues with AI systems, as reward hacking, safe exploration, the robustness and other negative side effects. All these problems have to be addressed for the system to be classified as trustworthy, and as for experience-based system, extensive testing is required.

Chapter 4

Flexible Guidance System for Waypoint-Following

There are many guidance laws already existing as those described in Section 2.5. LOS is often used for under-actuated vessels, controlling only the forward velocity and course angle. A position reference filter often used for DP vessels and models the vessel to stop at the waypoints by filtering the velocity. Velocity reference filters only generate references for velocity without taking direction into consideration. By combining LOS and reference filters one can achieve a guidance system which track positional waypoints as well as velocity references to maintain a desired forward velocity. By generating references to the DP, offsets from the path can be handled by the DP system. Hence, the steady-state cross-track error is compensated for by sway control action, instead of a steady-state heading non-parallel to the path as e.g. ILOS does. A novel guidance method being flexible for using input from an operator or machine learning algorithm has been proposed in this thesis where the key features are:

- **The desired velocity can vary and is maintained through the waypoints:** The velocity reference filter ensures that the desired velocity is kept, and that the vessel does not stop at waypoints.
- **It is easy to alter the route:** The next waypoint is only needed once the previous is reached.
- **DP compatible:** The references are created to be feasible for a DP system, which will handle offset from path due to for example environmental conditions.
- **The waypoint only need desired position in north and east direction:** The desired angle is calculated by the guidance system.

It is a flexible guidance system since the desired velocity can vary, and the next waypoint is not necessarily decided until the current one is reached. Consequently the waypoints can be decided during the run, instead of being decided beforehand.

4.1 Method

The main goal of this guidance system is for the vessel to follow waypoints at a desired velocity. The desired velocity can vary between some maximum value and zero as described in Equation (4.1).

$$0 \leq U_d \leq U_{max} \tag{4.1}$$

The waypoint consists of coordinates in north and east direction, which implies that the yaw reference has to be created by the guidance system. Intuitively it is desirable that that the heading is equal to the course direction. To use the existing DP-controller, references in 3-DOFs for position, velocity and acceleration are required.

The next waypoint is only known to the system, once the vessel is sufficiently close to the current waypoint. Towards the waypoint, the vessel is to converge to the path decided by line-of-sight such that the control objectives are formulated as:

$$\lim_{t \to +\infty} \psi = \psi_{los}, \tag{4.2}$$

$$\lim_{t \to +\infty} U = U_d, \tag{4.3}$$

and ultimately the waypoint should be reached.

4.1.1 Yaw Reference

Firstly the yaw reference will be determined. The yaw reference is important as it will decide the course angle, and the course angle determines the velocity components in north and east direction. LOS will be used for this guidance system as described in Section 2.5.3. This generates a heading reference that leads the vessel towards a straight line connecting the previous waypoint with the next.

Lookahead-based steering will be used at the expense of enclosure-based steering, since it is applicable for all cross-track errors and is less computationally demanding (Breivik & Fossen, 2009).

When calculating the cross track error using Equation (2.23), the previous reference is used instead of the actual position to avoid a link between the actual position and the guidance system. This implies that the guidance system is a stand-alone system independent of offsets the vessel might have. Any deviations should be handled by the DP system.

To obtain smooth reference signals for yaw, yaw rate and yaw acceleration, a standard position reference filter is applied as described in Section 2.5.1 for a scalar reference. The LOS-angle ψ_{los} is the desired setpoint and ψ the filtered reference. The reference filter is applicable as it is desirable to reach the referenced heading without to much oscillation, as opposed to maintaining a constant yaw rate.

4.1.2 Surge and Sway References

Once the yaw reference is decided, references for surge and sway can be calculated. The coordinates provided by the waypoints are not used directly by the reference model as commonly done. Instead a desired velocity is used to provide the total velocity while the heading reference decide the movement direction.

The velocity reference model from Section 2.5.2 is applied to achieve smooth signals with U_d as the desired velocity. U_d is set by the operator, and the reference U

generated by the filter is decomposed into two components in north and east direction such that Equation (4.4) is fulfilled.

$$U = \sqrt{\dot{x}^2 + \dot{y}^2} \tag{4.4}$$

$\dot{x}$ is the velocity component in north direction, and $\dot{y}$ is the velocity component in east direction. In order to reach the desired position with a constant velocity, the heading reference is used to decide the velocity components $\dot{x}$ and $\dot{y}$. The velocity components are then obtained by Equations (4.5) and (4.6) assuming that the course angle is the same as the heading.

$$\dot{x} = U\cos(\psi) \tag{4.5}$$

$$\dot{y} = U\sin(\psi) \tag{4.6}$$

From the velocity reference model, reference signals for position and velocity are obtained for north and east direction. Acceleration references are found by differentiating Equation (2.3), which results in Equations (4.7) and (4.8).

$$\ddot{x} = \dot{U}cos(\psi) - rUsin(\psi) \tag{4.7}$$

$$\ddot{y} = \dot{U}sin(\psi) + rUcos(\psi) \tag{4.8}$$

$\ddot{x}$ and $\ddot{y}$ are the acceleration references for north and east direction respectively, and r is the yaw rate reference. The references are then used as input for the DP-control system. A simplified block diagram of the guidance system is given in Figure 4.1. The reference filters only have feedback from the previous references and not actual positional data or velocity measurements from the vessel. The red blocks are decided by the operator, the yellow blocks belong to the yaw references, and the blue block is the velocity reference filter. By combining the course angle with the referenced velocity, all relevant references are obtained.

4.1.3 Circle of Acceptance

It is not necessary or desirable for the vessel to approach the waypoint at its exact position, as this may cause large deviations from the desired path. To avoid this problem a circle of acceptance is implemented. When the vessel is an acceptable distance away from the waypoint at a line parallel to the straight line between the waypoints, the vessel continues to the next waypoint.

To begin, the length of the line connecting two consecutive waypoints is calculated by Equation (4.9).

$$L = \sqrt{(y_{k+1} - y_k)^2 + (x_{k+1} - x_k)^2} \tag{4.9}$$

The along-track distance from the target is given by Equation (4.10).

$$|L - s(t)| \leq acceptable \tag{4.10}$$

In Equation (2.27), as well as for the cross-track error, the previous references are used instead of the actual position of the vessel. The acceptance value is decided by trial and error, and how large the risk is by having deviations from the waypoints and paths between them. The acceptance value should be large enough for the vessel to avoid large overshoots from the path, yet small enough to achieve the desired behaviour.

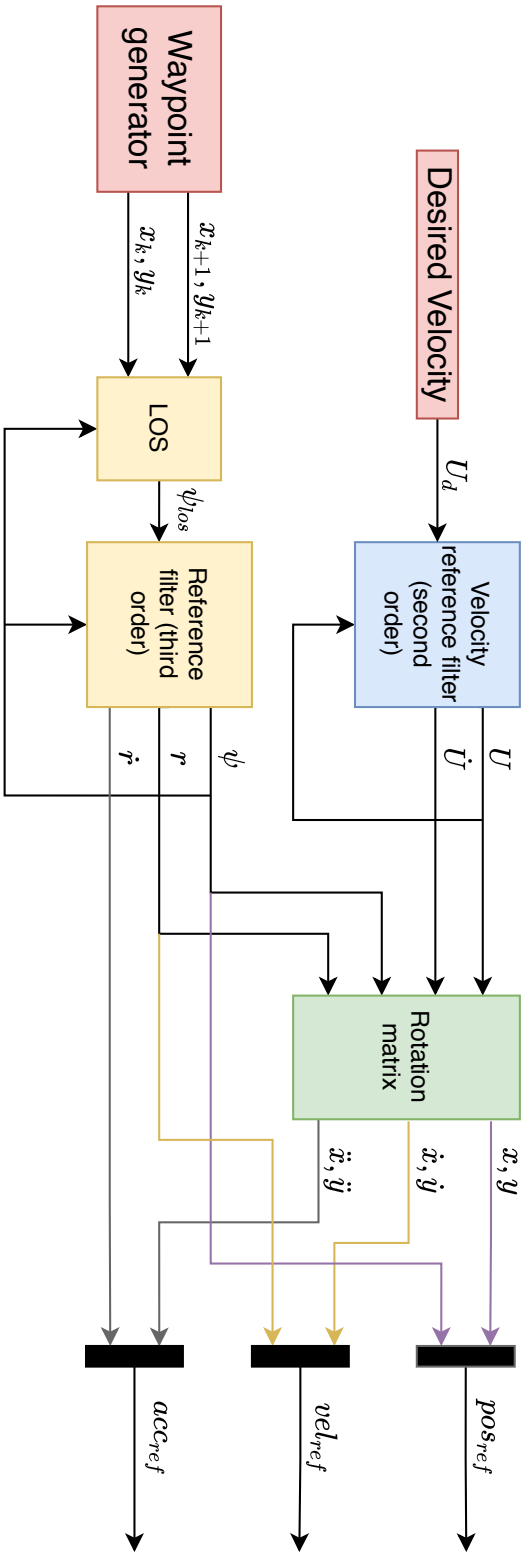


Figure 4.1: Block diagram of guidance system for waypoint-following

4.2 Results and Discussion of Simulations

To verify that the guidance law works as expected it has to be tested, and the process with testing and improving is described more detailed in Appendix A.

4.2.1 Offline Verification

Firstly an offline test was conducted in Jupyter Notebook. A path is generated between the waypoints in Equation (4.11) for a vessel originally in origo with zero heading. The goal was to verify that the desired velocity is maintained through the waypoints, and that the waypoints are reached.

$$WP = [(10, 10),\ (20, 0),\ (30, -10),\ (40, 0),\ (50, 10),\ (60, 0),\ (70, 0)] \tag{4.11}$$

Both the natural frequencies and the relative damping ratios were set to one as in Equation (4.12). The First values in the arrays correspond to the velocity reference, while the last values belong to the heading reference. The values were chosen to construct a base case for further online testing. A look-ahead distance of 7.5 meters was used, as a rule of thumb is to use 1.5 times the length of the vessel, and MilliAmpère is 5 meters long.

$$\Omega_\psi = \Omega_U = \Delta_\psi = \Delta_U = 1 \tag{4.12}$$

For this simple test the desired velocity was set to $1m/s$ and the acceptance value was set to 2.5 meters. Figure 4.2 shows the velocity reference signal. The desired velocity

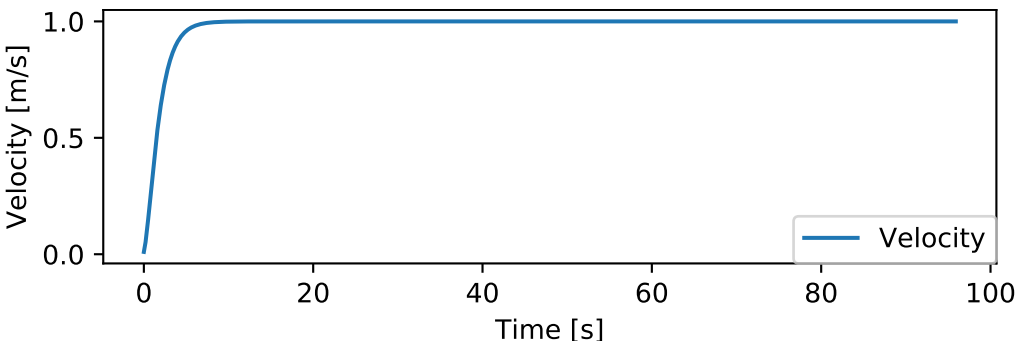


Figure 4.2: Offline velocity reference

of $1m/s$ is rapidly reached. The acceleration may be larger than a vessel realistic can achieve, and again it could lead to large overshoots and oscillations. By tuning the relative damping ratios and natural frequencies of both the heading and the velocity, more realistic references could be obtained.

Figure 4.3 shows the produced yaw reference. It shows that the transitions between different angles are smooth and that the target angles are quickly reached. The transitions may be to rapid for a real vessel. In the worst case a vessel may not be capable of producing a large enough yaw rate and acceleration. This could lead to the vessel not being able to follow the reference at all. In a milder case it could lead to large oscillations and overshoot. Figure 4.4 shows the generated trajectory between the waypoints. From the figure it is seen that the waypoints are easily reached. It can also be seen that with the used $\mathbf{\Omega}$ and $\mathbf{\Delta}$ the acceptance value of 2.5 meters is sufficient to change the course towards the next waypoint without a large overshoot.

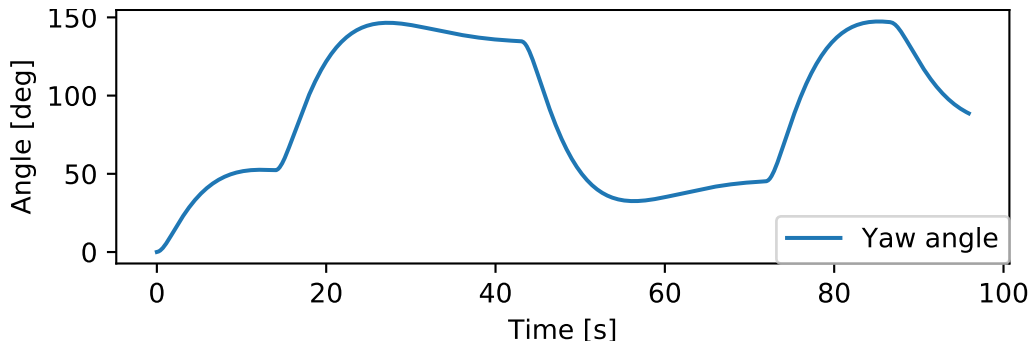


Figure 4.3: Offline yaw reference

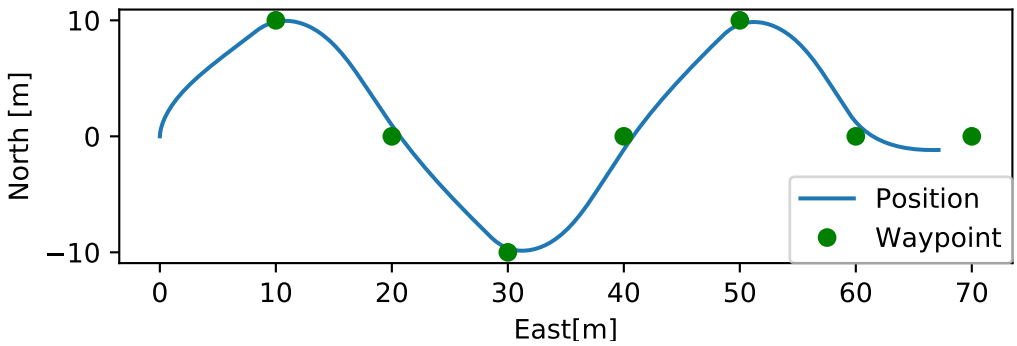


Figure 4.4: Offline generated path

The path stop some meters before the last waypoint due to the acceptance value and the trajectory has nice and smooth curves as desired.

This verification test has shown that the guidance law manages to create a trajectory between the predefined waypoints, and steer towards the waypoints at the LOS-angle. In addition the referenced velocity is constantly held to the desired velocity without trying to stop in the waypoints.

4.2.2 ROS Simulation with Constant Velocity

The guidance law has to be tested on a vessel in addition to the offline verification. Therefore simulations of the vessel were conducted using the robot operating system (ROS) model of MilliAmpère. For these simulations, the same movements as in Equation (4.11) were studied. However, the distance between the waypoints was doubled, as the vessel model has some constraints regarding turn rate and acceleration. As a consequence the vessel moves 20 meters in east direction and -20, 0 or 20 meters in north direction for each waypoint. This yields the waypoints given in Equation (4.13).

$$WP = [(20, 20),\ (0, 40),\ (-20, 60),\ (0, 80),\ (20, 100),\ (0, 120),\ (0, 140)] \tag{4.13}$$

It were examined if the guidance law provides the desired functionalities of being able to manoeuvre through waypoints at a constant velocity as well as stop when needed to. The desired velocity can also change during the run. This implies that the vessel can move at any velocity between $0m/s$ and $1m/s$ as well as stopping, providing many possibilities and thus considerable flexibility for navigation.

The guidance system was implemented as described above, and the look-ahead distance was 7.5 meters as it was in the verification test. For the mass-spring-damper system in the position and velocity reference filter to behave as expected, the relative damping ratios and natural frequencies were tuned so that the constraints on the maximum accelerations and velocities are inactive. In addition to avoiding saturation, tuning was done to try to achieve a fast response, without deviating to much from the reference. This yields Ω and Δ for the heading and velocity as given in Equation (4.14).

$$\Omega_\psi = 0.4, \quad \Omega_U = 0.5, \quad \Delta_\psi = 1, \quad \Delta_U = 3 \tag{4.14}$$

To compensate for some delay in the simulator, the acceptance value was increased from 2.5 meters to 3.0 meters, and for this first test the velocity will be held constant. Figure 4.5 shows the desired and referenced position in north and east direction, as

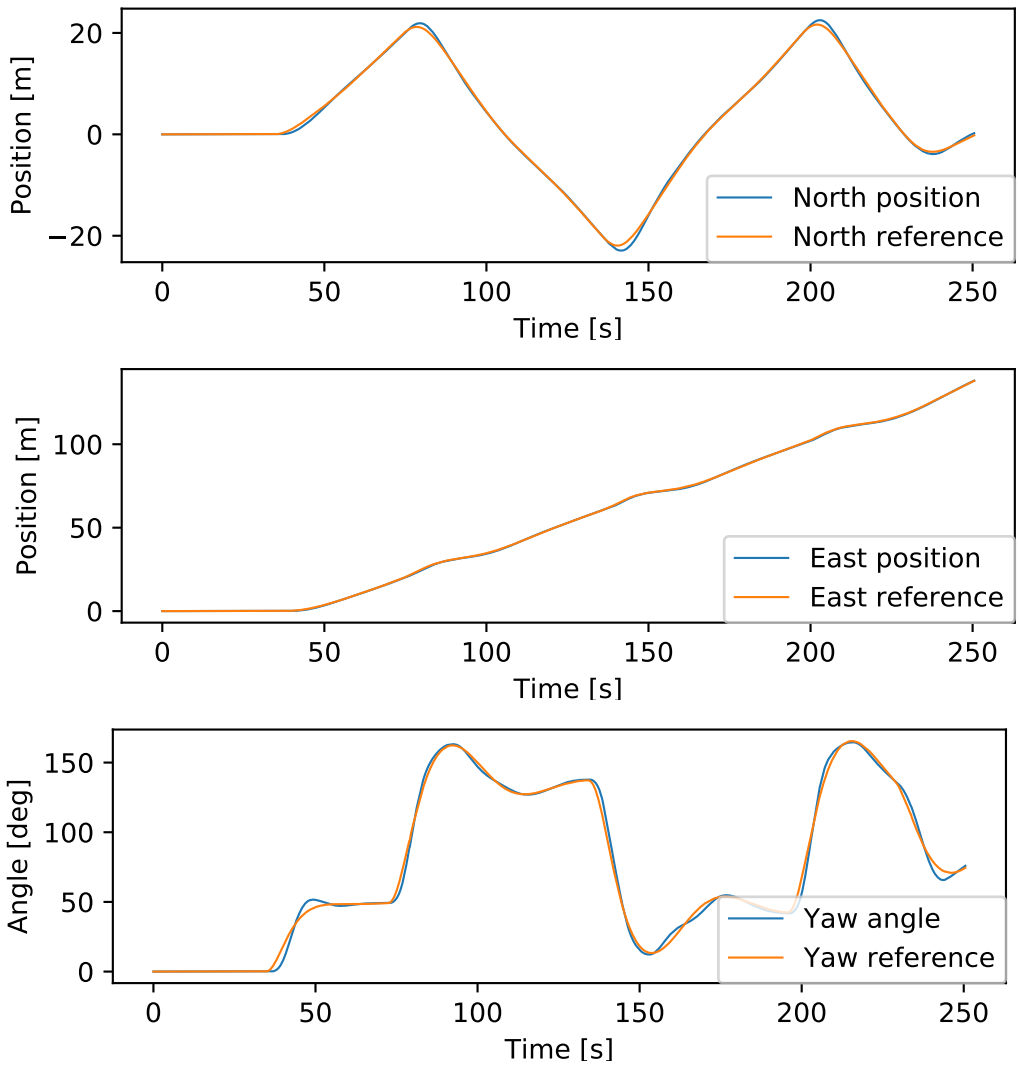


Figure 4.5: Simulation: Desired and actual position with constant desired velocity

well as the desired and referenced yaw angle. It is seen that the north and east position follow the referenced signal closely. The north position plot shows that the

vessel utilises a couple extra meters in the north direction as the vessel, as it almost reaches 22 meters. There is also a small offset from the reference in north direction after reaching the waypoint to make the turn toward the next waypoint. Just as for the position in north and east direction, the yaw angle reference is followed closely with a few exceptions. At about 50 seconds and 250 seconds the yaw angle deviates with approximately 5 degrees from the reference.

Figure 4.6 shows the forward velocity, acceleration and yaw turn rate. The deviations from the yaw angle correspond to the first and last overshoot in the yaw rate. However, there are overshoots in the yaw rate that does not lead to large deviations in the yaw angle as seen at approximately 80 and 200 seconds. The overshoot in the yaw rate can be reduced by decreasing the natural frequency. The downside of that is a slower turn rate for the vessel. Considering that the reference is followed closely for the most part, the current parameters seems like an acceptable compromise.

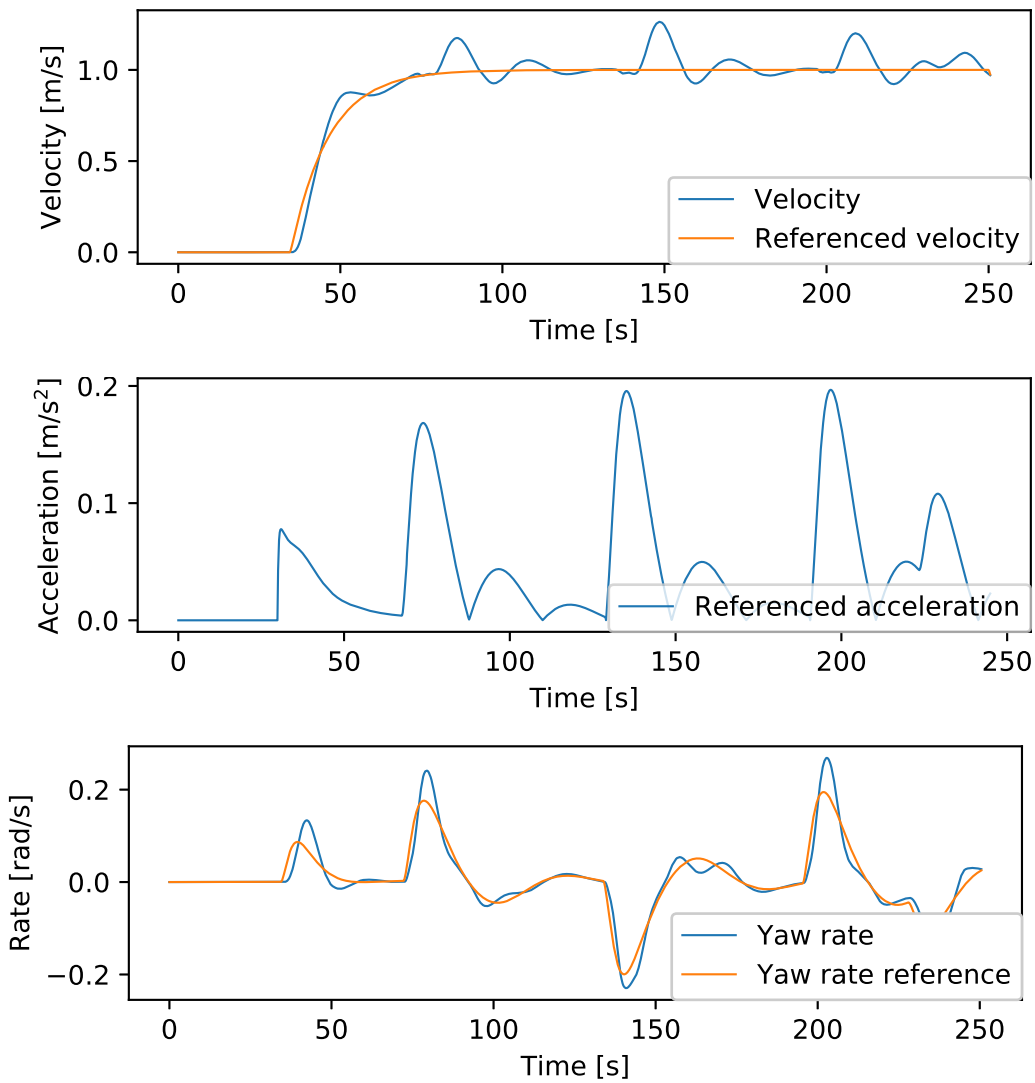


Figure 4.6: Simulation: Desired and actual velocity and turn rate with constant desired velocity

The forward velocity and acceleration are absolute values. The velocity signal is calculated by (4.4), and the same principle yields for the absolute acceleration. Firstly the goal of travelling through waypoints without stopping is reached. It manages to maintain the desired velocity trough the trajectory. The velocity quickly reaches the target velocity. There are some oscillation that may be due to the change of course direction and that the DP require the velocity reference in north and east direction, instead of one component. It is also seen that the acceleration reference is increasing very quickly. However, the signal is continuous, but to get a smoother acceleration the natural frequency could be decreased.

Figure 4.7 shows the resulting trajectory trough the waypoints, as well as the vessel heading. The waypoints are reached, but the vessel does not pass right through them. Comparing Figure 4.4 and 4.7 it is seen that the vessel has a path that goes further from the waypoints. This leads to rapid turns to obtain the LOS angle when the waypoint changes.

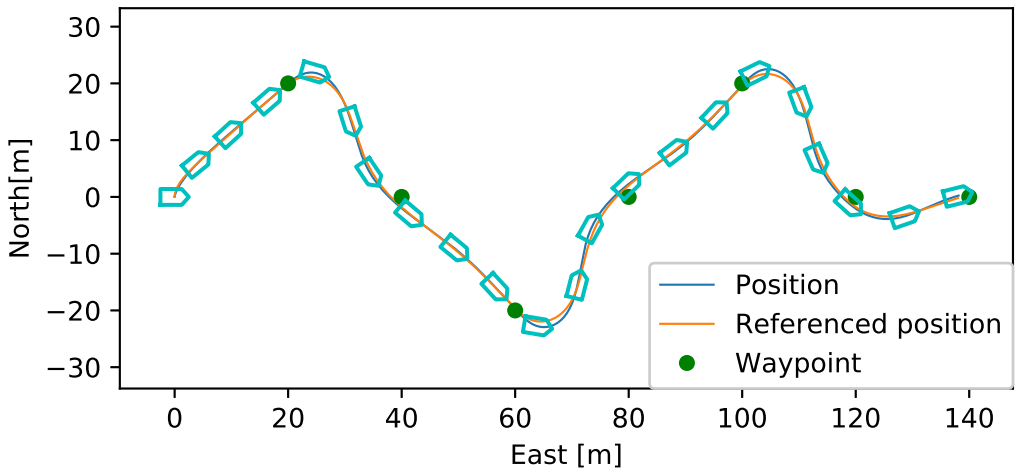


Figure 4.7: Simulation: Trajectory with constant desired velocity

The waypoints are followed, and the vessel follows the references reasonable well. To achieve a result with less deviations, the waypoints could be even further from each other and the damping increased. Other possibilities are to reduce the desired forward velocity, or increase the acceptance value, so the vessel can start the turn earlier and hence reduce the overshoot and need for high yaw rate.

4.2.3 ROS Simulation with Variable Velocity

For this test the desired velocity was changed during the run. Also the acceptance value was increased from three to eight, to check how it affected the results. As seen in Table 4.1 the desired velocity start at $1m/s$, then decreases to $0{,}7m/s$ before increasing to $1m/s$ again. When the last waypoint is reached, the vessel should stop.

The change in desired velocity is to examine the behaviour of the vessel when the referenced velocity change and see how fast the desired velocity is reached. In addition it will be investigated how fast the vessel manages to stop after the desired velocity is set to zero.

WP	1	2	3	4	5	6	7	stop
Desired Velocity $[m/s]$	1	1	1	0,7	0,7	1	1	0

Table 4.1: Desired velocity between the waypoints

Figure 4.8 shows the desired and actual position in north and east direction and the yaw angle for a test with variable velocity. As expected, the references are still followed closely for north and east direction. The position in north direction when the vessel turns have been reduced from 22 meters which gives an overshoot of 2 meters to 19,5 meters with the increased acceptance value. The change in velocity to $0m/s$ cause the plateau in east position at approximately 250-350 seconds.

When the plateau in east direction occur the vessel keeps a constant heading, which is desirable to avoid movement in an unexpected direction while the vessel stops. The second overshoot after approximately 250 seconds is now avoided. However, the first overshoot in yaw angle has not been reduced with higher acceptance value. It is still approximately five degrees.

Figure 4.9 shows the referenced and the actual forward velocity and yaw rate for the simulation test with variable velocity. It is seen that the yaw rate behaves almost identical as in the previous simulation, except at approximately 250 seconds where the reference now is smoother. This also underlines that the increased acceptance value did not reduce the first overshoot in yaw rate, as the yaw rate is the same.

The transition between a desired velocity of $1m/s$ to $0{,}7m/s$ is smooth, but takes a considerable amount of time. The vessel also manages to increase the velocity back to $1m/s$ after waypoint five. The oscillatory behaviour seems to be similar to that in Figure 4.6, and may therefore not be significantly affected by the change in desired velocity. From the vessels desired velocity is changed to zero and the vessel actually stops, it takes some time. For MilliAmpère using this guidance system it is approximately eight meters as seen in Equation (4.15), given that the vessel stops in $(0, 0,\ -144, 5)$ after waypoint seven.

$$\textit{Stop length} = \sqrt{(0-0)^2 + (148-144,5)^2} \approx 3,5m \tag{4.15}$$

A stop length of 3,5 meters should be acceptable, but depends on the mission of the vessel. By lowering the relative damping the vessel will stop quicker, but has an increased risk of oscillations.

In Figure 4.10 the generated trajectory trough the waypoints is shown. It is seen that the waypoints are easily followed, but the increased acceptance value cause the vessel to turn before the desired north position is reached. After waypoint three the desired velocity is reduced to $0.7m/s$, and that results in a trajectory further from the waypoints, with a smaller turn radius between waypoint five and six, compared to waypoint one and two. The larger acceptance value also contributes to a path that is generally tighter to the waypoints than in the first simulation test. A look-ahead distance of 7.5 meters seems to be sufficient to converge to the path after approximately 10 meters, while to much overshoot from the path generated by LOS is avoided.

This simulation test has shown that the flexible guidance system achieves the desired behaviour. The vessel manages to manoeuvre through a set of waypoints at a desired

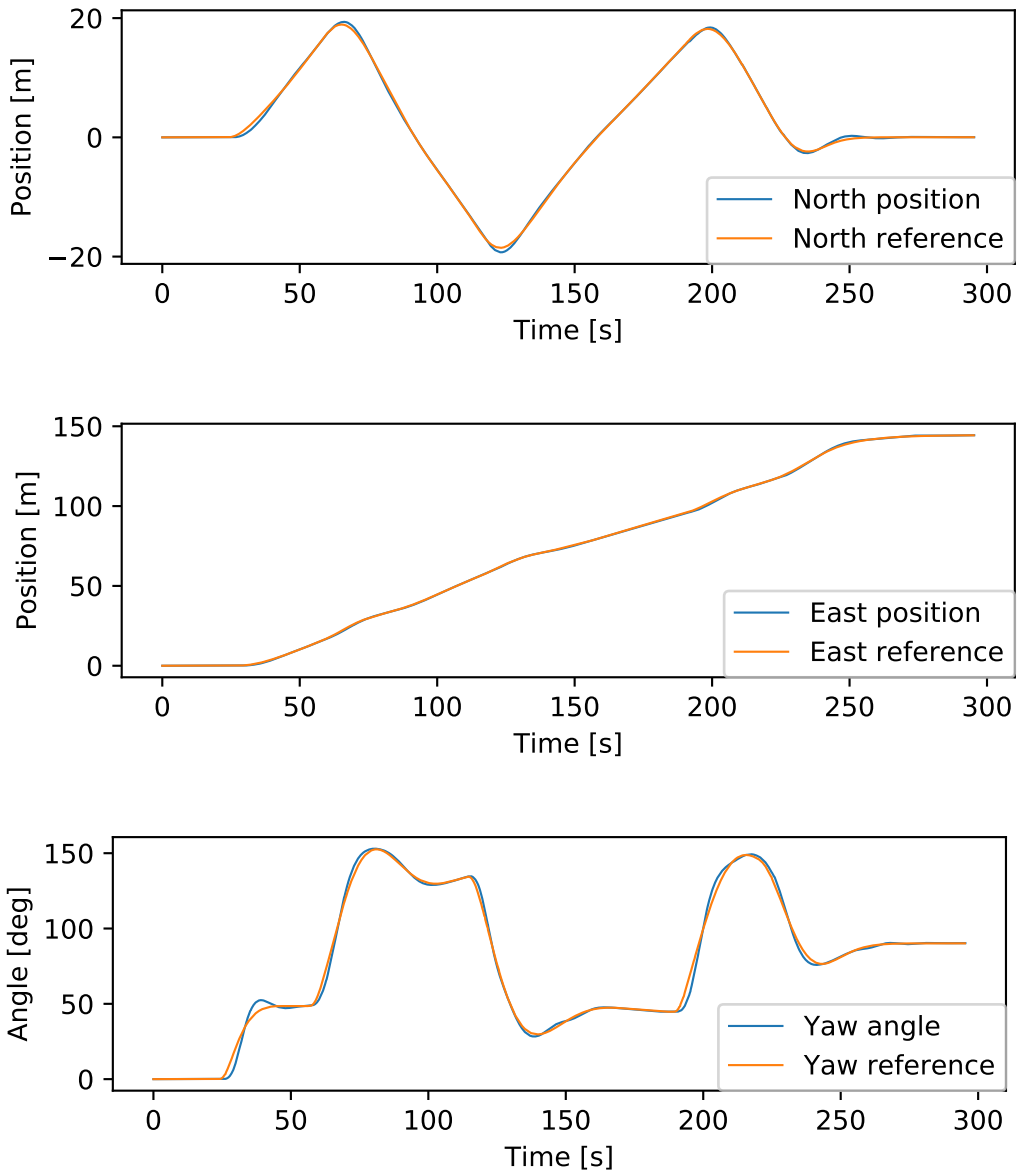


Figure 4.8: Simulation: Desired and actual position with variable desired velocity

velocity set by the operator. This is achieved while not knowing where the next waypoint is before the acceptance value of the current waypoint is reached. However, the response could be quicker and the overshoots smaller. To address these issues, variable look-ahead distance could be implemented with adaptive LOS, the acceptance value could change based on what the desired velocity is, and the reference filters could be replace by another reference model.

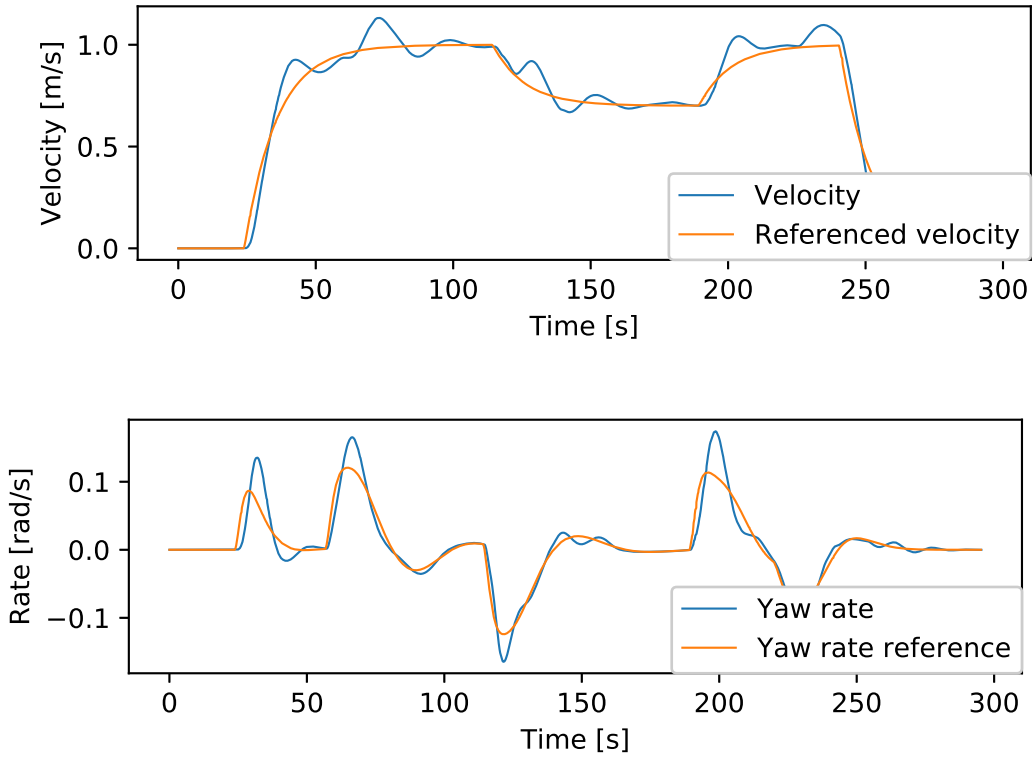


Figure 4.9: Simulation: Desired and actual velocity and turn rate with constant desired velocity

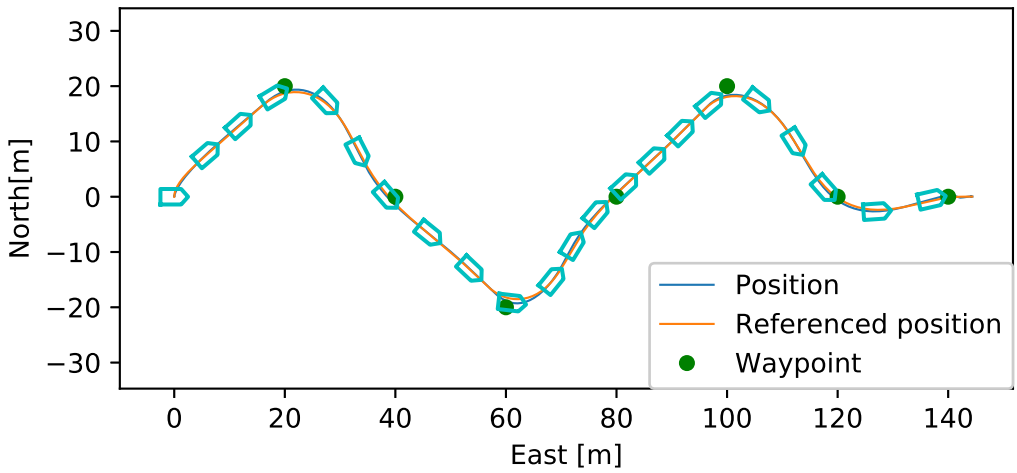


Figure 4.10: Simulation: Trajectory with variable desired velocity

4.3 Results and Discussion of Sea Trial

Following the promising results in the simulation study, full-scale tests with the autonomous marine vessel MilliAmpère were conducted. The tests was carried out close to "Hurtigbåtterminalen", Trondheim.

Table 4.2 shows the observations done of the test environment. Especially the wind estimate should be noted, as this created a significant environmental force on the ship. The weather was otherwise clear and sunny.

Observation	**Comment**
Temperature	$6°C$
Wind estimate	$5m/s$
Current estimate	$<0{,}5m/s$
Waves	small
Date	20.05.2020
Start time	10:20
End time	11:15

Table 4.2: Observations of test environment

More details about the test vessel, setup and results are found in Appendix B.

4.3.1 Test 1: Simple Path

First the vessel was tested on a straight simple path with no need for turning. Figure 4.11 shows the measured and referenced positions and yaw angle. It is seen that the produced references are smooth, and are followed closely by the vessel. The flexible guidance system is initialised after approximately ten seconds, which sets the yaw reference to the measured yaw angle.

The initial angle is approximately -114 °, and the range goes from -180° to 180°. There are some offsets between the referenced and measured yaw angle, as the measured angle oscillates around the reference. This is most likely due to MilliAmpère being slightly directional unstable.

Considering the significant wind present, it seems that the strategy of using DP to counteract the environmental forces instead of compensating for that in the guidance system is successful.

Figure 4.12 shows the measured and referenced total velocity and yaw rate. It should be noted that the velocity is not measured directly but rather estimated from the position signals. The desired velocity of $1m/s$ is slowly approached, before the decrease to $0m/s$. It is seen that there is a short delay from the referenced velocity increases or decreases, until the measured velocity follows. This can have many reasons e.g. that the reference is too abrupt and steep at the beginning, or that the control system operates with position control and not velocity control which leads to a delay. The velocity measurements oscillate as this is unfiltered measurement data.

For the yaw rate, the fact that MilliAmpère is a bit directional unstable is seen again. However, the deviations are small and the oscillations are about the referenced rate which is smooth and without sharp turns after initialisation. It should be noted that as the velocity measurements the measured yaw rate is unfiltered.

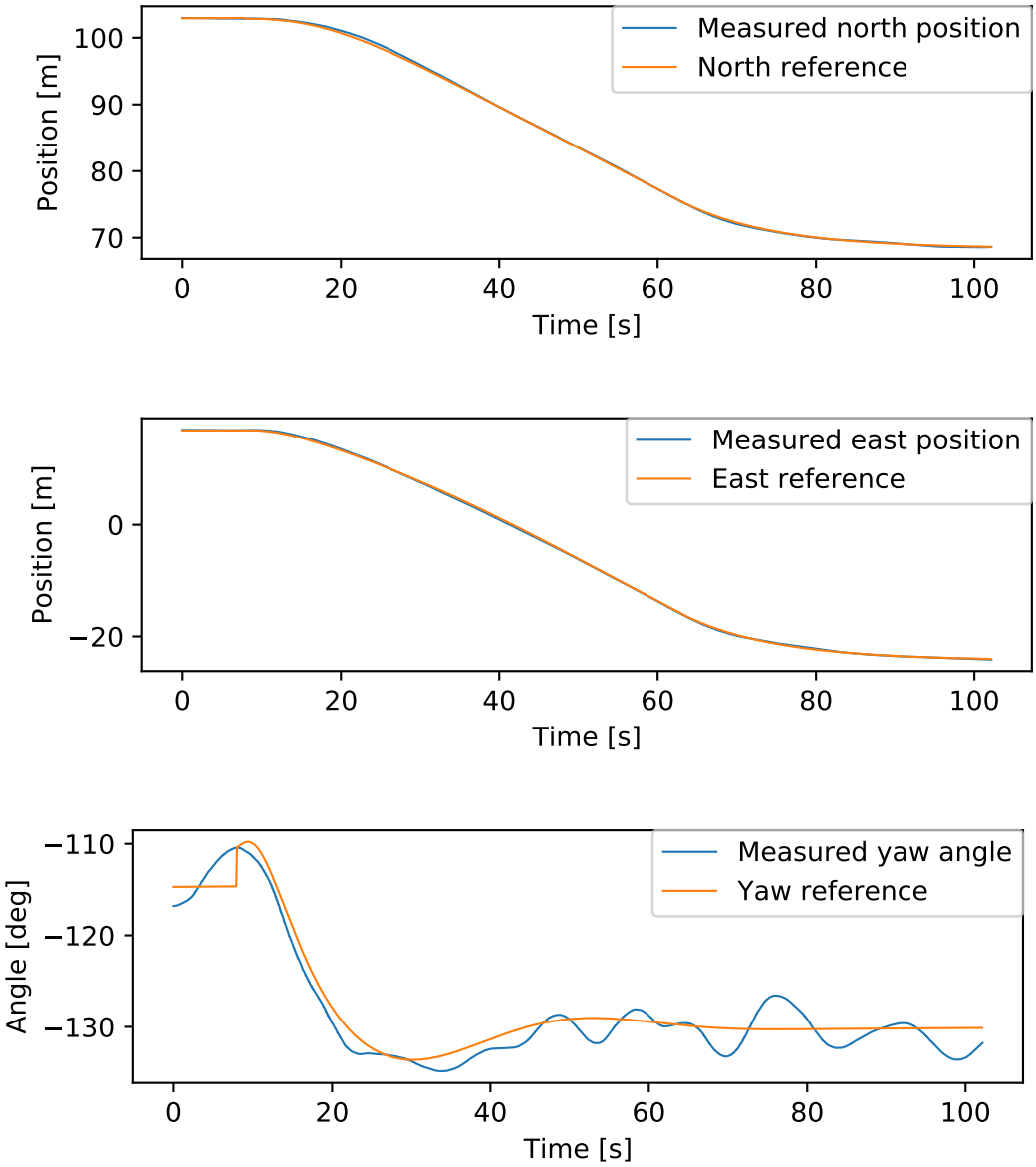


Figure 4.11: Full-scale: Desired and measured positions for straight line

4.3.2 Test 3: Zig-Zag - Variable Velocity

This test was to verify how the vessel manage turns and changes in desired velocity. The desired velocity start at $1m/s$ before first decreasing to $0.7m/s$ after waypoint 2 and then to $0.5m/s$ after waypoint 4. Figure 4.13 shows the generated trajectory and how the vessel follows the reference. The path is smooth, and the vessel follows easily without significant overshoots from the path. As seen in the simulations, also here the vessel turns before the waypoints are reached when the desired velocity is lower.

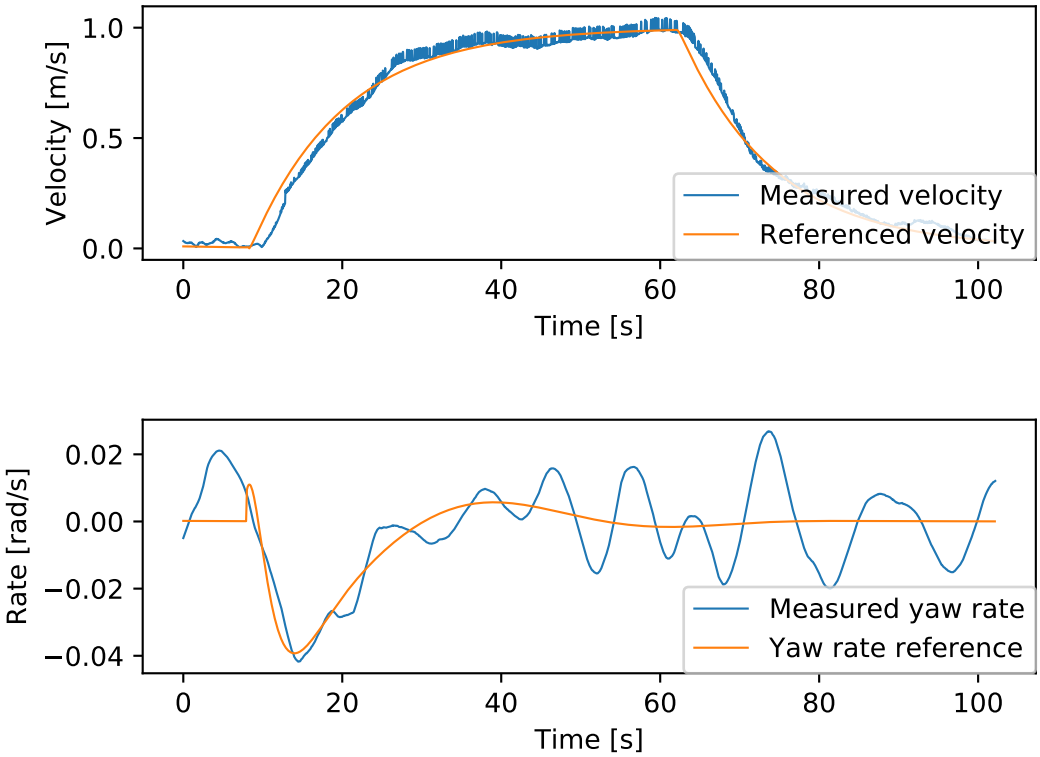


Figure 4.12: Full-scale: Desired and measured velocities for straight line

It is also seen that the first waypoint is disregarded. This is due to complications with transitioning between the DP with regular reference filter and the flexible guidance system. This should have been rectified, but due to COVID-19 which limited the test window significantly, this was not done. This complication also caused the rapid increase in velocity seen in Figure 4.15 which lead to a significant overshoot.

Figure 4.14 shows the measured and referenced yaw angle. The test is started after approximately ten seconds, the yaw reference is initialised to the actual yaw angle. From there on the produced reference is smooth and closely followed by the vessel.

Figure 4.15 shows the measured and referenced velocity and yaw rate. Beyond the abrupt change from $0m/s$ to approximately $0.7m/s$ before the velocity reference filter becomes active, the reference is smooth and followed nicely by the vessel. The reference seems to approach the decreased desired velocities smoothly and maintains the velocity. There is a wild point at approximately 300 seconds in the measured velocity. The vessel uses RTK and at this point the vessel went from floating to fixed RTK giving a correction in the measured position. Since the velocity measurements are derived directly from the position signals without signal processing this caused a wildpoint.

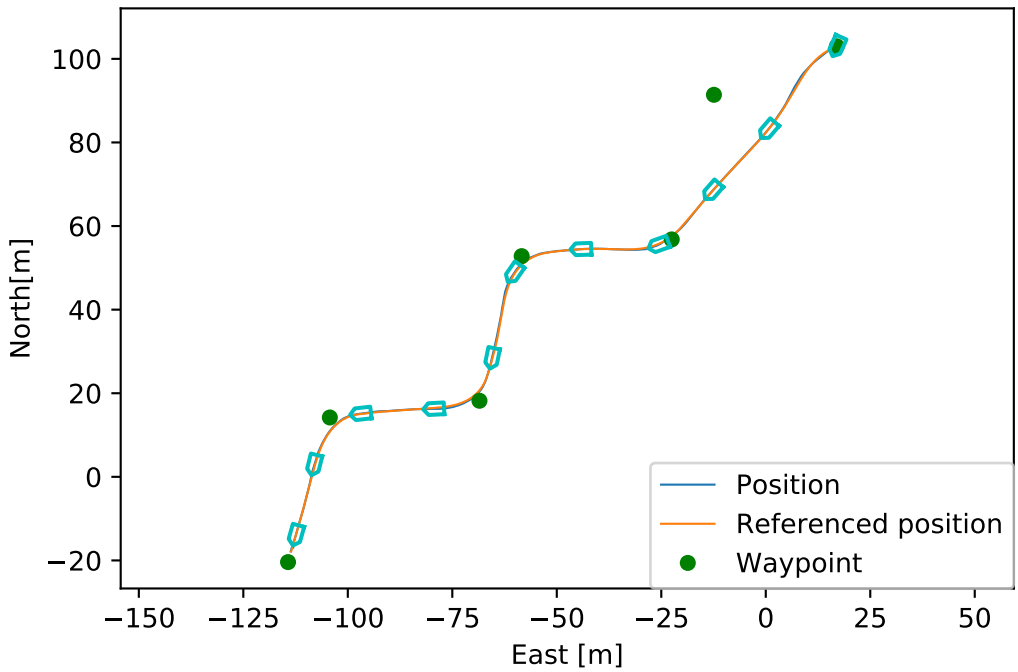


Figure 4.13: Full-scale: Trajectory for zig-zag test with variable desired velocity

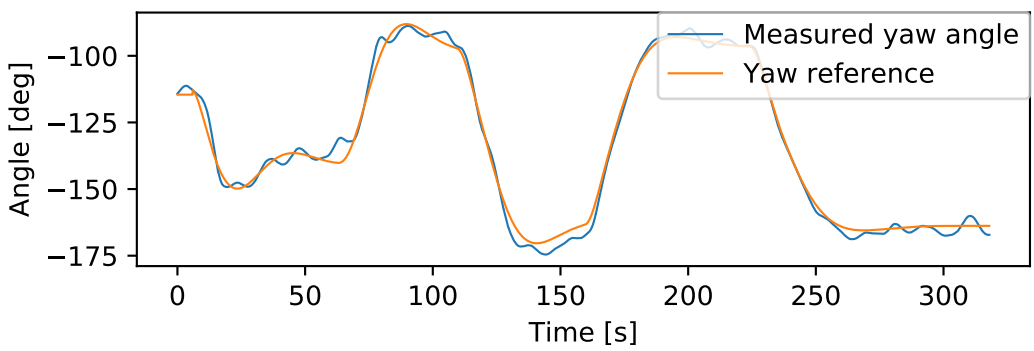


Figure 4.14: Full-scale: Desired and measured yaw angle for zig-zag test with variable desired velocity

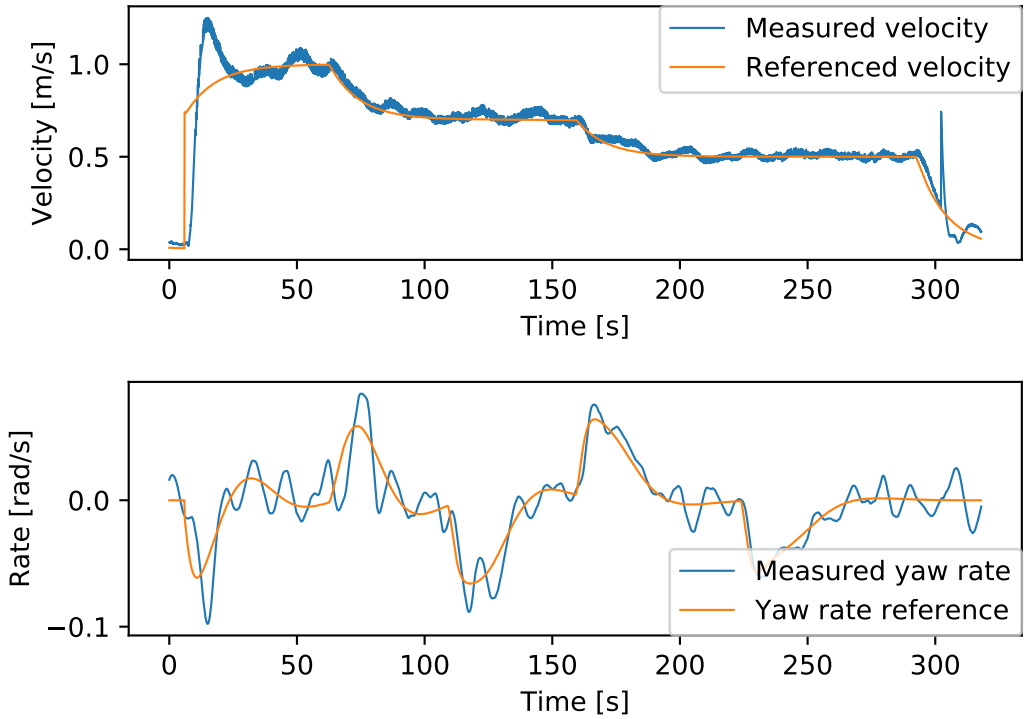


Figure 4.15: Full-scale: Desired and measured velocities for zig-zag test with variable desired velocity

The yaw rate reference is smooth and the vessel seems to follow it, though with the same oscillatory behaviour for the measurements as seen in the simple path test.

4.3.3 Test 4-7: Avoidance Manoeuvre

The point of test 4-7 were to test the vessels ability to do an evasive manoeuvre to avoid obstacles or other vessels. The manoeuvres deviated, 15°, 30°, 45° and 60° from the path. Also with these tests there were problems with the transitions. Hence only the avoidance test turning 15° reached the first waypoint before turning, while the other tests turn toward the deviating waypoint straight away. When the first waypoint falls out, the velocity reference filter takes some seconds before activating, giving some overshoot in the measured velocity compared to the reference. Even though it would have been interesting to see how the vessel turns away from the path, it is still interesting to investigate how it manages to resume the original path.

Figure 4.16 shows the trajectories for all the avoidance manoeuvres. As only the test where the vessel turn 15° go through the first waypoint, only the turn back to the third waypoint is discussed. It is seen that all the manoeuvres manage resume the path to reach the last waypoint. However, all the manoeuvres also overshoot when resuming the path, except for the manoeuvre with a 15° turn, but is limited to a maximum of 5 meters offset at the 60° turn. Considering that the vessel does not have massive overshoots and manage to resume the path, this guidance system could be applicable for avoidance manoeuvres.

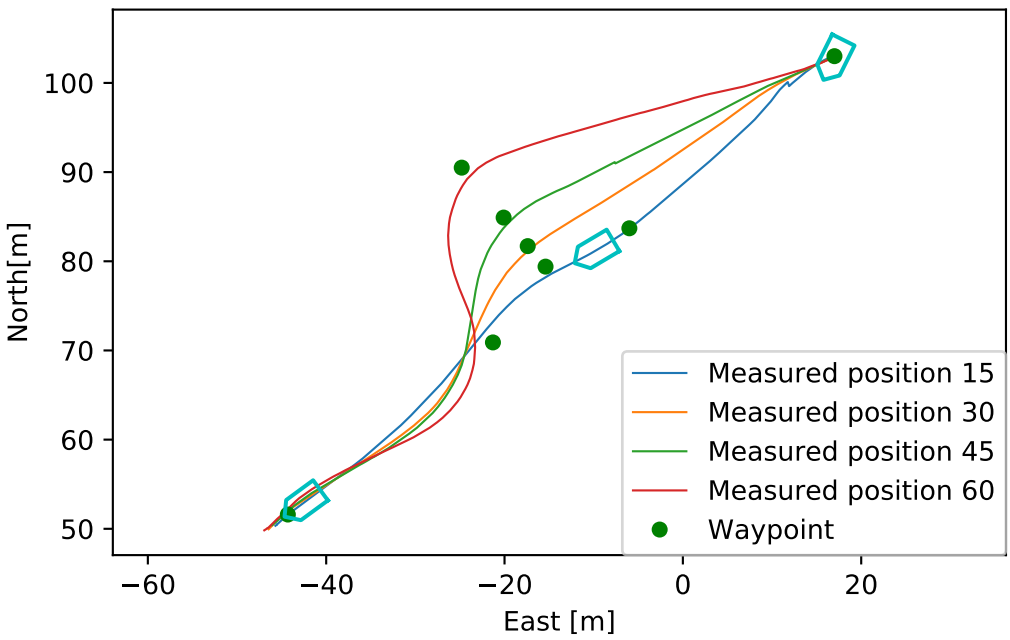


Figure 4.16: Full-scale: Trajectories for all avoidance manoeuvres

The sea trials have shown promising results of the guidance system and the performance in different scenarios. It has verified the results found in simulations and confirmed that the approach described in Appendix A can give good results in full-scale even with limit time to test and adjust.

Chapter 5

Autonomous Waypoint Navigation for MilliAmpère using Deep Reinforcement Learning

This approach with using machine learning as decision maker, is made possible due to the previously developed flexible guidance system described in Chapter 4. Remembering Figure 2.1 from Chapter 2, this method suggests to exchange the operator with a machine learning algorithm as shown in Figure 5.1. It is also assumed that the vessel is equipped with a functioning situational awareness system that can locate obstacles.

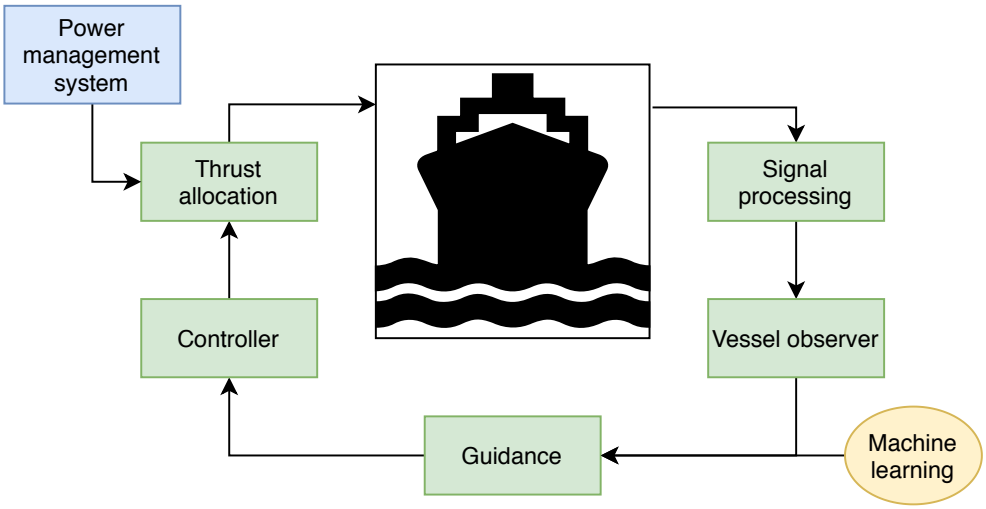


Figure 5.1: DP system where operator is substituted with machine learning

The method proposed in this chapter, utilising the guidance and DP system to execute actions chosen by a machine learning algorithm, has both pros and cons. The pros is that the vessel movements are stable, and the actuators are safely controlled. The

desired waypoints will be reached, as the DP system compensates for current and other environmental forces. This configuration also makes it easier to integrate it with other systems as collision avoidance as it is high-level control. The largest disadvantage is that every step has huge consequences. There are only limited leeway for bad actions as the vessel moves up 10 meters in east and north direction. This also makes the method more suitable where the distances are sufficient as opposed to fine manoeuvring. Another disadvantage is the reliability of the machine learning algorithm.

It will be investigated how far one can come with standard Q-learning, and if neural networks are reliable enough to be trusted. For this proposed approach, DDQN will be compared to standard Q-learning. The algorithms will be tested in a simulation study using two different environments with different level of risk. The simulations are carried out using the ROS model of the electric ferry MilliAmpère. However, before testing the ROS model has to be expanded to be applicable to deep reinforcement learning.

5.1 Method

To apply reinforcement learning the model needs an agent, and an environment. From the environment the agent gets information about the state and the received reward. The state is decided by observations which can be the position of the vessel or different sensor data. The agent then uses the observations to decide which actions to do next time. In this case, the vessel is the agent that is controlled with a machine learning algorithm. During testing, there will be different observations based on what is examined.

5.1.1 Expanding the Existing ROS Model

To expand the ROS model of MilliAmpère, the ROS framework has to be used. ROS is an open source software framework, which is used to control robots (Robotics, 2020). One model consists of different nodes communicating through topics. The nodes execute different tasks, as for MilliAmpère one node is the guidance or DP system. To give information to another node, data is published to a topic which the receiving node or nodes subscribe to. For the tests conducted in this case, the waypoint of the vessel will changed by the agent, while the existing model is responsible for controlling the vessel to the desired place. For the machine learning model to produce a waypoint, position data is required by the existing model. To achieve this relationship between the existing model and the machine learning expansion, a new node is constructed.

Figure 5.2 shows how the new node interact with the existing ROS model. The Milliampere_env node calls the machine learning algorithm. With positional information from the simulator, the algorithm decides what action to take, and the new node convert the action to a waypoint and informs the simulator.

To create a new node, the script, launch and configuration file have to be constructed. The launch file is needed to activate the node, while the configuration file define important variables and the necessary topics. In Figure 5.2 the relationship between the new node Milliampere_env and the existing model is shown. The existing ROS model is pictured as the blue ellipse. The topics are represented by rectangles, where the yellow are topics Milliampere_env publish to, while it subscribes to the blue topics. Of the topics, reset_position has to be implemented, while waypony, reset_integral and

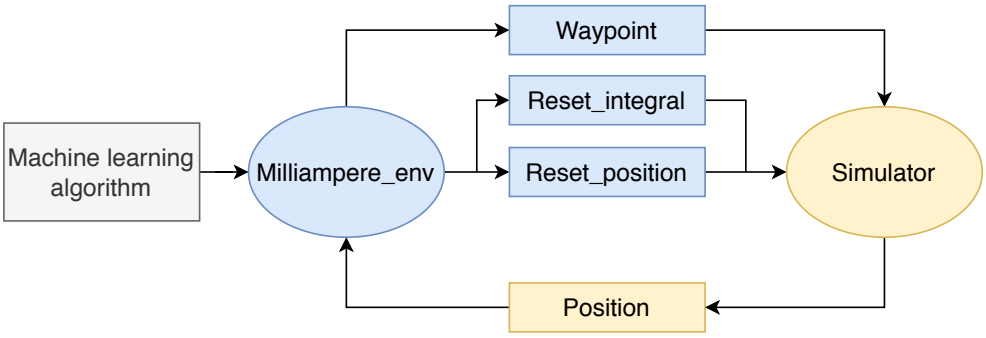


Figure 5.2: Flow chart with new node

positio already existed.

In the script file, the class MilliAmpere_env is defined. An object of this class is initialised with the initial position of the vessel, the range of actions that can be chosen, where the goal is, and more. The path of the topics are also defined to ensure that the node publishes and subscribes to the correct destinations.

The node is organised after the structure used by gym by openai (OpenAI, 2020). This include functions as reset, get_reward and the step function which is explained more thoroughly below.

To run several episodes in succession, the simulator has to be reset. This includes returning MilliAmpère and dynamic obstacles to a start position, set the waypoint to the initial position, and reset the integral gain of the controller. Since the goal position varies over the episodes this has to be decided as well.

The code of the reset function is given below. It was necessary to make the model wait 0.5 seconds between each time it publishes to a topic, to make sure the reset is done properly. Since the action depends on the previous action, the previous action is set to 0 to match the vessel heading.

```
def reset(self):
    #Reset position
    self.x_pos = 0
    self.y_pos = 0
    self.heading = 0

    #Reset way-point
    self.way.north = self.x_pos
    self.way.east = self.y_pos
    self.way.heading = self.heading

    #Reset last action
    self.prev_action = 0

    #Initilise distance to target
    self.prev_diff = [abs(self.goal[0]-self.eta[0]),...
                      abs(self.goal[1]-self.eta[1])]

    #Decide new goal location
```

```
    p = random.randrange(0, 3, 1)
        if p == 0:
            self.goal = [100, -40, 5]
        elif p == 1:
            self.goal = [100, 0, 5]
        elif p == 2:
            self.goal = [100, 40, 5]


    #Publish new values to simulator
    self.waypoint_pub.publish(self.way)
    rospy.sleep(0.5)
    self.reset_integral_pub.publish(True)
    rospy.sleep(0.5)
    self.reset_position_pub.publish(self.way)
    rospy.sleep(0.5)

    #Return current observation
    self.eta = self.eta.round()
    return [self.eta[0], self.eta[1],...
            self.goal[0], self.goal[1],0,0]
```

Only changes to the values in the class Environment_env is applicable, since the reset function is a member function. If there are values the rest of the ROS model need to be aware of, the values are published to the corresponding topic, hence, updating the values globally. Lastly the current observations are returned, as is convention in openai gym.

The reward function an important tool to achieve the desired behaviour. This is done by rewarding desirable behaviour and punish more risky behaviour. For this case the vessel should reach the goal as quickly as possible, without colliding or getting to close to an obstacle.

The reward function is as important for the DDQN as it is for Q-learning. Many different functions have been tested and to achieve desirable results, different reward functions have been used for the different cases. However, they all have in common the structure in Equation (5.1).

$$R = \begin{cases} R_T & \text{if } S = S_T \\ -R_T & \text{if } S \notin \mathcal{S} \ \vee \ \textit{collision} \\ f(S_t, S_{t-1}, A) & \text{if } S \in \mathcal{S} \end{cases} \tag{5.1}$$

R is the obtained reward, S_t is the current state, S_T is the terminal state, R_T is the reward received when an episode is ending, $\mathcal{S}$ is the state space, and $f(S_t, S_{t-1}, A)$ is a function calculating the reward when the terminal state is not reached, and the vessel is in the state space without hitting an obstacle. The function will penalise unnecessary turns, and reward if the vessel approach the goal.

An example of a used reward function is given below, for the first test with an obstacle and three different goal locations. For the vessel to behave as desired, a penalty will be given for manoeuvring out of the predefined space or moving away from the goal. The main functionality is:

- Rewards will be given when the goal is reached, and when the vessel moves towards the goal.
- The episode will terminate when the vessel is out of bounds, or when the goal is reached.
- To avoid the vessel from navigating slowly towards the goal to collect more rewards, the vessel is punished for moving away from the goal in any direction.
- A penalty of -0.5 is given if the vessel takes a turn, to avoid unpredictable navigation. This is done in the training script as the reward function does not posses information about which action is taken.

```
def get_reward(self):
    done = False
    reward = 0.5
    diff = [abs(self.goal[0]-self.eta[0]),...
            abs(self.goal[1]-self.eta[1])]
    if self.test_goal():
        done = True
        reward = 10
    elif self.test_collision():
        done = True
        reward = -10
    elif diff[0]>self.prev_diff[0] or ...
         diff[1]>self.prev_diff[1]:
        reward = -2

    self.prev_diff = diff
    return reward, done
```

At the end of the function the received reward is returned along with information to terminate or continue the episode.

To take into account that the vessel can collide or be to close to an obstacle during transition from one state to another, the step function also affects the resulting reward. For every meter the vessel moves, it is examined if it is too close to the obstacle. If it is, a penalty will be added.

The actual execution of the step, is done by the step function. The step function receives a desired action and executes it. It is convention that the step function also returns the new observation, received reward and if the episode should terminate or not. The main goal of the step function is to convert the action number to a specific action and take the vessel to a new state.

In order for the algorithm to know what its options are, specific actions have to be defined. Keeping in mind that the problem size increases by a factor equal to the action space, the number of possible actions are kept to a minimum. Therefore the vessel only has the option of turning left, right of continuing forward. This yields the action space [0, 1, 2], where 0 is left, 1 is forward and 2 is right. With this action space, the vessel moves at a constant velocity, and has no possibility to stop. The waypoint is then changed according to the action. MilliAmpère has both a DP and a guidance system that plans and navigates the vessel toward the waypoint at the desired velocity as described in Chapter 4.

```
def step(self,action):
    #Left
    if action == 0:
        action = self.prev_action - 1
        if action < 0 :
            action = 7
    #Forward
    elif action == 1:
        action = self.prev_action
    #Right
    elif action == 2:
        action = self.prev_action + 1
        if action > 7 :
            action = 0

    if (action == 0):
       self.way.north += 10
    elif (action == 1):
        self.way.north += 10
        self.way.east += 10
    elif (action == 2):
        self.way.east += 10
    elif (action == 3):
        self.way.east += 10
        self.way.north -= 10
    elif (action == 4):
        self.way.north -= 10
    elif (action == 5):
        self.way.north -= 10
        self.way.east -= 10
    elif (action == 6):
        self.way.east -= 10
    elif (action == 7):
        self.way.east -= 10
        self.way.north += 10

    self.waypoint_pub.publish(self.way)
        #While target position is not reached
        while (abs(self.s)>3.0) or i<5:
            if self.test_goal():
                break
            i = i + 1

    self.eta = self.eta.round()
    observation = [self.eta[0],self.eta[1],...
                   self.goal[0], self.goal[1],0,0]
    reward, done = self.get_reward()

    return observation, reward, done
```

Since the result of the chosen action depend on the previous action, the movement according to NED-frame has to be computed first. The actions in NED-frame are seen in Figure 5.3. A vessel with $\psi = 0$ is shown in the figure which means that action 7, 0 and 1 is available for this state. That the next action depends on the previous implies that the Markov property is not satisfied.

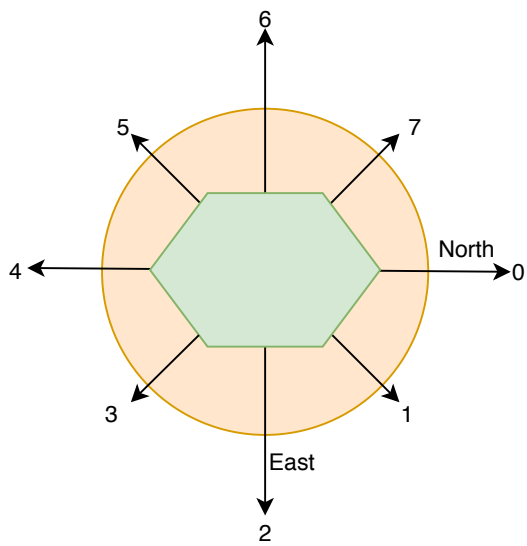


Figure 5.3: Actions in NED-frame

After OpenAI gym convention, the step function returns the observation, reward and if the episode is done or not. In order to obtain that information, get_reward() is called.

5.1.2 Implementing Q-learning

In this section, the Q-learn as the agent is defined. It is the agent that choose which action to execute. Q-learn is implemented as a class function with required member functions in the Milliampere_env node. Q-learn is dependant upon a discretized state and action space.

The states in this case will vary based on what information the vessel has available. For the first simple test, only the vessel position in north and east direction will be available. However, to show a simplified example of how the state space may look, it can be assumed that the only states are the positions in north and east direction as it is easy to visualise two dimensions. To use Q-learning the states have to be a discrete finite number, which is achieved by discretization of the space the vessel can manoeuvre in. To avoid struggles with huge dimensions, possible positions in north direction are [-10, 170] and [-60, 60] for east direction with dimension meter. Considering that one step for the vessel is ten meters, a spatial discretization of ten meters is used for both directions . This gives a total of states $18 \cdot 12 = 216$ possible states. Table 5.1 shows the distribution of the observations.

For the number of states to stay constant, the vessel is assumed not to be between states. As the state is required to be an integer, the DP system moves the vessel from its current location to the desired waypoint, and then it is assumed to be at

(-10,-60)	(0,-60)	(10,-60)	(20,-60)	(30,-60)	(40,-60)	···	(170,-60)
(-10,-50)	(0,-50)	(10,-50)	(20,-50)	(30,-50)	(40,-50)	···	(170,-50)
(-10,-40)	(0,-40)	(10,-40)	(20,-40)	(30,-40)	(40,-40)	···	(170,-40)
(-10,-30)	(0,-30)	(10,-30)	(20,-30)	(30,-30)	(40,-30)	···	(170,-30)
(-10,-20)	(0,-20)	(10,-20)	(20,-20)	(30,-20)	(40,-20)	···	(170,-20)
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
(-10,60)	(0,60)	(10,60)	(20,60)	(30,60)	(40,60)	···	(170,60)

Table 5.1: Discretised space

that exact location. The current state is calculated by Equation (5.2), where x is position in north-direction and y is position in east-direction divided by 10. When a new observation is added, the state is calculated by multiplying the previous amount of states with the amount of states the new observations has.

$$State = x + y \cdot 19 \tag{5.2}$$

This is the process of going from observations that the environment sends out, to a state. An example is that vessel has state 5 when in position [50, 0] and state 95 for position [0, 50]. Once the states are defined, Q-learning has to be implemented.

The class function that contains the actual reinforcement learning agent is called Qlearn. As the class is initialised, relevant learning parameters and variables are defined. It is designed after the theory in Section 3.3.

```
class Qlearn:
    def __init__(self, actions, epsilon, alpha, gamma, q):
        self.q = q                  # reward matrix
        self.epsilon = epsilon      # exploration constant
        self.alpha = alpha          # learning rate
        self.gamma = gamma          # discount factor
        self.actions = actions      # possible actions
```

Since a penalty is given for both unnecessary turns and steps that lead away from the goal, the discount factor is set to 0,95. This is to encourage the algorithm to reach the target quickly, while at the same time keeping the goal reward large enough for it to be important. Both the exploration constant and the learning rate where found through trial and error resulting with $\alpha = 0,1$ and $\epsilon = 0,1$. These parameters ensures that the state space is explored sufficiently enough for the algorithm to converge to an optimal solution.

5.1.3 Implementing Double Deep Q-learning

The implementation of DDQN will be slightly different than for the basic Q-learning algorithm. The possible actions are the same, but the states are stored in a neural network instead of a table. However, the training of the weights are done according do the same policy as for Q-learning.

The network is constructed using several different layers in sequence using Keras. Keras is an application programming interaface (API), and is a user-friendly and intuitive way to construct NN (Keras, 2020). Even though CNN has given promising

results for decision making, it is inefficient use of sensor data to first construct a visual representation of the environment it operates in, to then make a decision based on the visuals. Therefore the sensor data will be used directly as input to the algorithm. For the vessel to be able to understand the vessels own actions RNN is tried, making it Deep Recurrent Q-learning (DRQN). In order to get full benefit of transfer learning for the different tests, it is desirable to use the same network configuration for all the tests. The network configuration is decided using trial and error, as well as the hyperparameters given below.

```
class DQNAgent:
    def __init__(self, state_size, action_size):
        self.state_size = state_size        # observation size
        self.action_size = action_size      # number of actions
        self.memory = deque(maxlen=2000)    # length of replay
                                            # memory
        self.gamma = 0.95                   # discount rate
        self.epsilon = 1.0                  # exploration rate
                                            # start
        self.epsilon_min = 0.01             # minimum exploration
                                            # rate
        self.epsilon_decay = 0.995          # exploration
                                            # discount
        self.alpha = 0.001                  # learning rate
        self.learning_start = 500           # build up replay
                                            # memory
        self.batch_size = 64                # batch size to train
                                            # network
        self.model = self.build_model()
        self.target_model = self.build_model()
```

A discounted ϵ-greedy function will be used. It works the same as a ϵ-greedy except that it decreases with every step the agent takes until a minimum value is reached.

An example network is constructed in Keras below. ReLu is used as the activation function for the dense layers, as it has showed promising results. Also softmax is the activation function in the output layer as it is a multiclass classification problem. Simulations with LSTM layers were also attempted, and in order to make it converge the output activation function had to be changed to a linear activation function. Mean square error is used as the loss function and Adam is used as the optimiser, as these are the most commonly used.

```
def build_model(self):
        model = Sequential()
        model.add(SimpleRNN(70, input_shape=(1,6)))
        model.add(Dense(100,activation='relu'))
        model.add(Dense(100,activation='relu'))
        model.add(Dense(action_size, activation = 'softmax'))
        model.compile(loss='mse',...
                      optimizer=Adam(lr=learning_rate))
        return model
```

The deep Q-learning will benefit from the same simplifications as the Q-learning. In addition to that, the input should be processed to ease the training of the neural net and increase the chance for convergence.

The state space will be larger for DDQN than for Q-learning. The DDQN will observe the actual vessel and goal position, as the number of states is not as crucial, and it could potentially connect the goal location with the location of the vessel. For simplicity the positional data is rounded to the closest meter.

Normalisation is often used to make sure that a change in one state, is as significant as another regardless the value range the state operates in. Equation (5.3) is used to normalise the states.

$$normalised = \frac{data - min}{max - min} \tag{5.3}$$

data is the state value, *min* is the lowest value the state can have while *max* is the maximum state value. However, for this case all the states are positional data in the same reference frame, and therefor normalisation is disregarded.

5.1.4 Visualisation

To evaluate the algorithm and check that everything is working properly, it is expedient with a visualisation. For this purpose a simple 2D visualisation with North and East dimensions will be used. It is constructed using the Python package Pygame. Figure 5.4 shows how it looks. It is easy to alter the obstacles and update the image to see the vessel movements.

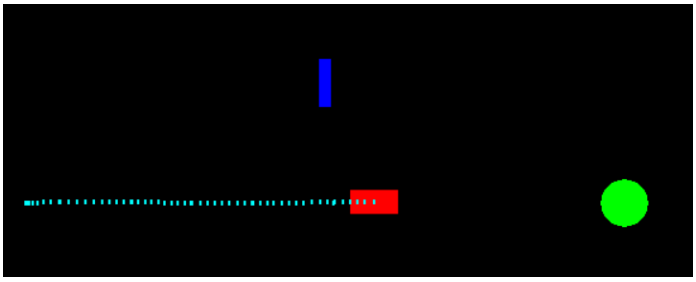
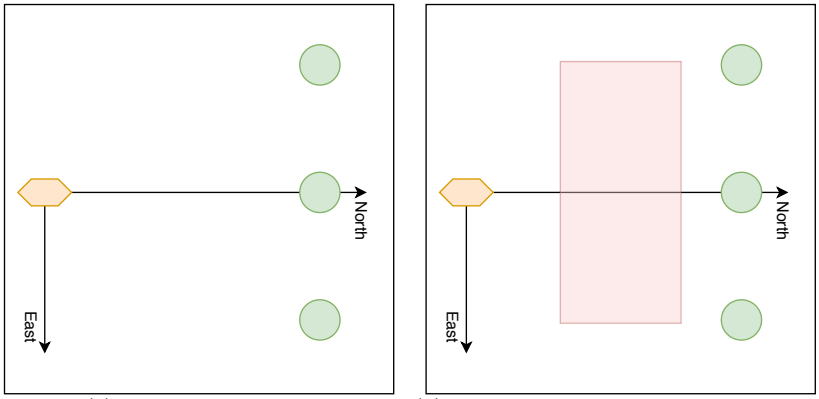


Figure 5.4: Pygame visualisation

The red rectangle is the vessel, and the green circle is the goal location. The obstacle is represented by the blue rectangle and the path is tracked by the light blue spots.

5.1.5 Obstacles and Environmental Difficulty

The risk the vessel is subject to is different depending on what environment the vessel operates in and how autonomous the vessel is as described in Section 1.2. For this case, the vessel will be defined as a highly autonomous vessel. All decisions will be taken by the vessel, and it is expected to manoeuvre safely from start to goal without human intervention. This requires the vessel to manage occurring challenges by itself, as for example avoid an obstacle. The mission complexity is quite low, as the operation only require the vessel to move from A to B without hitting an obstacle.



(a) Tree different goals

(b) Three different goals and obstacle present

Figure 5.5: Different environmental difficulty

When it comes to the complexity of the operation, this will vary. The human indecency is constant, in the way that it is fully independent. The environmental difficulty will be limited, as environmental forces will be neglected and the scenarios will be constrained to those in Figure 5.5. It is obvious that the environmental difficulty increases when the obstacle is introduced.

5.1.6 Training the Model

The algorithms have to be trained in order to work efficiently. It requires long time to run episodes on the simulation model of MilliAmpère as it is online. Therefore offline training will be done first in Jupyter Notebook. This decrease training time significantly, as the vessel position can be moved instantly to the waypoint instead of waiting for the vessel to get there. The work flow is the same here as with the guidance system, and described in Appendix A.

The maximum number of steps per episode, influence how long the training takes, as well as how much the algorithm get to explore. As less than 10 steps are required to reach the goal, 100 is used as a ceiling value. This gives room for exploration, but avoid unnecessary time spent on the vessel for example going circle.

When the algorithm has shown satisfactory results offline, it will be evaluated online. The online verification is done using the ROS model of MilliAmpère. This simulator is special made for MilliAmpère, and should have the same behaviour as the full-scale vessel. If the testing online shows promising results, the code can be run directly on the actual vessel as well.

For Q-learning the training and testing will be very similar, as it is limited to the defined states. This means that the algorithm will observe where it is supposed to be after a step, instead of the actual position. Therefore it is expected to perform as well in the online simulator as during the offline training. For DDQN the situation is different. It will observe the actual position of the vessel, which can be different than those observed in training. Therefore some training online may be necessary as well for DDQN.

5.2 Results and Discussion

In this section Q-learning and DDQN will be trained and tested, to evaluate the performance of the proposed method for navigation. To remember some key features of the algorithms while reviewing the results, they are summarised in Table 5.2.

Algorithm	**States**	**Markov property**	**Black box**	**Transfer learning**
Q-learning	Limited	Required	No	No
DDQN	Unlimited	Required	Yes	Yes
DDQN with RNN	Unlimited	Not Required	Yes	Yes

Table 5.2: Key features of the machine learning algorithms

States refer to the number of states the algorithm can handle, Markov property is if it is required for the method to guarantee a global optimal solution with unlimited training data. The black box column state if it is realistic to understand why the algorithm takes the decisions it does and the transfer learning column is if the method in theory should be applicable for transfer learning.

5.2.1 Find One of Three Goals

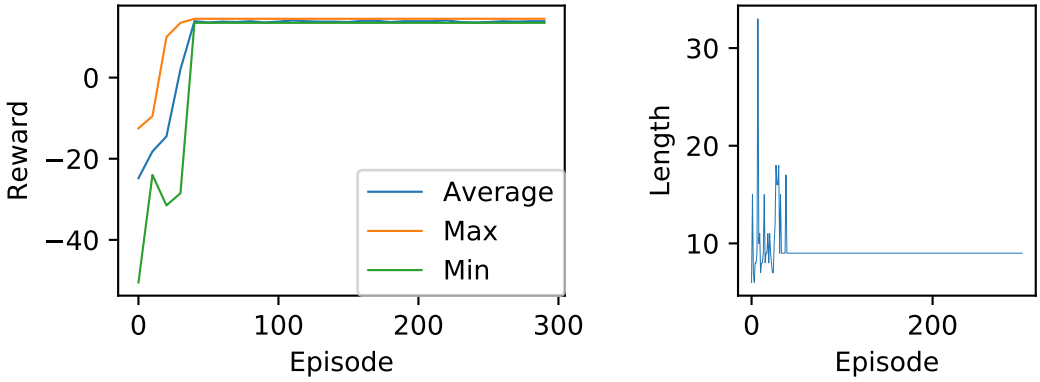
For the first challenge, the vessel should navigate to one of three goal locations. The vessel starts in (0,0) with zero heading. The goal locations are (100,-40),(100,0) and (100,40). Since transfer learning is used for the networks, this test is the basis for everything else. To increase the chance of this test being successful, three different networks structures will be tested. A successful training includes reaching the goal as fast as possible and without any unnecessary turns.

Q-learning will observe the vessel waypoints, as an approximate position, and a number stating which goal is active. This is done to keep the number of states to a minimum, as the algorithm has no ability to benefit from knowing where the goal actually is. Hence, the state space consists of three values. The total number of states are given in Equation (5.4).

$$n_{states} = 19 \cdot 13 \cdot 3 = 741 \tag{5.4}$$

19, and 13 are the amount of achievable positions in north and east direction respectively, while there are three different goal locations.

The results of the offline training for Q-learning are shown in Figure 5.6. Figure 5.6(a) shows the average, maximum an minimum received reward for every 10 episode. Figure 5.6(b) shows the length of each episode, which is how many steps the algorithm needs to find the goal or go out of bounds per episode. For the first episodes before convergence, the length varies between under five to over 30 steps. This is normal while the algorithm search for an optimal solution. It is seen that the algorithm converges rapidly, and the result is that nine steps is executed in each episode to reach the goal. The average reward varies between the minimum and maximum after convergence, due to different maximum rewards depending on where the goal is. This is because a high or low goal position require turning, while the goal in the center does not.



(a) Episode reward

(b) Episode Length

Figure 5.6: First test offline training: Three different goals with Q-learning

For the online testing the algorithm will still observe the waypoint positions instead of the vessels actual position, because Q-learning requires a fixed number of states. This make the online testing very similar to the offline training, and hence very similar behaviour as in offline training should be expected. Figure 5.7 is a screen shot of the visualisation created during online testing. For illustrative purposes, the inactive goals have been outlined. The light blue dots indicate all the positions the vessel has possessed during testing. It shows the paths that the algorithm took with no random actions. There are three different paths, one for each goal location. It is easily seen that the algorithm is deterministic, and chooses the same path towards each of the three goals every time. However, the globally optimal solution is not found, as the vessel takes two turns to reach the target. The best solution is when the turn to the bottom and top goal is done at the right time, giving only on turn.

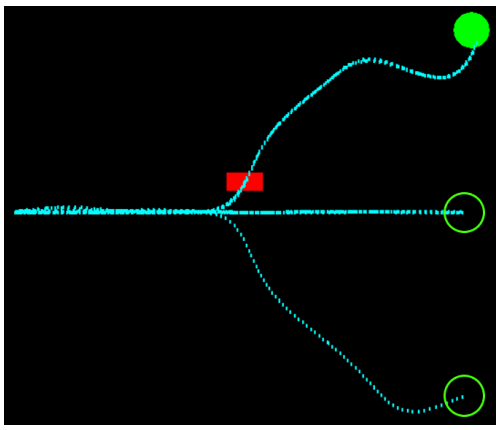


Figure 5.7: First test online: Three different goals with Q-learning

It should also be noticed that the vessel has some overshoot from the waypoints, caused by the guidance system. This is because the waypoints are close to each other. By enhancing the guidance system or increasing the distance moved between the waypoints, the overshoot could be reduced. In this application with machine

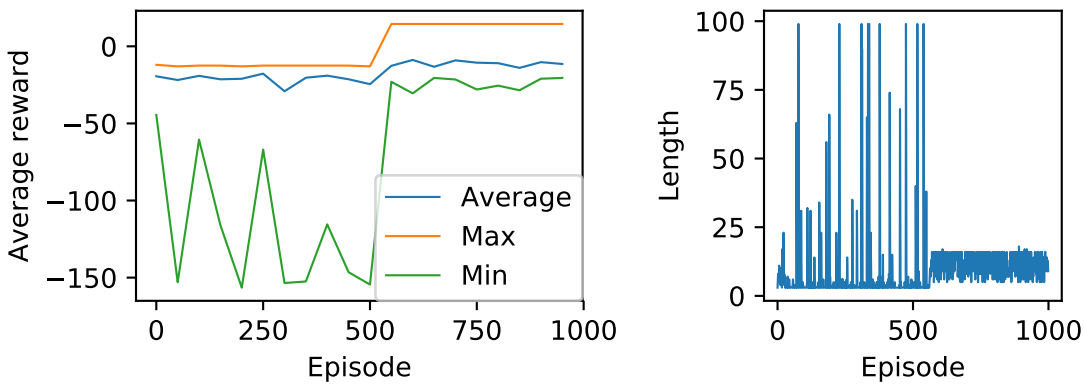
learning, it was considered important to keep the distance between waypoints as small as possible, to maintain manoeuvrability.

A consequence of the heading not being directly controlled through the waypoint, is that the vessel may approach the goal from different angles. This is seen in Figure 5.7 where the algorithm has converged to reaching the goal from below to the top goal while the bottom goal is approached head on. Both solutions contain only two turns, making them both optimal solutions, though probably not the paths a human would take.

Q-learning managed to converge and obtain a descent result. Next, the DDQN algorithm will be tested. For DDQN the vessel will observe two states that are constantly zero in addition to the vessel position and goal location. This is to make the network the same size as when an obstacle is observed, to facilitate transfer learning. This yields a total of six states as shown in Equation (5.5).

$$Observation = [x_{vessel},\ y_{vessel},\ x_{goal},\ y_{goal},\ 0,\ 0] \tag{5.5}$$

x_{vessel} and y_{vessel} are the positions of the vessel in north and east direction respectively, while x_{goal} and y_{goal} are the goal location.



(a) Episode reward

(b) Episode length

Figure 5.8: First test offline training: Three different goals with Dense network

Figure 5.8 shows the results of the offline training for a network without recurrent layers, with the structure in Figure B.1(a). Also here the average, maximum and minimum reward is plotted in Figure 5.8(a), but as the algorithm requires more episodes to converge, the values are calculated for every 50 episodes. It is seen that the network has a higher variance in received reward as the distance between maximum and minimum reward is much higher than for Q-learning. It seems the average reward is closer to the minimum than the maximum reward, indicating that the goal is reached less than half of the episodes. Figure 5.8(b) shows the length of each reward. It is seen that the algorithm converges after approximately 500 episodes which is slower than Q-learning, but still quite quick. While Q-learning had zero variance after convergence, DDQN has some variety in the episode length. This may be due to the episodes where the goal is not found, and that the path towards the different goals has a different length or amount of turns. Another explanation could be that the paths the algorithm

has converged to, are not robust. Meaning that when a random action is taken, it does not manage to resume to the initial plan.

In addition to the dense network, the LSTM network given in Figure B.1(b) and a network similar to the dense, except that the first layer is substituted by a SimpleRNN layer were tested.

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 100)	700
dense_2 (Dense)	(None, 100)	10100
dense_3 (Dense)	(None, 100)	10100
dense_4 (Dense)	(None, 3)	303

Total params: 21,203
Trainable params: 21,203
Non-trainable params: 0

(a) Dense network

Layer (type)	Output Shape	Param #
lstm_1 (LSTM)	(None, 1, 16)	1344
lstm_2 (LSTM)	(None, 1, 16)	2112
lstm_3 (LSTM)	(None, 16)	2112
dense_1 (Dense)	(None, 100)	1700
dense_2 (Dense)	(None, 20)	2020
dense_3 (Dense)	(None, 3)	63

Total params: 9,351
Trainable params: 9,351
Non-trainable params: 0

(b) LSTM

Figure 5.9: Network configurations for Dense network and LSTM network

It seems that all the networks give approximately the same performance, with a high variance in both reward and length of episodes. The Dense network with a recurrent layer requires more episodes to converge than the pure dense network, while the LSTM network needs the least amount of episodes. This may be due to the LSTM network having half the amount of trainable parameters as the other networks. Even though all networks converge, it is uncertain if the behaviour is acceptable as the average reward is so low. It should be possible to reach the goal without going out of bounds almost every third run. This has to be tested in online simulations with the vessel.

For the online testing of the network, the algorithm will observe the actual position of the vessel instead of the waypoints. This means that the networks will be tested in how well they handle slight changes in the states, testing if the network can transfer the learned behaviour and the networks robustness. Figure 5.10 shows the resulting path when the network is run without random actions. In this figure, the low goal position shown in Figure 5.5(a) is active. The picture is taken such that only the lower half of the environment is showed, leaving the light blue path and the red vessel in the middle of the environment. All the networks showed the same result.

It is seen that the algorithm has converged to only taking the straight path straight

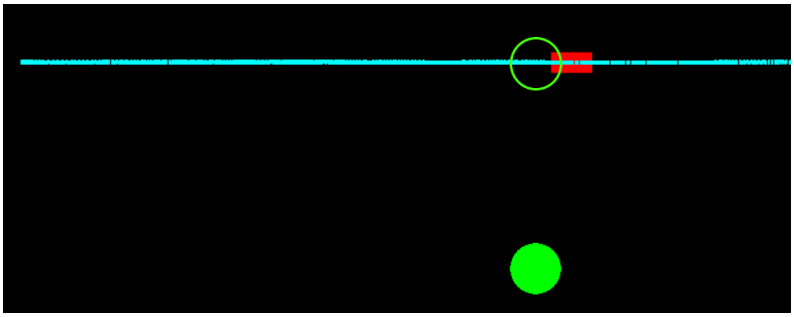
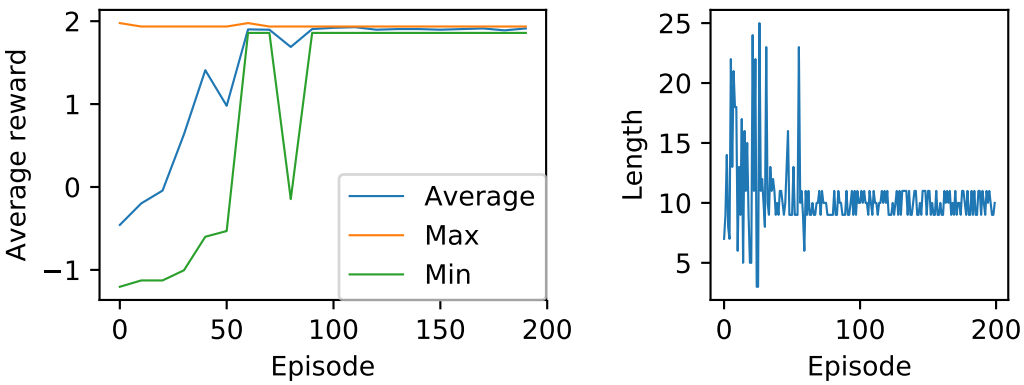


Figure 5.10: Path for networks

forward. This is showed by the light blue line that indicate the positions the vessel has possessed, which is dense and horizontal for all episodes regardless of where the goal location is. This is in accordance with Figure B.1(a) where the average is at about one third of the maximum value. On average the goal should be at the middle position every third episode. It is obvious that the same reward function for Q-learning and DDQN does not necessarily yield acceptable results. That the algorithm has converged, indicate that it has found a local maximum. This indication is strengthened since all the networks converge to the same behaviour. Once a local maximum is found, it is often difficult for an optimisation function to keep exploring to find the global maximum. Hence, the training will have to be executed differently in order for the algorithm to converge to a more satisfactory result.

As an attempt to achieve better results, the Dense network was trained with a different reward function and a slowly increasing distance to the goal. The method with increasing the distance to the goal is to ensure that the algorithm trains on many episodes where the goal is reached. This will encourage the algorithm to reach for that solution, as it knows it is better than crashing or always going straight forward. The reward function is given in Appendix C, and does not include a penalty for turning. This is to avoid algorithm being to restrictive to turn the vessel. The algorithm was trained in 15 rounds with the distance from the goal increasing between each round and the number of episodes varying between 200 to 2000 depending on how long it took to converge.

The results from the last offline training round is given in Figure 5.11.



(a) Episode reward

(b) Episode length

Figure 5.11: First test offline training: Three different goals with Dense network, alternative reward function and training regime

It is seen that the algorithm performs significantly better with the new reward function and training regime. The reward values between the different reward functions cannot be compared as the reward value has different ranges in the to functions. However, the relationship between average, maximum and minimum value can be compared. From Figure 5.11(a) it is seen that the algorithm finds a way towards each of the different goals every time, as the reward converges to straight under two for both the maximum, minimum and average reward.

Figure 5.11(b) shows that length of the episodes vary between 9 and 11 after con-

verging, depending on what goal position is active. The consequence of the different lengths is that the vessel movement is not straight towards the goal, but takes inefficient steps for some of the goal locations. Still, this is a better result than Figure 5.8, where every step was straight forward no matter where the goal was.

For the online training it is executed 100 episodes. Figure 5.12 shows the behaviour. It is seen that the algorithm once again converges to only going straight forward as the path in the middle is by far the thickest. That the algorithm does not turn back to the middle path after leaving it, indicate that the algorithm has concluded that the best action to take is going forward. The turns are due to random actions.

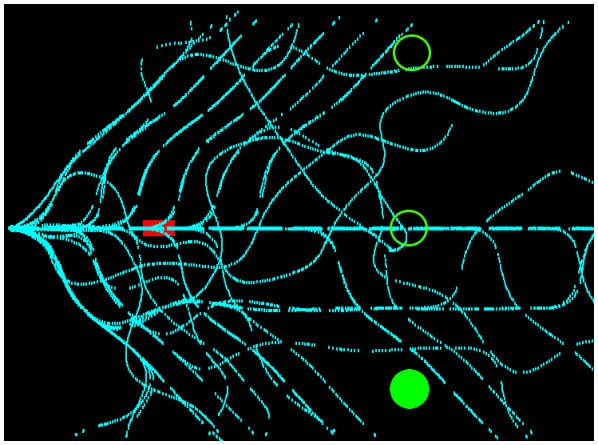


Figure 5.12: Online training Dense network with alternative reward function and training regime

That the results are much better offline than online, is most likely due to overfitting. The algorithm is not able to transfer the good training results to slightly different state values in the online environment. To avoid overfitting, different measures can be taken. This includes changing how the network is constructed, and how the training is done. There seems to be a fine balance between avoiding local optimums and adjusting the training so much to the situation that overfitting occurs.

A simulation with a more complex environment will be tested for both algorithms, even though the results for DDQN are unsatisfactory. This is both to see if the Q-learning algorithm can handle the increased amount of states and to check the offline performance of the Dense network with alternative reward and training procedure. As it is not valuable to see a vessel going straight ahead no matter what the situation is, the network will not be tested online.

5.2.2 Find One of Three Goals with Stationary Obstacle Present

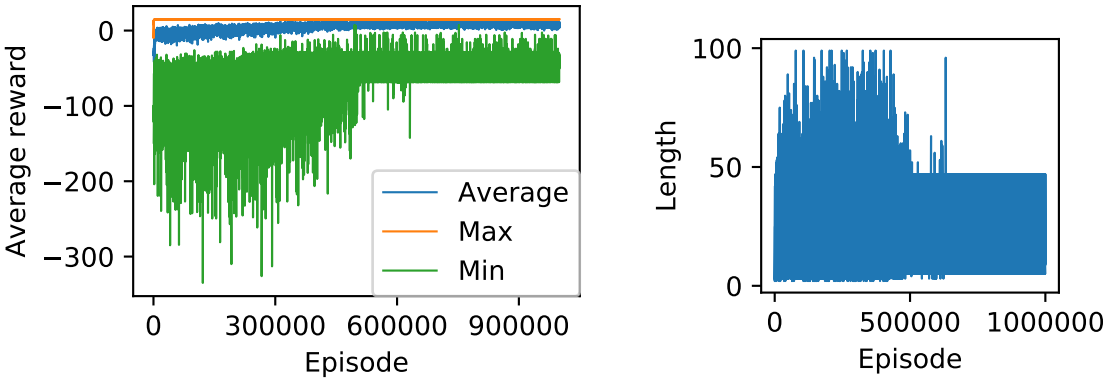
For this test, a stationary obstacle will be randomly placed in the area shown in Figure 5.5(b). The vessel still has to reach one of three goal locations, but should now also avoid getting closer than 10 meters to the obstacle. If the vessel is closer than 3 meters to the obstacle, it is regarded as a collision and the episode is terminated.

The obstacle is summoned within [30,70] in north direction and [-40,40] in east direction with a step interval of 5. The step interval is chosen assuming that a situational awareness system will be able to process Lidar data to localise an obstacle within a

failure radius of 2,5 meters. This is considered an acceptable uncertainty, as the vessel should keep 10 meters from the detected obstacle position at any time. It is assumed that the vessels situational awareness system discovers the obstacle at 50 meters distance. By multiplying the number of obstacle states with the existing states, the total number of states is achieved. This is done in Equation (5.6).

$$n_{states} = 741 \cdot (21 \cdot 17 + 1) = 265278 \tag{5.6}$$

21 and 17 is the number of locations in east and north direction while the extra state is for when the obstacle is undetected.



(a) Episode reward

(b) Episode length

Figure 5.13: Second test offline training: Three different goals with Q-learning and stationary obstacle

Figure 5.13 shows the results of the offline training for Q-learning. It is obvious that the algorithm has not found the optimal route for each scenario. For Q-learning to converge to an optimal solution it has to visit every state an infinite amount of times. With the high amount of states it is not able to to that. However, the average reward is quite high, so it seems to succeeds more than it fails and it manages to reach a level where most of the worst actions are avoided.

Considering that the algorithm only needed 100 episodes to converge for the first test, the need for training exploded with the appearance of the obstacle. After 500.000 episodes it converges to get a more stable result, but the variance is still high. The high variance may be due to the indeterminate states. In addition the algorithm may have problems since the problem does not satisfy the Markov property as described in Section 3.2.1. The obstacle force the vessel to do more turns, and the probability that the vessel reaches the same position from different directions increases. When different actions yield different results, Q-learning has no means to separate the situations. This could be solved by introducing the heading as a state, but this would drastically increase the amount of states by multiplying the existing states by eight. If it were set to train for several weeks, it could probably find good solutions for all cases, but that is not realistic.

The results of the online testing with Q-learning is shown in Figure 5.14. It seems that the algorithm suffers from *The curse of dimensionality* and acts almost random,

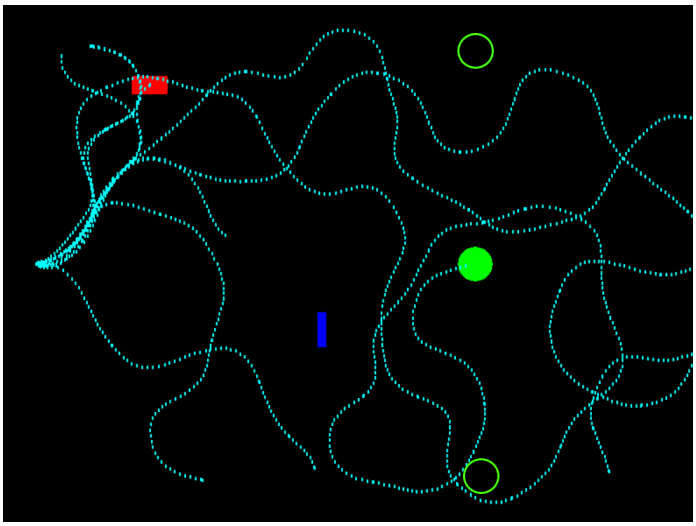
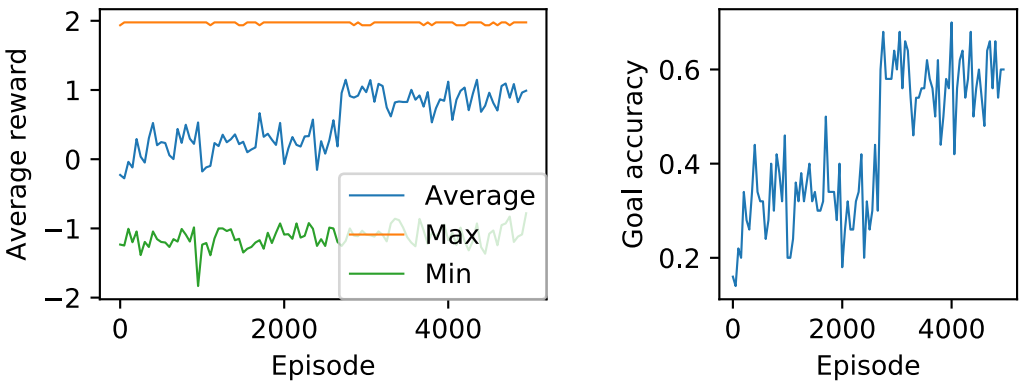


Figure 5.14: Second test online: Three different goals and obstacle with Q-learning

as the paths looks like a chaotic mess. It also has a tendency to collide in the upper left corner. During the 10 test runs it collided with the obstacle once, and reached the goal only once. Three times it went out of the state space in the upper left corner. This behaviour is not acceptable for an autonomous ship where a collision has high consequences for both the environment and human life.

DDQN has also been trained in this scenario. Figure 5.15 shows the results of the offline training with the dense network. It is clear that the tables have turned and that DDQN now converges much faster than Q-learning.



(a) Episode reward

(b) Accuracy

Figure 5.15: Second test offline training: Three different goals and obstacle with Dense network

Figure 5.15(a) shows the received reward with the alternative reward function. Comparing the received rewards, it seems that the average reward for Q-learning is closer to the maximum value than that of DDQN. Even though the average reward makes

Q-learning seem safer than DDQN, this is most likely not the case. In theory, DDQN has the ability to transfer known knowledge to new situations, and will therefor not act random when the obstacle is in a new position as Q-learning will. In a risk perspective Q-learning is therefore completely unreliable, considering it has not converged to an optimal action for all possible states. It would be irresponsible to use the algorithm, knowing it can take dangerous actions anytime.

Figure 5.15(b) shows that DDQN ensures that the goal is reached $\approx$ 60% of the episodes. It would be beneficial to know if the remaining episodes end with collision or simply slightly missing the target. Also DDQN with a dense network suffers from the fact that the process does not satisfy the Markov property. The dense network, in contrary to a recurrent network, has no memory of what happened in the previous steps. This cause the same action in the same states to yield a different output, as for Q-learning. This could be some of the cause of the low accuracy. However, it is not acceptable for a vessel to only guarantee safe voyage 60% of the time. No ferry passengers would accept that, and no kayaks would feel safe in waters where the ferry operates.

5.2.3 Advise for Further Work

Even though the approach tested in this case did not pull through, it does not mean that the concept of decision making with deep reinforcement learning for autonomous ships is unreachable. More research and testing are needed to conclude, and this section propose some possible approaches.

That the alternative dense network performs so much better in an offline training situation than online is an important observation. It questions how the training and verification of networks should be done, and emphasises the importance of realistic simulation environments. It is not sufficient to have good results in a test environment, if the results do not transfer to real-life situations. The algorithm masters well the scenarios it has been trained on, so the training situation has to be as close to the real situation as possible. Still, it has to be able to handle new scenarios as well for the vessel to be regarded safe. To increase the reliability of the algorithm one could conduct a hazard identification. Then the results can be used as a starting point for designing the reward function and training regime. However, black swans are still a risk. Even if the algorithm where to reach 90% accuracy, that would not be acceptable if the 10% lead to collisions, dangerous situations or the algorithm does not manage to transfer simulation results to real-life situations. If close to 100% accuracy is not achievable, it indicates that neural networks as the sole decision maker is unacceptable. To provide guiding on this topic DNV GL and other class societies should define clear demands and test regimes to verify the use of machine learning on autonomous ship.

Integrating the navigation system with an online risk management system could be beneficial. One large disadvantage with this approach that cannot be changed, is the need for a sufficient length between the waypoints for the guidance system to follow. This cause every step the algorithm takes to have a large impact, due to the distance between the waypoints. Therefore the algorithm has to be certain what the best action is, and if it does not, an online risk management system should be activated.

Chapter 6

Conclusions and Further Work

This chapter concludes the work done in this master thesis, and suggests further work.

6.1 Conclusions

In this master thesis two research questions have been investigated through theoretic development, model expansions, simulation studies and a full-scale test.

The first research question that was examined is *"How to develop a guidance system with configurable velocity for waypoint tracking, preparing for a machine learning decision maker?"* The proposed approach was to use a machine learning algorithm to set the waypoints as the vessel progresses. This created a need for a customised guidance, navigation and control system. A ROS model of the over-actuated vessel MilliAmpère was used, and it was already implemented with a DP system. However, the guidance system was a simple position reference filter, developed for station-keeping. Therefore a new guidance system which combines LOS with reference filters was developed. This resulted in a flexible guidance system that only needs set points in north and east directions, since the heading reference is calculated by the guidance system using LOS. In addition it maintained the desired velocity, not stopping at the waypoints, and gave the possibility to vary the velocity. The next waypoint was needed only once the previous one was reached, implying that they can be changed during the run.

To verify that the guidance system managed to keep a desired velocity and reach the waypoints, an offline simulation test was executed. As the offline simulations showed promising results a simulation study was conducted on the ROS model to investigate if the same behaviour was maintained. The simulation study showed that the guidance system generates references as expected. It guided the vessel through the waypoints at the desired velocity. However, the overshoot in yaw angle was at approximately 5 degrees, and the reference filter made the transition from one velocity to another slow in addition to slow turns. When the next waypoint caused the vessel to turn, the vessel needed some time to converge to the LOS angle.

As a final test, to check the robustness of the guidance system in a real life environment, a full-scale test was conducted. The sea trial confirmed the results from

simulation, which is promising.

The next step towards using machine learning as a decision maker, once the guidance system was developed and tested, was to adapt the vessel model. This lead to research question two: *"How to use machine learning to define waypoints as input for guidance and navigational systems?"* For the ROS model of MilliAmpère to be applicable for machine learning, it had to be expanded. The machine learning algorithm would set the waypoints, and therefore needed the relevant information to choose an action. After an action was chosen, the algorithm has to be able to communicate the waypoint to the vessel model. The expansion was structured in the same way as used by gym by openai. In addition to altering the ROS model, two different algorithms were implemented: Q-learning and DDQN.

To analyse the proposed approach, a simulation study was conducted. Both Q-learning and DDQN was trained offline before they were tested with the ROS model of MilliAmpère. The vessel was to reach a goal which was in one of three positions. A more challenging test was also conducted, where a stationary obstacle spawned at a random position within a predefined area. Q-learning showed to master the simple test well, but suffered from *The curse of dimensionality* when the stationary obstacle was implemented. DDQN converged to a local optimum for the simple test, where it choose to go straight forward no matter where the goal was. This resulted in reaching the goal at approximately every third episode. To try to circumvent this problem, another reward function and training regime were implemented. This lead to good results offline, but the results were not transferable to online testing which implies overfitting. DDQN was also trained offline in the advanced environment, which showed an accuracy of approximately 60%.

The last research question is *"Is the proposed application of machine learning reliable for autonomous ships?"* In this thesis the results are not good enough to conclude that the method is reliable. 60% accuracy is not near high enough to ensure passengers that the voyage is safe, when remaining 40% could lead to collisions, or transfer to the wrong location. The proposed method did not perform acceptable in this case study. Q-learning could not handle the amount of states, and DDQN did not yield good results online and had too low accuracy after offline training. Suggestions for further work on this method have been presented and is summarised in the next section.

6.2 Further Work

This project has proposed a method for decision making for autonomous vessels using machine learning, including a new guidance law. In order to improve the guidance system itself, there are several changes that could be done. The simulations and sea trials gave some indications of what future work could be done:

- Adaptive look-ahead distance
- Adaptive acceptance value
- Change the reference model
- Improve integration and initialisation for sea trials

Where both the adaptive look-ahead distance and adaptive acceptance value, are to accommodate different velocities, while a quicker reference model would give quicker turns and faster transitions between different velocities. If these suggestions were to be implemented successfully, it would result in a guidance system that makes rapid

enough turns to keep the course, manages to change velocities quick and is adequately adapted for different velocity setpoints. For the guidance system to be a permanent part of an autonomous system, also the integration and initialisation have to be improved.

To use deep reinforcement learning as a decision maker with more success, one should consider the following:

- Either use a recurrent network or change the action space to make the process satisfy the Markov property. If the recurrent network is chosen, the algorithm should also be able to handle dynamic obstacles, while a dense network still would be limited to stationary obstacles.
- Construct a case study that has appropriate distances for the guidance system.
- Conduct a training regime for the algorithm that is adapted to the reward function. If the rewards are sparse, consider training with increasing distance to the goal to encourage the algorithm to reach the goal.
- Implement the network with means to avoid overfitting.
- Use machine learning algorithms as an integrated part of a larger system. Including collision avoidance, and online risk handling. If this is not possible, it could be beneficial to reduce the level of autonomy.
- Consider using a different machine learning algorithm to investigate if they yield better results.

These are some suggestions for future work to improve the performance of the method.

Bibliography

Alpaydın, E. (2010). *Introduction to Machine Learning Second edition*. Adaptive Computation and Machine Learning. The MIT press.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P. F., Schulman, J., & Mané, D. (2016). Concrete problems in AI safety. *CoRR*, *abs/1606.06565*.

Aven, T. (2012). The risk concept—historical and recent development trends. *Reliability Engineering System Safety*, *99*, 33–44.

Aven, T. (2013). Practical implications of the new risk perspectives. *Reliability Engineering System Safety*, *115*, 136–145.

Beard, R. W., & McLain, T. W. (2012). *Small unmanned aircraft : theory and practice*. Princeton University Press.

Borhaug, E., Pavlov, A., & Pettersen, K. (2008). Integral los control for path following of underactuated marine surface vessels in the presence of constant ocean currents. In *2008 47th IEEE Conference on Decision and Control*, (pp. 4984 – 4991).

Breivik, M., & Fossen, T. (2009). *Guidance Laws for Autonomous Underwater Vehicles*, chap. 4. IntechOpen.

Caharija, W., Candeloro, M., Pettersen, K. Y., & Sørensen, A. J. (2012). Relative velocity control and integral los for path following of underactuated surface vessels. In *IFAC Proceedings Volumes*, vol. 45, (pp. 380 – 385). 9th IFAC Conference on Manoeuvring and Control of Marine Craft.

Caharija, W., Pettersen, K. Y., Bibuli, M., Calado, P., Zereik, E., Braga, J., Gravdahl, J. T., Sørensen, A. J., Milovanović, M., & Bruzzone, G. (2016). Integral line-of-sight guidance and control of underactuated marine vehicles: Theory, simulations, and experiments. *IEEE Transactions on Control Systems Technology*, *24*(5), 1623–1642.

Dey, R., & Salem, F. M. (2017). Gate-variants of gated recurrent unit (GRU) neural networks. *CoRR*, *abs/1701.05923*.

DNV GL (2019). Trustworthy industrial ai systems. https://www.dnvgl.com/publications/trustworthy-industrial-ai-systems-164957. Accessed: 2020-26-03.

Fei-Fei, L., Deng, J., Russakovsky, O., Berg, A., & Li, K. (2016). Imagenet. http://www.image-net.org/. Accessed: 2020-20-03.

Figueiredo, J. M. P., & Rejaili, R. P. A. (2018). Deep reinforcement learning algorithms for ship navigation in restricted waters. *Mecatrone*, *3*(2), 1–10.

Fossen, T., Breivik, M., & Skjetne, R. (2003). Line-of-sight path following of underactuated marine craft. In *IFAC Proceedings Volumes*, vol. 36, (pp. 211–216).

Fossen, T. I. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control*. Chichester, UK: John Wiley & Sons, Ltd.

Gamrian, S., & Goldberg, Y. (2018). Transfer learning for related reinforcement learning tasks via image-to-image translation. *CoRR*, *abs/1806.07377*.

Giel, A., & Diaz, R. (2015). Recurrent neural networks and transfer learning for action recognition.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Gupta, P., Malhotra, P., Vig, L., & Shroff, G. (2018). Transfer learning for clinical time series analysis using recurrent neural networks. *CoRR*, *abs/1807.01705*.

Gurney, K. (1997). *An introduction to neural networks*. UCL Press.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., & Davidson, J. (2019). Learning latent dynamics for planning from pixels. *International Conference on Machine Learning (ICML)*.

Hasselt, H. (2010). Double q-learning. In *Advances in Neural Information Processing Systems 23*, (pp. 2613–2621). Curran Associates, Inc.

Hasselt, H., Guez, A., & Silver, D. (2015). Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, (pp. 2094–2100). AAAI press.

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

Huang, H.-M. (2006). The autonomy levels for unmanned systems (alfus) framework interim results. *National Institute of Standards and Technology*.

IMO (1972). Convention on the international regulations for preventing collisions at sea,(colregs). http://www.imo.org/en/About/conventions/listofconventions/pages/colreg.aspx. Accessed: 2020-06-05.

IMO (2018). IMO takes first steps to address autonomous ships. `http://www.imo.org/en/MediaCentre/PressBriefings/Pages/08-MSC-99-MASS-scoping.aspx`. Accessed: 2019-11-19.

Insaurralde, C. C., & Lane, D. M. (2012). Autonomy-assessment criteria for underwater vehicles. *IEEE/OES Autonomous Underwater Vehicles (AUV)*.

Isherwood, R. M. (1972). Wind resistance of merchant ships. *The Royal Institution of Naval Architects*, *115*, 327–338.

ISO (2019). Ships and marine technology - Terminology related to automation of Maritime Autonomous Surface Ships (MASS). *Manuscript in preparation*.

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

Kaplan, S., & Garrick, J. (1981). On the quantitative definition of risk. *Risk Analysis*, *1*(1), 11–27.

Kendall, A., Grimes, M., & Cipolla, R. (2015). Posenet: A convolutional network for real-time 6-dof camera relocalization. In *2015 IEEE International Conference on Computer Vision (ICCV)*, (pp. 2938–2946).

Keras (2020). Keras: The python deep learning library. https://keras.io. Accessed: 2020-17-03.

Khan, A., Sohail, A., Zahoora, U., & Qureshi, A. S. (2019). A survey of the recent architectures of deep convolutional neural networks. `https://arxiv.org/ftp/arxiv/papers/1901/1901.06032.pdf`. Accessed: 2019-10-07.

Khan, A., Zhang, C., Atanasov, N., Karydis, K., Lee, D. D., & Kumar, V. (2017). End-to-end navigation in unknown environments using neural networks. *CoRR*, *abs/1707.07385*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Commun. ACM*, *60*(6), 84–90.

Kufoalor, D. K. M., Wilthil, E. F., Hagen, I., Brekke, E. F., & Johansen, T. A. (2019). Autonomous colregs-compliant decision making using maritime radar tracking and model predictive control. *2019 18th European Control Conference (ECC)*, (pp. 2536–2542).

Le Pham, T., Layek, A., Vien, N., & Chung, T. (2017). Deep reinforcement learning algorithms for steering an underactuated ship. (pp. 602–607).

Lei, X., Zhang, Z., & Dong, P. (2018). Dynamic path planning of unknown environment based on deep reinforcement learning. *Journal of Robotics*, *2018*.

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

March, J. G. (1991). Exploration and exploitation in organizational learning. *Organization Science*, *2*(1), 71–87.

MIT (2015). places. http://places.csail.mit.edu/. Accessed: 2020-20-03.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. A. (2013). Playing atari with deep reinforcement learning. *CoRR*, *abs/1312.5602*.

NIST (2008). Autonomy levels for unmanned systems (alfus) framework. *Volume I: Terminology, Version 2.0*.

NOAA (2012). Lidar 101: An introduction to lidar technology, data, and applications. `https://coast.noaa.gov/data/digitalcoast/pdf/lidar-101.pdf`. Accessed: 2019-12-05.

NTNU (2019). Autoferry. `https://www.ntnu.edu/autoferry`. Accessed: 2019-11-19.

OpenAI (2020). Gym. `https://gym.openai.com/`. Accessed: 2020-02-12.

Otterlo, M., & Wiering, M. (2012). Reinforcement learning and markov decision processes. *Reinforcement Learning: State of the Art*, (pp. 3–42).

Peng, J. (2018). *Understanding of the Convolutional Neural Networks with Relative Learning Algorithms*. 3rd International Conference on Electromechanical Control Technology and Transportation (ICECTT 2018).

Robotics, O. (2020). Ros. `https://www.ros.org/`. Accessed: 2020-02-12.

Rødseth, Ø. J., & Nordahl, H. (2017). Definitions for autonomous merchant ships. *Norwegian forum for autonomous ships (NFAS)*.

Rødseth, (2018). Defining ship autonomy by characteristic factors. *International Conference on Maritime Autonomous Surface Ships*, (pp. 19–26).

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

SNAME (1950). *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.

Sutton, R., & Barto, A. (2015). *Reinforcement Learning: An Introduction*. The MIT Press.

Sørensen, A. J. (2011). A survey of dynamic positioning control systems. *Annual Reviews in Control*, *35*(1), 123 – 136.
URL `http://www.sciencedirect.com/science/article/pii/S1367578811000095`

Sørensen, A. J. (2018). Marine cybernetics : Towards autonomous marine operations and systems : Lecture notes.

Taleb, N. N. (2007). *The Black Swan: The Impact of the Highly Improbable*. Random House Group.

Thyri, E. (2019). *A Path-Velocity Decomposition Approach to Collision Avoidance for Autonomous Passenger Ferries*. Master's thesis, NTNU.

Torben, T. (2019). *Control allocation and observer design for autonomous ferries*. Master's thesis, NTNU.

Utne, I. B., Sørensen, A. J., & Schjølberg, I. (2017). Risk management of autonomous marine systems and operations. *Proceedings of the 36th International Conference on Ocean, Offshore & ArcticEngineering*.

Watkins, C. J. C. H. (1989). Learning from delayed reward. *PhD Thesis, University of Cambridge*.

Watkins, C. J. C. H., & Dayan, P. (1992). Technical note q-learning. *Machine learning*, *8*, 279–292.

Widrow, B., & Hoff, M. (1960). Adaptive switching circuits. In *1960 IRE WESCON Convention Record, Part IV*, (pp. 96–104). IRE.

Zhang, X., Wang, C., Liu, Y., & Chen, X. (2019). Decision-making for the autonomous navigation of maritime autonomous surface ships based on scene division and deep reinforcement learning. *Sensors*, *19*, 4055.

Appendix A

Research Method

The work with developing the guidance system and the use of machine learning as decision maker has been an iterative process. Figure A.1 illustrates the work flow. The colours range from green, through orange to red which defines different phases in the process.

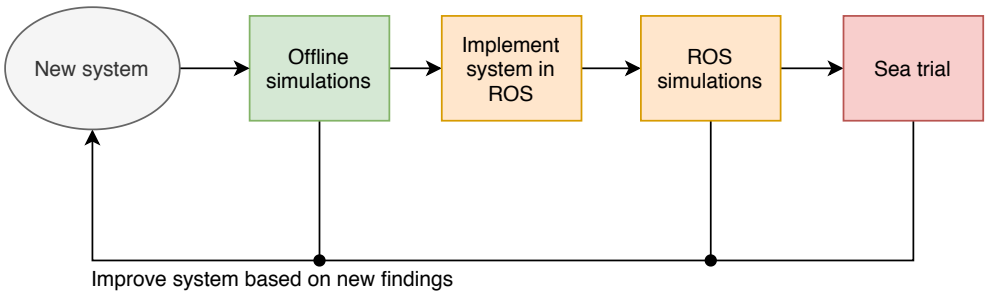


Figure A.1: Iterative process

Firstly an idea is developed into a new system. This is then tested with offline simulations, to verify that it behaves as expected. If the offline simulations uncover flaws such that improvements are needed, alterations are done to the system before testing again. Once the offline simulations shows promising results, effort is done to implement it on the ROS model of MilliAmpère. The ROS model of MilliAmpère is an exact copy of the full-scale ferry MilliAmpère, with the same systems implemented. This means that once the system is implemented in the ROS model, it is simple to implement it on the vessel. The ROS simulations provide a safe and controlled environment to perform realistic testing of the system. Since the system is the same as one the vessel, compatibility with the other systems on the vessel is tested simultaneously as the performance of the new system giving fewer surprises when conducting a sea trial.

Once both the offline simulations and the ROS simulations give acceptable results, a sea trial can be conducted. The new ROS model, is simply uploaded to the vessel and is ready for testing. A sea trial will reveal if the system is robust enough for a real-life

situation, as sensor data can be noisy, and environmental conditions are affecting the vessel.

After the sea trial, the performance of the system is evaluated again giving three different stages of testing:

- **Offline simulation:** Verify that the behaviour is as expected and desired
- **ROS simulation:** Check how the system interacts with the vessel system and the performance with the limitations posed by the vessel
- **Sea trial:** Test the system in a real-life situation

This is a beneficial way to work, as most of the challenges and errors of the system can be identified early in the process. Sea trials are both time consuming and expensive, and this method enables that as much debugging as possible before sea trial.

Appendix B

Test Setup and Additional Results Sea Trial

In this Appendix the test setup and vessel will be described. In addition some plots not included in Section 4.3 is showed here.

B.1 Test Vessel

The full-scale testing will be conducted on MilliAmpère, which is a small passenger ferry. It is equipped with two azimuth thrusters, mounted along the ship longitudinal axis as described in Section 2.4. Key features of the vessel are shown in Table B.1.

Parameter	**Value**
Hull length	5.0m
Hull width	2.8m
Displacement	1667kg
Number of thrusters	2
Max thrust	500.7N
Max azimuth turn rate	$30\frac{deg}{s}$
$L_{1,x}$	-1.8m
$L_{2,x}$	1.8m

Table B.1: MilliAmpère specifications

Here, $L_{1,x}$ and $L_{2,x}$ are the azimuth position along the x-axis. See also Figure 2.3. The following systems are currently implemented on MilliAmpère:

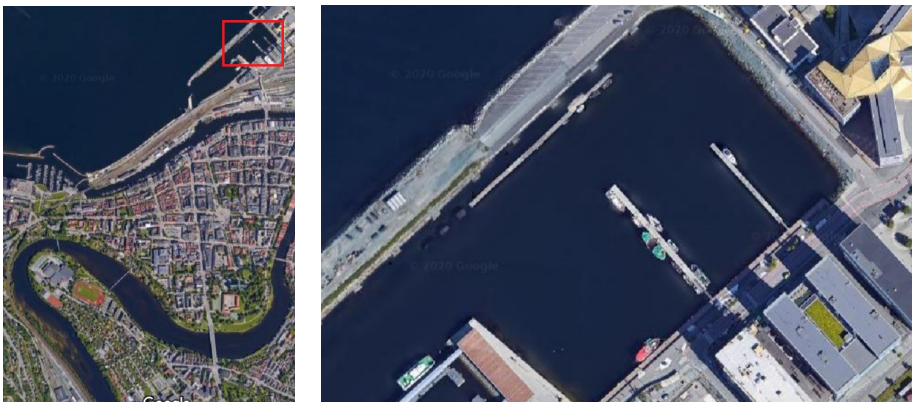
- The DP system described in Section 2.2.2
- The thrust allocation briefly introduced in Section 2.4
- The guidance system developed earlier in Chapter 2.5.

In addition MilliAmpère is equipped with a GNSS receiver, IMU with orientation measurement, radar and Lidar.

B.2 Test Setup

As the simulation study was conducted using a ROS model of MilliAmpère, the code can be run directly on the passenger ferry MilliAmpère. This simplifies the process of doing full-scale testing, and is the main reason that the code has been implemented in ROS.

Figure B.1 shows the area in Trondheim where the test was carried out. The tests were conducted close to the pier, sheltered from the roughest waves, yet still exposed to wind.



(a) Trondheim (b) Test area

Figure B.1: Overview of the test area

Origin of the NED frame is placed at the pier next to the green vessel in Figure B.1 at (63.4389029083N, 10.39908278E).

Different tests were executed, to test the applicability of the guidance system. The test configurations are given in Table B.2.

WP_n	vel_d	vel_{end}	Δ	*acceptance*	*angle*
Test 1: Simple path					
1	$1m/s$	$0m/s$	7,5m	8	
Test 2: Curvy path					
2	$1m/s$	$0m/s$	7.5m	8	
Test 3: Curvy path - variable velocity					
2	0.5m/s - 1m/s	$0m/s$	7.5m	8	
Test 4: Avoidance manoeuvre: angle 1					
3	$1m/s$	$0m/s$	7.5m	8	15°
Test 5: Avoidance manoeuvre: angle 2					
3	$1m/s$	$0m/s$	7.5m	8	30°
Test 6: Avoidance manoeuvre: angle 3					
3	$1m/s$	$0m/s$	7.5m	8	45°
Test 7: Avoidance manoeuvre: angle 4					
3	$1m/s$	$0m/s$	7.5m	8	60°

Table B.2: Test matrix

Between the tests, the vessel was driven by direct actuator control before switching to DP with a regular position reference filter to obtain the desired start position (103, 17) with the heading at -2 radians. It was necessary to implement transitions between the DP with regular position reference filter and the flexible guidance system to be able to conduct the tests efficiently. Hence, all the tests were conducted consecutively.

Test 1 followed the waypoints in Table B.3.

Waypoint	**0**	**1**
North [m]	103	70,9
East [m]	17	-21,3

Table B.3: Waypoints for simple path

Table B.4 shows the waypoints for test 2-3. The curvy path was constructed with deviations of 10 meters in each direction from the the straight line through the coordinates (103,17),(-10,-113).

Waypoint	**0**	**1**	**2**	**3**	**4**	**5**	**6**
North [m]	103	91,4	56,8	52,8	18,2	14,2	-20,4
East [m]	17	-12,4	-22,5	-58,4	-68,5	-104,3	-114,3

Table B.4: Waypoints for curvy path

Test 4-7 followed the waypoints in Table B.5. The tests are avoidance manoeuvres with 15°-60° turn from a straight path.

Waypoint	**1**	**2**	**3**	**4**	**5**
North [m]	103	83,7	79,4	70,9	51,6
East [m]	17	-6	[-15,4,-24,8]	-21,3	-44,3

Table B.5: Waypoints for avoidance manoeuvre

B.3 Additional Test Results

Here, plots for north and east direction, trajectory yaw angle, yaw rate and total velocity is given for tests 1-3 where these is not included in Section 4.3. For tests 4-7 plots for yaw will be included as these are the most relevant.

B.3.1 Test 1: Simple Path

Figure B.2 shows the trajectory for test 1.

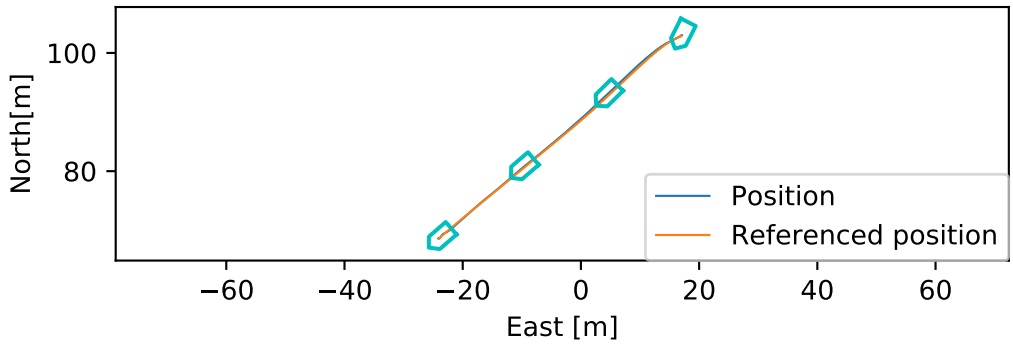


Figure B.2: Full-scale: Trajectory for straight line

B.3.2 Test 2: Curvy Path

In Figure B.3-B.5 the results from test 2 is shown.

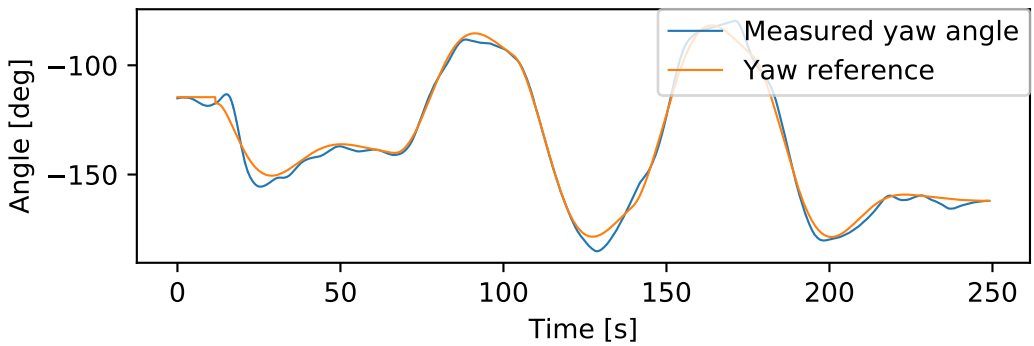


Figure B.3: Full-scale: Desired and measured positions for curvy path

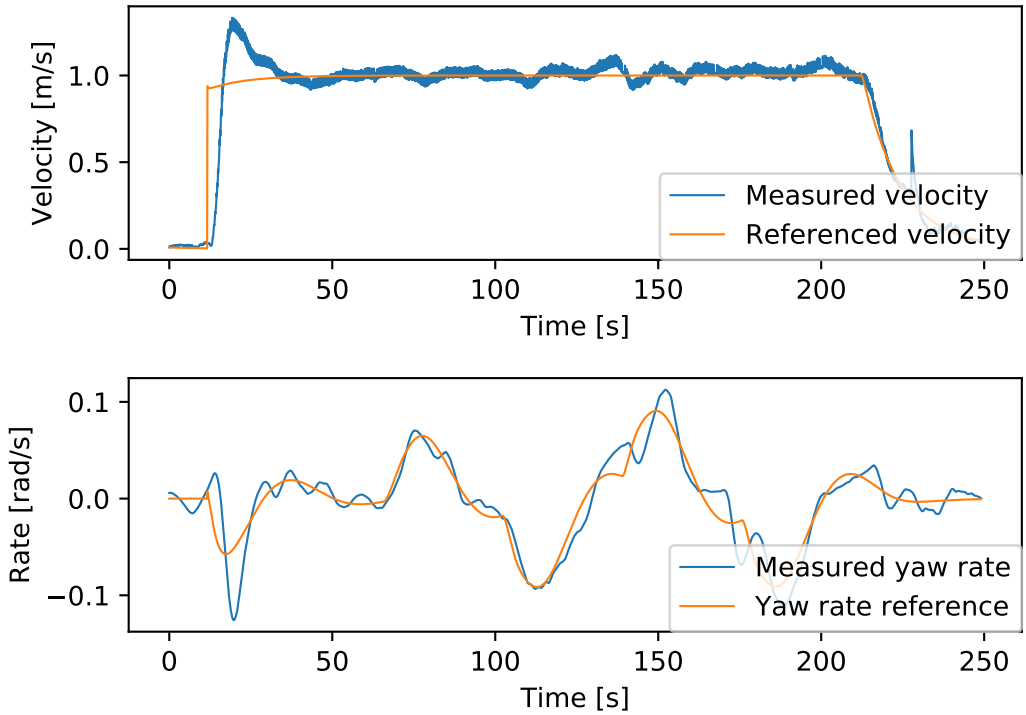


Figure B.4: Full-scale: Desired and measured velocities for curvy path

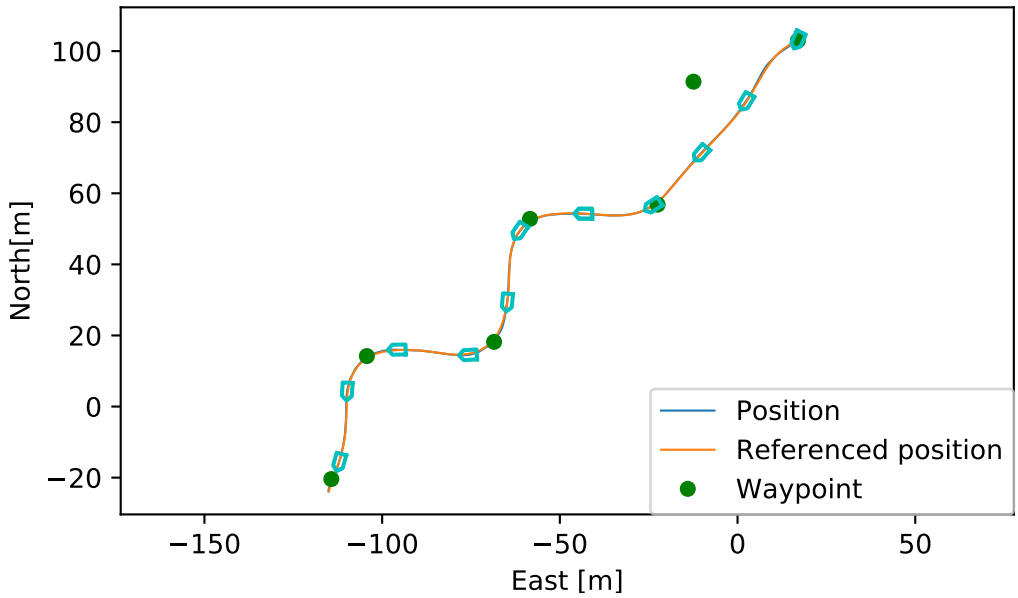


Figure B.5: Full-scale: Trajectory for curvy path

B.3.3 Test 3: Curvy Path with Variable Velocity

Figure B.6 shows additional results for test 3.

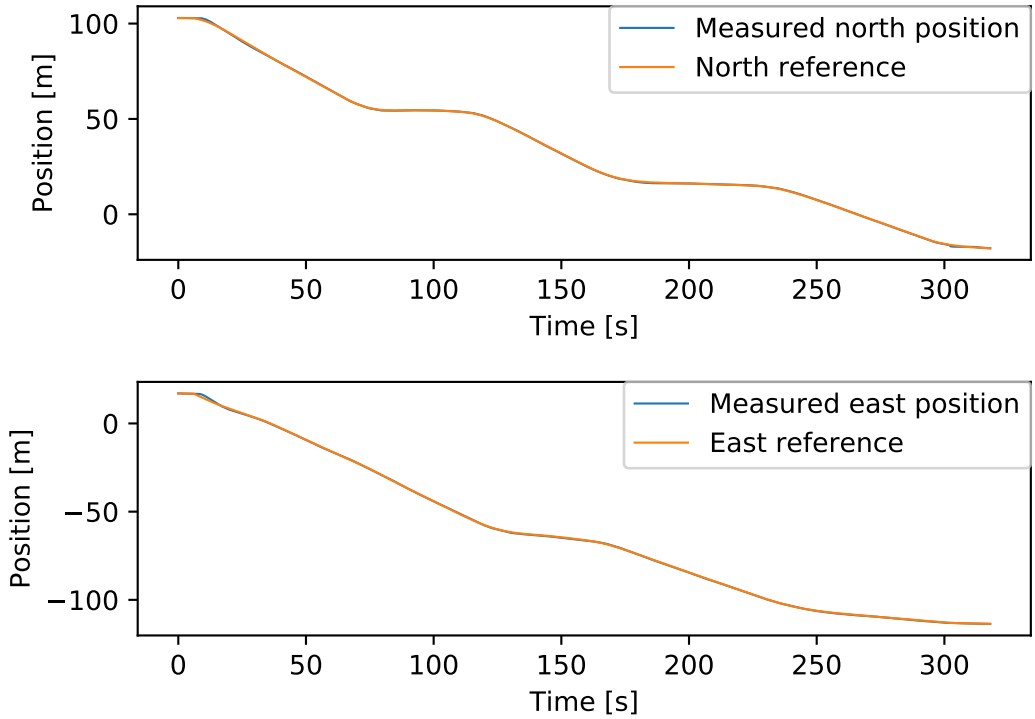


Figure B.6: Full-scale: Desired and measured north and east positions for curvy path with variable velocity

B.3.4 Test 4: Avoidance Manoeuvre 15°

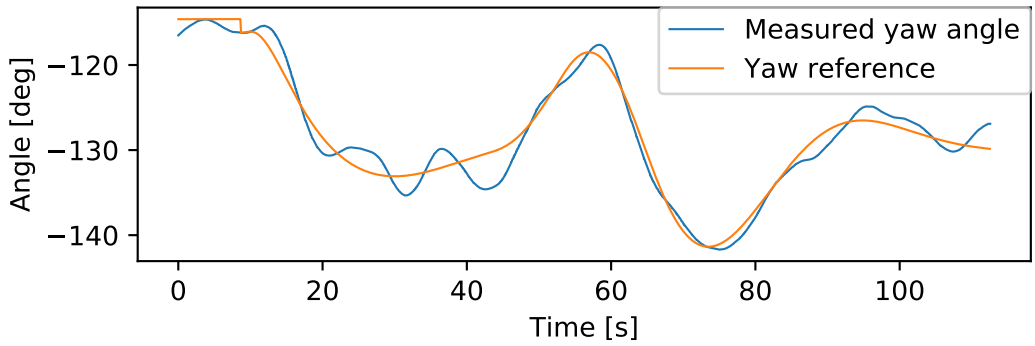


Figure B.7: Full-scale: Desired and measured yaw angle for avoidance manoeuvre with 15° turn

B.3.5 Test 5: Avoidance Manoeuvre 30°

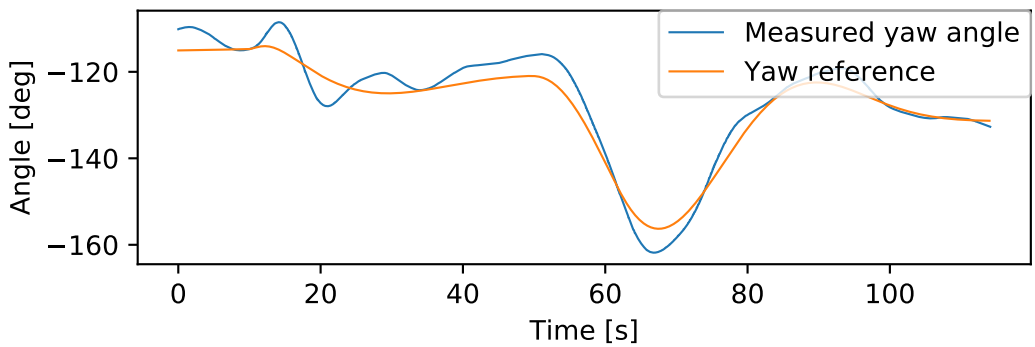


Figure B.8: Full-scale: Desired and measured yaw angle for avoidance manoeuvre with 30° turn

B.3.6 Test 6: Avoidance Manoeuvre 45°

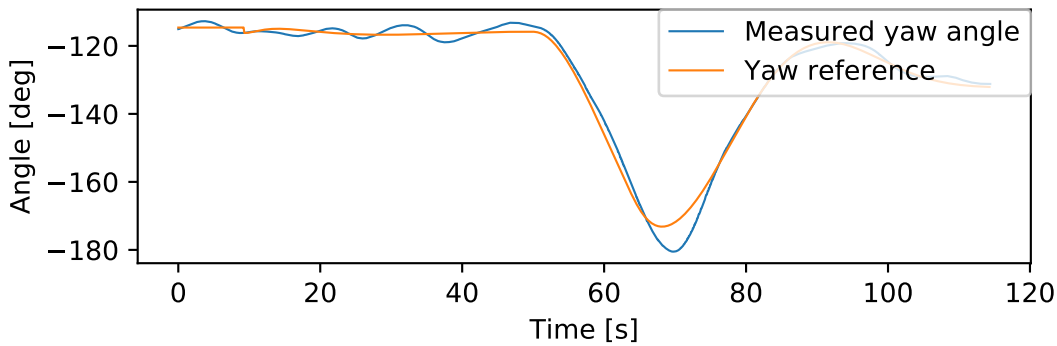


Figure B.9: Full-scale: Desired and measured yaw angle for avoidance manoeuvre with 45° turn

B.3.7 Test 7: Avoidance Manoeuvre 60°

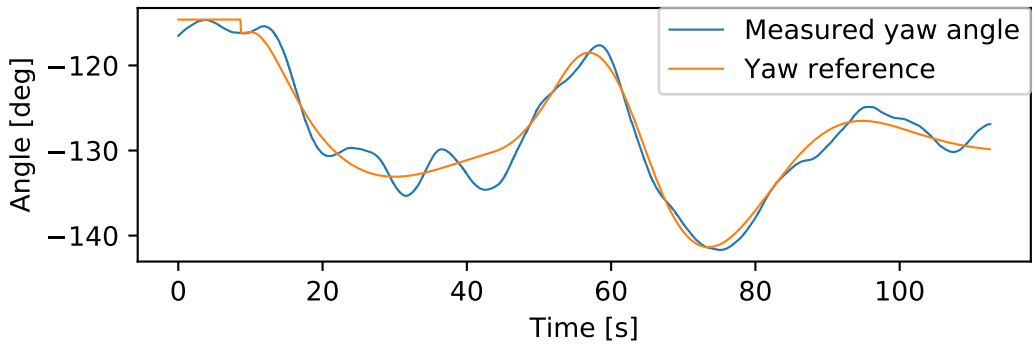


Figure B.10: Full-scale: Desired and measured yaw angle for avoidance manoeuvre with 60° turn

Appendix C

Alternative Reward Function

```
def get_reward(self):
    #If no collision or goal reached, continue
     done = False

    #Calculate distance from vessel to goal
    dist = get_distance(self.goal,[self.eta[0],self.eta[1]])

    if self.test_goal():
        done = True
        reward = 1
    elif self.test_collision():
        done = True
        reward = -1
    else:
        reward = (self.prev_dist-dist)/100
    self.prev_dist = dist
    return reward, done
```

Appendix D

Appended Paper

Here the paper on the new flexible guidance system is attached.

Waypoint-Based Trajectory Tracking with Experimental Results ⋆

Karianne Skudal Tjøm * **Tobias Valentin Rye Torben** *
Børge Rokseth * **Asgeir J. Sørensen** *

* *Centre for Autonomous Marine Operations (NTNU AMOS), Department of Marine Technology, Norwegian University of Science and Technology (NTNU), Otto Nielsens vei 10, 7052 Trondheim, (e-mail: [tobias.torben; borge.rokseth; asgeir.sorensen]@ntnu.no).*

Abstract: In this paper a novel guidance system for fully-actuated ships with dynamic positioning (DP) is proposed. The flexible guidance system is DP compatible and generates the heading reference, such that it does not have to be specified by the operator. The proposed method combines line-of-sight (LOS) and reference filters to ensure that the desired velocity is maintained through waypoints. Through simulation trials it was found that the guidance system performed well and that the key properties are obtained. A successful sea trial was conducted with the NTNU small passenger ferry MilliAmpère to verify the simulation results and test the method with real environmental forces.

Keywords: Guidance systems, Trajectory tracking, Autonomous vehicles

1. INTRODUCTION

This paper is about the development of a flexible guidance system for ships. With increased autonomy for vessels, more flexibility is required of the guidance and navigation systems of ships. The idea with this guidance system, is that the operator sets the waypoints and the desired velocity setpoint. The guidance system takes care of the trajectory generation. Often DP is used for transitions to a desired point for station-keeping, while here it is used for trajectory tracking at a desired velocity.

There are many guidance systems already existing. LOS and integral line-of-sight (ILOS) are often used for under-actuated vessels, controlling only the forward velocity and course angle (Fossen et al., 2003). ILOS introduced integral effect in the guidance system to counteract mean and slowly varying environmental forces (Caharija et al., 2016). Other guidance systems are reference filters which are often used for over-actuated vessels (Fossen, 2011). There are both positional and velocity reference filters. A position reference filter models the vessel to stop at the waypoints by filtering the velocity. Velocity reference filters only generate references for velocity without taking direction into consideration. Nađ et al. (2015) has proposed a method for fully-actuated line following using a constant predefined heading.

The main contribution in this paper is a novel guidance system. By combining LOS and reference filters we can achieve a guidance system which track position waypoints as well as velocity references to maintain a desired forward velocity. The steady-state cross-track error is compensated for by sway control action, instead of a steady-state heading non-parallel to the path as e.g. ILOS does. It is a flexible guidance system since the desired velocity can vary, and the next waypoint is not necessarily decided until the current one is reached. Consequently the waypoints can be decided during the run, instead of being decided beforehand. This is an advantage when it is necessary to change the path during the operation or if the ship operates in an unknown area.

This paper is organised as follows. Preliminaries are given in Chapter 2. Chapter 3 gives the problem statement, before the development of the flexible guidance law is described in Chapter 4. Chapter 5 presents and discusses the results while conclusion and further work come in Chapter 6.

2. PRELIMINARIES

2.1 Kinematics

To describe vessel movements, one commonly uses two frames of reference: the inertial North-East-Down (NED) and the body-fixed reference frame (SNAME, 1950). A vessel only operates at the sea surface, which leads to the 6-DOF model generally being reduced to a 3-DOF. NED velocities frame is related to BODY velocities by the kinematic equation

$$\dot{\boldsymbol{\eta}} = \mathbf{R}(\psi)\boldsymbol{\nu}, \tag{1}$$

where $\mathbf{R}$ is the rotation matrix given by

$$\mathbf{R} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2}$$

$\dot{\boldsymbol{\eta}}$ is the velocity vector in NED-frame, $\boldsymbol{\nu}$ is the velocity vector in body-fixed-frame and ψ is the heading (Fossen, 2011).

⋆ This work was supported by the Research Council of Norway through the Centres of Excellence funding scheme, project number 223254 - NTNU AMOS, and the KPN ORCAS project, project number 280655.

2.2 Vessel model

In this work an over-actuated DP vessel is used. A DP vessel is a vessel which maintains a stationary position or follows a pre-determined path using only active thrusters. Figure 1 shows a simplified structure of a DP system based on Sørensen (2011). From the vessel, the system receives

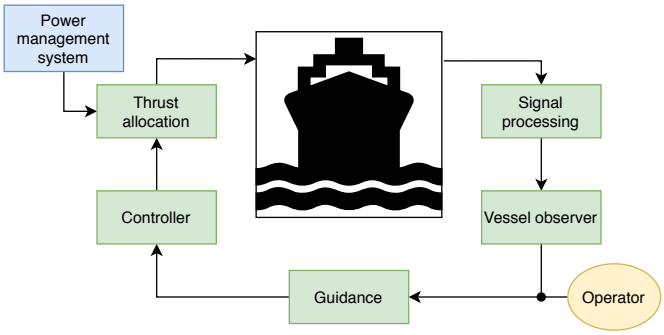


Fig. 1. Simplified representation of a dynamic positioning system

measurements, which are processed to ensure proper data quality before the vessel observer uses the data to do wave filtering and estimate states. The operator sets the waypoints, and the guidance system generates trajectories from one waypoint to another. The controller commands the desired thrust in the available DOF's, and the thrust allocation decides the setpoints for thrusters to achieve the desired thrust.

2.3 Line-of-sight

The LOS guidance law navigates towards a point on the path at a constant distance Δ from the vessel along the path. Δ is referred to as the look-ahead distance. The path $\mathcal{P}$ has to be linear and is given between the starting point WP_k and end point WP_{k+1}. The parameters are defined in Figure 2 in addition to $e(t)$ which is the cross-track

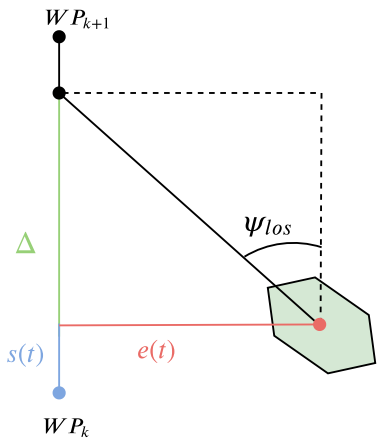


Fig. 2. Line-of-sight principles

error, $s(t)$ is the along-track distance. ψ_{los} is the LOS-angle, which often is the desired course angle.

LOS is often used for under-actuated vehicles, since it only demand control action in surge and yaw as demonstrated in Fossen et al. (2003), Borhaug et al. (2008) and, Caharija et al. (2012). The main goal is to make the cross-track error

$$\lim_{t \to +\infty} e(t) = 0, \tag{3}$$

go to zero. The cross-track error is found by

$$e(t) = -(x(t) - x_k) \cdot sin(\alpha) + (y(t) - y_k) \cdot cos(\alpha). \tag{4}$$

$x(t)$ and $y(t)$ are the current position in north and east direction.

α is the path-tangential angle, and is calculated by

$$\alpha = atan2\left(\frac{y_{k+1} - y_k}{x_{k+1} - x_k}\right), \tag{5}$$

where y_{k+1} and x_{k+1} are the north and east coordinates of the current waypoint in NED-frame while y_k and x_k are the coordinates of the previous waypoint. $atan2(\,)$ is used instead of $arctan(\,)$ to enable waypoints in all quadrants, ensuring that the correct direction is obtained.

The desired course angle for the vessel also depends on the velocity-path relative angle. The vessel has a point on the desired path at a look-ahead distance, which the vessel movement is directed towards due to the velocity-path relative angle. The velocity-path relative angle is given by

$$\chi_r = atan2\left(\frac{-e}{\Delta}\right), \tag{6}$$

where Δ is the look-ahead distance.

All components of the desired course angle χ_{LOS} is then given by

$$\chi_{los} = \chi_r + \alpha. \tag{7}$$

In addition to the cross-track error, the along-track distance $s(t)$ is often calculated to measure how far along the course from the previous waypoint the vessel is. The along-track distance is found by

$$s(t) = (x(t) - x_k) \cdot cos(\alpha) + (y(t) - y_k) \cdot sin(\alpha). \tag{8}$$

2.4 Reference filter

To obtain references that are two times differentiable, references are filtered. A standard position reference filter for continuous signals for position, velocity and acceleration as described by Fossen (2011) is given by

$$\boldsymbol{\eta_d}^{(3)} + (2\boldsymbol{\Delta} + \mathbf{I})\boldsymbol{\Omega}\ddot{\boldsymbol{\eta}}_{\boldsymbol{d}} + (2\boldsymbol{\Delta} + \mathbf{I})\boldsymbol{\Omega}^2\dot{\boldsymbol{\eta}}_{\boldsymbol{d}} + \boldsymbol{\Omega}^3\boldsymbol{\eta}_{\boldsymbol{d}} = \boldsymbol{\Omega}^3\mathbf{r}^n. \tag{9}$$

$\boldsymbol{\eta_d}$ is the referenced position vector, $\boldsymbol{\Delta}$ is the relative damping ratio matrix, $\boldsymbol{\Omega}$ is the natural frequency matrix, $\boldsymbol{r}^n$ is the desired position setpoint vector and $\boldsymbol{I}$ is the identity matrix.

When the reference is a velocity instead of a position, a velocity reference filter is used. Fossen (2011) describes a velocity reference filter that is constructed as a second order low-pass filter to avoid steps in the velocity and acceleration references and is described by

$$\ddot{\boldsymbol{\nu}}_{\boldsymbol{d}} + 2\boldsymbol{\Delta}\boldsymbol{\Omega}\dot{\boldsymbol{\nu}}_{\boldsymbol{d}} + \boldsymbol{\Omega}^2\boldsymbol{\nu}_{\boldsymbol{d}} = \boldsymbol{\Omega}^2\boldsymbol{r}^b. \tag{10}$$

Here $\boldsymbol{\nu_d}$ is the referenced velocity, and $\boldsymbol{r}^b$ is the desired velocity setpoint vector.

2.5 Controller

To reach the waypoints, a nonlinear PID with reference feedforward by Fossen (2011) is used as the control law as follows

$$\boldsymbol{\tau} = -\mathbf{R}^\top(\psi)\mathbf{K_p}(\hat{\boldsymbol{\eta}} - \boldsymbol{\eta_d}) - \mathbf{K_d}(\hat{\boldsymbol{\nu}} - \boldsymbol{\nu_d}) \\ -\mathbf{R}^\top(\psi)\mathbf{K_i}\int(\hat{\boldsymbol{\eta}} - \boldsymbol{\eta_d})dt + \mathbf{M}\boldsymbol{a_d} + \mathbf{D}\boldsymbol{\nu_d}, \tag{11}$$

where ψ is the measured heading. The proportional, derivative and integral gains are given by the non-negative matrices $\mathbf{K_p}$, $\mathbf{K_d}$, $\mathbf{K_i} \in \mathbb{R}^{3\times3}$. $\boldsymbol{\eta_d}$, $\boldsymbol{\nu_d}$ and $\boldsymbol{a_d}$ are the reference signal vectors for position, velocity and acceleration, respectively. The hatted symbols indicate that the states are estimated using an observer as e.g. Kalman-filter or nonlinear passive observer.

3. PROBLEM STATEMENT

Consider a DP vessel having waypoints with reference positions in North and East direction and a desired surge velocity setpoint. It lacks a yaw reference and the transitional behaviour from one waypoint to another. This reference has to be created by the guidance system, and intuitively it is desirable that that the heading is equal to the course direction.

The main goal of this guidance system is to produce continuous and differentiable position, velocity and acceleration references. This is for a fully-actuated vessel to follow waypoints at a desired velocity. The desired velocity can vary between some maximum value and zero as described by

$$0 \leq U_d \leq U_{max}. \tag{12}$$

The next waypoint is only known to the system, once the vessel is sufficiently close to the current waypoint. Towards the waypoint, the vessel is to converge to the path decided by line-of-sight such that the control objectives are formulated as

$$\lim_{t\to+\infty} \psi = \psi_{los}, \tag{13}$$

$$\lim_{t\to+\infty} U = U_d, \tag{14}$$

and ultimately the waypoint should be reached.

4. FLEXIBLE GUIDANCE LAW FOR WAYPOINT-FOLLOWING

This chapter describes the derivation of the propesed guidance method.

4.1 Yaw reference

The yaw reference is important as it will decide the course angle, and the course angle determines the velocity components in north and east direction. LOS will be used as described in Section 2.3. This generates a heading reference that leads the vessel towards a straight line connecting the previous waypoint with the next. ILOS was considered but was deemed superfluous due to the integrator effect in the control system.

Lookahead-based steering will be used at the expense of enclosure-based steering, since it is applicable for all cross-track errors and is less computationally demanding (Breivik and Fossen, 2009).

When calculating the cross track error with (4) the previous reference is used instead of the actual position to avoid a link between the actual position and the guidance system. This implies that the guidance system is a stand-alone system independent of offsets the vessel might have. Those deviations should be handled by the DP system.

To obtain smooth reference signals for yaw, yaw rate and yaw acceleration, a standard position reference filter is applied as described in section 2.4 for a scalar reference with the LOS-angle ψ_{los} as the desired setpoint and ψ the filtered reference.

4.2 Surge and sway references

The coordinates provided by the waypoints are not used directly by the reference model as commonly done. Instead a desired velocity is used to provide the total velocity while the heading reference decide the velocity components in north and east direction.

The velocity reference model from Section 2.4 is applied to achieve smooth signals with U_d as the desired velocity. U_d is set by the operator, and the reference U generated by the filter, is decomposed into two components in north and east direction such that

$$U = \sqrt{\dot{x}^2 + \dot{y}^2}, \tag{15}$$

is fulfilled. $\dot{x}$ is the velocity component in north direction, and $\dot{y}$ is the velocity component in east direction.

In order to reach the desired position with the desired velocity, the heading reference ψ is used to decide the velocity components $\dot{x}$ and $\dot{y}$. The velocity components are then obtained by

$$\dot{x} = U\cos(\psi), \tag{16}$$

$$\dot{y} = U\sin(\psi), \tag{17}$$

assuming that the course angle is the same as the heading. From the velocity reference model, reference signals for position and velocity are obtained for north and east direction. Acceleration references are found by differentiating (1), which results in

$$\ddot{x} = \dot{U}cos(\psi) - rUsin(\psi), \tag{18}$$

$$\ddot{y} = \dot{U}sin(\psi) + rUcos(\psi), \tag{19}$$

where $\ddot{x}$ and $\ddot{y}$ are the acceleration references in north and east direction respectively and r is the yaw rate reference. The references are then used as input for the DP control system.

A simplified block diagram of the guidance system is shown in Figure 3. The reference filters only have feedback from the previous references and not actual positional data or velocity measurements from the vessel. The red blocks are decided by the operator, the yellow blocks belong to the yaw references, and the blue block is the velocity reference filter. By combining the course angle with the referenced velocity, all necessary references are obtained.

4.3 Circle of acceptance

It is not necessary or desirable for the vessel to approach the waypoint at its exact position. This may cause large deviations from the desired path. To avoid this problem a circle of acceptance is implemented. When the vessel is an acceptable distance away from the waypoint at a line parallel to the straight line between the waypoints, the vessel continues to the next waypoint.

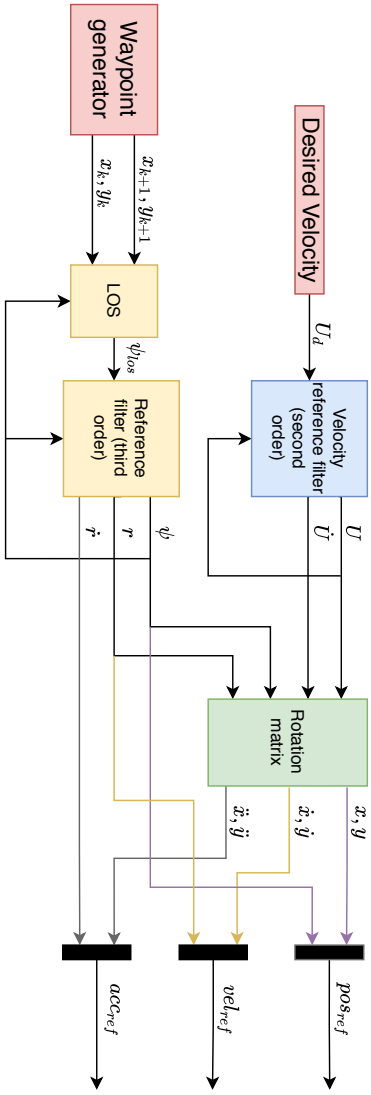


Fig. 3. Block diagram of the novel guidance system

To begin, the length of the line connecting two consecutive waypoints is calculated by

$$L = \sqrt{(y_{k+1} - y_k)^2 + (x_{k+1} - x_k)^2}. \tag{20}$$

The along-track distance is measured tangential to the path, and the along-track distance from the target is given by

$$|L - s(t)| \leq acceptable. \tag{21}$$

In (8), as well as for the cross-track error, the previous references are used instead of the actual position of the vessel. The acceptance value is decided by trial and error, and how large the risk is by having deviations from the waypoints and paths between them. The acceptance value should be large enough for the vessel to avoid large overshoots from the path, yet small enough to achieve the desired behaviour.

5. SIMULATIONS AND EXPERIMENTS

To verify that the guidance law works as expected, both simulations and full-scale experiments were conducted. The goal is to verify that the desired velocity is maintained through the waypoints, and that the waypoints are reached and that the generated references are sufficiently smooth.

The NTNU small passenger ferry MilliAmpère is used both in simulations and experiments.

The full-scale testing was conducted in Trondheim. MilliAmpère was equipped with two azimuth thrusters, mounted along the ship longitudinal axis. Key features of the vessel are shown in Table 1. Here, $L_{1,x}$ and $L_{2,x}$ are the azimuth position along the x-axis.

Table 1. MilliAmpère specifications

Parameter	Value
Hull length	5.0m
Hull width	2.8m
Displacement	1667kg
Number of thrusters	2
Max thrust	500.7N
Max azimuth turn rate	$30\frac{deg}{s}$
$L_{1,x}$	-1.8m
$L_{2,x}$	1.8m

5.1 Simulation results and discussion

The simulations were executed in a SiL simulator with a ROS-based control system for MilliAmpère. The vessel shall track waypoints while maintaining first a constant velocity, then a variable velocity. The damping ratios and natural frequencies are

$$\Omega_\psi = 0.4, \quad \Delta_\psi = 1, \quad \Omega_U = 0.5, \quad \Delta_U = 3. \tag{22}$$

The acceptance value is set to 3.0 meters for the first simulation with constant velocity. Figure 4 shows the desired and actual position and yaw angle of the vessel as it tracks the waypoints.

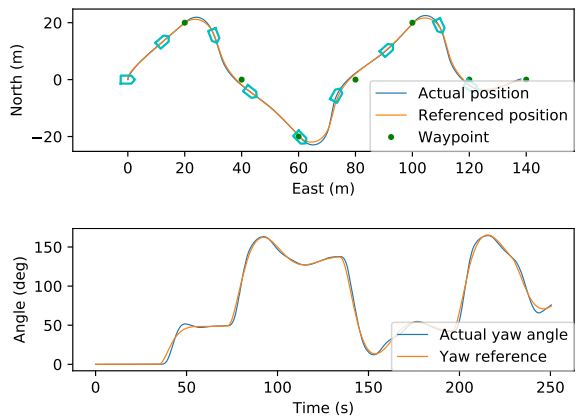


Fig. 4. Simulation: Trajectory and desired and actual yaw angle with constant velocity

The waypoints are reached, but the vessel does not pass right through them. The yaw angle reference is followed closely with a few exceptions. At about 50 seconds and 250 seconds the yaw angle deviates with approximately 5 degrees from the reference.

Figure 5 shows the referenced and the actual forward velocity and yaw rate in addition to the referenced absolute acceleration. The forward velocity and acceleration are absolute values. The velocity signal is calculated by (15), and the same principle yields for the absolute acceleration. The forward velocity quickly reaches the target velocity, and the reference is constantly at the desired velocity.

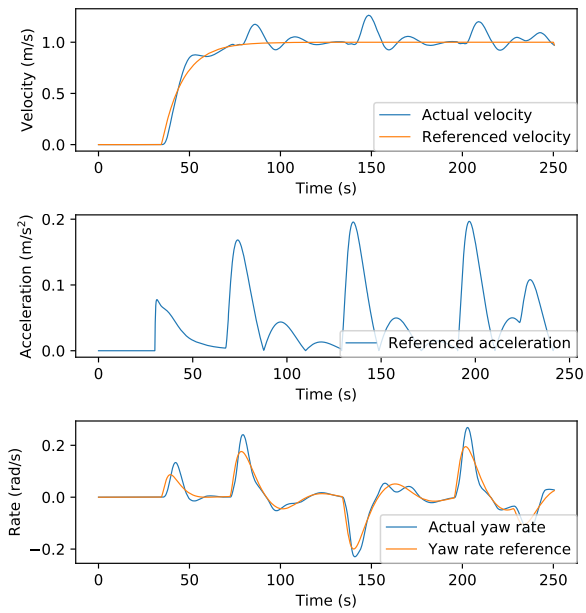


Fig. 5. Simulation: Desired and actual absolute velocity with constant desired velocity

There are some oscillation that may be due to the change of course direction and that the DP require the velocity reference in north and east direction, instead of one component. It is seen that the acceleration reference is increasing very quickly. However, the signal is continuous but to get a smoother acceleration the natural frequency could be decreased.

The actual yaw rate follows the reference closely, while there are some overshoot when the rate change quickly. This can be reduces by decreasing the natural frequency, but the downside of that is a slower turn rate for the vessel. Considering that the reference is followed closely for the most part, the current parameters seems like an acceptable compromise.

To achieve a result with less deviations, the waypoints could be even further from each other or the damping increased. Other possibilities are to reduce the desired forward velocity, or increase the acceptance value, so the vessel can start the turn earlier and hence reduce the overshoot and need for high yaw rate. For the next simulation with variable velocity, the acceptance value will be increased to eight, while all other parameters stays the same.

Figure 6 shows the trajectory and velocity for the simulation with variable velocity. It is seen that the transition from waypoint 1-2 is more direct with the increased acceptance value, while when the velocity is lower, the ship turns too far away from the waypoint not reaching it at all. The transition between a desired velocity of $1m/s$ to $0{,}7m/s$ is smooth, but takes a considerable amount of time. The vessel also manages to increase the velocity back to $1m/s$ after waypoint five. The oscillatory behaviour seems to be similar to that in Figure 5, and may therefore not be significantly affected by the change in desired velocity.

This simulation test has shown that the flexible guidance system achieves the desired behaviour. However, the response could be quicker and the overshoots smaller. To

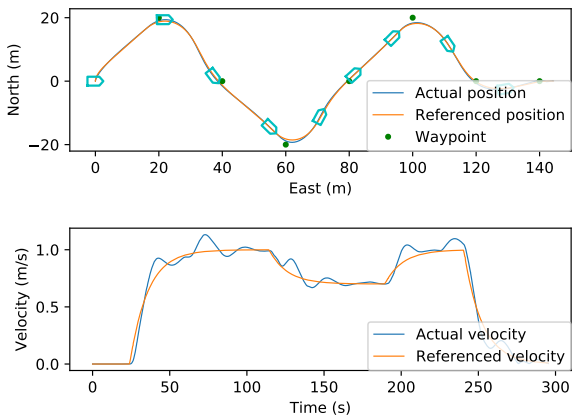


Fig. 6. Simulation: Trajectory and desired and actual velocity with variable desired velocity

address these issues, variable look-ahead distance could be implemented with adaptive LOS, the acceptance value could change based on what the desired velocity is, and the reference filters could be replace by another reference model.

5.2 Full-scale experimental setup, results and discussion

Table 2 shows the observations done of the test environment. Especially the wind estimate should be noted, as this created a significant environmental force on the ship. The weather was otherwise clear and sunny. First a zig-

Table 2. Observations of test environment

Observation	Comment
Temperature	$6°C$
Wind estimate	$5m/s$
Current estimate	$<0{,}5m/s$
Waves	small
Date	20.05.2020
Start time	10:20
End time	11:15

zag test was conducted to compare the performance in real-life to the simulation. Figure 7 shows the trajectory and measured and referenced velocity for the curvy path with variable desired velocity.

The generated path is smooth, and the vessel follows easily without significant overshoots from the path. Considering the significant wind present, it seems that the strategy of using DP to counteract the environmental forces instead of compensating for that in the guidance system is successful. As seen in the simulations, also here the vessel turns before the waypoints are reached when the desired velocity is lower. It is also seen that the first waypoint is disregarded. This is due to complications with transitioning between the DP with regular reference filter and the flexible guidance system. This complication also caused the rapid increase in velocity which lead to a significant overshoot. It should be noted that the velocity is not measured directly but rather estimated from the position signals. The desired velocity start at $1m/s$ before first decreasing to $0.7m/s$ after waypoint 2 and then to $0.5m/s$ after waypoint 4.

Beyond the abrupt change from $0m/s$ to approximately $0.7m/s$ before the velocity reference filter becomes active,

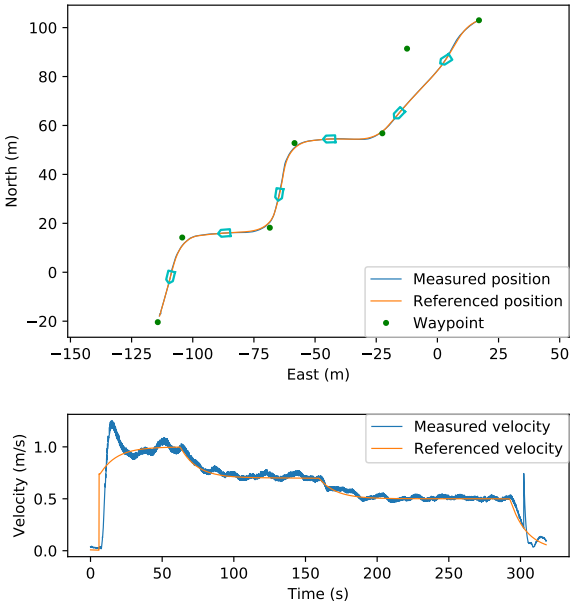


Fig. 7. Sea trial: Trajectory and desired and actual velocity with variable desired velocity

the reference is smooth and followed nicely by the vessel. The reference seems to approach the decreased desired velocities smoothly and maintains the velocity. There is a wildpoint at approximately 300 seconds in the measured velocity. The vessel uses RTK and at this point the vessel went from floating to fixed RTK giving a correction in the measured position. Since the velocity measurements are derived directly from the position signals without signal processing this causes a wildpoint.

In addition the guidance system was tested on how well it is suited for avoidance manoeuvres. The manoeuvres will deviate, 15°, 30°, 45° and 60° from the path. Also with these tests there were problems with the transitions. Hence only the avoidance test turning 15° reach the first waypoint before turning, while the other tests turn toward the deviating waypoint straight away. Therefore the discussion focuses on the turn back to the path. When the first waypoint falls out, the velocity reference filter takes some seconds before activating, giving some overshoot in the measured velocity compared to the reference. Even though it would have been interesting to see how the vessel turns away from the path, it is still interesting to investigate how it manages to resume the original path.

Figure 8 shows the trajectories for all the avoidance manoeuvres. It is seen that all the manoeuvres manage resume the path to reach the last waypoint. However, all the manoeuvres also overshoot when resuming the path, except for the manoeuvre with a 15° turn, but is limited to a maximum of 5 meters offset at the 60° turn. Considering that the vessel does not have massive overshoots and manage to resume the path, this guidance system could be applicable for avoidance manoeuvres.

6. CONCLUSION AND FURTHER WORK

This paper has presented and tested a novel guidance method for over-actuated DP ships. The sea trial confirmed the results found in the simulations and the proposed method accomplished the desired behaviour. Use of adaptive look-ahead distance and acceptance value may improve the performance even more. If these suggestions were to be implemented successfully, it would result in a guidance system that makes rapid enough turns to keep the course, manages to change velocities quick and is adequately adapted for different velocity setpoints.

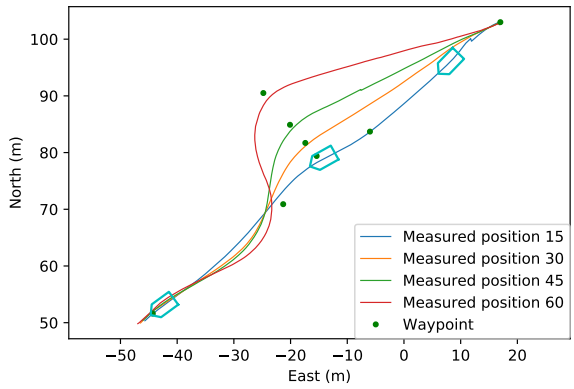


Fig. 8. Sea trial: Avoidance manoeuvres with four waypoints common and one individual for all tests

REFERENCES

Borhaug, E., Pavlov, A., and Pettersen, K.Y. (2008). Integral los control for path following of underactuated marine surface vessels in the presence of constant ocean currents. In *2008 47th IEEE Conference on Decision and Control*, 4984 – 4991.

Breivik, M. and Fossen, T. (2009). *Guidance Laws for Autonomous Underwater Vehicles*, chapter 4. IntechOpen.

Caharija, W., Pettersen, K.Y., Bibuli, M., Calado, P., Zereik, E., Braga, J., Gravdahl, J.T., Sørensen, A.J., Milovanović, M., and Bruzzone, G. (2016). Integral line-of-sight guidance and control of underactuated marine vehicles: Theory, simulations, and experiments. *IEEE Transactions on Control Systems Technology*, 24(5), 1623–1642.

Caharija, W., Candeloro, M., Pettersen, K.Y., and Sørensen, A.J. (2012). Relative velocity control and integral los for path following of underactuated surface vessels. In *IFAC Proceedings Volumes*, volume 45, 380 – 385. 9th IFAC Conference on Manoeuvring and Control of Marine Craft.

Fossen, T.I. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons, Ltd, Chichester, UK.

Fossen, T., Breivik, M., and Skjetne, R. (2003). Line-of-sight path following of underactuated marine craft. In *IFAC Proceedings Volumes*, volume 36, 211–216.

Nađ, D., Mišković, N., and Mandić, F. (2015). Navigation, guidance and control of an overactuated marine surface vehicle. *Annual Reviews in Control*, 40.

SNAME (1950). *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.

Sørensen, A.J. (2011). A survey of dynamic positioning control systems. *Annual Reviews in Control*, 35(1), 123 – 136.

Karianne Skudal Tjøm

Guidance and Decision Making using Machine Learning

NTNU

Norwegian University of
Science and Technology