Master's thesis

2020

Maiken Berthelsen

Master's thesis

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

NTNU
Norwegian University of
Science and Technology

Maiken Berthelsen

Current Estimation for Small Autonomous Passenger Ferry

June 2020

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

Current Estimation for Small Autonomous Passenger Ferry

Maiken Berthelsen

Marine Technology

Submission date:	June 2020
Supervisor:	Asgeir J. Sørensen
Co-supervisor:	Børge Rokseth
	Tobias R. Torben

Norwegian University of Science and Technology
Department of Marine Technology

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

MASTER THESIS IN MARINE CYBERNETICS

SPRING 2020

FOR

STUD. TECHN. MAIKEN BERTHELSEN

Current Estimation for Small Autonomous Passenger Ferry

Work description (short description)
It is assumed that autonomous ships will enable safer, cheaper, and more environmentally friendly operations at sea. New technologies are rapidly emerging and applying these can help improve the safety of autonomous ships. To enable maneuvering of a ship with surrounding traffic it is essential with a well-working system for situation awareness.

The aim of this thesis is to improve situation awareness by estimating the current forces that a vessel is exposed to. This will be done using to different methods; one based on the extended Kalman filter and one using a machine learning algorithm. With the environmental forces considered, the path planning and collision avoidance system of an autonomous ship can be further improved. A framework for how the current estimates should be considered together with obstacles or thruster capacity will be proposed. In this thesis the small passenger ferry MilliAmpère is considered, both in the form of its simulator and through a full-scale experiment.

Scope of work
1. Review necessary literature within the field of current estimation on a vessel and the extended Kalman filter.
2. Review necessary literature within the field of machine learning.
3. Develop two models for estimating the current forces on MilliAmpère, one using the extended Kalman filter and one using machine learning.
4. Test the sensitivity of the EKF with respect to noise and parameter sensitivity.
5. Test the methods for current estimation by performing a full-scale experiment using the MilliAmpère ferry.
6. Evaluate the proposed methods and experimental results.

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: Dr. Børge Rokseth, Tobias Torben

Asgeir J. Søren

Professor Asgeir J. Sørensen
Supervisor

Preface

This thesis is written by Maiken Berthelsen during the spring of 2020. It is the final piece of the fulfillment of the Master's degree in Marine Technology, with a specialization in Marine Cybernetics, at the Norwegian University of Science and Technology (NTNU).

The main goal of this thesis was to develop methods that would provide current estimation onboard the small autonomous passenger ferry MilliAmpère. A simulator of MilliAmpère was given, and the concepts were first tested and evaluated using this, before the current estimation methods were tested during full-scale experiments.

Since there are many interesting topics related to the field of marine cybernetics, I decided to look into two methods to perform the current estimation. One of the methods proposed is developed using an algorithm commonly used within control theory, the extended Kalman filter, and the other method is more commonly used within other fields, and is based on machine learning methods. Performing a thorough study into two very different fields proved challenging at times, but it was also intriguing to investigate how both ways could lead to the same result.

The full-scale experiments were first planned to be performed in two rounds, which would have made it possible to implement further improvements to the second round. However, due to the extraordinary circumstances around COVID-19, the experiments were greatly postponed, and only one round of experiments was performed. Even though the results from the experiment were quite satisfying, there was no time to adjust the changes done to the extended Kalman filter to the results obtained from the simulator.

The work presented, including theoretical studies, numerical analysis, and experimental trials, is solely done by me, unless otherwise stated in the text.

Trondheim, June 8th, 2020

Maiken Berthelsen

Maiken Berthelsen

Acknowledgments

First of all, I would like to express my gratitude to my supervisor, Professor Asgeir Johan Sørensen, for all your guidance throughout the year. Thank you for always being optimistic and invested in the problems I faced, and always making extra time for me in order to solve them. During a spring where nothing seemed to be going according to plan, you were a great contributor to ensuring that progress was made, which made it possible for me to finish my thesis on time.

I would also like to thank my co-supervisors, Dr. Børge Rokseth and Tobias Rye Torben. Thank you, Børge, for helping me with the process and guiding me towards what could be interesting to investigate further, along with providing invaluable feedback on my report. Thank you, Tobias, for always helping me find the solutions, especially regarding my programming problems. I would also like to thank you for helping us with MilliAmpère, without you, the experiment would not have been possible to conduct.

Further, I would like to thank the class of 2020, for many great memories and fun moments, making it possible to get through the not so fun ones, and also for keeping in touch through these extraordinary circumstances during the spring of 2020. At last, I would also like to thank my family for much support throughout my master's.

Abstract

The autonomy within maritime industries is improving, and several innovative projects are thus emerging. Just as autonomous buses are entering the streets, the idea of using autonomous water buses has also been proposed. Water buses will make it possible to create cheaper urban connections, and this without blocking other waterway transport.

One challenge related to autonomous vessels is that they need to obtain the same level of perception and interpretation of the surrounding environment as a human being. Many techniques for obstacle detection, path planning, and collision avoidance are under development, in order to improve the situation awareness. Technologies created for other industries, especially the automobile industry, can also be applied to autonomous vessels, though some problems remain to be solved.

Traditionally, it has proven difficult to obtain good estimates of the velocity of the water current. Therefore, the current is usually incorporated in the form of a bias in dynamic positioning systems, which also includes the unmodeled dynamics. The main topic of this thesis has thus been to further investigate methods of estimating the current velocity, along with how these current estimates could be relevant for a path planning and collision avoidance algorithm of an autonomous vessel.

A proposed architecture explaining the relevance of the current estimate has been developed. This architecture explains how the estimates of the current, together with typical knowledge regarding the surroundings of the vessel and the ship status of the vessel, can be used to improve the path planning and collision avoidance algorithm. Necessary information regarding the surroundings includes knowledge about potential obstacles, obtained through various sensors. The ship status should include information about the maximum capacity of the thrusters and the total energy available for propulsion.

Two types of methods for performing current estimation have also been developed. One using the extended Kalman filter and one using machine learning. Within machine learning, two methods were studied, consisting of either deep densely connected neural networks, or radial basis function networks. The extended Kalman filter gave satisfactory results on both simulated and experimental data but proved difficult to tune. It was also demonstrated how exact knowledge of the control forces and the Coriolis, centripetal, and damping matrices are essential to obtain an accurate estimate of the current. This was also found to be true for the machine learning methods. The best-performing machine learning models were either a deep densely connected neural network, consisting of three hidden layers of 500 neurons and dropout layers with a dropout rate of 0.2, or a radial basis function network with 100 neurons, where the widths belonging to each center of the neurons were varying for each feature. The advantage of the radial basis function network is that the results are obtained using fewer parameters.

Seeing that both methods demonstrated promising results on experimental data, the current estimates achieved are assumed to be of value and further improve an autonomous vessel's situation awareness.

Sammendrag

Autonomien i maritime næringer blir stadig bedre, noe som gjør at flere innovative prosjekter settes i gang. Nå som selvkjørende busser har blitt en realitet på veien, har det også blitt åpnet opp for idéen om selvkjørende vannbusser. Disse vannbussene vil gjøre det mulig å skape billigere urbane transportforbindelser, uten at de er til hinder for annen vanntransport.

En utfordring med autonome skip er at de må oppnå samme nivå av oppfatning og tolkning av det omliggende miljøet som et menneske. Flere teknikker for hindringsdeteksjon, baneplanlegging og kollisjonsunngåelse utvikles derfor for å forbedre situasjonsforståelsen til et fartøy. Teknologier fra andre industrier, spesielt bilindustrien, kan også bli anvendt på autonome skip, men det er fortsatt noen problemer som gjenstår å løse.

Tidligere har det vist seg vanskelig å oppnå gode estimater for strømhastigheten i vann, og i dynamiske posisjoneringssystemer blir strøm ofte tatt hensyn til i form av en bias. Denne biasen inkluderer også umodellert dynamikk. Hovedtemaet i denne masteroppgaven har derfor vært å undersøke metoder for å estimere strøm, i tillegg til å se nærmere på hvordan strømestimat kan være nyttig for baneplanlegging og kollisjonsunngåelses-algoritmen for et autonomt fartøy.

Masteroppgaven forselår en arkitektur som beskriver relevansen av et strømestimat. Denne arkitekturen forklarer hvordan strømestimatene, sammen med annen kunnskap om omgivelsene og skipets status, kan forbedre baneplanleggings og kollisjonsunngåelses-algoritmen. Nødvendig data om skipets omgivelser består av informasjon om potensielle hindringer, som er oppnådd ved bruk av forskjellige sensorer. Informasjon om skipets status innebærer informasjon om maksimal thrusterkapasitet og total mengde energi tilgjengelig for propulsjon.

To typer metoder har blitt utviklet for å utføre strømestimering. Én ved bruk av det utvidede Kalman-filteret, og en ved bruk av maskinlæring. Innen maskinlæring ble det sett på to forskjellige typer nettverk, som bestod av enten dype nevrale nett eller radielle basisfunksjonsnettverk. Det utvidede Kalman-filteret ga gode resultater for både simulert og eksperimentell data, men det viste seg å være vanskelig å tune. Det ble også demonstrert hvordan eksakt kunnskap om de sanne kontrollkreftene og Coriolis, sentripetal og dempningsmatrisene er avgjørende for nøyaktige strømestimater. Dette viste seg å stemme for maskinlæringsmetodene også. Maskinlæringsmetodene som ga de beste strømestimatene bestod av enten et tett koblet dypt nevralt nett bestående av tre skjulte lag, med 500 nevroner, i

tillegg til mellomliggende *"dropout"* lag med en rate på 0.2, eller et radielt basisfunksjonsnettverk av 100 neuroner, hvor bredden tilhørende senteret av hvert nevron varierte over variablene i input dataen. Fordelen med det radielle basisfunksjonsnettverket er at resultater oppnås med færre parametere.

Begge metodene viste lovende resultater på eksprimentell data. De oppnådde strømestimatene antas derfor å være av verdi, og til å kunne videre forbedre situasjonsfortåelsen til et autonomt fartøy.

Table of Contents

List of Tables

List of Figures

Abbreviations

AI	=	Artificial Intelligence
AIS	=	Automatic Identification System
AUV	=	Autonomous Underwater Vehicle
CO	=	Coordinate origin
DOF	=	Degrees of Freedom
DP	=	Dynamic Positioning
EKF	=	Extended Kalman Filter
GNSS	=	Global Navigation Satellite Systems
INS	=	Inertial Navigation System
LOA	=	Level of Autonomy
LOS	=	Line-of-Sight
MAE	=	Mean Absolute Error
MSE	=	Mean Squared Error
NED	=	North-East-Down
RBFN	=	Radial Basis Function Network
ROS	=	Robot Operating System
RTK	=	Real-Time Kinematic
SA	=	Situation Awareness
SGD	=	Stochastic Gradient Descent

Chapter 1

Introduction

This thesis presents new methods for estimating the current velocity and direction onboard a marine vessel. It covers how the estimates of the current, and elements needed to be considered together with these estimates, can be used to improve the path planning and collision avoidance of autonomous vessels. The required knowledge of the true model of the vessel used to perform current estimation, and how noise in the model affects the results, are also discussed. The methods used to provide the current estimates are validated through both simulation and experimental studies.

1.1 Background and Motivation

Over the years, machines and robots have been developed to perform tasks previously performed by humans. In the maritime industry, autonomous ships have gained a lot of interest for some time now. According to DNV GL (2018), the main incentives behind the development of autonomous ships are economy, safety, and the environment. Both building and operational costs can be reduced with unmanned ships. For large merchant ships, the benefits of not having a crew onboard are many. There is no need for accommodation, resulting in a more simplified structures as well as less power consumption, no need for safety equipment and no expenses connected to manning a crew (Hoem et al. 2019). An example of such a ship is the fully electric and autonomous container ship YARA Birkeland, shown in Figure 1.1. According to KONGSBERG (2019), the ship will be delivered by the shipyard during the year of 2020, and will gradually move from manned operation to fully autonomous operation by 2022.

Figure 1.1: Illustration of the autonomous ship YARA Birkeland (KONGSBERG 2019)

Another example of autonomous solutions being looked at are the ferries. Examples of projects in the industry consist of Wärtsilä developing solutions for the car ferry *MF Folgefonn*, (Stensvold 2018), Rolls-Royce Marine (now Kongsberg) looking at the *Fjord1* ferries (Rosbach 2018), and Kongsberg working with solutions for the ferries of *Bastø Fosen* (Lorentzen 2018). The Norwegian University of Science and Technology (NTNU)-established firm Zeabuz looks at the concept of having zero-emission water buses. These water buses can replace potential pedestrian overpasses, making it cheaper and creating urban connections without blocking other waterway transport. A test ferry of such a concept is the small passenger ferry, MilliAmpère (Trana 2019), shown in Figure 1.2.

The regulating and certification bodies are also preparing rules and regulations for autonomous ships. The International Maritime Organization, IMO, have defined the term MASS as being Maritime Autonomous Surface Ships, and has the current main strategic direction for autonomous ships: *Integrate new and advancing technologies in the regulatory framework* (IMO 2019). Several consisting regulations, such as *Safety of Life at Sea*, (SOLAS), and *International Regulations for Preventing Collisions at Sea*, (COLREG), are now being looked at to determine how the operation of MASS can be introduced in IMO instruments in a safe, secure, and environmentally friendly manner (IMO 2019). The main requirement to obtain regulatory approval is that the autonomous ships need to be at least as safe as conventional ships (Jokioinen et al. 2016).

An aspect of information often overlooked, both when considering autonomous and traditional vessels, is the effects of the current velocity, which is usually counteracted rather than estimated. Obtaining more information about the current velocity and its direction may be of relevance in several situations. For autonomous

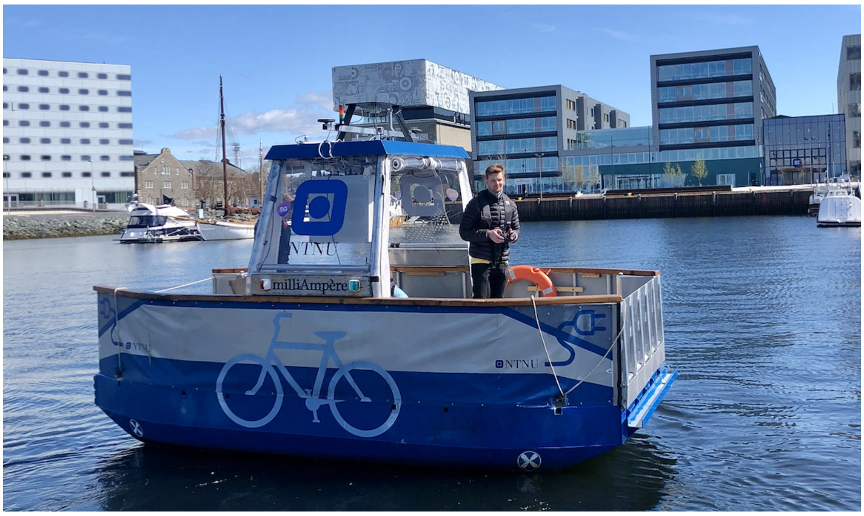

Figure 1.2: Picture of Tobias Torben onboard MilliAmpère, NTNU's autonomous ferry. Picture taken when doing experiments on the 20th of May 2020

ships it may give a wider insight when performing path planning and collision avoidance, making better paths both in terms of energy efficiency and safety. In case of an engine failure, the risk of collision can be reduced by passing an obstacle with a larger distance if the current is pushing the vessel towards the obstacle. In terms of freely floating obstacles, the direction of the current will also give a better estimate in predicting the motion of the obstacles.

Estimating the current velocity precisely may be difficult if an exact model of the vessel is not available. However, there may be value in the estimates even if the exact current values are not known precisely. Path planning of an autonomous vessel can be improved by obtaining information about the approximate direction of the current. If a lot of resources are needed to obtain a good estimate of the current, it may also be interesting to look at how long such an estimate will be valid, and how much the current velocities changes during a day due to influences from the wind and tide.

A great deal of effort has already gone into making MilliAmpère more autonomous, and developing models which satisfactorily represent the passenger ferry. The full-scale MilliAmpère ferry and the simulator describing it will, therefore, be used as a case study to evaluate the proposed current estimation methods. The simulator has been developed using the Robot Operating System (ROS), a framework suitable for writing robot software as it enables a seamless transition between the system developed for the simulator and the physical vessel. ROS works with several programming languages, and the simulator of MilliAmpère is written in Python.

1.2 Research Question and Contributions

The research questions of this thesis are:

- *How can an estimate of the current velocity have an impact on the path planning and collision avoidance of a highly autonomous vessel?*
- *Is it possible to obtain decent current estimates onboard a real ship knowing its mathematical model?*
- *How does noise in the mathematical ship model affect the accuracy of the current estimates?*

The contributions of this thesis can be summarized as:

- The proposal of a framework explaining how the current estimates together with other elements can play an important part in the path planning and collision avoidance algorithm.
- Two main methods for estimating the current:
  - one based on the extended Kalman filter, and
  - one based on machine learning methods, which is further divided into traditional deep neural networks and radial basis function networks.
- The validation of the methods is achieved through simulation and full-scale experiments, and the simulations are tested with noise to investigate the sensitivity of the current estimation methods.

The extended Kalman filter is programmed to be implemented directly into MilliAmpère's simulator, and is therefore written in ROS using Python. The machine learning models are stand-alone methods, and use the same parameters of MilliAmpère as given in the simulator, but are written in Python using the machine learning framework Keras, and Tensorflow as backend engine.

1.3 Thesis Outline

This thesis is divided into six chapters, which consist of:

Chapter 2 - Background Material:
This chapter covers relevant background material needed to perform the current estimation using an autonomous vessel. The first two sections elaborate on the classification of autonomous vessels into different levels of autonomy, and the aspects of information relevant for situation awareness. Secondly, a brief literature review on methods for current estimation is presented, before the mathematical modeling of a ship and the environmental forces affecting the ship are presented. The last two sections present the theory behind the extended Kalman filter and relevant machine learning methods, which are used to perform current estimation on the simulator in Chapter 4 and the full-scale MilliAmpère in Chapter 5.

Chapter 3 - Proposed Approaches for the Application and Methods of Performing Current Estimation:
The proposed architecture for how estimates of the current can contribute to improved path planning and collision avoidance is presented in this chapter. The specifications of MilliAmpère are also presented, both in terms of its full-scale dimensions and the parameters used in the simulator representing MilliAmpère. This chapter also presents how the extended Kalman filter is developed in order to estimate the current, which is simulated in Section 4.1. It is also explained how the filter is further expanded to improve the current estimates onboard a real vessel, which is used to obtain the results presented in Chapter 5. The method of generating data used as input for machine learning models, together with the methods of building deep neural networks and radial basis function networks are also presented. These trained neural networks are used on simulated data in Section 4.2 and experimental data in Chapter 5.

Chapter 4 - Simulation Results and Discussion:
This chapter presents the results and discussion of the current estimation performed using simulated data. In the first section, the extended Kalman filter is tested on the MilliAmpère simulator, and how the accuracy of the current estimates is affected by adding noise in the simulator is presented and discussed. In the next section, deep neural networks and radial basis function networks of various sizes are tested and evaluated on stationary data generated from the mathematical model of MilliAmpère.

Chapter 5 - Full-Scale Experiment Results and Discussion:
The results from performing current estimation on experimental data are presented in this chapter. Information about the experiment and how the true values for the current are obtained are presented first. Based on these true values, the current estimates obtained when MilliAmpère is floating freely, maintaining its position using dynamic positioning, or moving with constant velocity are evaluated. The current estimates are obtained using the trained neural networks and the extended Kalman filter. A discussion on the performance of the methods along with potential sources of error are also presented.

Chapter 6 - Conclusions and Recommendations for Further Work:
This chapter presents the concluding remarks of the thesis and elaborates on the main improvements and possibilities available for further work.

Chapter 2

Background Material

This chapter covers the necessary background material needed to perform an estimation of the current. The first two sections present a way of classifying an autonomous ship depending on its level of autonomy, and a brief overview of the information needed for a ship to have situation awareness. The next section presents a brief literature review on methods used for current estimation, and some machine learning methods used to estimate similar topics within fields outside the maritime industry. The next two sections elaborate on how to mathematically model a ship and how to model the environmental forces acting on a ship. In the last two sections, the theory behind the extended Kalman filter and how it is used to estimate different states of a system is described, along with relevant machine learning methods.

2.1 Levels of Autonomy

When developing regulations for autonomous ships, it is convenient to classify systems depending on their level of autonomy (LOA). However, defining these levels has proved to be a complicated task, and several classifications of LOAs, with various amount of levels, have been defined. A traditional way of defining LOA is to classify how the responsibility is shared between a human and the automation system, and how independently the automation system is able to operate. In Rødseth (2019), however, a new classification is proposed where the LOA is defined in terms of *operational complexity*, *degree of automation*, and *operator presence*, as shown in Figure 2.1. The term automation is here used to describe the abilities a system has to implement functions traditionally performed by humans, while autonomy characterizes a system that to some level can operate independently of human operators.

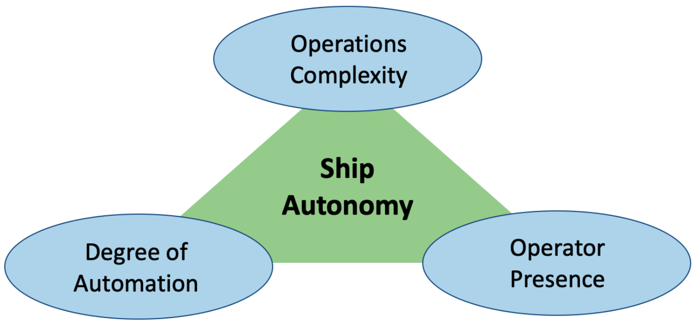

Figure 2.1: Autonomy as a function of three main factors (Rødseth 2019)

Many systems can have a high degree of automation if the tasks are simple enough, and the complexity of the operation is included to take this into account. Rødseth (2019) defines the concept of the Operational Design Domain, as state-space containing all the expected system states, and the Dynamic Ship Tasks, as the set of tasks that the automation system or operator must be able to perform to satisfy the Operational Design Domain. The degree of automation is further proposed to be divided into five cases, defined by the need for the presence of a human at the control station, and not by which tasks the human has at the control station.

DA0 – Operator Controlled: The human is always in control of the operations, but some limited automation and decision support systems are available.

DA1 - Automatic: The human must use its own judgment to decide how long he or she may be away from the control station. Some more advanced automation systems are available.

DA2 - Partial Autonomy: The degree of automation is higher than at level DA1, and the human operator must evaluate how much attention is required.

DA3 - Constrained Autonomous: Similar degree of automation as in level DA2, but limits are set to define the capabilities of the system, which again enables the system to detect when these limits are exceeded. The exceedance of a limit results in an alert to the operator, which has a maximum time limit before he or she needs to take control of the system.

DA4 - Fully Autonomous: The full Operational Design Domain can be handled by the ship automation, and fallback to a minimum risk condition can be done without the intervention of a crew.

Unmanned ships are assumed to have some remote monitoring and control, and the operator presence is therefore involved in the characterization of ship autonomy. The presence of a crew is in Rødseth (2019) defined by four categories, where a combination of both a local and a remote crew is assumed, and the level of the remote crew may be different from the level of the local crew. If both a remote and a local crew are used, one of them should be assigned as the one in charge, having the main responsibility to intervene if needed.

0 - None: There is nobody available to man the control position.

1 - Backup: Someone is available to operate the control position but they are not present. There will, therefore, be a latency as they need to be called in.

2 - Available: Someone is available at the control position, but is not actively controlling the ship. They may be in charge of monitoring several ships, and some latency is expected, but shorter than at level 1.

3 - In control: Someone is in charge of actively controlling the ship.

2.2 Situation Awareness

Achieving autonomous ships, which are at least as safe as conventional ships, necessitate a sufficient perception of the surroundings in a way it has previously been achieved by humans. Situation awareness (SA) can be defined as *"the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future"*, (Endsley 1988). Three levels of SA have also been defined, consisting of *perception*, *comprehension*, and *projection* (Endsley and Garland 2000).

Based on the various levels of SA, Sharma et al. (2019) have analyzed the information requirements for navigators at each SA level. The same requirements will likely be present to achieve SA for an autonomous vessel. At level one, perception, information about the ship status, for instance, position and speed, the operational status of equipment, the planned route together with traffic and obstacles, and weather conditions are needed. The second level of SA, comprehension, consists of interpreting the information received at the first level. This may be achieved by looking at the deviations between the current and ideal state and understand the impact of certain events. At level three, projection, the position of the vessel, and the surrounding traffic as well as the weather condition, need to be predicted. In Figure 2.2, Sharma et al. (2019) have illustrated the connection between the three levels of SA.

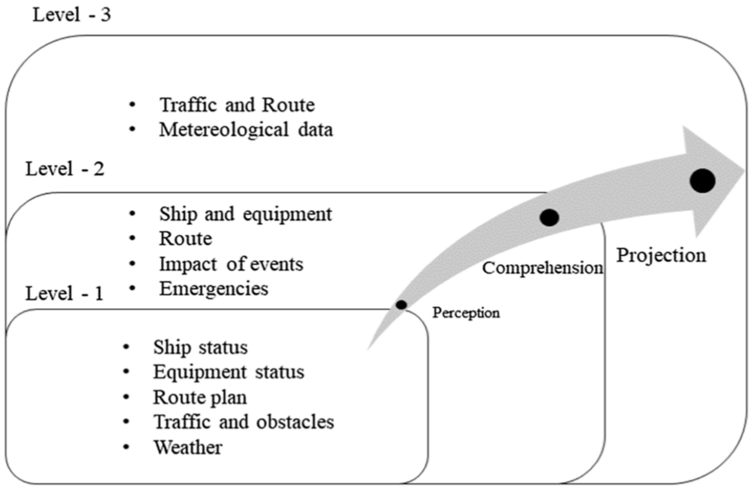

Figure 2.2: Illustration of the information requirements for the three levels of SA (Sharma et al. 2019)

2.3 Methods for Current Estimation

Although the current estimation is often ignored due to the difficulty of distinguishing it from unmodeled dynamics, an estimate of the current velocity should be a part of the perception of the weather, categorized as level 1 SA by Sharma et al. (2019). Some efforts have been made with regards to current estimation, but the goal is often to counteract the disturbance rather than knowing its velocity and direction. This section presents a brief literature review of methods used to estimate the current, together with machine learning methods used to estimate similar problems within other fields, which might be applicable to perform the current estimation.

In traditional dynamic positioning (DP) control, the current is often included in the modeling of the bias. The bias is often modeled by a Markov process. This process approximates the unmodeled dynamics, as well as the mean and slowly-varying forces due to wave, wind, and currents (Hassani et al. 2012). This approach is well suited to estimate position and velocity states used in the controllers to generate the desired control forces, but may give imprecise information regarding the velocity and direction of the current.

In Refsnes et al. (2007), the current estimation is performed for a slender body underwater vehicle. This is done by creating two control plant models, where the first one consists of a traditional nonlinear model, and the second one is an approximated current-induced vessel model created to account for the main effects of the current loads. Two separate Luenberger observers are then used for each control plant model, where one of them gives an estimate for the current velocities. The incorporation of the current estimates results in a model with more robustness related to the unknown current disturbance.

To perform straight-line path following in environments with unknown current disturbances, a current estimation algorithm can be developed by looking at the cross-track error in a modified line-of-sight (LOS) guidance algorithm together with an adaptive observer for the current (Lekkas and Fossen 2014). In Paliotta and Pettersen (2016), a Luenberger observer is used for current estimation, and based on a model including both the kinematics and dynamics of the vehicle and using cascaded systems theory, almost semi-global asymptotic stability (almost-SGAS) has been proved. Both of these methods take advantage of the structure of the guidance algorithm in order to estimate the current, which unfortunately makes the current estimation algorithm difficult to implement for systems using other guidance techniques.

For autonomous underwater vehicles (AUVs), efforts have been made in terms of current estimation to provide possibilities of model-aided inertial navigation system (INS), to avoid getting a large position error drift. The velocity of the AUV is estimated using a mathematical model, which again depends on accurate parameters representing the AUV and external forces acting on it. In Kim et al. (2018), a real-time model identification of the mathematical AUV model is performed in order to account for the change in the parameters due to the environmental forces. A nonlinear high-gain observer is further used to estimate the relative velocities, which through the knowledge of the vehicle's velocities leads to en estimate of the current velocity. Hegrenæs et al. (2007) propose a least-squares identification algorithm for maneuvering characteristics of an AUV. By including the unknown current in the model, an estimate of the current velocity will be identified simultaneously. In Martinez et al. (2015) a simplified 3 DOF dynamic model of an AUV is used together with a Kalman filter to develop a model-aided INS. The sea current's characteristics must be estimated by experiment before each mission in order to have a satisfactory performance of the dynamic model, but an observer is further used to provide a real-time estimation of the current velocity.

2.3.1 Estimation Using Machine Learning

Machine learning techniques are being more and more adopted in the industry, and some of these methods are employed to measure the environment around or disturbances affecting a system. These methods may be relevant in terms of estimating the current velocities.

In Goff et al. (2000) the airspeed around a helicopter is estimated using a neural network where the inputs consist of various internal measurements such as control positions and airframe attitudes and rates. The advantage of using these inputs is that it avoids relying on the pitot-static system to measure the airspeed, as this measuring is only possible in the direction of the helicopter and the airflow has to be sufficiently high. Another example is for a small unmanned aircraft, found in Borup (2018), where small low-cost MEMS-based pressure sensors are used together with both a linear regression method and a neural network method to estimate the airspeed. Radial Basis Function Networks (RBFNs) have also gained a lot of interest in terms of disturbance observers. In Lee and Blaabjerg (2007) an RBFN is used to measure the disturbance of a servo system, and Li et al. (2014) use an RBFN to design a multi-input-multi-output neural network disturbance observer.

2.4 Mathematical Ship Modeling

The dynamics of a system can be divided into two parts. The kinematics, which takes the geometrical aspects of motion into account, and kinetics, which analyses the forces creating the motion (Fossen 2011).

2.4.1 Kinematics

Reference Frames
To explain how a vessel behaves, different reference frames are often used depending on the situation. The most common reference frames for a ship operating in

relatively small distances are the North-East-Down (NED) coordinate system and the body-fixed reference frame.

In the NED reference frame, the position of the vessel is defined relative to a chosen origin o_n. The NED coordinate frame is then defined as a tangent plane on the surface of the Earth, with the x-axis pointing towards the North, y-axis pointing towards the East, and the z-axis pointing downwards normal to the surface of the Earth (Fossen 2011). For a vessel moving over small distances, the NED reference frame can be used for navigation and is often called flat-Earth navigation. The reference frame can then be assumed to be inertial, allowing Newton's laws to still be valid.

The body-fixed reference frame moves along with the vessel, and the origin, o_b, often referred to as CO (coordinate origin), is defined at a fixed location on the vessel. The x-axis, x_b, is positive forwards, the y-axis, y_b, is positive to the starboard, and the z-axis is positive downwards. The relationship between the NED and body-fixed reference frame in two dimensions is shown in Figure 2.3, where $\boldsymbol{p}_{b/n}^{n}$ is the position vector describing the distance between the NED and body-fixed coordinate systems, expressed in NED coordinates.

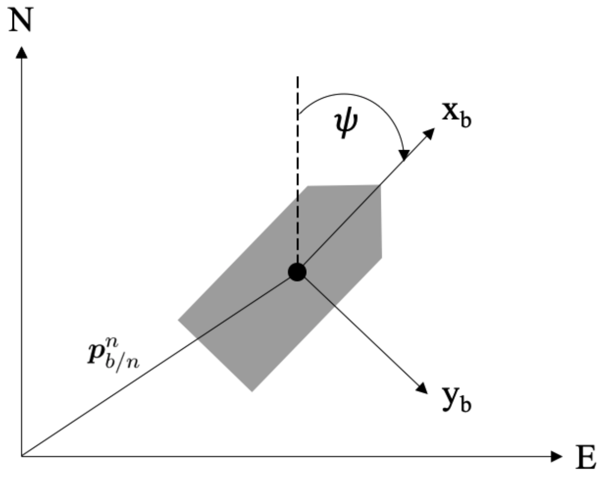

Figure 2.3: Relationship between NED and body-fixed reference frame in two dimensions

The Earth-fixed position and orientation, as well as the body-fixed translation and rotation velocities of a vessel free to move in six degrees of freedom (DOFs), can be expressed using the notation defined by SNAME (1950). The notation is presented in Table 2.1 and the velocities in the body-fixed reference frame are illustrated in Figure 2.4.

Table 2.1: The Notation of Marine Vessels (SNAME 1950).

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

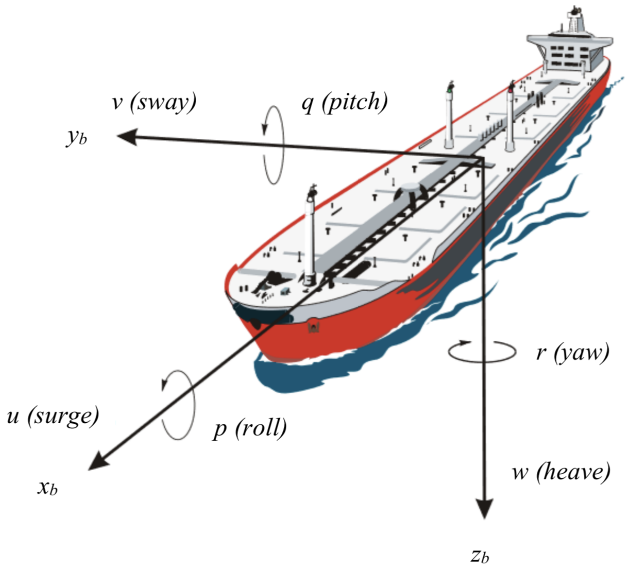

Figure 2.4: The linear and angular velocities of a six DOF vessel in the body-fixed reference frame (Fossen 2011).

In Fossen (2011), the notation explained above is conveniently expressed as vectors. The symbols corresponding to the vectors are explained in Table 2.2, where

$\mathbb{R}^3$ is the *Euclidean space* of dimension three and the set $\mathcal{S}^3$ is a sphere with three angles defined in the interval $[0, 2\pi]$.

Table 2.2: Vessel notations expressed in vectors (Fossen 2011)

NED position	$\boldsymbol{p}^n_{b/n} = \begin{bmatrix} N \\ E \\ D \end{bmatrix} \in \mathbb{R}^3$	Attitude (Euler Angles)	$\boldsymbol{\Theta}_{nb} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \in \mathcal{S}^3$
Body-fixed linear velocity	$\boldsymbol{v}^b_{b/n} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \in \mathbb{R}^3$	Body-fixed angular velocity	$\boldsymbol{\omega}^b_{b/n} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \in \mathbb{R}^3$

The motion of a 6 DOF vessel can then be expressed by the two following vectors

$$\boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{p}^n_{b/n} \\ \boldsymbol{\Theta}_{nb} \end{bmatrix}, \quad \boldsymbol{\nu} = \begin{bmatrix} \boldsymbol{v}^b_{b/n} \\ \boldsymbol{\omega}^b_{b/n} \end{bmatrix} \tag{2.1}$$

A connection between the two coordinate systems can be made using rotation matrices. The rotation matrices are in the *special orthogonal group of order 3, SO(3)*, with the properties of being orthogonal and having a determinant equal to one. The *SO(3)* group is also a subset of all *orthogonal group of order 3, O(3)*, which in addition to orthogonality has the property that $\mathbf{R}\mathbf{R}^\top = \mathbf{R}^\top\mathbf{R} = \mathbf{I}$, which leads to $\mathbf{R}^{-1} = \mathbf{R}^\top$.

Rotation about one axis is described by the principal rotation matrices

$$\mathbf{R}_{x,\phi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathrm{c}\phi & \mathrm{s}\phi \\ 0 & \mathrm{s}\phi & \mathrm{c}\phi \end{bmatrix}, \quad \mathbf{R}_{y,\theta} = \begin{bmatrix} \mathrm{c}\theta & 0 & \mathrm{s}\theta \\ 0 & 1 & 0 \\ -\mathrm{s}\theta & 0 & \mathrm{c}\theta \end{bmatrix}, \quad \mathbf{R}_{z,\psi} = \begin{bmatrix} \mathrm{c}\psi & -\mathrm{s}\psi & 0 \\ \mathrm{s}\psi & \mathrm{c}\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.2}$$

where $\mathrm{c}(\cdot) = \cos(\cdot)$ and $\mathrm{s}(\cdot) = \sin(\cdot)$. By using the Euler angles and the principal rotation matrices, the rotation matrix decomposing the velocity vectors from the body-fixed reference frame to NED, $\mathbf{R}^n_b(\boldsymbol{\Theta}_{nb})$, is conventionally performed by rotations about the z-, y-, and x-axes.

$$\mathbf{R}^n_b(\boldsymbol{\Theta}_{nb}) := \mathbf{R}_{z,\psi}\mathbf{R}_{y,\theta}\mathbf{R}_{x,\phi} = \begin{bmatrix} \mathrm{c}\psi\mathrm{c}\theta & -\mathrm{s}\psi\mathrm{c}\phi + \mathrm{c}\psi\mathrm{s}\theta\mathrm{s}\phi & \mathrm{s}\psi\mathrm{s}\phi + \mathrm{c}\psi\mathrm{c}\phi\mathrm{s}\theta \\ \mathrm{s}\psi\mathrm{c}\theta & \mathrm{c}\psi\mathrm{c}\phi + \mathrm{s}\psi\mathrm{s}\theta\mathrm{s}\phi & -\mathrm{c}\psi\mathrm{s}\phi + \mathrm{s}\theta\mathrm{s}\psi\mathrm{c}\phi \\ -\mathrm{s}\theta & \mathrm{c}\theta\mathrm{s}\phi & \mathrm{c}\theta\mathrm{c}\phi \end{bmatrix} \tag{2.3}$$

The body-fixed velocity vector, $\boldsymbol{v}_{b/n}^{b}$, can be expressed by NED velocity vector $\dot{\boldsymbol{p}}_{b/n}^{n}$, by performing the following transformation

$$\dot{\boldsymbol{p}}_{b/n}^{n} = \mathbf{R}_{b}^{n}(\boldsymbol{\Theta}_{nb})\boldsymbol{v}_{b/n}^{b} \tag{2.4}$$

From Fossen (2011) the relationship between the body-fixed angular velocity vector $\boldsymbol{\omega}_{b/n}^{b} = [p, q, r]^{\top}$ and the Euler rate vector $\dot{\boldsymbol{\Theta}}_{nb} = [\dot{\phi}, \dot{\theta}, \dot{\psi}]^{\top}$ is found through the transformation matrix $\boldsymbol{T}_{\Theta}(\boldsymbol{\Theta}_{nb})$

$$\dot{\boldsymbol{\Theta}}_{nb} = \boldsymbol{T}_{\Theta}(\boldsymbol{\Theta}_{nb})\boldsymbol{\omega}_{b/n}^{b}, \quad \boldsymbol{T}_{\Theta}(\boldsymbol{\Theta}_{nb}) = \begin{bmatrix} 1 & \mathrm{s}\phi\mathrm{t}\theta & \mathrm{c}\phi\mathrm{t}\theta \\ 0 & \mathrm{c}\phi & -\mathrm{s}\phi \\ 0 & \mathrm{s}\phi/\mathrm{c}\theta & \mathrm{c}\phi/\mathrm{c}\theta \end{bmatrix}, \tag{2.5}$$

where $\mathrm{c}(\cdot) = \cos(\cdot)$, $\mathrm{s}(\cdot) = \sin(\cdot)$, and $\mathrm{t}(\cdot) = \tan(\cdot)$. $\boldsymbol{T}_{\Theta}(\boldsymbol{\Theta}_{nb})$ is not defined for a pitch angle of $\theta = \pm 90^{\circ}$, and therefore $\boldsymbol{T}_{\Theta}^{-1}(\boldsymbol{\Theta}_{nb}) \neq \boldsymbol{T}_{\Theta}^{\top}(\boldsymbol{\Theta}_{nb})$.

These transformations can be expressed in vector form as

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}_{\Theta}(\boldsymbol{\eta})\boldsymbol{\nu} \tag{2.6a}$$

$$\begin{bmatrix} \dot{\boldsymbol{p}}_{b/n}^{n} \\ \dot{\boldsymbol{\Theta}}_{nb} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{b}^{n}(\boldsymbol{\Theta}_{nb}) & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \boldsymbol{T}_{\Theta}(\boldsymbol{\Theta}_{nb}) \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_{b/n}^{b} \\ \boldsymbol{\omega}_{b/n}^{b} \end{bmatrix} \tag{2.6b}$$

A common simplification for surface vessels is to express them in terms of 3 DOF. The motions considered are only in surge, sway, and yaw, and are based on the assumption that ϕ and θ are small. The transformation matrices can then be expressed as $\mathbf{R}_{b}^{n}(\boldsymbol{\Theta}_{nb}) = \mathbf{R}_{z,\psi}\mathbf{R}_{y,\theta}\mathbf{R}_{x,\phi} \approx \mathbf{R}_{z,\psi} = \mathbf{R}(\psi)$ and $\boldsymbol{T}_{\Theta}(\boldsymbol{\Theta}_{nb}) \approx \boldsymbol{I}_{3\times 3}$. By neglecting heave, roll and pitch, and using the vectors $\boldsymbol{\eta} = [N, E, \psi]^{\top}$ and $\boldsymbol{\nu} = [u, v, r]^{\top}$ we can express Equation (2.6) as

$$\dot{\boldsymbol{\eta}} = \boldsymbol{R}(\psi)\boldsymbol{\nu}, \tag{2.7}$$

where $\boldsymbol{R}(\psi) = \boldsymbol{R}_{z,\psi}$ in Equation (2.2).

2.4.2 Kinetics

From Fossen (2011) the kinetic equation can be expressed as in Equation (2.8), where the symbols are explained in Table 2.3.

$$\underbrace{\boldsymbol{M}_{RB}\dot{\boldsymbol{\nu}} + \boldsymbol{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu}}_{\text{rigid-body forces}} + \underbrace{\boldsymbol{M}_{A}\dot{\boldsymbol{\nu}}_r + \boldsymbol{C}_{A}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \boldsymbol{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r}_{\text{hydrodynamic forces}} + \underbrace{\boldsymbol{g}(\boldsymbol{\eta}) + \boldsymbol{g}_0}_{\text{hydrostatic forces}} = \boldsymbol{\tau} + \boldsymbol{\tau}_{wind} + \boldsymbol{\tau}_{wave} \quad (2.8)$$

The relative velocity vector, $\boldsymbol{\nu}_r$, is given by subtracting the current velocity vector, $\boldsymbol{\nu}_c$,

$$\boldsymbol{\nu}_r = \boldsymbol{\nu} - \boldsymbol{\nu}_c. \quad (2.9)$$

For irrotational ocean currents, we get the relative velocity $\boldsymbol{\nu}_r = [u - u_c, v - v_c, w - w_c, p, q, r]^\top$.

Table 2.3: Description of symbols in Equation (2.8) (Fossen 2011)

Symbol	Explanation
$\boldsymbol{\nu} \in \mathbb{R}^{6\times1}$	velocity vector in body-frame
$\boldsymbol{\nu}_r \in \mathbb{R}^{6\times1}$	relative velocity vector in body-frame
$\boldsymbol{M}_{RB} \in \mathbb{R}^{6\times6}$	rigid body inertia matrix
$\boldsymbol{M}_{A} \in \mathbb{R}^{6\times6}$	added mass matrix
$\boldsymbol{C}_{RB}(\boldsymbol{\nu}) \in \mathbb{R}^{6\times6}$	rigid body Coriolis and centripetal matrix
$\boldsymbol{C}_{A}(\boldsymbol{\nu}_r) \in \mathbb{R}^{6\times6}$	added mass Coriolis and centripetal matrix
$\boldsymbol{D}(\boldsymbol{\nu}_r) \in \mathbb{R}^{6\times6}$	damping matrix
$\boldsymbol{g}(\boldsymbol{\eta}) \in \mathbb{R}^{6\times1}$	vector of gravitational and buoyancy forces and moments
$\boldsymbol{g}_0 \in \mathbb{R}^{6\times1}$	vector pretrimming and ballast forces and moments
$\boldsymbol{\tau} \in \mathbb{R}^{6\times1}$	vector of control inputs
$\boldsymbol{\tau}_{wind} \in \mathbb{R}^{6\times1}$	vector of wind forces
$\boldsymbol{\tau}_{wave} \in \mathbb{R}^{6\times1}$	vector of wave induced forces

For surface vessels, Equation (2.8) can be written as the maneuvering model

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{N}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r = \boldsymbol{\tau} + \boldsymbol{\tau}_{\text{wind}} + \boldsymbol{\tau}_{\text{wave}}, \quad (2.10)$$

where $\boldsymbol{N}(\boldsymbol{\nu}_r) = \boldsymbol{C}_A(\boldsymbol{\nu}_r) + \boldsymbol{D}(\boldsymbol{\nu}_r)$. According to Fossen (2011), Equation (2.10) can be expressed completely with only the relative velocity if the rigid-body Coriolis and centripetal matrix, $\boldsymbol{C}_{RB}(\boldsymbol{\nu}_r)$, is parametrized independent of linear velocity. If the ocean current is irrotational and constant as well, the rigid-body kinetic satisfies

$$\mathbf{M}_{RB}\dot{\boldsymbol{\nu}} + \mathbf{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu} = \mathbf{M}_{RB}\dot{\boldsymbol{\nu}}_r + \mathbf{C}_{RB}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r. \tag{2.11}$$

This results in the equation of motion being written as

$$\boldsymbol{M}\dot{\boldsymbol{\nu}}_r + \boldsymbol{N}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r = \boldsymbol{\tau} + \boldsymbol{\tau}_{\text{wind}} + \boldsymbol{\tau}_{\text{wave}}, \tag{2.12}$$

where $\boldsymbol{M} = \boldsymbol{M}_A + \boldsymbol{M}_{RB}$ and $\boldsymbol{N}(\boldsymbol{\nu}_r) = \boldsymbol{C}_A(\boldsymbol{\nu}_r) + \boldsymbol{C}_{RB}(\boldsymbol{\nu}_r) + \boldsymbol{D}(\boldsymbol{\nu}_r)$.

Fossen (2011) then states that for DP vessels and ships moving on a straight-line path, $\boldsymbol{\omega}^b_{b/n} \approx \mathbf{0}$, the acceleration of the current is negligible, $\dot{\boldsymbol{\nu}}_c \approx \mathbf{0}$, resulting in the following equation of motion

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{N}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r = \boldsymbol{\tau} + \boldsymbol{\tau}_{\text{wind}} + \boldsymbol{\tau}_{\text{wave}} \tag{2.13}$$

The 3 DOF kinematic and kinetic equation can then be summarized as

$$\dot{\boldsymbol{\eta}} = \mathbf{R}(\psi)\boldsymbol{\nu} \tag{2.14a}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{N}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r = \boldsymbol{\tau} + \boldsymbol{\tau}_{\text{wind}} + \boldsymbol{\tau}_{\text{wave}} \tag{2.14b}$$

For a 3 DOF model, we have the rigid-body inertia matrix and the added mass matrix

$$\boldsymbol{M}_{RB} = \begin{bmatrix} m & 0 & -my_g \\ 0 & m & mx_g \\ -my_g & mx_g & I_z \end{bmatrix}, \quad \boldsymbol{M}_A = \begin{bmatrix} -X_{\dot{u}} & -X_{\dot{v}} & -X_{\dot{r}} \\ -Y_{\dot{u}} & -Y_{\dot{v}} & -Y_{\dot{r}} \\ -N_{\dot{u}} & -N_{\dot{v}} & -N_{\dot{r}} \end{bmatrix} \tag{2.15}$$

These matrices can be combined in a total mass matrix

$$\boldsymbol{M} = \begin{bmatrix} m - X_{\dot{u}} & -X_{\dot{v}} & -my_g - X_{\dot{r}} \\ -Y_{\dot{u}} & m - Y_{\dot{v}} & mx_g - Y_{\dot{r}} \\ -my_g - N_{\dot{u}} & mx_g - N_{\dot{v}} & I_z - N_{\dot{r}} \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \tag{2.16}$$

If it is assumed that a vessel has *xz*-plane symmetry, and a homogeneous mass distribution, the CO can be set at the centerline of the vessel, resulting in $y_g = 0$ (Fossen 2011). If it is also assumed that the added mass is computed in CO, the added mass terms $X_{\dot{v}}$, $X_{\dot{r}}$, $Y_{\dot{u}}$, and $N_{\dot{u}}$ are zero, resulting in a surge-decoupled mass matrix, where surge is decoupled from sway and yaw. The total rigid-body and added mass matrix can then be presented as

$$\boldsymbol{M} = \begin{bmatrix} m - X_{\dot{u}} & 0 & 0 \\ 0 & m - Y_{\dot{v}} & mx_g - Y_{\dot{r}} \\ 0 & mx_g - N_{\dot{v}} & I_z - N_{\dot{r}} \end{bmatrix}. \tag{2.17}$$

From Equation (3.60), (6.45), and (6.46) in Fossen (2011) we have that

$$\boldsymbol{C}_{RB}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & -m(x_g r + v) \\ 0 & 0 & -m(y_g r - u) \\ m(x_g r + v) & m(y_g r - u) & 0 \end{bmatrix} \tag{2.18a}$$

$$\boldsymbol{C}_{A}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & Y_{\dot{u}}u + Y_{\dot{v}}v + Y_{\dot{r}}r \\ 0 & 0 & -X_{\dot{u}}u - X_{\dot{v}}v - X_{\dot{r}}r \\ \begin{matrix}-Y_{\dot{u}}u - Y_{\dot{v}}v \\ -Y_{\dot{r}}r\end{matrix} & \begin{matrix}X_{\dot{u}}u + X_{\dot{v}}v \\ +X_{\dot{r}}r\end{matrix} & 0 \end{bmatrix} \tag{2.18b}$$

$$\boldsymbol{C}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & \begin{matrix}-m(x_g r + v)+ \\ Y_{\dot{u}}u + Y_{\dot{v}}v + Y_{\dot{r}}r\end{matrix} \\ 0 & 0 & \begin{matrix}-m(y_g r - u)- \\ X_{\dot{u}}u - X_{\dot{v}}v - X_{\dot{r}}r\end{matrix} \\ \begin{matrix}m(x_g r + v)- \\ Y_{\dot{u}}u - Y_{\dot{v}}v - Y_{\dot{r}}r\end{matrix} & \begin{matrix}m(y_g r - u)+ \\ X_{\dot{u}}u + X_{\dot{v}}v + X_{\dot{r}}r\end{matrix} & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -m_{21}u - m_{22}v - m_{23}r \\ 0 & 0 & m_{11}u + m_{12}v + m_{13}r \\ \begin{matrix}m_{21}u + m_{22}v \\ +m_{23}r\end{matrix} & \begin{matrix}-m_{11}u - m_{12}v \\ -m_{13}r\end{matrix} & 0 \end{bmatrix} \tag{2.19}$$

A surge-decoupled 3 DOF Coriolis and centripetal matrix is also presented in Fossen (2011), where the terms y_g, $Y_{\dot{u}}$, $X_{\dot{v}}$, and $X_{\dot{r}}$ are set to zero in Equations (2.18) and (2.19).

A linear velocity-independent parametrization of $\boldsymbol{C}_{RB}$ can be represented as

$$\boldsymbol{C}_{RB}(\boldsymbol{\nu}) = \begin{bmatrix} m\boldsymbol{S}(\boldsymbol{\nu}_2) & -m\boldsymbol{S}(\boldsymbol{\nu}_2)\boldsymbol{S}(\boldsymbol{r}_g^b) \\ m\boldsymbol{S}(\boldsymbol{r}_g^b)\boldsymbol{S}(\boldsymbol{\nu}_2) & \boldsymbol{S}(\boldsymbol{I}_b^b\boldsymbol{\nu}_2) \end{bmatrix} \tag{2.20}$$

where $\boldsymbol{\nu}_2 = [p, q, r]^\top$, $\boldsymbol{r}_g^b = [x_g, y_g, z_g]^\top$ is the center of gravity, and $\boldsymbol{I}_b^b$ is the inertia tensor with respect to the origin, and $\boldsymbol{S}$ is the cross product

$$\boldsymbol{I}_b^b = \begin{bmatrix} I_x & -I_{xy} & -I_{xz} \\ -I_{yx} & I_y & -I_{yz} \\ -I_{zx} & -I_{zy} & I_z \end{bmatrix}, \quad \boldsymbol{S}(a) = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{2.21}$$

For a 3 DOF vessel, we then get a rigid-body Coriolis and centripetal matrix independent of the linear velocity, which is useful in the presence of currents

$$\boldsymbol{C}_{RB}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & -mr & -mx_g r \\ mr & 0 & -my_g r \\ mx_g r & my_g r & 0 \end{bmatrix} \tag{2.22}$$

From Equation (6.51) in Fossen (2011), we have that the added mass Coriolis and centripetal matrix with regards to the relative velocity can be written as

$$\boldsymbol{C}_A(\boldsymbol{\nu}_r) = \begin{bmatrix} 0 & 0 & Y_{\dot{v}}v_r + Y_{\dot{r}}r \\ 0 & 0 & -X_{\dot{u}}u_r \\ -Y_{\dot{v}}v_r - Y_{\dot{r}}r & X_{\dot{u}}u_r & 0 \end{bmatrix} \tag{2.23}$$

Giving the resulting Coriolis and centripetal matrix

$$\boldsymbol{C}(\boldsymbol{\nu}_r) = \boldsymbol{C}_{RB}(\boldsymbol{\nu}_r) + \boldsymbol{C}_A(\boldsymbol{\nu}_r) = \begin{bmatrix} 0 & -mr & -mx_gr + Y_{\dot{v}}v_r + Y_{\dot{r}}r \\ mr & 0 & -my_gr - X_{\dot{u}}u_r \\ mx_gr - Y_{\dot{v}}v_r - Y_{\dot{r}}r & my_gr + X_{\dot{u}}u_r & 0 \end{bmatrix} \tag{2.24}$$

It is often convenient to divide the hydrodynamic damping matrix into one linear damping matrix $\boldsymbol{D}$, which comes from potential damping and possible skin friction, and the nonlinear damping matrix, $\boldsymbol{D}_n(\boldsymbol{\nu}_r)$, which is due to quadratic damping and higher-order terms (Fossen 2011). For a 3 DOF vessel with decoupled surge dynamics, we get

$$\boldsymbol{D}(\boldsymbol{\nu}_r) = \boldsymbol{D} + \boldsymbol{D}_n(\boldsymbol{\nu}_r) \tag{2.25a}$$

$$\boldsymbol{D} = -\begin{bmatrix} X_u & 0 & 0 \\ 0 & Y_v & Y_r \\ 0 & N_v & N_r \end{bmatrix} \tag{2.25b}$$

$$\boldsymbol{D}_n(\boldsymbol{\nu}_r) = -\begin{bmatrix} \begin{matrix} X_{|u|u}|u_r| + \\ X_{uuu}u_r^2 \end{matrix} & 0 & 0 \\ 0 & \begin{matrix} Y_{|v|v}|v_r| + Y_{|r|v}|r| \\ +Y_{vvv}v_r^2 \end{matrix} & Y_{|v|r}|v_r| + Y_{|r|r}|r| \\ 0 & N_{|v|v}|v_r| + N_{|r|v}|r| & \begin{matrix} N_{|v|r}|v_r| + N_{|r|r}|r| \\ +N_{rrr}r^2 \end{matrix} \end{bmatrix} \tag{2.25c}$$

2.5 Environmental Forces and Moments

Environmental forces and disturbances will always be present in a real system. Information about these forces and moments are needed to obtain improved situation awareness and is needed to explain the kinetics of the system, as expressed in Equation (2.8). Common forces and moments acting on a marine surface vessel stems from wind, waves, and ocean currents. In this thesis, the studied vessel is assumed to be operating in a channel, and the modeling of the waves will, therefore, not be further discussed. The modeling of wind and current forces and moments are presented below.

2.5.1 Wind

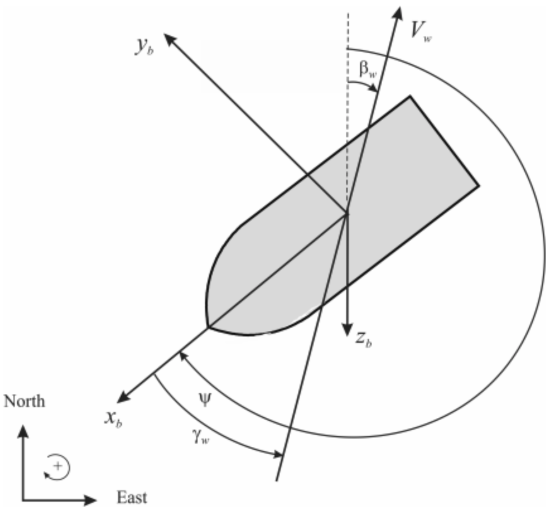

Figure 2.5: Illustration of the wind speed, V_w, wind direction, β_w, and the wind angle of attack γ_w (Fossen 2011).

From Figure 2.5, we see that the wind speed, V_w, can be defined with regards to the direction the wind is going to, β_w, in the NED-frame, and the angle of attack is called γ_w. For a moving ship, it is useful to define the wind forces in terms of the relative wind speed, V_{rw}, and the relative angle of attack γ_{rw}. For a 3-DOF vessel, the forces in the body-fixed reference frame can according to Fossen (2011) be expressed as

$$\boldsymbol{\tau}_{wind} = \frac{1}{2}\rho_a V_{rw}^2 \begin{bmatrix} C_X(\gamma_{rw})A_{Fw} \\ C_Y(\gamma_{rw})A_{Lw} \\ C_N(\gamma_{rw})A_{Lw}L_{oa} \end{bmatrix}, \tag{2.26}$$

where A_{Fw} and A_{Lw} are the frontal and lateral projected areas of the vessel above the waterlines, L_{oa} is the length of the vessel, and ρ_a is the air density. The relative wind speed, V_{rw}, and relative angle of attack, γ_{rw}, are defined in terms of the relative velocities u_{rw} and v_{rw}.

$$V_{rw} = \sqrt{u_{rw}^2 + v_{rw}^2} \tag{2.27a}$$
$$\gamma_{rw} = -\text{atan2}(v_{rw}, u_{rw}) \tag{2.27b}$$
$$u_{rw} = u - u_w, \;\; u_w = V_w \text{cos}(\beta_w - \psi) \tag{2.27c}$$
$$v_{rw} = v - v_w, \;\; v_w = V_w \text{sin}(\beta_w - \psi) \tag{2.27d}$$

For symmetrical ships with respect to the xz- and yz-plane, the wind coefficients can be approximated as in Equation (2.28), where the ranges for c_x, c_y, and c_n are found through experiments (Fossen 2011).

$$C_X(\gamma_{rw}) \approx -c_x \text{cos}(\gamma_{rw}), \;\; c_x \in [0.50, 0.90] \tag{2.28a}$$
$$C_Y(\gamma_{rw}) \approx c_y \text{sin}(\gamma_{rw}), \;\; c_y \in [0.70, 0.95] \tag{2.28b}$$
$$C_N(\gamma_{rw}) \approx c_n \text{sin}(2\gamma_{rw}), \;\; c_n \in [0.05, 0.20] \tag{2.28c}$$

2.5.2 Current

Ocean currents occur because of gravity, wind friction, and variation in the density of the water (Fossen 2011). The modeling of current can be divided into two parts, consisting of the surface current, to use when modeling surface vessel response, and full current profile, to use when modeling risers and anchor lines, among other things (Sørensen 2013). For surface vessels, it is sufficient to model the current in two dimensions, which can be described by its magnitude, V_c and its direction in the NED frame ψ_c. The velocity vector of the current in NED coordinates can then be written as

$$\boldsymbol{\nu}_c = \begin{bmatrix} V_c \text{cos}(\psi_c) \\ V_c \text{sin}(\psi_c) \\ 0 \end{bmatrix} = \begin{bmatrix} V_{c,x} \\ V_{c,y} \\ 0 \end{bmatrix} \tag{2.29}$$

To simulate the ocean currents, the ocean current velocity can be modeled as a first-order *Gauss-Markov process* (Fossen 2011)

$$\dot{V}_c + \mu V_c = w, \tag{2.30}$$

where w is Gaussian white noise and $\mu \geq 0$ is a constant. The current velocity should also be restricted by saturating limits to avoid unrealistic values

$$V_{c,min} \leq V_c \leq V_{c,max}. \tag{2.31}$$

Variation in the direction of the current can in a similar manner also be implemented as a *Gauss-Markov process*

$$\dot{\psi}_c + \mu_2 \psi_c = w_2, \tag{2.32a}$$

$$\psi_{c,min} \leq \psi_c \leq \psi_{c,max}. \tag{2.32b}$$

2.6 Observers

In order to control a system, for example a ship explained by the equations presented in the previous chapters, measurements of the system are needed. These measurements provide information about the different states of the system, such as its position or velocity. However, it may not be possible to measure all the desired states, and observers are therefore used. According to Strand (1999), the main objectives of observers are to estimate the velocity, the bias term, and provide wave filtering. When estimating the velocity, only the position measurements are assumed known, and velocity estimates must be created without any measurements. The bias term consists of slowly-varying environmental loads as well as unmodeled dynamics and needs to be estimated to avoid steady-state offsets. Wave filtering consists of dividing the motions of a marine vessel into two parts, one consisting of the wave-frequency motion and the other of the low-frequency motion. A control system cannot counteract the fast dynamics of the wave-frequency part, and the wave-frequency motions should, therefore, be filtered out. Commonly used observers for marine vessels are Kalman filters and nonlinear passive observers. In this thesis the Kalman filter, and in particular, the extended Kalman filter (EKF) will be further studied.

2.6.1 The Kalman Filter

The derivation of the Kalman filter is retrieved from Chapter 7 in Mastro (2013) if not stated otherwise, where some modifications in terms of the notation of variables have been done. A linear system can be described in the form of the state space equations

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{E}\boldsymbol{w} \tag{2.33a}$$

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{v}. \tag{2.33b}$$

The objective of the Kalman filter is to calculate the unobserved variables x, from the observed variables y. For a discrete system, we get the discretized system model

$$\boldsymbol{x}_k = \boldsymbol{A}_k \boldsymbol{x}_{k-1} + \boldsymbol{B}_k \boldsymbol{u}_k + \boldsymbol{E}_k \boldsymbol{w}_k \tag{2.34a}$$

$$\boldsymbol{y}_k = \boldsymbol{H}_k \boldsymbol{x}_k + \boldsymbol{v}_k, \tag{2.34b}$$

where $\boldsymbol{x}$ is the state vector, $\boldsymbol{y}$ is the output vector, $\boldsymbol{u}$ is the input or control vector, $\boldsymbol{w}$ is the process noise vector, $\boldsymbol{v}$ is the measurement noise vector, $\boldsymbol{A}$ is the state or system matrix, $\boldsymbol{B}$ is the input matrix, $\boldsymbol{E}$ is the process noise gain matrix, and $\boldsymbol{H}$ is the output matrix. The process noise, $\boldsymbol{w}_k$, has a zero-mean Gaussian distribution with covariance matrix $\boldsymbol{Q}_k = \mathbb{E}[\boldsymbol{w}_k \boldsymbol{w}_k^\top]$. Similarly, the measurement noise $\boldsymbol{v}_k$, has a zero-mean Gaussian distribution with covariance matrix $\boldsymbol{R}_k = \mathbb{E}[\boldsymbol{v}_k \boldsymbol{v}_k^\top]$.

The Kalman filter recursively cycles through two stages, which is usually referred to as the prediction or propagation stage and the update or correction stage. The filter estimates the state at time step k using the information given at the previous time steps $(1, \ldots, k-1)$. The state estimate is then given by

$$\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{A}_k \hat{\boldsymbol{x}}_{k-1|k-1} + \boldsymbol{B}_k \boldsymbol{u}_k. \tag{2.35}$$

The predicted observation can then be found by inserting the state estimation in the measurement equation

$$\hat{\boldsymbol{y}}_{k|k-1} = \boldsymbol{H}_k \hat{\boldsymbol{x}}_{k|k-1}. \tag{2.36}$$

Update or Correction Step

From this, a new state estimate $\hat{\boldsymbol{x}}_{k|k}$, often called the state estimate update, can be made by adjusting the previous state estimation, $\hat{\boldsymbol{x}}_{k|k-1}$, with the error between the actual observation and the predicted observation, $\boldsymbol{y}_k - \hat{\boldsymbol{y}}_{k|k-1}$, weighted by the Kalman gain, $\boldsymbol{K}_k$.

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k \left(\boldsymbol{y}_k - \hat{\boldsymbol{y}}_{k|k-1} \right) \tag{2.37a}$$

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k \left(\boldsymbol{y}_k - \boldsymbol{H}_k \hat{\boldsymbol{x}}_{k|k-1} \right) \tag{2.37b}$$

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} \left(\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k \right) + \boldsymbol{K}_k \boldsymbol{y}_k \tag{2.37c}$$

The difference between the true state and the state estimate can be denoted as $\boldsymbol{e}_k = \boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k}$, often called the *a posteriori* error estimate. The *a posteriori* error covariance matrix $\boldsymbol{P}_{k|k}$ is then given by

$$\boldsymbol{P}_{k|k} = \operatorname{cov}\left(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k} \right) = \mathbb{E}\left[\boldsymbol{e}_k \boldsymbol{e}_k^\top \right] = \mathbb{E}\left[\left(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k} \right) \left(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k} \right)^\top \right] \tag{2.38}$$

Substituting the measurement equation, Equation (2.34b), into Equation (2.37)

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1}\left(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k\right) + \boldsymbol{K}_k\left(\boldsymbol{H}_k\boldsymbol{x}_k + \boldsymbol{v}_k\right) \tag{2.39a}$$

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1}\left(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k\right) + \boldsymbol{K}_k\boldsymbol{H}_k\boldsymbol{x}_k + \boldsymbol{K}_k\boldsymbol{v}_k. \tag{2.39b}$$

Inserting this into Equation (2.38)

$$\begin{aligned}\boldsymbol{P}_{k|k} =& \mathbb{E}[\big(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k-1}(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k) - \boldsymbol{K}_k\boldsymbol{H}_k\boldsymbol{x}_k - \boldsymbol{K}_k\boldsymbol{v}_k\big)\\ &\big(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k-1}(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k) - \boldsymbol{K}_k\boldsymbol{H}_k\boldsymbol{x}_k - \boldsymbol{K}_k\boldsymbol{v}_k\big)^\top]\end{aligned} \tag{2.40a}$$

$$\begin{aligned}\boldsymbol{P}_{k|k} =& \mathbb{E}[\big((\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k-1}) - \boldsymbol{K}_k\boldsymbol{v}_k\big)\\ &\big((\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k-1}) - \boldsymbol{K}_k\boldsymbol{v}_k\big)^\top]\end{aligned} \tag{2.40b}$$

$$\begin{aligned}\boldsymbol{P}_{k|k} =& (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\mathbb{E}\left[(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k-1})(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k-1})^\top\right]\\ &(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)^\top + \boldsymbol{K}_k\mathbb{E}\left[\boldsymbol{v}_k\boldsymbol{v}_k^\top\right]\boldsymbol{K}_k^\top\end{aligned} \tag{2.40c}$$

$$\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{P}_{k|k-1}(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)^\top + \boldsymbol{K}_k\boldsymbol{R}_k\boldsymbol{K}_k^\top \tag{2.40d}$$

where in the last transition we have used the relations $\boldsymbol{P}_{k|k-1} = \text{cov}(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k-1}) = \mathbb{E}[(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k-1})(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k-1})^\top]$ and $\boldsymbol{R}_k = \text{cov}(\boldsymbol{v}_k\boldsymbol{v}_k^\top) = \mathbb{E}[\boldsymbol{v}_k\boldsymbol{v}_k^\top]$.

The Kalman gain is defined as the gain minimizing the expected mean-squared error $\mathbb{E}[(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k})^2]$. The mean square error terms can further be found in the diagonal elements of the state error covariance matrix $\boldsymbol{P}_{k|k} = \text{cov}(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k})$. Since the trace (tr) is the sum of the diagonal elements and the diagonal elements in $\boldsymbol{P}_{k|k}$ consists of the variances, tr$(\boldsymbol{P}_{k|k})$ is the sum of the variances. The minimum mean squared error can, therefore, be found by minimizing tr$(\boldsymbol{P}_{k|k})$, and the optimal Kalman gain $\boldsymbol{K}_k$ is found by taking the first derivative of tr$(\boldsymbol{P}_{k|k})$ with regards to $\boldsymbol{K}_k$ and setting it to zero. The derivative is found by first rewriting the error covariance matrix as

$$\begin{aligned}\boldsymbol{P}_{k|k} = \boldsymbol{P}_{k|k-1} - \boldsymbol{K}_k\boldsymbol{H}_k\boldsymbol{P}_{k|k-1} - \boldsymbol{P}_{k|k-1}\boldsymbol{K}_k^\top\boldsymbol{H}_k^\top +&\\ \boldsymbol{K}_k(\boldsymbol{H}_k\boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^\top + \boldsymbol{R}_k)\boldsymbol{K}_k^\top&.\end{aligned} \tag{2.41}$$

The trace of the error covariance matrix and its derivative is then given as

$$\begin{aligned}\text{tr}\left[\boldsymbol{P}_{k|k}\right] =& \text{tr}\left[\boldsymbol{P}_{k|k-1}\right] - \text{tr}\left[\boldsymbol{K}_k\boldsymbol{H}_k\boldsymbol{P}_{k|k-1}\right] - \text{tr}\left[\boldsymbol{P}_{k|k-1}\boldsymbol{K}_k^\top\boldsymbol{H}_k^\top\right] \\ &+ \text{tr}\left[\boldsymbol{K}_k(\boldsymbol{H}_k\boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^\top + \boldsymbol{R}_k)\boldsymbol{K}_k^\top\right].\end{aligned} \tag{2.42a}$$

$$\frac{\partial\text{tr}\left[\boldsymbol{P}_{k|k}\right]}{\partial\boldsymbol{K}_k} = -2(\boldsymbol{H}_k\boldsymbol{P}_{k|k-1})^\top + 2\boldsymbol{K}_k(\boldsymbol{H}_k\boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^\top + \boldsymbol{R}_k) = 0 \tag{2.42b}$$

Solving this for the Kalman gain, $\boldsymbol{K}_k$, gives

$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^\top\left(\boldsymbol{H}_k\boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^\top + \boldsymbol{R}_k\right)^{-1} \tag{2.43}$$

Propagation or Prediction Step

The prediction of the state vector, or state estimate propagation, is given by

$$\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{A}_k\hat{\boldsymbol{x}}_{k-1|k-1} + \boldsymbol{B}_k\boldsymbol{u}_k \tag{2.44}$$

Defining the *a priori* error estimate as $\boldsymbol{e}_{k|k-1} = \boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k-1}$, we get

$$\boldsymbol{e}_{k|k-1} = \boldsymbol{A}_k\boldsymbol{x}_{k-1} + \boldsymbol{B}_k\boldsymbol{u}_k + \boldsymbol{E}_k\boldsymbol{w}_k - \boldsymbol{A}_k\hat{\boldsymbol{x}}_{k-1|k-1} - \boldsymbol{B}_k\boldsymbol{u}_k \tag{2.45a}$$

$$\boldsymbol{e}_{k|k-1} = \boldsymbol{A}_k\left(\boldsymbol{x}_{k-1} - \hat{\boldsymbol{x}}_{k-1|k-1}\right)\boldsymbol{E}_k\boldsymbol{w}_k \tag{2.45b}$$

$$\boldsymbol{e}_{k|k-1} = \boldsymbol{A}_k\boldsymbol{e}_{k-1|k-1} + \boldsymbol{E}_k\boldsymbol{w}_k \tag{2.45c}$$

The *a priori* estimate of the error covariance, $\boldsymbol{P}_{k|k-1}$, is

$$\begin{aligned}\boldsymbol{P}_{k|k-1} =& \mathbb{E}\left[\boldsymbol{e}_{k|k-1}\boldsymbol{e}_{k|k-1}^\top\right] \\ =& \mathbb{E}\left[(\boldsymbol{A}_k\boldsymbol{e}_{k-1|k-1} + \boldsymbol{E}_k\boldsymbol{w}_k)(\boldsymbol{A}_k\boldsymbol{e}_{k-1|k-1} + \boldsymbol{E}_k\boldsymbol{w}_k)^\top\right]\end{aligned} \tag{2.46a}$$

$$\begin{aligned}\boldsymbol{P}_{k|k-1} =& \boldsymbol{A}_k\mathbb{E}\left[(\boldsymbol{e}_{k-1|k-1})(\boldsymbol{e}_{k-1|k-1})^\top\right]\boldsymbol{A}_k^\top \\ &+ \boldsymbol{E}_k\mathbb{E}\left[\boldsymbol{w}_k\boldsymbol{w}_k^\top\right]\boldsymbol{E}_k^\top\end{aligned} \tag{2.46b}$$

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{A}_k\boldsymbol{P}_{k-1|k-1}\boldsymbol{A}_k^\top + \boldsymbol{E}_k\boldsymbol{Q}_k\boldsymbol{E}_k^\top \tag{2.46c}$$

2.6.2 The Extended Kalman Filter

The EKF can be applied to nonlinear systems of the form

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{E}\boldsymbol{w} \tag{2.47a}$$

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{v}. \tag{2.47b}$$

According to Mastro (2013), the EKF relies on a linearization of the nonlinear state function, allowing it to operate based on a set of prediction and correction equations similar to the traditional Kalman filter. For a nonlinear system, the state estimate propagation step given in Equation (2.44), can be written in terms of the discrete-time quantity $\mathcal{F}\left(\hat{\boldsymbol{x}}(k), \boldsymbol{u}(k)\right)$. $\mathcal{F}\left(\hat{\boldsymbol{x}}(k), \boldsymbol{u}(k)\right)$, along with the discrete-time quantities $\boldsymbol{\Phi}(k)$ and $\boldsymbol{\Gamma}(k)$, can, according to Fossen (2011), be found using *forward Euler* integration

$$\mathcal{F}\left(\hat{\boldsymbol{x}}(k), \boldsymbol{u}(k)\right) \approx \hat{\boldsymbol{x}}(k) + h\left[\mathbf{f}(\hat{\boldsymbol{x}}(k)) + \mathbf{B}\boldsymbol{u}(k)\right] \tag{2.48a}$$

$$\boldsymbol{\Phi}(k) \approx \mathbf{I} + h\frac{\partial \mathbf{f}(\boldsymbol{x}(k), \boldsymbol{u}(k))}{\partial \boldsymbol{x}(k)}\bigg|_{\boldsymbol{x}(k)=\hat{\boldsymbol{x}}(k)} \tag{2.48b}$$

$$\boldsymbol{\Gamma}(k) \approx h\mathbf{E}. \tag{2.48c}$$

By noticing the change in notation for the Kalman filter equations between Mastro (2013) and Fossen (2011) presented in Table 2.4, a summary of the discrete-time EKF algorithm is presented in Table 2.5 (Fossen 2011).

Table 2.4: Mapping of notation between Mastro (2013) and Fossen (2011)

Name	Mastro (2013)	Fossen (2011)
Discrete-time state	$\boldsymbol{A}_k$	$\boldsymbol{\Phi}(k)$
Discrete-time input matrix	$\boldsymbol{B}_k$	$\boldsymbol{\Delta}(k)$
Discrete-time process noise gain matrix	$\boldsymbol{E}_k$	$\boldsymbol{\Gamma}(k)$
Discrete-time output matrix	$\boldsymbol{H}_k$	$\boldsymbol{H}(k)$
Kalman gain matrix	$\boldsymbol{K}_k$	$\boldsymbol{K}(k)$
State estimate update	$\hat{\boldsymbol{x}}_{k\|k}$	$\hat{\boldsymbol{x}}(k)$
State estimate propagation	$\hat{\boldsymbol{x}}_{k\|k-1}$	$\bar{\boldsymbol{x}}(k)$
Error covariance update	$\boldsymbol{P}_{k\|k}$	$\hat{\boldsymbol{P}}(k)$
Error covariance propagation	$\boldsymbol{P}_{k\|k-1}$	$\bar{\boldsymbol{P}}(k)$

Table 2.5: Equations of the discrete-time extended Kalman filter. Table 11.3 in Fossen (2011).

Initial state:	$\bar{\boldsymbol{x}}(0) = \boldsymbol{x}_0$
Initial covariance error:	$\bar{\mathbf{P}}(0) = \mathbf{P_0}$
Kalman gain:	$\mathbf{K}(k) = \bar{\mathbf{P}}(k)\mathbf{H}^\top(k)\left[\mathbf{H}(k)\bar{\mathbf{P}}(k)\mathbf{H}^\top(k) + \mathbf{R}(k)\right]^{-1}$
State estimate update:	$\hat{\boldsymbol{x}}(k) = \bar{\boldsymbol{x}}(k) + \mathbf{K}(k)\left(\boldsymbol{y}(k) - \mathbf{H}(k)\bar{\boldsymbol{x}}(k)\right)$
Error covariance update:	$\hat{\mathbf{P}}(k) = \left[\mathbf{I} - \mathbf{K}(k)\mathbf{H}(k)\right]\bar{\mathbf{P}}(k)\left[\mathbf{I} - \mathbf{K}(k)\mathbf{H}(k)\right]^\top + \mathbf{K}(k)\mathbf{R}(k)\mathbf{K}^\top(k)$
State estimate propagation:	$\bar{\boldsymbol{x}}(k+1) = \mathcal{F}\left(\hat{\boldsymbol{x}}(k), \boldsymbol{u}(k)\right)$
Error covariance propagation:	$\bar{\mathbf{P}}(k+1) = \boldsymbol{\Phi}(k)\hat{\mathbf{P}}(k)\boldsymbol{\Phi}^\top(k) + \boldsymbol{\Gamma}(k)\mathbf{Q}(k)\boldsymbol{\Gamma}^\top(k)$

2.7 Machine Learning

Artificial intelligence (AI) can be defined as *the effort to automate intellectual tasks normally performed by humans* and has gained a lot of interest in recent years, even though the concept was born in the 1950s (Chollet 2018). As the definition of AI is wide, it encompasses many different technologies, where many are employed in different industries today. Different aspects of AI may be suitable for different tasks, and this thesis focuses on methods that may be applicable for the estimation of current velocity. The following theory presented in this section is found in Chollet (2018) unless stated otherwise.

2.7.1 Artificial Intelligence, Machine Learning & Deep Learning

In the beginning, the dominant part of AI consisted of *symbolic AI*, where programmers obtained results by incorporating explicit rules and human knowledge into computer programs. Machine learning, on the other hand, was developed by trying to see if the computer could do even more than what programmers knew how to make the computer do. In symbolic AI, the input consists of human-specified rules and data to apply these rules on, and the output consists of the answers. The input to machine learning, however, consists of the data combined with the answers from these data, and the output consists of the rules. These new rules can then be tested on new data in order to generate answers corresponding to these data. Three things are needed in order to perform machine learning; input data points, examples of the expected output, and a measure for how well the algorithm is performing. The goal of the machine learning algorithm is to learn how the input data can be transformed to output data, and this transformation is *learned* by exposing the algorithm to input examples where the true output is known. The machine learning algorithms searches for the optimal transformation between the input and output, and the optimal transformation is found by guiding from a feedback signal. In deep learning, the goal is to learn successive layers of such transformations. How many layers the model is defined by is called the depth of the model, and is the reason for the name *deep* learning. The deep learning model is often called a neural net and will be further explained in the next section. In Figure 2.6 the relationship between artificial intelligence, machine learning, and deep learning is illustrated.

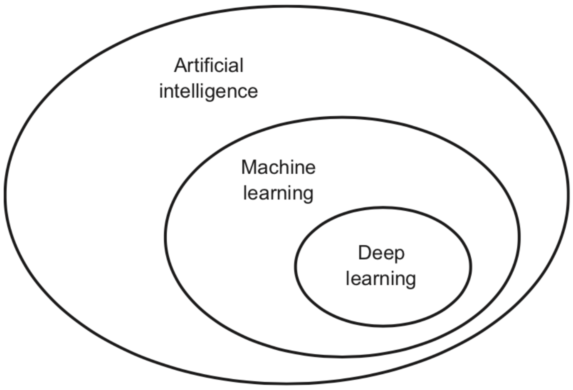

Figure 2.6: Artificial intelligence, machine learning, and deep learning (Chollet 2018).

Machine learning can, according to Chollet (2018), be categorized into four categories:

Supervised learning:
Consists of mapping input data to known targets. Mostly used for classification and regression, such as speech recognition, image classification, and language translation. This is the most common type of machine learning.

Unsupervised learning:
The goal is to find an interesting transformation of the input data without knowing any targets. It is often done in order to visualize the data, perform data compression, or to better understand the data correlation. Two popular categories are dimensionality reduction and clustering.

Self-supervised learning:
The same as supervised learning, but without human-annotated labels. The labels are still included in the learning process but are typically generated from the input data.

Reinforcement learning:
An agent learns, by repeating trial and error, to choose actions that will maximize some reward by receiving information about its environment.

2.7.2 Deep Learning

Almost all deep learning methods commonly used today can be classified as supervised learning, where the goal is to map inputs to targets. In deep learning, a deep sequence of simple data transformations generates this mapping. These transformations are further learned by looking at examples. Each of these data transformation sequences is called a layer. In an artificial neural network, as shown in Figure 2.7, there exist three different types of layers. The input layer, which passes the input data to the system, the output layer, which provides the output of the system, and the hidden layers in between. The hidden layers presented here are two *Dense* layers. Dense layers are neural layers which are densely or fully connected.

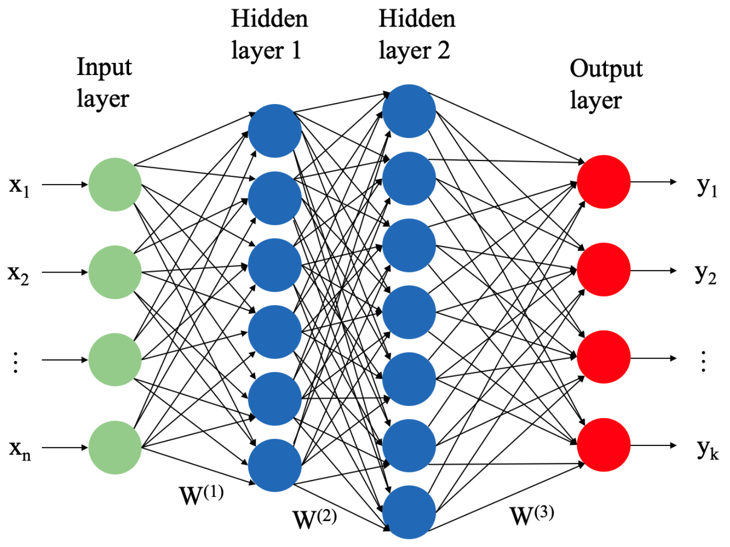

Figure 2.7: Illustration of a deep neural network.

The data transformation performed at a layer is specified by its weights, W. The goal is to find the correct weights so the network maps the inputs to their associated targets, and the *learning* is the process of finding these weights. In order to find the best weights, some measure is needed in order to see how the output of the function is compared with the expected output. This measure specifies how well the network is performing and is often called the *loss function* or the *objective function* of the network. A common loss function used in regression problems is the mean squared error, and the task of the loss function is to calculate a loss score based on how far away the predicted outputs are from the true targets. This loss score should then be used in order to update the weights, which is done by the *optimizer*. The network starts out with random weights, which results in a high loss score, but adjusts the weights a little by little with every example that the network processes. This is called the *training loop* and is presented in Figure 2.8.

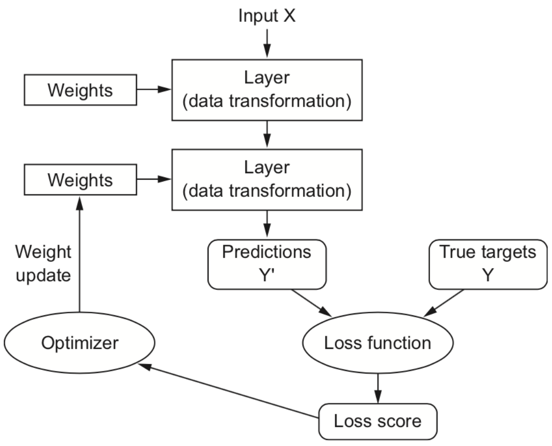

Figure 2.8: Illustration of the deep learning algorithm (Chollet 2018).

At each layer, the transformation is done by combining the weights with the input. A naive approach would be to take the dot product between the input, x, and the weights, W, and add a bias term, b, output $= dot(W, x) + b$, where the goal is to find the best W's and b's. However, since this is only two linear functions, a dot product, and an addition, the layer can then only learn *linear transformations* of the input data. Therefore, a non-linearity, or an *activation function* is used, resulting in output = activation function$(dot(W, x)+b)$. Common activation functions consist of the rectified linear unit (ReLU), tanh, and sigmoid, shown in Figure 2.9.

The method of finding the best weights resulting in the smallest loss is linked to finding the minimum of a function, where the smallest value will be at a point where the derivative of the function is zero. The derivative of the loss function is called the *gradient*, and for a loss function $f(W)$, the goal is to solve gradient$(f(W)) = 0,$ for W. For a neural network with a large number of parameters, this operation is not suitable. However, since the loss function is differentiable, it is possible to calculate it's gradient in order to update the weights. A

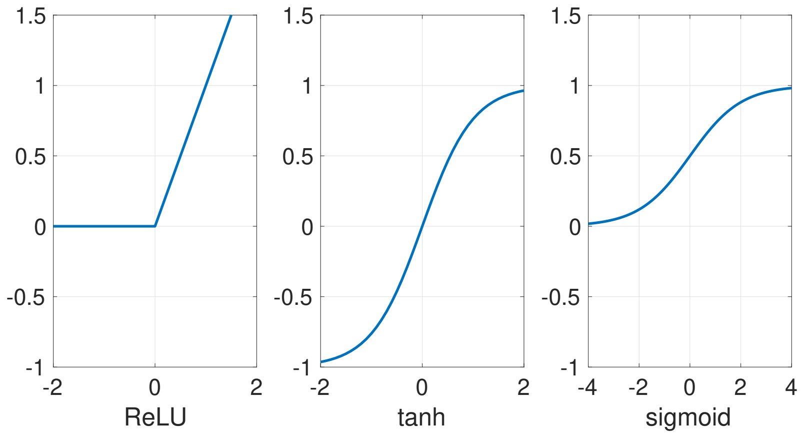

Figure 2.9: Activation functions

neural net is composed of many operations chained together, where each operation has a simple derivative. The computation of the gradient values of a neural net is done by an algorithm called *Backpropagation*, which starts with the final loss value and move backward from the top to the bottom layers. At each layer, the chain rule is applied to calculate each parameter's contribution to the loss value. This computation of the gradient is for each step done on a small random batch of data, and this method for updating the weights is called *mini-batch stochastic gradient descent* and is in Chollet (2018) summarized as:

1. Draw a batch of training samples x and corresponding targets y.
2. Run the network on x to obtain predictions y_{pred}.
3. Compute the loss of the network on the batch, a measure of the mismatch between y_{pred} and y.
4. Compute the gradient of the loss with regard to the network's parameters (a backward pass).
5. Move the parameters a little in the opposite direction from the gradient — for example $W -= \texttt{step} \cdot \text{gradient}$ — thus reducing the loss on the batch a bit.

A reasonable size for the `step` factor is important, where a too small one would lead to many iterations, and a too large may lead to completely random locations on the curve. Each time the algorithm iterates over all the training data, and the gradients are calculated for each batch of samples, it is called an *epoch*. Since the gradients, which in term update the weights, are only calculated for a small set of the data and this is an iterative process, the entire dataset needs to be sent through the model several times, and how many times is specified by the number of epochs.

One issue with the stochastic gradient descent (SGD) method, is that it may get stuck in a local minima. This issue has been avoided by using momentum, where the update is done by looking at both the current gradient and the previous weight updates. SDG with momentum is a type of *optimizer*, and many different types of optimizers have been developed. Some examples are Adagrad, RMSProp, and Adam, where the following descriptions are found in Ruder (2016). Adagrad is a gradient-based algorithm that adopts the learning rate based on its parameters. It has low learning rates for parameters belonging to features that are occurring frequently and high for infrequent features. RMSProp solves the problem of diminishing learning rates by dividing the learning rates by an exponentially decaying average of squared gradients. The Adam (Adaptive Moment Estimation)-optimizer combines the Adagrad and RMSprop, and computes adaptive learning rates by storing an exponentially decaying average of both the past gradients and squared gradients.

Many parameters needs to be decided when developing a deep neural network, for example the number of hidden layers, the size of the hidden layer, the activation functions, the loss function, and the optimizer. Chollet (2018) claims that *a fundamental issue in machine learning is the tension between optimization and generalization*. Optimization involves the process of updating the model so that it works well on the training data, while generalization is a measure of how well the model performs when it encounters new data which it has never seen before. A problem with machine learning models is that it may be memorizing the input data, which leads to a model performing very well on the training data, but very poorly on new data. This is called *overfitting*. This is especially a risk if you have a large model or little training data. However, the model needs to be large enough to sufficiently learn the mapping from input to targets, if not the model is said to be *underfitting*. The first step in making the model more generalizable is to provide it with more training data, but other measures can be taken.

Weight Regularization

Weight regularization is based on the concept that simpler models are less likely to overfit than complex ones. The complexity of the model is, therefore, constrained by forcing the weights to take small values. This is done by adding an additional cost related to having large weights, either in the form of L1 or L2 regularization.

L1 regularization has a cost proportional to the absolute value of the weight coefficients, and L2 has a cost proportional to the square of the value of the weight coefficients.

Dropout
Dropout consists of randomly setting a number of output features to zero during training, these features are said to be dropped out. The amount of dropout, called the dropout rate, is the fraction of features that are set to zero and is usually between 0.2 and 0.5. Since no features are dropped during testing, the output values of the layer are then scaled down by the same factor as the dropout rate. The main idea behind setting random features to zero is to apply noise, which can break up insignificant patterns, and avoid that model memorizes these patterns.

2.7.3 Radial Basis Function Networks

Another type of neural networks is the Radial Basis Function Networks (RBFNs), which is a special type of neural networks using radial basis functions as their activation functions (Faris et al. 2017). The RBFNs are popular due to their simple structure and have a much faster training process compared to deep neural networks (Du and Swamy 2006). In RBFNs there are only three layers, the input layer, one hidden layer, and the output layer, as shown in Figure 2.10. A radial function is specified by its center, and the output depends on how far away the argument is from this center (Vidnerova et al. 2008).

Many different functions can be used as the radial basis function, but a commonly used is the Gaussian function

$$\phi_i(||x - c_i||) = \exp\left(-\frac{||x - c_i||^2}{2\sigma_i^2}\right), \tag{2.49}$$

where $||\cdot||$ is the Euclidean norm, c_i is the center of the hidden neuron i and σ_i is the variance of the Gaussian function or width of the hidden neuron i (Faris et al. 2017). The output of the node k, y_k, is then given as

$$y_k = \sum_{i=1}^{n} \omega_{ik}\phi_i(||x - c_i||), \tag{2.50}$$

where ω_{ik} is the weights that needs to be learned.

The traditional way of training RBFNs is to first decide the centers and the widths, and then use these when learning the weights. The centers and widths can be determined by some unsupervised clustering algorithms, for example, the K-means

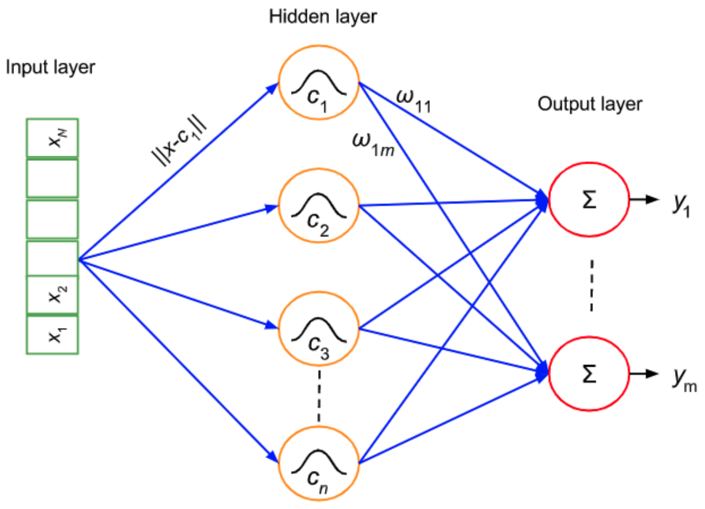

Figure 2.10: Illustration of the radial basis function network (Faris et al. 2017).

algorithm, or be randomly initialized. The K-means clustering algorithm is based on deciding the number of clusters, k, and starts with k randomly initialized centroids. Each data point is then allocated to the nearest center, and the goal is to keep the clusters as small as possible. At each step, the centroid of the cluster is updated to the average position of all the data points assigned to it (Dabbura 2020). This iteration continues until there is no change in the centroids, or if a maximum iteration limit is reached.

2.7.4 Keras and TensorFlow

The deep learning framework used in this thesis is Keras, which is written in Python (Keras 2020). Keras consists of high-level building blocks for machine learning, making it user-friendly to develop machine learning models, but it relies on a *backend engine* to perform low-level operations such as tensor manipulation and differentiation (Chollet 2018). The most common backend implementation is TensorFlow (TensorFlow 2020), which is an open-source machine learning platform, and it is the backend engine that will be used in this thesis.

Chapter 3

Proposed Approaches for the Application and Methods of Performing Current Estimation

In this chapter, the relevance of obtaining current estimates is put into context by proposing an architecture for how these estimates can be included to improve an autonomous vessels' ability to perform safe and efficient path planning and collisions avoidance. The specifications of the small autonomous passenger ferry MilliAmpère are also presented. This vessel will be used as a case study, both in the form of its simulator and the full-scale vessel, to evaluate the proposed methods of performing the current estimation. At last, the methods for performing current estimation using the extended Kalman filter and machine learning methods are presented.

3.1 Proposed Architecture for Taking the Current Estimates into Account

To succeed with autonomous ships, the system onboard the ships needs to be able to perform the same maneuvers and make the same decisions that humans traditionally have done, in a given situation. This can only be achieved if the vessel is able to interpret and understand highly dynamic and complex environments. In other words, the vessel must achieve the situation awareness level previously achieved by humans. This is illustrated by Patrón et al. (2008) in Figure 3.1, where the autonomous vessel needs to obtain all the SA in order to be fully autonomous.

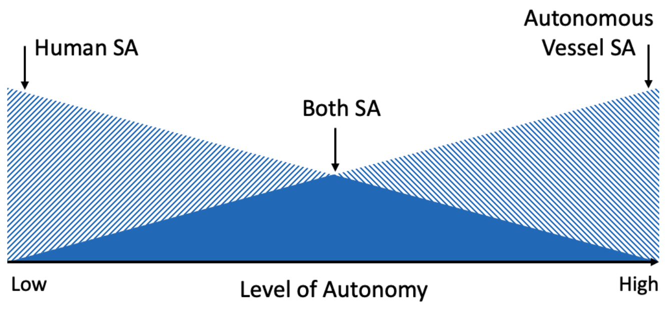

Figure 3.1: Illustration of the SA obtained in humans and autonomous vessels across the levels of autonomy (Patrón et al. 2008)

Perera (2018) purposes that the autonomous ship can be implemented using an agent-based system, and describes an agent as a system located in a specific environment, which interact with the environment by intelligent decisions and actions to satisfy design objectives. There can also be multiple agents in the same environment that needs to interact together. One such intelligent agent on an autonomous vessel can be in charge of planning the path for the vessel, and this thesis purposes a simplified architecture explaining some aspects of how the estimation of the current velocity may affect the path planning and scenarios where the current velocity might be taken into account.

In Figure 3.2, an influence diagram of the purposed architecture is shown, and it presents how various parameters and information obtained should be taken into account by a final *Path planning and collision avoidance algorithm*. The green nodes can be described as level 1 SA, which is mainly about gathering information, while the blue nodes can be characterized as level 2 SA, which is about interpreting the information obtained at level 1.

In the node called *Current estimate*, the current velocity and direction are estimated by using algorithms such as, for example, the EKF, machine learning methods, or by other means. These methods may require a lot of computation or specific maneuvers in order to achieve a good estimate, and it is, therefore, assumed that real-time estimates of the current velocity and direction may not be accessible. Assuming that an estimate of the current was obtained at some time instant, more information about this estimate may be given by looking at the *Time of day* node. This node may obtain information about how long ago the current estimate was ob-

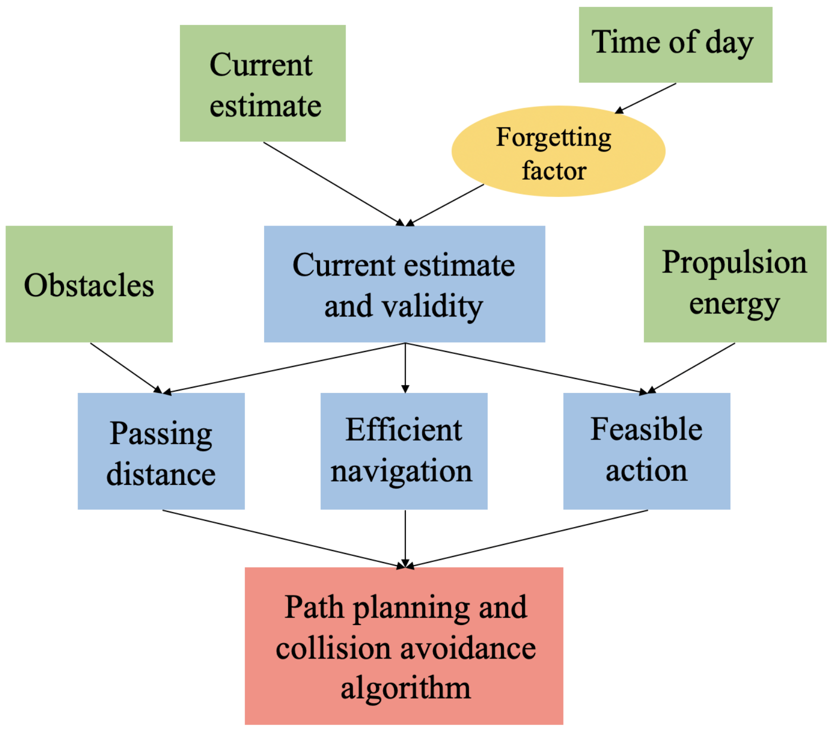

Figure 3.2: Illustration of elements to be included in an intelligent agent providing path planning for an autonomous vessel

tained, together with information about the tidal variations in the local area. Data about how the tides are changing provides more information about how the current is assumed to change, and therefore how long a current estimate can be assumed to be of value. Information about other local effects, such as the presence of a dam or a hydropower plant and their activity, which may cause a significant change in the current estimates, should also be included if possible. These factors contribute to the calculation of the *Forgetting factor*, which should be combined with the current estimate into the node *Current estimate and validity*. The forgetting factor may on some levels be characterized as level 3 SA, as it can be argued that it in some ways predicts how much the current has changed or will change, but on the other hand it is not designed to create a new current estimate.

Given this information about the current estimates and its validity, some navigation patterns might be more efficient than others, resulting in the node *Efficient navigation*. This may consist of information about for example an optimal heading that best takes advantage of the forces from the current, leading to less power consumption. However, the current estimate may be of relevance when the autonomous vessel should perform other tasks not traditionally associated with current estimation.

The *Obstacles* node contains information about objects around the vessel, and there might be a need to move around either a still or a moving object to perform a certain task. If the direction of the current is pointing so that, if not counteracted, the current pushes the vessel towards the object, it would be advantageous to pass the object with a greater distance, which leads to the node *Passing distance*. This will result in a smaller risk with regards to engine failure, but also provides some extra safety in terms of the uncertainty of the current estimate. If the control system is not counteracting the disturbance as well as initially assumed, a greater distance to the object provides an extra safety margin when passing the object.

Another input to consider is described by the *Propulsion energy* node. In the case of really strong currents, the thrusters may not be strong enough to counteract the current forces in a satisfactory manner, and the planned task may not be possible to perform. The energy consumption will also be greater in the case of a large current, and one needs to be certain that the energy carried by the ship is sufficient. For autonomous vessels either driven by fuel or being fully electric, a lower limit needs to be determined as to how low the energy level can be before it is not permissible for the vessel to leave the dock. This lower limit should be higher when the currents are strong, and these considerations results in the *Feasible action* node, which determines if a specific task, such as for example crossing a channel, is possible to accomplish or not.

The factors presented in this section describes some aspects of what an intelligent agent needs to consider when calculating the path and making decisions such as determining whether to leave the dock. In Figure 3.3 the complete architecture of an autonomous system is proposed, where the main structure is inspired from Sørensen (2013). The intelligent agent consists of an observer, which among other things may perform and estimation of the current velocity. The ship status information node consists of general information about the vessel, such as energy available for propulsion or other constant parameters as the total thruster capacity. The obstacle information block keeps track of obstacles by using sensors such as Automatic Identification System (AIS), Radar, camera, and Lidar, and consists of some obstacle detection system. Digital maps may also be an input to the system as some paths may be infeasible if the depth is to shallow. This depth may also vary a lot depending on whether it is ebb or flow. This, along with other weather forecasts should also be included in the path planning and collision avoidance algorithm, if available.

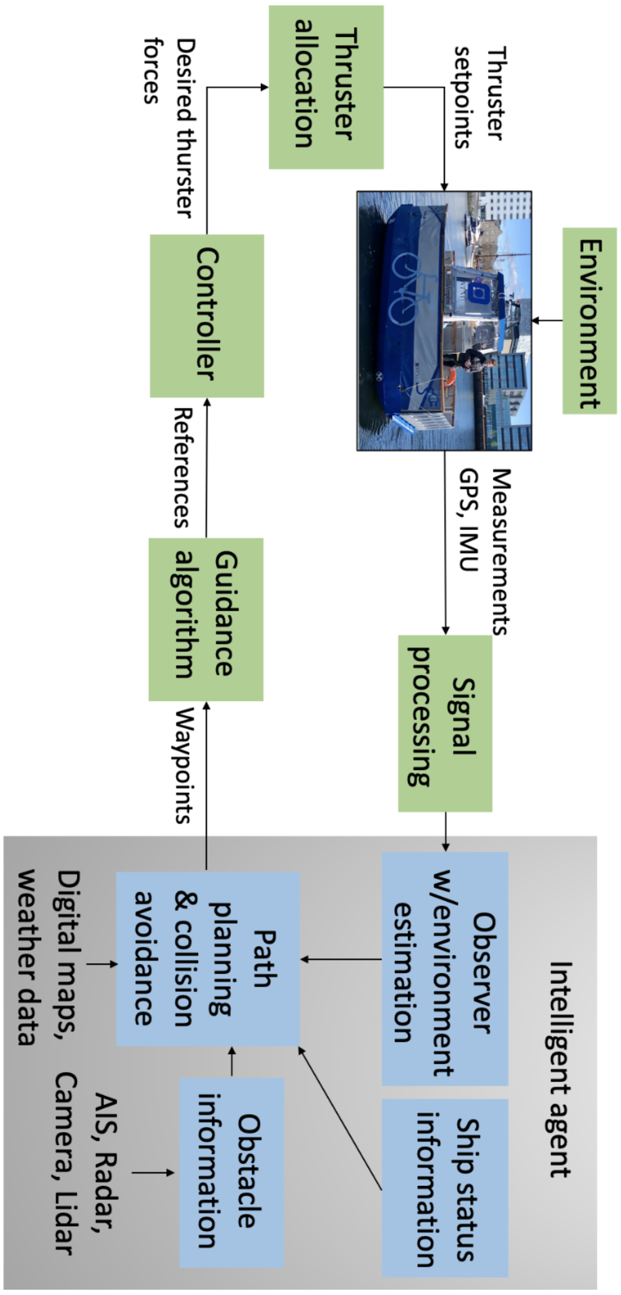

Figure 3.3: Proposed architecture of an autonomous system with an intelligent agent performing path planning and collision avoidance

3.2 Specifications of the MilliAmpère Ferry

The MilliAmpère ferry is a small double-ended passenger ferry with the general specifications presented in Table 3.1, where L_1 and L_2 are the locations of the azimuth thrusters along the x-axis. It is equipped with a *Vector VS330* positioning device, which consists of a GNSS (Global Navigation Satellite Systems) with real-time kinematic (RTK), resulting in precise positioning and heading measurements. These measurements are further differentiated into velocities, and these velocities are used as if MilliAmpère is equipped with velocity sensors. In Pedersen (2019),

Table 3.1: Parameter values of the MilliAmpère ferry

Parameter	Value	Unit
Length overall	5.0	m
Beam	2.8	m
Displacement	1667	kg
Max thrust	500.7	N
L_1	-1.8	m
L_2	1.8	m

mathematical models of the MilliAmpère ferry were developed through system identification techniques using experimental data. The simulator of MilliAmpère was developed using this experimental data, and performing a new curve fitting analysis, and is implemented using the ROS framework and Python programming language. This simulator is used as a basis to test the concepts developed in this thesis. The parameters used in the simulator are presented in Table 3.2. Ignoring the terms equal to zero, the Coriolis, centripetal, and damping matrices used in the simulator are presented in Equations (3.1) and (3.2). The terms Y_{ur}, N_{uv}, and N_{ur} was included to get more stable sway/yaw dynamics.

$$\boldsymbol{C}(\boldsymbol{\nu}_r) = \begin{bmatrix} 0 & 0 & -m_{22}v_r - \frac{1}{2}(m_{23}+m_{32})r \\ 0 & 0 & m_{11}u_r \\ m_{22}v_r + \frac{1}{2}(m_{23}+m_{32})r & -m_{11}u_r & 0 \end{bmatrix} \tag{3.1}$$

$$\boldsymbol{D}(\boldsymbol{\nu}_r) = \begin{bmatrix} \begin{matrix} -X_u - X_{|u|u}|u_r| \\ -X_{uuu}u_r^2 \end{matrix} & 0 & 0 \\ 0 & \begin{matrix} -Y_v - Y_{|v|v}|v_r| \\ -Y_{|r|v}|r| - Y_{vvv}v_r^2 \end{matrix} & \begin{matrix} -Y_r - Y_{|v|r}|v_r| \\ -Y_{|r|r}|r| - Y_{ur}u_r \end{matrix} \\ 0 & \begin{matrix} -N_v - N_{|v|v}|v_r| \\ -N_{|r|v}|r| - N_{uv}u_r \end{matrix} & \begin{matrix} -N_r - N_{|v|r}|v_r| \\ -N_{|r|r}|r| - N_{ur}u_r \end{matrix} \end{bmatrix} \tag{3.2}$$

Table 3.2: Parameter values in the MilliAmpère simulator

Parameter	Value	Unit
m_{11}	2390	kg
m_{12}	0	kg
m_{13}	0	kgm
m_{21}	0	kg
m_{22}	2448	kg
m_{23}	268.1	kgm
m_{31}	0	kgm
m_{32}	-23.84	kgm
m_{33}	4862	kgm^2
X_u	-106.6	kg/s
$X_{\|u\|u}$	-21.39	kg/m
X_{uuu}	-37.43	kgs/m^2
Y_v	62.58	kg/s
$Y_{\|v\|v}$	-172.9	kg/m
Y_{vvv}	-1.338	kgs/m^2
$Y_{\|r\|v}$	-1517	kg
Y_r	62.58	kgm/s
$Y_{\|v\|r}$	488.7	kg
$Y_{\|r\|r}$	-198.2	kgm
N_v	7.34	kgm/s
$N_{\|v\|v}$	-4.352	kg
$N_{\|r\|v}$	437.8	kgm
N_r	-142.7	kgm^2/s
$N_{\|r\|r}$	-831.7	kgm^2
N_{rrr}	0.0	kgm^2s
$N_{\|v\|r}$	-122	kgm
Y_{ur}	77.58	kg
N_{uv}	-90.97	kg
N_{ur}	178.5	kgm

3.3 The Extended Kalman Filter for Current Estimation

The continuous system representing MilliAmpère can be represented as

$$\dot{\boldsymbol{\eta}} = \mathbf{R}(\psi)\boldsymbol{\nu} \tag{3.3a}$$

$$\dot{\boldsymbol{\nu}} = \boldsymbol{M}^{-1}\left[-\mathbf{N}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \boldsymbol{\tau}\right] \tag{3.3b}$$

$$\dot{\mathbf{V}}_c = \mathbf{0}, \tag{3.3c}$$

where

$$\boldsymbol{\eta} = \begin{bmatrix} N \\ E \\ \psi \end{bmatrix}, \quad \boldsymbol{\nu} = \begin{bmatrix} u \\ v \\ r \end{bmatrix}, \quad \mathbf{V_c} = \begin{bmatrix} V_{c,x} \\ V_{c,y} \\ V_{c,n} \end{bmatrix}, \quad V_{c,n} = 0 \tag{3.4}$$

The last three states are included to model the current. The current is assumed to be irrotational and has negligible acceleration. The last state, $V_{c,n}$ is, therefore, only included in terms of dimensional purposes, but is assumed to be zero.

Since the model will not represent the MilliAmpère ferry perfectly, white noise is added to the states. Noise added to the acceleration equation, $\dot{\boldsymbol{\nu}}$ will directly be included in $\dot{\boldsymbol{\eta}}$, and is, therefore, not added again. The resulting states are represented as

$$\dot{\boldsymbol{\eta}} = \mathbf{R}(\psi)\boldsymbol{\nu} \tag{3.5a}$$

$$\dot{\boldsymbol{\nu}} = \mathbf{M}^{-1}\left[-\mathbf{N}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \boldsymbol{\tau} + \boldsymbol{w}_1\right] \tag{3.5b}$$

$$\dot{\mathbf{V}}_c = \boldsymbol{w}_2 \tag{3.5c}$$

where $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ are zero-mean Gaussian white noise. Since we ignore the last state of $\boldsymbol{V_c}$ completely, $\boldsymbol{w}_2$ is assumed to be zero in the last state.

Defining the state vector as $\boldsymbol{x} = [N, E, \psi, u, v, r, V_{c,x}, V_{c,y}, 0]^\top$, the system can be written in the form of the nonlinear state-space presented in Equation (2.47), where $\mathbf{f}(\boldsymbol{x}) = [\mathbf{f}_1^\top, \mathbf{f}_2^\top, \mathbf{f}_3^\top]^\top$, $\boldsymbol{u} = \boldsymbol{\tau}$, and

$$\mathrm{f}_1 = \dot{\boldsymbol{\eta}} = \mathbf{R}(\psi)\boldsymbol{\nu} \tag{3.6a}$$

$$\mathrm{f}_2 = \dot{\boldsymbol{\nu}} = -\mathbf{M}^{-1}\mathbf{N}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r \tag{3.6b}$$

$$= \mathbf{M}^{-1}\left[-\mathbf{N}(\boldsymbol{\nu}_r)\boldsymbol{\nu} + \mathbf{N}(\boldsymbol{\nu}_r)\mathbf{R}^T(\psi)\boldsymbol{V}_c\right] \tag{3.6c}$$

$$\mathrm{f}_3 = \dot{\mathbf{V}}_c = \mathbf{0_{3\times1}} \tag{3.6d}$$

$$\boldsymbol{B} = \begin{bmatrix} \mathbf{0_{3\times3}} \\ \mathbf{M}^{-1} \\ \mathbf{0_{3\times3}} \end{bmatrix}, \quad \boldsymbol{H} = \begin{bmatrix} \boldsymbol{I_{3\times3}} & \mathbf{0_{3\times3}} & \mathbf{0_{3\times3}} \\ \mathbf{0_{3\times3}} & \boldsymbol{I_{3\times3}} & \mathbf{0_{3\times3}} \end{bmatrix} \tag{3.6e}$$

$$\boldsymbol{E} = \begin{bmatrix} 0_{3\times3} & 0_{3\times3} & 0_{3\times2} & 0_{3\times1} \\ 0_{3\times3} & \mathrm{M}^{-1} & 0_{3\times2} & 0_{3\times1} \\ 0_{2\times3} & 0_{2\times3} & \boldsymbol{I}_{2\times2} & 0_{2\times1} \\ 0_{1\times3} & 0_{1\times3} & 0_{1\times2} & 0_{1\times1} \end{bmatrix}, \tag{3.6f}$$

Due to the extra dimension in the current, the $\boldsymbol{E}$ matrix is augmented with a row and a column of zeros. The output vector is defined as $\boldsymbol{y} = [N, E, \psi, u, v, r]^\top$.

Based on the theory presented in Chapter 2, an illustration of the extended Kalman filter loop can be illustrated as in Figure 3.4.

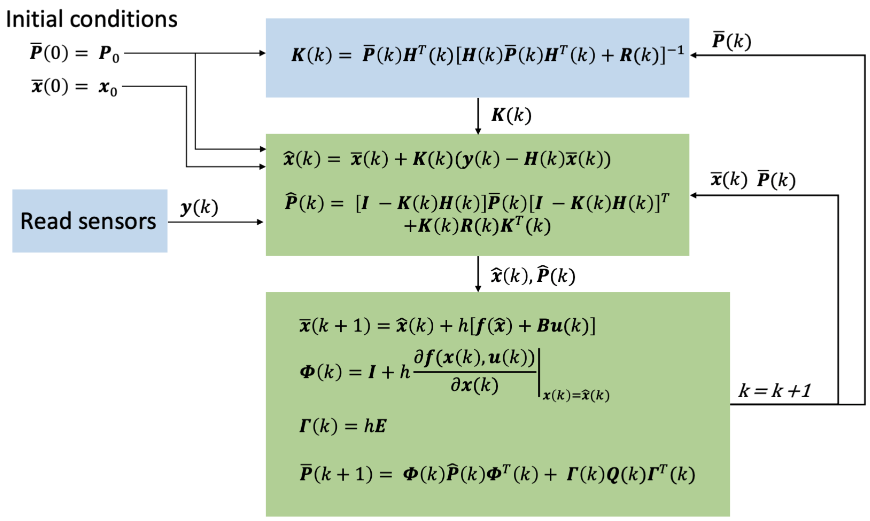
Initial conditions
$\bar{P}(0) = P_0$
$\bar{x}(0) = x_0$
Read sensors
$y(k)$
$K(k) = \bar{P}(k)H^T(k)[H(k)\bar{P}(k)H^T(k) + R(k)]^{-1}$
$\bar{P}(k)$
$K(k)$
$\hat{x}(k) = \bar{x}(k) + K(k)(y(k) - H(k)\bar{x}(k))$
$\hat{P}(k) = [I - K(k)H(k)]\bar{P}(k)[I - K(k)H(k)]^T + K(k)R(k)K^T(k)$
$\bar{x}(k)\ \bar{P}(k)$
$\hat{x}(k), \hat{P}(k)$
$\bar{x}(k+1) = \hat{x}(k) + h[f(\hat{x}) + Bu(k)]$
$\Phi(k) = I + h\frac{\partial f(x(k), u(k))}{\partial x(k)}\Big|_{x(k)=\hat{x}(k)}$
$\Gamma(k) = hE$
$\bar{P}(k+1) = \Phi(k)\hat{P}(k)\Phi^T(k) + \Gamma(k)Q(k)\Gamma^T(k)$
$k = k+1$

Figure 3.4: Illustration of the extended Kalman filter loop

From Equation (2.48a), we get the state estimate propagation, as presented in Equation (3.7), where the desired ψ, and not its estimate, is used in the rotation matrices (Sørensen 2018).

$$\begin{bmatrix} \bar{N}(k+1) \\ \bar{E}(k+1) \\ \bar{\psi}(k+1) \end{bmatrix} = \begin{bmatrix} \hat{N}(k) \\ \hat{E}(k) \\ \hat{\psi}(k) \end{bmatrix} + h \cdot \boldsymbol{R}(\psi) \begin{bmatrix} \hat{u}(k) \\ \hat{v}(k) \\ \hat{r}(k) \end{bmatrix} \tag{3.7a}$$

$$\begin{aligned}\begin{bmatrix}\bar{u}(k+1)\\ \bar{v}(k+1)\\ \bar{r}(k+1)\end{bmatrix} &= \begin{bmatrix}\hat{u}(k)\\ \hat{v}(k)\\ \hat{r}(k)\end{bmatrix} - h\cdot \boldsymbol{M}^{-1}\boldsymbol{N}(\hat{\boldsymbol{\nu}}_r)\begin{bmatrix}\hat{u}(k)\\ \hat{v}(k)\\ \hat{r}(k)\end{bmatrix}\\ &+ h\cdot \boldsymbol{M}^{-1}\boldsymbol{N}(\hat{\boldsymbol{\nu}}_r)\boldsymbol{R}^\top(\psi)\begin{bmatrix}\hat{V}_{c,x}(k)\\ \hat{V}_{c,y}(k)\\ 0\end{bmatrix} + h\cdot \boldsymbol{M}^{-1}\begin{bmatrix}\tau_x(k)\\ \tau_y(k)\\ \tau_n(k)\end{bmatrix}\end{aligned} \tag{3.7b}$$

$$\begin{bmatrix}\bar{V}_{c,x}(k+1)\\ \bar{V}_{c,y}(k+1)\\ 0\end{bmatrix} = \begin{bmatrix}\hat{V}_{c,x}(k)\\ \hat{V}_{c,y}(k)\\ 0\end{bmatrix} \tag{3.7c}$$

To calculate the error covariance propagation, we need to calculate $\boldsymbol{\Phi}$, and from Equation (2.48b), we see that the Jacobi matrix is used. The Jacobi matrix of $\mathbf{f}$ is presented in Equation (3.8), and since the calculation of this is quite complex the detailed calculation of the submatrices are presented in Appendix A. These equations should further be evaluated at $\boldsymbol{x}(k) = \hat{\boldsymbol{x}}(k)$ at each time step.

$$\boldsymbol{J} = \frac{\partial \mathbf{f}(\boldsymbol{x},\boldsymbol{u})}{\partial \boldsymbol{x}} = \begin{bmatrix}\frac{\partial \mathbf{f}_1}{\partial \boldsymbol{\eta}} & \frac{\partial \mathbf{f}_1}{\partial \boldsymbol{\nu}} & \frac{\partial \mathbf{f}_1}{\partial \boldsymbol{V}_c}\\ \frac{\partial \mathbf{f}_2}{\partial \boldsymbol{\eta}} & \frac{\partial \mathbf{f}_2}{\partial \boldsymbol{\nu}} & \frac{\partial \mathbf{f}_2}{\partial \boldsymbol{V}_c}\\ \frac{\partial \mathbf{f}_3}{\partial \boldsymbol{\eta}} & \frac{\partial \mathbf{f}_3}{\partial \boldsymbol{\nu}} & \frac{\partial \mathbf{f}_3}{\partial \boldsymbol{V}_c}\end{bmatrix} \tag{3.8}$$

The extended Kalman filter is connected to the simulator as illustrated in Figure 3.5. The simulator was extended to include current forces. These current forces are set using a disturbance node, which creates the inputs to the simulator, $V_{c,x}$ and $V_{c,y.}$ The position vector, $\boldsymbol{\eta}$, the velocity vector, $\boldsymbol{\nu}$, the control forces and moments, $\boldsymbol{\tau}$, as well as the desired heading, ψ_{ref}, are used as inputs to the EKF. The EKF then estimates the position vector, $\hat{\boldsymbol{\eta}}$, the velocity vector, $\hat{\boldsymbol{\nu}}$, and the current velocity in north and east direction, that is $\hat{V}_{c,x}$ and $\hat{V}_{c,y}$ respectively, which can be further transformed into a total velocity and direction for the current estimates.

3.3.1 Approach for Evaluating the Performance of the EKF

The exact inputs given to the model in the simulator are also available to the EKF, which means that the filter is operating under perfect circumstances. The robustness of the filter is, therefore, tested by applying noise in the simulation model, and

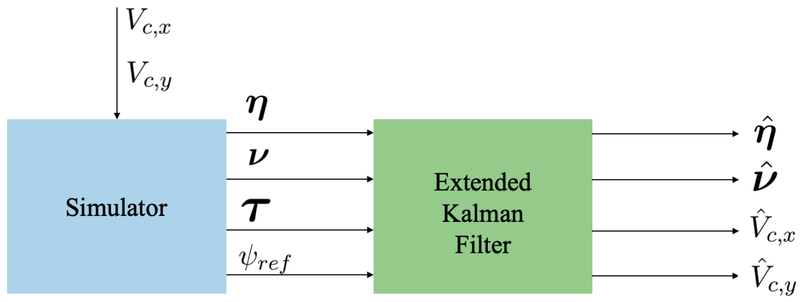

Figure 3.5: Implementation of the EKF with the MilliAmpère simulator

the metric used to evaluate how well the EKF is performing is the mean absolute error (MAE) which can be described by

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |x_i - \hat{x}_i| \tag{3.9}$$

where n is the number of samples, x_i is the true value and $\hat{x}_i$ is the estimated value. The noise added to the model is given in the form of a constant additive noise with a different percentage of noise. The new value will, therefore be value $= (1 + \%$ noise / 100$)\cdot$ value. In Section 4.1 the performance of the EKF is first shown when the simulator has no noise, and then the change in MAE between the true and estimated current is presented when noise is present. The sensitivity of the EKF is evaluated by adding noise to three different parameters, the control forces and moment, $\boldsymbol{\tau}$, the Coriolis, centripetal, and damping matrix, $\boldsymbol{N}(\boldsymbol{\nu}_r)$, and the mass matrix, $\boldsymbol{M}$. The sensitivity of the current estimation may be dependent on the current direction and will be presented in polar plots, inspired by how capability plots express the station-keeping capabilities of a vessel in DP. These polar plots will present the MAE between the true and the estimated current, given the direction of the current.

3.3.2 Additions to the EKF for Physical Experiments

In the simulator, it is possible to know the exact forces and moments acting on the ferry, however, the exact forces and moments commanded by the DP system of the real vessel may not be the exact forces and moments given by the propeller.

There will be some time delay from when the forces are commanded to when they are acting on the ferry, and the commanded forces will oscillate more to adjust for small changes than what is possible for the propeller. The control forces and moment are therefore included as state variables, and the new state vector becomes $\boldsymbol{x} = [N, E, \psi, u, v, r, V_{c,x}, V_{c,y}, 0, \tau_x, \tau_y, \tau_n]^\top$. The state equation for $\boldsymbol{\tau}$ is given as a first-order low-pass filter between the commanded, $\boldsymbol{\tau}_{DP}$, and actual thrust, $\boldsymbol{\tau}$, where the time constant, T, is set to 1 second.

$$\dot{\boldsymbol{\tau}} = \frac{1}{T}(\boldsymbol{\tau}_{DP} - \boldsymbol{\tau}) + \boldsymbol{w}_3 \tag{3.10}$$

Written in the form of the nonlinear state-space presented in Equation (2.47), the system can now be written as in Equation (3.11), where $\mathbf{f}(\boldsymbol{x}) = [\mathbf{f}_1^\top, \mathbf{f}_2^\top, \mathbf{f}_3^\top, \mathbf{f}_4^\top]^\top$ and $\boldsymbol{u} = \boldsymbol{\tau}_{DP}$.

$$\mathrm{f}_1 = \dot{\boldsymbol{\eta}} = \mathbf{R}(\psi)\boldsymbol{\nu} \tag{3.11a}$$
$$\mathrm{f}_2 = \dot{\boldsymbol{\nu}} = \mathbf{M}^{-1}\left[-\mathbf{N}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \boldsymbol{\tau}\right] \tag{3.11b}$$
$$= \mathbf{M}^{-1}\left[-\mathbf{N}(\boldsymbol{\nu}_r)\boldsymbol{\nu} + \mathbf{N}(\boldsymbol{\nu}_r)\mathbf{R}^T(\psi)\boldsymbol{V}_c + \boldsymbol{\tau}\right] \tag{3.11c}$$
$$\mathrm{f}_3 = \dot{\mathbf{V}}_c = \mathbf{0}_{3\times1} \tag{3.11d}$$
$$\mathrm{f}_4 = \dot{\boldsymbol{\tau}} = -\frac{1}{T}\boldsymbol{\tau} \tag{3.11e}$$

$$\boldsymbol{B} = \begin{bmatrix} \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} \\ \frac{\mathbf{1}}{\boldsymbol{T}}_{3\times3} \end{bmatrix}, \quad \boldsymbol{H} = \begin{bmatrix} \boldsymbol{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \boldsymbol{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix} \tag{3.11f}$$

$$\boldsymbol{E} = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times2} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{M}^{-1} & \mathbf{0}_{3\times2} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{2\times3} & \mathbf{0}_{2\times3} & \boldsymbol{I}_{2\times2} & \mathbf{0}_{2\times1} & \mathbf{0}_{2\times3} \\ \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & \mathbf{0}_{1\times2} & \mathbf{0}_{1\times1} & \mathbf{0}_{1\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times2} & \mathbf{0}_{3\times1} & \boldsymbol{I}_{3\times3} \end{bmatrix}, \tag{3.11g}$$

Equation (3.7b) is now updated to Equation (3.12a), and Equation (3.12b) is added.

The Jacobi matrix is also augmented to Equation (3.13).

$$\begin{bmatrix}\bar{u}(k+1)\\ \bar{v}(k+1)\\ \bar{r}(k+1)\end{bmatrix} = \begin{bmatrix}\hat{u}(k)\\ \hat{v}(k)\\ \hat{r}(k)\end{bmatrix} - h\cdot \boldsymbol{M}^{-1}\boldsymbol{N}(\hat{\boldsymbol{\nu}}_r)\begin{bmatrix}\hat{u}(k)\\ \hat{v}(k)\\ \hat{r}(k)\end{bmatrix} + h\cdot \boldsymbol{M}^{-1}\boldsymbol{N}(\hat{\boldsymbol{\nu}}_r)\boldsymbol{R}^\top(\psi)\begin{bmatrix}\hat{V}_{c,x}(k)\\ \hat{V}_{c,y}(k)\\ 0\end{bmatrix} + h\cdot\boldsymbol{M}^{-1}\begin{bmatrix}\hat{\tau}_x(k)\\ \hat{\tau}_y(k)\\ \hat{\tau}_n(k)\end{bmatrix} \tag{3.12a}$$

$$\begin{bmatrix}\bar{\tau}_x(k+1)\\ \bar{\tau}_y(k+1)\\ \bar{\tau}_y(k+1)\end{bmatrix} = \begin{bmatrix}\hat{\tau}_x(k)\\ \hat{\tau}_x(k)\\ \hat{\tau}_x(k)\end{bmatrix} + h\cdot\frac{1}{T}\left(\begin{bmatrix}\tau_{DP,x}\\ \tau_{DP,y}\\ \tau_{DP,n}\end{bmatrix} - \begin{bmatrix}\hat{\tau}_x(k)\\ \hat{\tau}_x(k)\\ \hat{\tau}_x(k)\end{bmatrix}\right) \tag{3.12b}$$

$$\boldsymbol{J} = \frac{\partial \mathbf{f}(\boldsymbol{x},\boldsymbol{u})}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial \mathbf{f}_1}{\partial \boldsymbol{\eta}} & \frac{\partial \mathbf{f}_1}{\partial \boldsymbol{\nu}} & \frac{\partial \mathbf{f}_1}{\partial \boldsymbol{V}_c} & \frac{\partial \mathbf{f}_1}{\partial \boldsymbol{\tau}} \\ \frac{\partial \mathbf{f}_2}{\partial \boldsymbol{\eta}} & \frac{\partial \mathbf{f}_2}{\partial \boldsymbol{\nu}} & \frac{\partial \mathbf{f}_2}{\partial \boldsymbol{V}_c} & \frac{\partial \mathbf{f}_2}{\partial \boldsymbol{\tau}} \\ \frac{\partial \mathbf{f}_3}{\partial \boldsymbol{\eta}} & \frac{\partial \mathbf{f}_3}{\partial \boldsymbol{\nu}} & \frac{\partial \mathbf{f}_3}{\partial \boldsymbol{V}_c} & \frac{\partial \mathbf{f}_3}{\partial \boldsymbol{\tau}} \\ \frac{\partial \mathbf{f}_4}{\partial \boldsymbol{\eta}} & \frac{\partial \mathbf{f}_4}{\partial \boldsymbol{\nu}} & \frac{\partial \mathbf{f}_4}{\partial \boldsymbol{V}_c} & \frac{\partial \mathbf{f}_4}{\partial \boldsymbol{\tau}} \end{bmatrix} \tag{3.13}$$

To remove noise, the measurements will also be passed through a low-pass filter before entering the EKF. The results of the current estimation after applying the EKF on experimental data obtained using the full-scale MilliAmpère ferry are presented in Chapter 5.

3.4 Machine Learning for Current Estimation

The goal of the machine learning methods is to provide an estimate for the current velocity and direction using the body-fixed velocities of the vessel together with the control forces. The first section explains how the training data for the machine learning models are developed, before the method of building machine learning models is presented. Two types of machine learning models will be looked into, consisting of traditional deep neural networks, and radial basis function networks.

3.4.1 Generating Data

The machine learning methods are developed to estimate the current velocities in the body-fixed reference frame. This is because fewer parameters are needed as input when one only works in the body-fixed reference frame. The data for the machine learning model is generated by assuming a steady-state model, where the model is created using the parameters of the MilliAmpère simulator. In the steady-state case, the accelerations are zero, and we get

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} = -\boldsymbol{C}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r - \boldsymbol{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \boldsymbol{\tau} = 0 \tag{3.14a}$$

$$\boldsymbol{\tau} = \boldsymbol{C}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \boldsymbol{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r \tag{3.14b}$$

$$\boldsymbol{\nu}_r = \boldsymbol{\nu} - \boldsymbol{\nu}_c \tag{3.14c}$$

Random values are then generated for $\boldsymbol{\nu}$ and $\boldsymbol{\nu}_c$ to find the corresponding $\boldsymbol{\tau}$. This process is illustrated in Figure 3.6. The training data for the machine learning models is generated from random data having a uniform distribution, where the limits for the different variables are shown in Table 3.3.

Table 3.3: Limits for the parameters generating the test data

Parameter	Limits	Unit
u	$[-2, 2]$	m/s
v	$[-0.5, 0.5]$	m/s
r	$[-0.5, 0.5]$	rad/s
V_c	$[0, 1.2]$	m/s
ψ_c	$[-\pi, \pi]$	rad

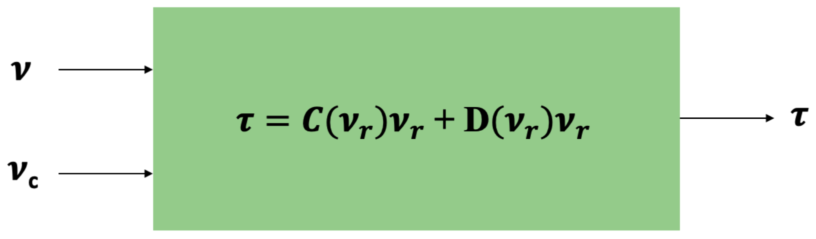
$\boldsymbol{\nu}$
$\boldsymbol{\nu}_c$
$\boldsymbol{\tau} = \boldsymbol{C}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \mathbf{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r$
$\boldsymbol{\tau}$

Figure 3.6: Illustration of the model for generating data

Since the goal of the machine learning algorithm is to estimate the current velocities, the input data to the machine learning model consists of $\boldsymbol{\nu}$ and $\boldsymbol{\tau}$, and the output is the current velocities expressed in body-frame. This process is shown in Figure 3.7. Each of the inputs, u, v, r, τ_x, τ_y, and τ_n is called a feature.

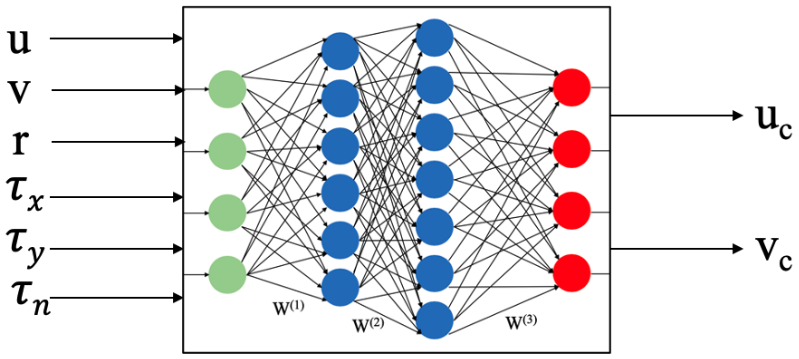

Figure 3.7: Illustration of the deep neural network showing the inputs and outputs

3.4.2 Building a Deep Neural Network

The steps of building a traditional deep neural network consist of adding layers, deciding the number of neurons in each layer as well as the number of layers, and then adding dropout or weight regularization to prevent overfitting.

The first step of building a neural network usually consists of preprocessing the data. For text data, the text might need to be mapped to integers or there might be missing data in the dataset. Since the data used in this thesis is generated, the only preprocessing needed consists of normalizing each feature. This is done by calculating the mean of each feature and subtracting this from the feature and divide by the feature's standard deviation. The resulting input data has a mean of zero and a standard deviation of one.

To verify how the model is doing it is common to split the data into a set of training data and a set of validation data. If the model performs significantly better on the training data than on the validation data, it is a sign of overfitting. For regression problems, a common loss function is the mean squared error (MSE), and one must, therefore, pay attention to how this evolves for both the training and validation data. Another common metric to pay attention to is the mean absolute error (MAE). To avoid information leaking from the validation data to the model, the mean and the standard deviation used during the preprocessing can only be calculated for the training data, and the values from the training data must be used when normalizing the validation data.

The deep neural network created in this thesis is developed by adding dense layers, and varying how many layers and how many neurons in each layer there will be. In Keras, this is implemented by `Dense(number of neurons)`. The activation function must be specified when adding a layer, as the default activation function is a linear. This is done by `Dense(number of neurons, activation = 'sigmoid')`, if the desired activation function is the sigmoid, as presented in Chapter 2. A weight regularizer can be added to the dense layer by the function `Dense(number of neurons,activation = 'sigmoid, kernel_regularizer=l2(size))`, which in this example applies a penalty proportional to the square of the value of the weight, L2, of a given size. Dropout layers can be added between the traditional dense layers by adding `Dropout(dropout rate).`

When adjusting these parameters and evaluating how the network performs on the validation data, some information from the validation data leaks into the model (Chollet 2018). It is, therefore, common to have a separate set of data, the test data, for which the model may be evaluated for at the end, in order to measure the generalization of the model. The same preprocessing based on the training data must be applied to the test data.

3.4.3 Building an RBFN

The preprocessing steps described for the deep neural networks are the same as the ones used when building an RBFN, but there are no default radial basis function layers in Keras. The RBF-layer in this thesis is, therefore, implemented using the source code given in Vidnerova (2019). The hyperparameters to be specified consist of how many neurons the RBF-layer should have, and how the initial values for the centers and widths of these neurons are specified. The locations of the centers can be initialized either as randomly chosen samples of the dataset or by a K-means algorithm, and the number of neurons and the initial widths need to be further investigated. In Vidnerova (2019), the radial basis function is defined as Equation (3.15), instead of the one presented in Equation (2.49), and the locations of the centers and their widths are the parameters that are being trained when training the neural network, and not a separate set of weights.

$$\phi_i(\boldsymbol{x} - \boldsymbol{c}_i) = \exp\left(-\beta_i \cdot (\boldsymbol{x} - \boldsymbol{c}_i)^2\right), \tag{3.15}$$

This method has a constant width for each center, but it is also possible to have different widths for each feature for the different centers. This results in more parameters needed to define the model but gives more flexibility to learn the connections between each feature. To distinguish these two cases, and keep the notation of how it is implemented in Vidnerova (2019), the β is now thought of as radii, and will be denoted $\boldsymbol{r}$ instead. The radial basis function is now implemented as

$$\phi_i(\boldsymbol{x} - \boldsymbol{c}_i) = \exp\left(-\left(\frac{\boldsymbol{x} - \boldsymbol{c}_i}{\boldsymbol{r}_i}\right)^2\right), \tag{3.16}$$

The results from training different machine learning models of different sizes, with input data generated by the simulation model illustrated in Figure 3.6, is presented in Section 4.2. Here, a data set is generated and divided into a training and validation set, used to evaluate the performance of the models. Next, two test sets are generated, where one of them contains data generated by adding noise to the simulation model, to evaluate how the trained machine learning models behave on unseen data. These trained models are further applied to full-scale experimental data obtained using MilliAmpère, and the results from predicting the current velocity and direction using these models are presented in Chapter 5.

Chapter 4

Simulation Results and Discussion

In this chapter, the results from two case studies are presented and discussed, where the results are obtained through simulation. In the first section, a sensitivity analysis of the extended Kalman filter is performed, and the change in the accuracy of the current estimates when noise is added to the simulation model is examined. In the next section, the results from performing the current estimation using two types of machine learning models are presented and discussed. The source code for the developed methods are presented in Appendix B.

4.1 Case Study: Sensitivity Analysis of the Extended Kalman Filter

The developed EKF, explained in Section 3.3, is tested using the simulator of MilliAmpère, and the results are presented and discussed in this section. The simulations are run by exposing the model to currents with different velocities and directions, where the estimates from the EKF are the North and East component of the current velocity, which is further mapped to total current velocity and direction. Noise is added to $\boldsymbol{\tau}$, $\boldsymbol{N}(\boldsymbol{\nu}_r)$, and $\boldsymbol{M}$, separately, and the MAEs between the true and estimated current are shown.

4.1.1 Simulation of Current Estimation Using the EKF

To show the performance of the EKF when no noise is present, two cases are simulated. In the first case the current velocity is $V_c = 0.3$ m/s and current direction is $\psi_c = 45°$, and for the second case $V_c = 1.0$ m/s and $\psi_c = 150°$. The simulated vessel is in DP, where the desired set-points are (0, 0, 0), corresponding to the desired set-points in North, East, and heading. The covariance matrices in the EKF are given as

$$\boldsymbol{Q} = \text{diag}(1.0, 1.0, 0.01, 1.0, 1.0, 0.01, 0.01, 0.01, 0.0) \cdot 10^{-1}$$
$$\boldsymbol{R} = \text{diag}(1.0, 1.0, 0.01, 0.1, 0.1, 0.01) \cdot 10^{-3}.$$

As explained in Section 3.3, a third variable of the current is only added for simplicity with regards to dimensions, and is constantly zero. The gain corresponding to this state in the $\boldsymbol{Q}$ matrix will have no impact on the filter, and is set to zero. In Figure 4.1 the estimated current velocities and directions are given for the two cases, together with the true values of the current. Figure 4.2 shows the position and heading estimates along with the true values, and in Figure 4.3 the estimated and true velocities are shown. From the simulations, it is clear that the filter is able to estimate the states in a satisfactory manner. The figures also show that when the current is stronger the simulated vessel is oscillating more around the given set-points, which results in a larger oscillation in the current velocity estimates.

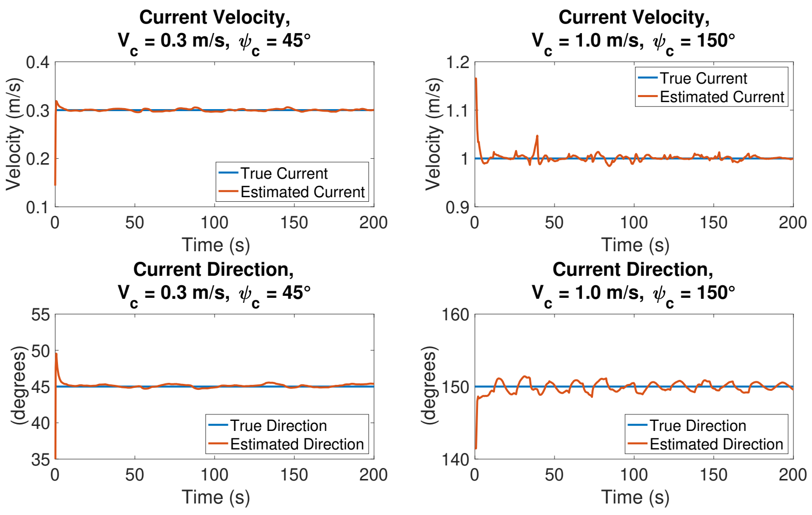

Figure 4.1: Extended Kalman filter estimation of current velocity and direction

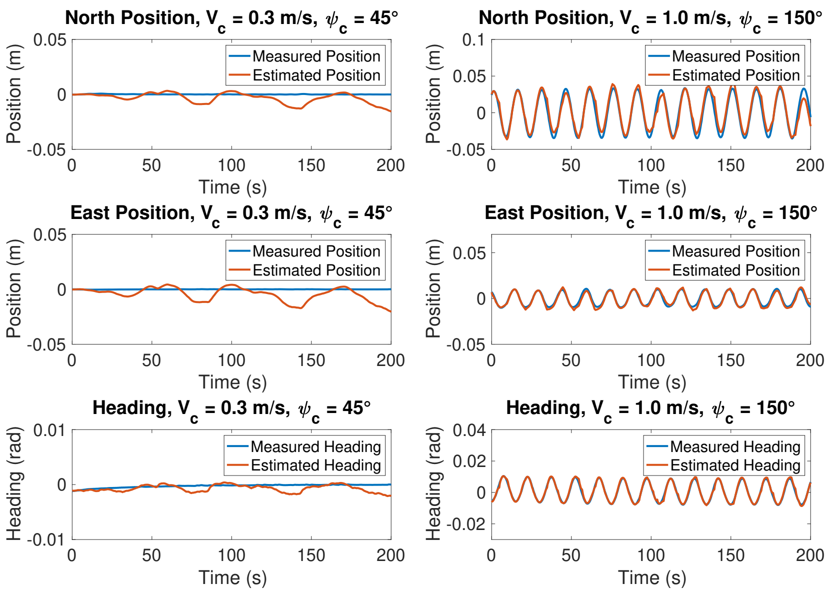

Figure 4.2: Extended Kalman filter estimates of position

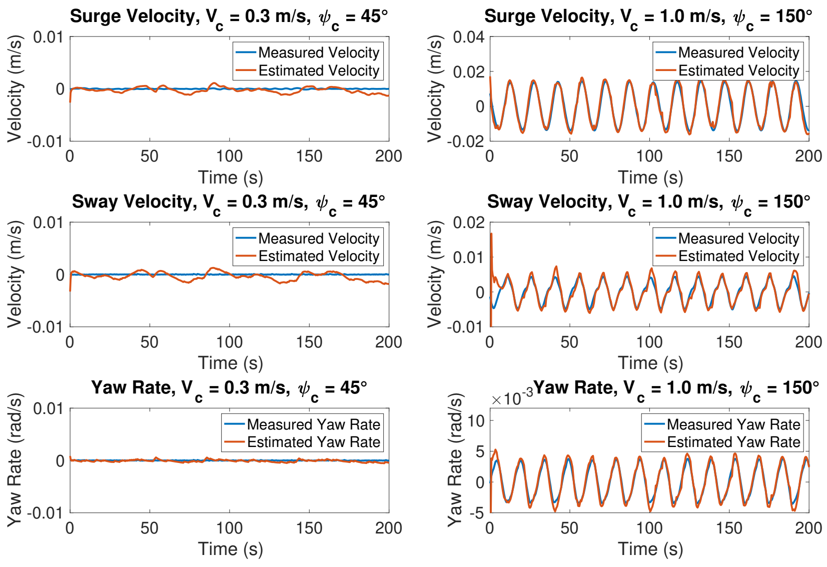

Figure 4.3: Extended Kalman filter estimates of velocities

4.1.2 MAE in the Current Estimation Using the EKF when the Model is Exposed to Noise

The sensitivity of the current estimates using the EKF is evaluated by adding additive noise to three parameters, separately. Noise is added to the control forces and moment,$\boldsymbol{\tau}$, to the Coriolis, centripetal, and damping matrix, $\boldsymbol{N}(\boldsymbol{\nu}_r)$, and to the mass matrix, $\boldsymbol{M}$. The additive noise is 1, 5, 10, or 25 % of the original value. The simulations are performed by having the vessel in DP, with the desired set-points (0, 0, 0). The specified current is then applied, and when the vessel is in the desired position, the simulated data is recorded for approximately 200 seconds. The current velocity, V_c has the magnitude of either 0.3 m/s or 1.0 m/s, and the current direction, ψ_c is 15, 30, 45, 60, 80, 120, 135, or 150°.

In Figures 4.4 to 4.9, the MAEs between the true and estimated current velocities and directions are shown, where the degrees of the polar plot specifies the direction of the current. Figures 4.4 and 4.5 show the MAE when noise is added in $\boldsymbol{\tau}$, Figures 4.6 and 4.7 show the MAE when noise is added in $\boldsymbol{N}(\boldsymbol{\nu}_r)$, and Figures 4.8 and 4.9 show the MAE when noise is added in $\boldsymbol{M}$.

4.1.3 Discussion

As expected, the MAE is larger when the total current velocity is larger. For the control forces and moment, $\boldsymbol{\tau}$ and the Coriolis, centripetal, and damping matrix, $\boldsymbol{N}(\boldsymbol{\nu}_r)$, it is also a clear trend showing that the more disturbance added, the larger the MAE. The MAE for the estimate of the current velocity has the approximate same value independent of the direction of the current. In the current direction estimate, however, the MAEs have some variation across the different directions of the current. For the mass matrix, $\boldsymbol{M}$, it is evident that the MAE of the current estimates is almost unaffected by the added noise, and no clear trend can be found. This makes sense since the data is for an approximately stationary case, and the mass matrix only affects the acceleration.

Figures 4.10 and 4.11 show the MAE for the estimates in the current velocities and directions when the simulated vessel is exposed to currents with directions of 15, 45, 120, and 150 degrees. The figures show how the amount of noise added in $\boldsymbol{\tau}$, $\boldsymbol{N}(\boldsymbol{\nu}_r)$, and $\boldsymbol{M}$ affects the MAE. The change in noise $\boldsymbol{\tau}$ and $\boldsymbol{N}(\boldsymbol{\nu}_r)$ affects the MAE of the current velocities almost identically, while the MAE is almost unaffected by the noise in $\boldsymbol{M}$, as previously noted. In the MAE of the current directions, there are no clear trends, and the MAE-values are quite similar for when noise is added to the different parameters. The MAE-values are not particularly large, even when the noise is 25 %, which indicates that the current estimates given by EKF should be of value even if the true $\boldsymbol{\tau}$ and $\boldsymbol{N}(\boldsymbol{\nu}_r)$ is not known exactly.

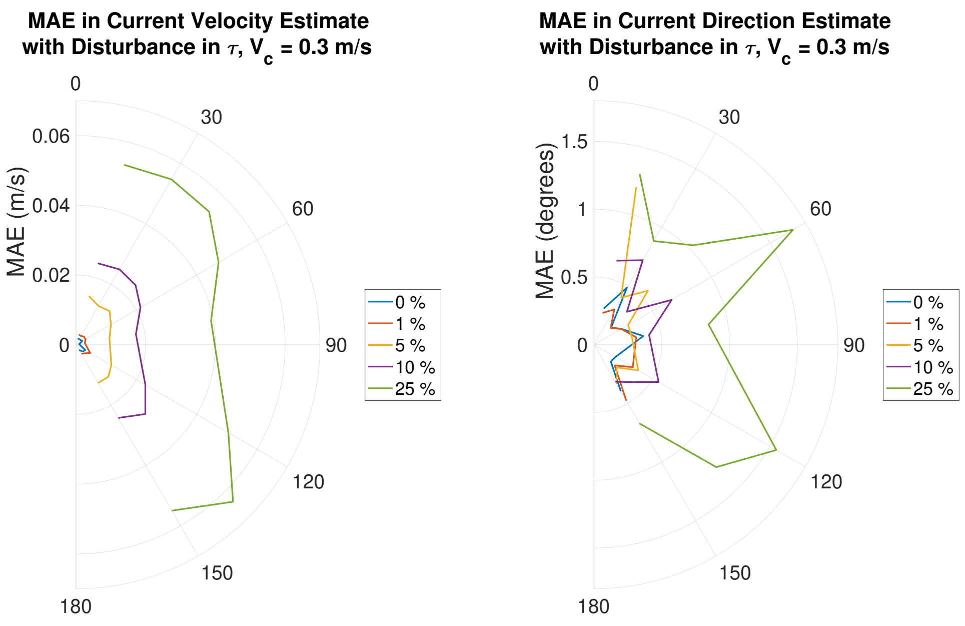

Figure 4.4: MAE for the estimated current velocity (left) and direction (right) shown for different additive disturbances in τ, where $V_c = 0.3$ m/s

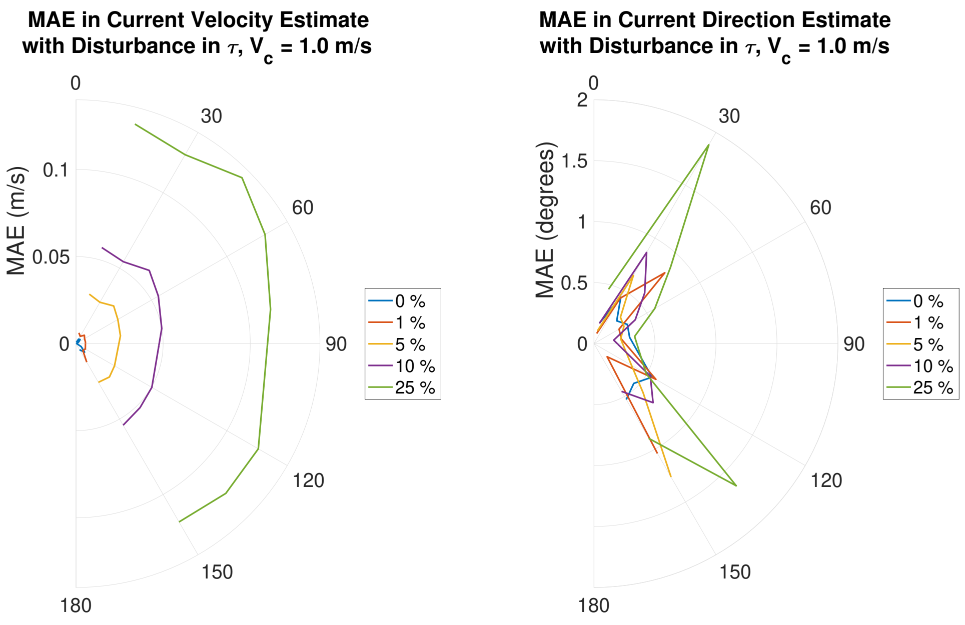

Figure 4.5: MAE for the estimated current velocity (left) and direction (right) shown for different additive disturbances in τ, where $V_c = 1.0$ m/s

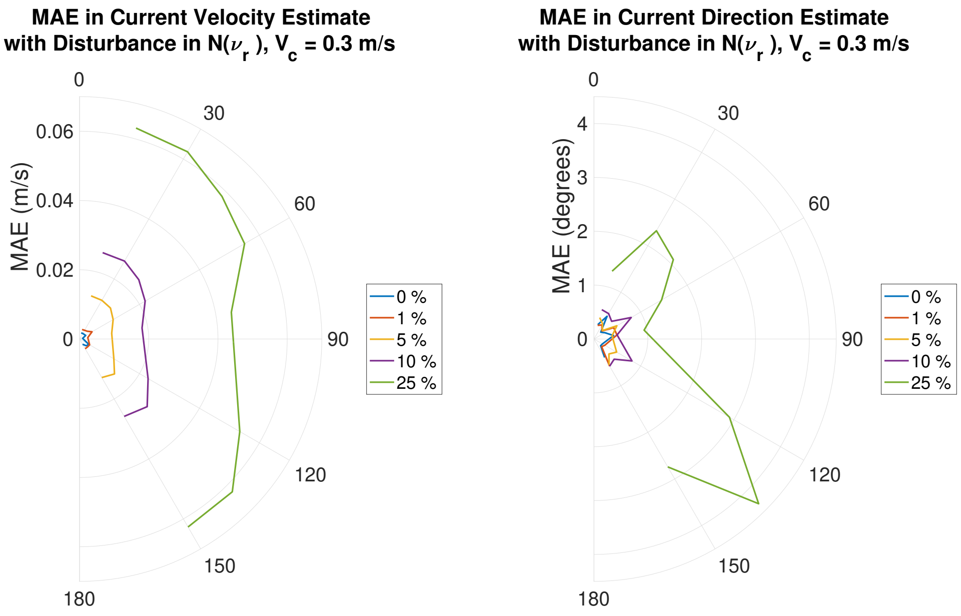

Figure 4.6: MAE for the estimated current velocity (left) and direction (right) shown for different additive disturbances in $\boldsymbol{N}(\boldsymbol{\nu}_r)$, where $V_c = 0.3$ m/s

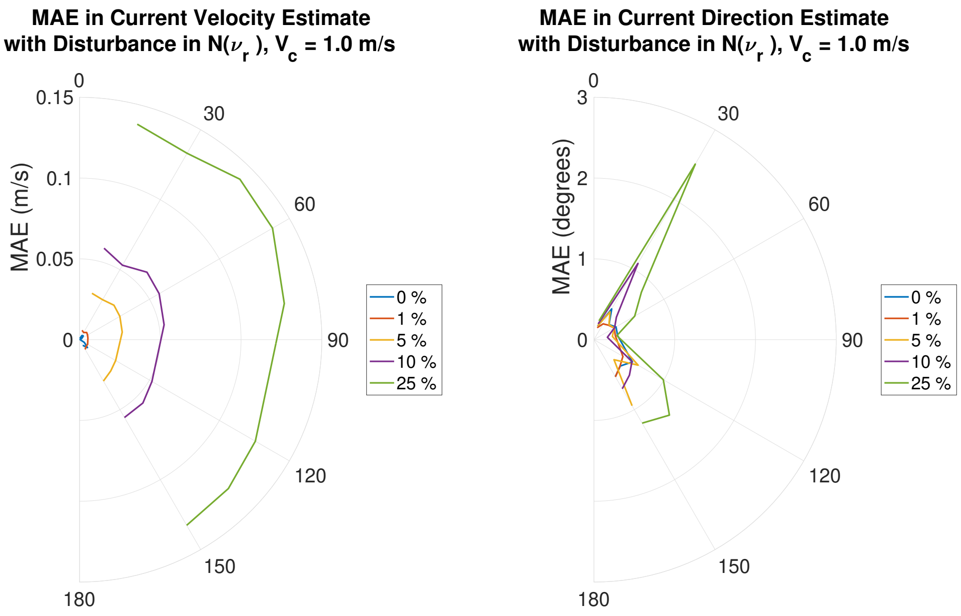

Figure 4.7: MAE for the estimated current velocity (left) and direction (right) shown for different additive disturbances in $\boldsymbol{N}(\boldsymbol{\nu}_r)$, where $V_c = 1.0$ m/s

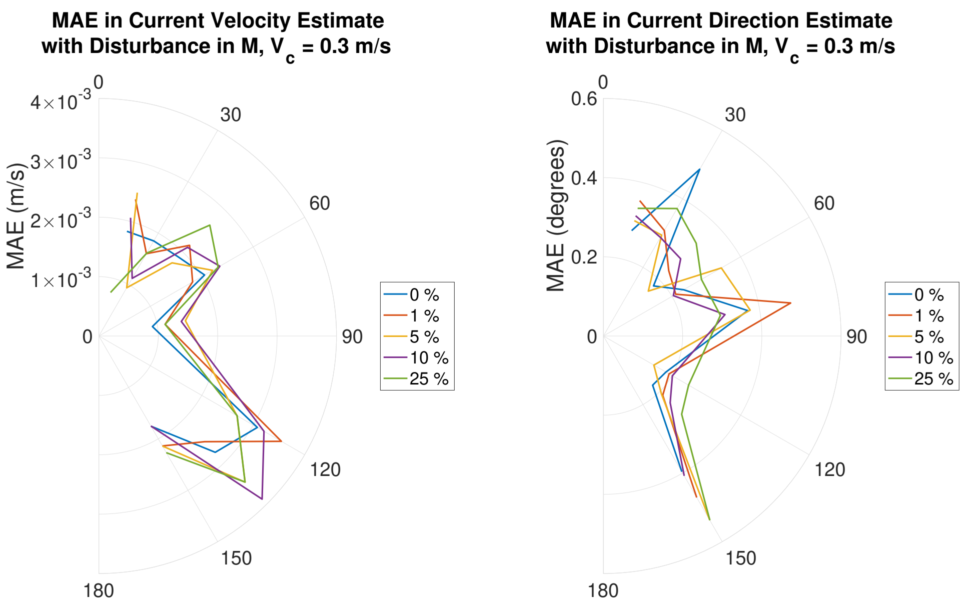

Figure 4.8: MAE for the estimated current velocity (left) and direction (right) shown for different additive disturbances in M, where $V_c = 0.3$ m/s

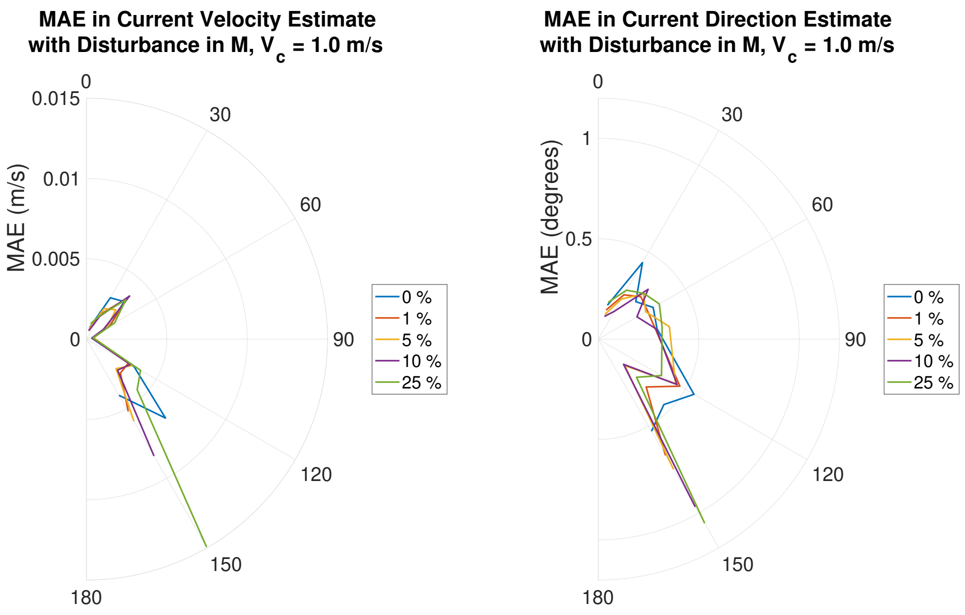

Figure 4.9: MAE for the estimated current velocity (left) and direction (right) shown for different additive disturbances in M, where $V_c = 1.0$ m/s

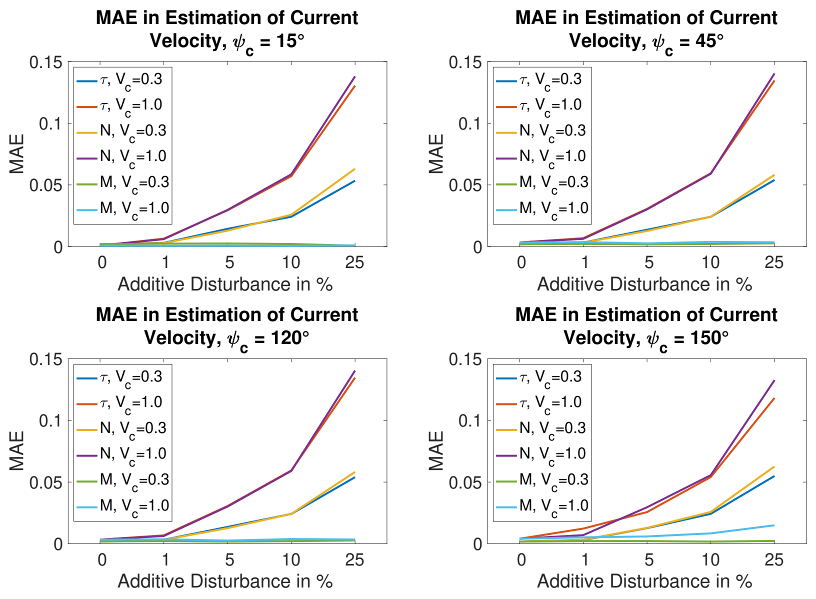

Figure 4.10: MAE in estimated current velocity depending on noise in $\boldsymbol{\tau}$, $\boldsymbol{N}(\boldsymbol{\nu}_r)$, or $\boldsymbol{M}$.

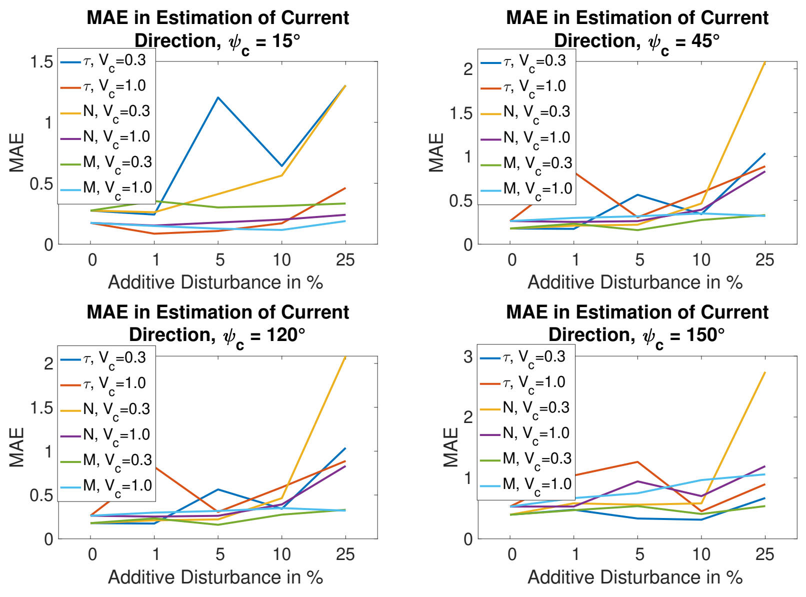

Figure 4.11: MAE in estimated current direction depending on noise in $\boldsymbol{\tau}$, $\boldsymbol{N}(\boldsymbol{\nu}_r)$, or $\boldsymbol{M}$.

4.2 Case Study: Current Estimation with Neural Networks

In this section, the results from the current estimation using two different types of machine learning models are presented and discussed. Using the parameters given in the simulator of MilliAmpère, random data is generated for the steady-state model as explained in Section 3.4. 100 000 data-points are generated, which is further divided into a training set of 80 000 data-points and a validation set of 20 000 data-points. A separate test set of 20 000 data-points is also generated, which has the same purpose as the validation set but is created as a separate set to make sure no information has leaked from the validation set to the model. At last, a test set with noise is generated, consisting of 20 000 samples, where the noise has a normal distribution, $\mathcal{N}(0, 0.05)$. The noise is added to the Coriolis, centripetal, and damping matrix when the data is generated, resulting in noise in $\boldsymbol{\tau}$, which is given as input to the model. In Figure 4.12 some samples of input data from the test set with and without noise are shown. The following section presents the results obtained after training deep neural networks and the next section presents the results obtained after training radial basis function networks.

4.2.1 Testing of Deep Neural Networks for Current Estimation

Deep neural network models are developed by choosing the number of layers and the numbers of neurons within each layer, together with the activation functions and the optimizer. Three different sizes of neural networks are tested, where the first neural network is set to have one densely connected layer with 100 neurons, the second is set to have three sequential densely connected layers with 100 neurons at each layer, and the third is set to have three sequential layers with 500 neurons at each layer. Each of the models also has one final densely connected output layer, consisting of two neurons. The hidden layers have the activation function *sigmoid*, as presented in Chapter 2, and the output layer has a linear activation function. The optimizer used is *Adam*, and the number of epochs is set to 100. The training and validation loss, in the form of mean squared error, is presented together with the mean absolute error in Figure 4.13.

For machine learning models, it is common that the model overfits on the training data if the model is too large, resulting in a greater validation loss compared to the training loss. Although this cannot be seen from Figure 4.13, two measures to counteract overfitting is tested. One consists of a dropout layer, with a dropout rate of 0.2 after each layer, and the other consists of weight regularization. The weight

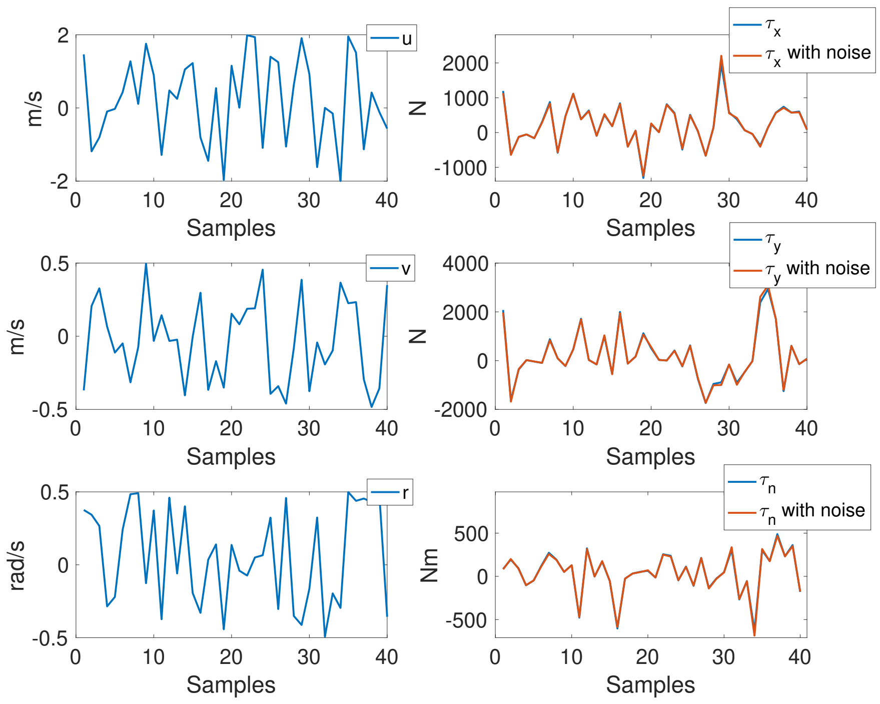

Figure 4.12: Samples of the random input data, body-fixed velocities (left) and control forces and moment (right)

regularization is added in the form of L2, which means it has a cost proportional to the square of the value of the weight coefficients and has a value of 10^{-4}. The MSE and MAE for the training, validation, test, and test set with noise are shown in Table 4.1, for the three original models, as well as the original models with measures to counteract overfitting. The predictions of some samples are shown in Figure 4.14, where the predictions are done by the model with three layers and 100 neurons and by the model with three layers and 500 neurons in each and dropout between the layers. The samples are from the test sets with and without noise, and the input data to these samples are presented in Figure 4.12.

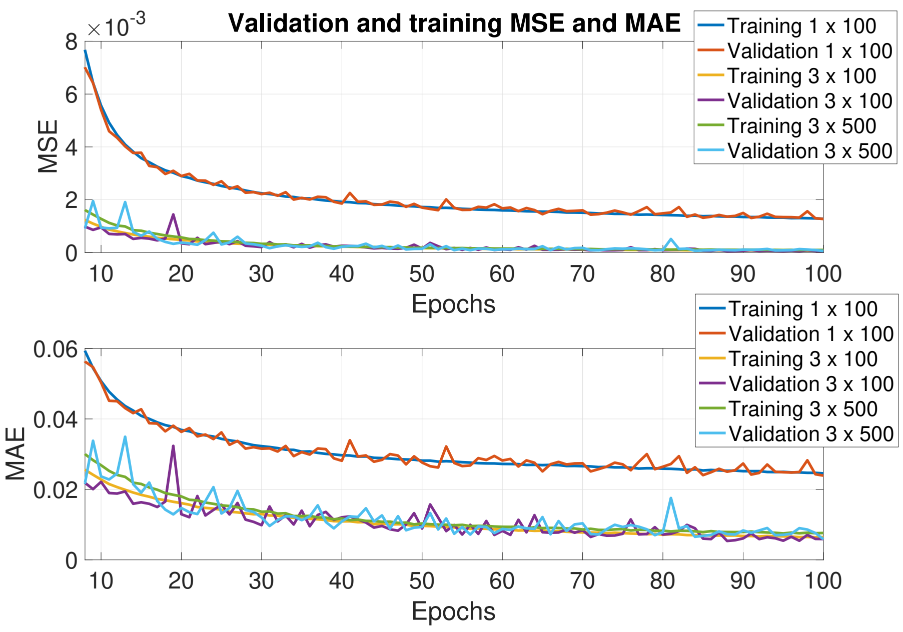

Figure 4.13: MSE and MAE for training and validation data for different deep densely connected neural networks

4.2.2 Testing of RBFNs for Current Estimation

Two different types of RBFNs are tested, one where the widths belonging to a center is constant for all the features, and one where the widths vary depending on the features. As for the deep neural network the MSE and MAE for the training, validation, test, and test data with noise are shown in Table 4.2. The parameters to be varied are the number of neurons, how the initial values of the centers are set, and the initial widths, in the form of β when the widths are identical across the features and in the form of $\boldsymbol{r}$ when the widths vary across the features. It was found that the results after running the model for several epochs were almost independent of the initial values of the widths, and these are therefore set to one for all the following tests. The numbers of neurons are either 50, 100, or 200, and the initial centers are decided as either random samples of the training data, or through the K-means algorithm explained in Chapter 2. The optimizer used is *Adam*, and the number of epochs is also here set to 100. The prediction for some input samples using one model with 200 neurons and constant width across the features, β, and one model with 100 neurons and varying width across the features, $\boldsymbol{r}$ is shown in Figure 4.15. The input data to these predictions are shown in Figure 4.12.

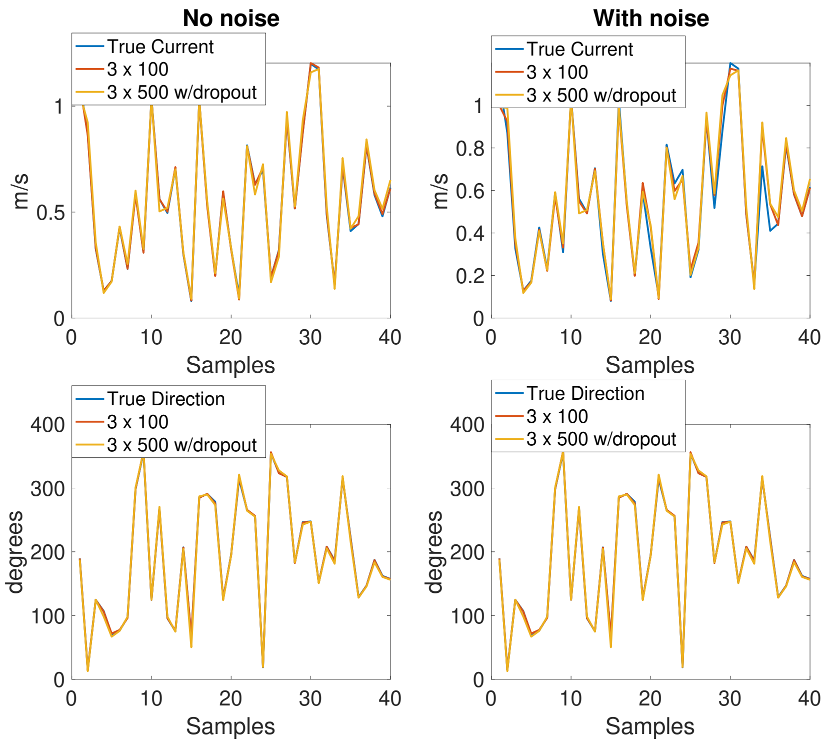

Figure 4.14: Prediction of current velocity (top) and direction (bottom) for samples from the test sets with and without noise using the models of three densely connected layers and 100 neurons and three densely connected layers and 500 neurons with dropout

4.2.3 Discussion

As shown in Figure 4.13, the deep neural network models do not seem to overfit, but rather stagnates at approximately the same value for both the training and validation data. From the values presented in Table 4.1, it is clear that having three layers with 100 neurons each or 500 neurons each does not significantly improve the model. The model also performs poorer when adding dropout, and the dropout especially affects the model with the layers of 100 neurons, which can imply that with dropout, the model is not complex enough to sufficiently learn a good representation of the data. The change in loss between the test set and the test set with noise, however, is small for the model with dropout, which may indicate that this model is more robust. The values for the performance when the model is tested on the validation and the test data is almost identical, and also very close the performance values for the training data, which is expected since the data is generated

Table 4.1: MSE and MAE for training, validation, and test data using a deep densely connected neural networks (DDNNs)

DDNN Model	Training MSE (MAE)	Validation MSE (MAE)	Test MSE (MAE)	Test w/noise MSE (MAE)
1×100	$1.2 \cdot 10^{-3}$ (0.0237)	$1.3 \cdot 10^{-3}$ (0.0239)	$1.3 \cdot 10^{-3}$ (0.0241)	$2.8 \cdot 10^{-3}$ (0.0379)
3×100	$5.3 \cdot 10^{-5}$ (0.0053)	$5.5 \cdot 10^{-5}$ (0.0054)	$5.5 \cdot 10^{-5}$ (0.0054)	$1.4 \cdot 10^{-3}$ (0.0253)
3×100 with dropout	$1.9 \cdot 10^{-3}$ (0.0320)	$2.0 \cdot 10^{-3}$ (0.0325)	$2.0 \cdot 10^{-3}$ (0.0323)	$3.1 \cdot 10^{-3}$ (0.0411)
3×100 w/weight regularization	0.0231 (0.0458)	0.0232 (0.0463)	0.0232 (0.0463)	0.0247 (0.0571)
3×500	$6.0 \cdot 10^{-5}$ (0.0058)	$6.2 \cdot 10^{-5}$ (0.0059)	$6.3 \cdot 10^{-5}$ (0.0059)	$1.4 \cdot 10^{-3}$ (0.0249)
3×500 with dropout	$6.8 \cdot 10^{-4}$ (0.0195)	$7.0 \cdot 10^{-4}$ (0.0198)	$7.0 \cdot 10^{-4}$ (0.0198)	$1.9 \cdot 10^{-3}$ (0.0312)
3×500 w/weight regularization	0.0261 (0.0525)	0.0262 (0.0528)	0.0262 (0.0532)	0.0276 (0.0623)

from deterministic equations, and the model seems to learn this deterministic relationship very well. This may be why overfitting was not seen in Figure 4.13. Even though the models without any measures for preventing overfitting also results in the best estimates for the test data with noise, it is worth noticing that the change in MSE and MAE is significantly larger for these models compared to the models with weight regularization and dropout. This indicates that adding weight regularization or dropout leads to more robust models, but there is a significantly larger loss in the models with weight regularization.

Table 4.2: MSE and MAE for training, validation, and test data using RBFNs

RBFN Model	Training MSE (MAE)	Validation MSE (MAE)	Test MSE (MAE)	Test w/noise MSE (MAE)
50 neurons random c_{init} $\boldsymbol{\beta} = \mathbf{1}_{50\times 1}$	0.0119 (0.0772)	0.0123 (0.0783)	0.0123 (0.0780)	0.0139 (0.0851)
50 neurons K-means c_{init} $\boldsymbol{\beta} = \mathbf{1}_{50\times 1}$	0.0125 (0.0785)	0.0128 (0.0795)	0.0128 (0.0790)	0.0144 (0.0861)
100 neurons random c_{init} $\boldsymbol{\beta} = \mathbf{1}_{100\times 1}$	$5.5 \cdot 10^{-3}$ (0.0505)	$5.8 \cdot 10^{-3}$ (0.0516)	$5.8 \cdot 10^{-3}$ (0.0518)	$7.5 \cdot 10^{-3}$ (0.0619)
200 neurons random c_{init} $\boldsymbol{\beta} = \mathbf{1}_{200\times 1}$	$2.6 \cdot 10^{-3}$ (0.0340)	$2.8 \cdot 10^{-3}$ (0.0347)	$2.9 \cdot 10^{-3}$ (0.0355)	$4.5 \cdot 10^{-3}$ (0.0481)
200 neurons K-means c_{init} $\boldsymbol{\beta} = \mathbf{1}_{200\times 1}$	$2.7 \cdot 10^{-3}$ (0.0342)	$2.8 \cdot 10^{-3}$ (0.0348)	$2.9 \cdot 10^{-3}$ (0.0353)	$4.5 \cdot 10^{-3}$ (0.0479)
50 neurons random c_{init} $\boldsymbol{r} = \mathbf{1}_{50\times 6}$	$7.2 \cdot 10^{-4}$ (0.0193)	$7.5 \cdot 10^{-4}$ (0.0196)	$7.6 \cdot 10^{-4}$ (0.0197)	$2.2 \cdot 10^{-3}$ (0.0339)
50 neurons K-means c_{init} $\boldsymbol{r} = \mathbf{1}_{50\times 6}$	$6.9 \cdot 10^{-4}$ (0.0192)	$7.1 \cdot 10^{-4}$ (0.0194)	$7.3 \cdot 10^{-4}$ (0.0196)	$2.2 \cdot 10^{-3}$ (0.0339)
100 neurons random c_{init} $\boldsymbol{r} = \mathbf{1}_{100\times 6}$	$2.6 \cdot 10^{-4}$ (0.0114)	$2.7 \cdot 10^{-4}$ (0.0116)	$2.7 \cdot 10^{-4}$ (0.0117)	$1.7 \cdot 10^{-3}$ (0.0287)
200 neurons random c_{init} $\boldsymbol{r} = \mathbf{1}_{200\times 6}$	$1.5 \cdot 10^{-4}$ (0.0090)	$1.6 \cdot 10^{-4}$ (0.0091)	$1.6 \cdot 10^{-4}$ (0.0091)	$1.6 \cdot 10^{-3}$ (0.0276)
200 neurons K-means c_{init} $\boldsymbol{r} = \mathbf{1}_{200\times 6}$	$9.6 \cdot 10^{-5}$ (0.0070)	$1.2 \cdot 10^{-4}$ (0.0072)	$1.0 \cdot 10^{-4}$ (0.0072)	$1.6 \cdot 10^{-3}$ (0.0269)

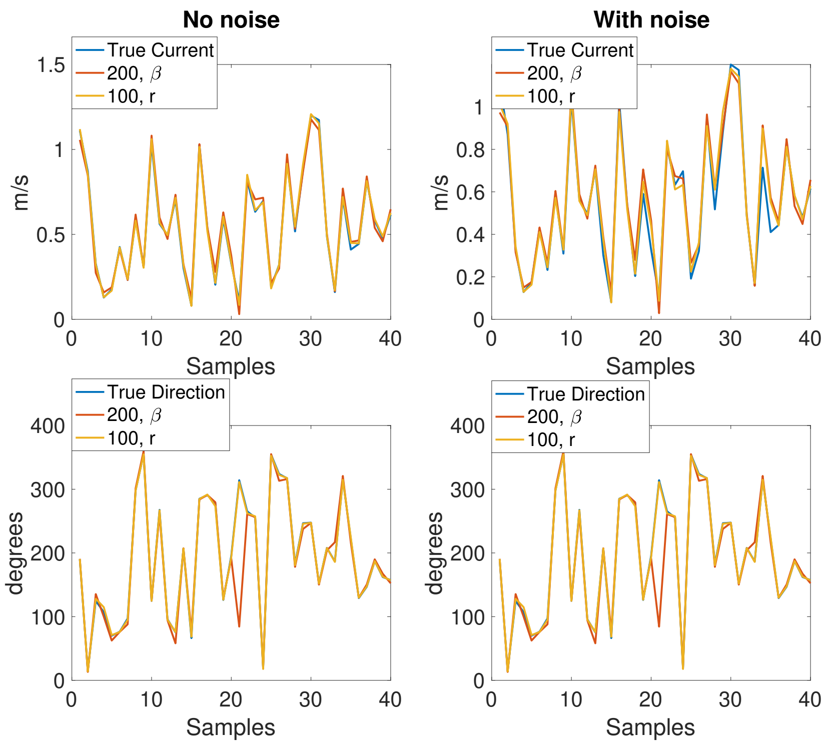

Figure 4.15: Prediction of current velocity (top) and direction (bottom) for samples from the test sets with and without noise using the RBFN model with 200 neurons and constant width across features, β, and the RBFN model with 100 neurons and varying widths across features, $\boldsymbol{r}$

When using an RBFN model to predict the current, it is also here seen that the model performs almost identically on the training, validation, and test set, as it also did with the deep neural networks, which again can be explained by the deterministic equations generating the data. From the data in Table 4.2, it is also seen that adding more neurons leads to a smaller MSE and MAE. However, the model with just 50 neurons performs significantly better when the widths of the radial basis function vary across each feature. When looking at the trainable parameters, this is not surprising, since an RBFN with 50 neurons and a constant width across the features generates 350 parameters in the RBF-layer, $50 \times 6 = 300$, for the centers of each neuron across 6 dimensions, and 50 for the widths belonging to each center. The amount of parameters is, therefore, almost doubled when the widths vary across the features, and the layer has a total of 600 parameters, $50 \times 6 = 300$ for the centers, and $50 \times 6 = 300$ for the widths. Even though the models with varying widths across features performs better than the models with constant widths on both the test data with and without noise, it is worth noticing that the change in MSE and MAE between the two test sets are greater for the models with varying widths. This means that these models may not perform better on totally unseen data which varies some from the data generated by the deterministic equations.

Running a K-means algorithm to decide the initial values of the centers seems to have some improvement of the MSE and MAE for the models where the widths vary across the features, but it does not have any effect for the models with constant widths. Initializing the centers using the K-means algorithm does, however, not seem to have any effect on how the model behaves when evaluated on test data with noise, and this may be a sign that the K-means algorithm leads to some overfitting. Since the test data is generated at random, it makes sense that there may not be any clear clusters in the dataset, which means that performing a K-means clustering on the initial centers will not lead to improved learning for the models. The RBF-layer implemented by Vidnerova (2019) learn the desired centers and widths when the network is trained, and not the weights corresponding to given centers and weights. The initial values of the parameters may therefore not be as important, and the models initialized by the K-means algorithm will not be considered further.

The performance of the deep neural networks and the RBFNs are very similar, but a slightly smaller loss is achieved for the test data for the deep densely connected neural networks. For the test data with noise, the best performing models obtain almost the same loss. The RBFNs use significantly fewer parameters compared to the deep neural networks which often end up with thousands of parameters when multiple layers are used. This makes the RBFNs quicker to train.

Chapter 5

Full-Scale Experiment Results and Discussion

Three full-scale experiments were performed, at different locations and times. These are further referred to as experiment 1, 2, and 3. The experiments were performed using the MilliAmpère ferry, and took place in Nidelva, Trondheim, on the 20th of May 2020. The locations for the three experiments are shown in Figure 5.1. The specifications of the test environment, along with the time intervals for each of the experiments, are shown in Table 5.1. The wind velocity is given according to the weather forecast, but since MilliAmpère is not equipped with an anemometer, the wind velocity is not taken further into account. The tide table for Trondheim on the 20th of May 2020 is shown in Table 5.2.

Each experiment started with letting MilliAmpère float freely for some time, in order to measure an estimate for the velocity and direction of the current at that time instant. After the true values for the current velocity and direction was found for each experiment, multiple tests were performed where the DP system implemented on MilliAmpère was used to either keep MilliAmpère still at a given location with a given heading or move with a constant velocity. The recorded data of MilliAmpère's movement when floating freely is shown in Figures 5.2 to 5.4, for experiments 1, 2, and 3, respectively. The figures also show the approximately stationary areas for which the mean values for the current velocity and direction are given. These mean values, which will further be assumed to be the true values for the current, are shown in Table 5.3. It can be seen that the true values for the current velocities for experiments 1 and 2 are quite different, even though there is not a lot of time between the two experiments. The location between experiments

1 and 2, however, is significant. This indicates that there are large local variations in the current.

In the following sections the current estimates obtained for the experimental data, using both the machine learning models, presented in Section 3.4, and the EKF, presented in Section 3.3, are presented. The control forces used as input for both of these methods are the control forces specified by the DP system of MilliAmpère. The results obtained from a test where the current estimates from both the EKF and the machine learning models are unsatisfactory are also presented before the results are discussed. The source code for the functions used to obtain these results are presented in Appendix B.

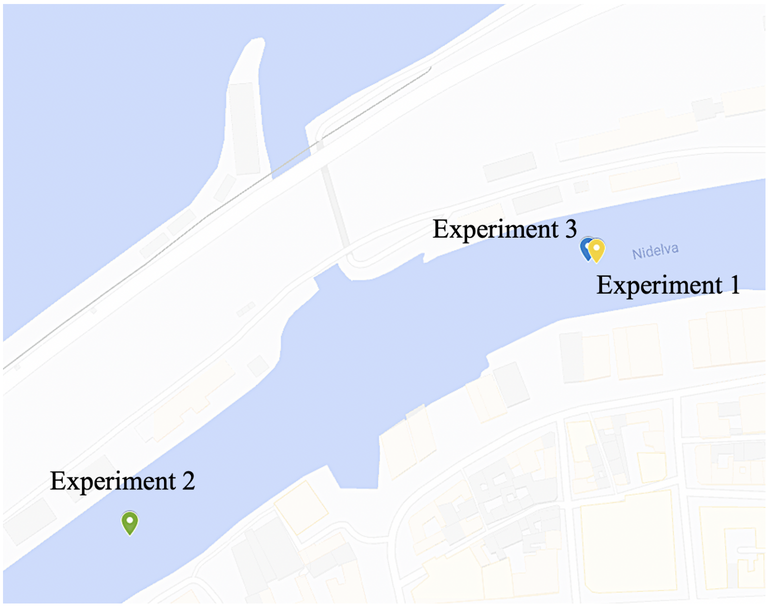

Figure 5.1: Map showing the approximate locations of the three experiments

Table 5.1: Specifications of the test environment

Date	20th of May 2020
Wind estimate	3-5 m/s
Temperature	$6°C$
Experiment 1	09:33 - 09:46
Experiment 2	10:07 - 10:12
Experiment 3	14:07 - 14:25

Table 5.2: Tide times for Trondheim, 20th of May 2020

Tide	Time	Height
Low tide	05:13	0.77 m
High tide	11:21	2.51 m
Low tide	17:25	0.54 m
High tide	23:48	2.62 m

Table 5.3: The mean true values for velocity and direction of the current

Experiment	1	2	3
Velocity (m/s)	0.48	0.29	0.43
Direction (degrees)	262.9	226.0	238.0

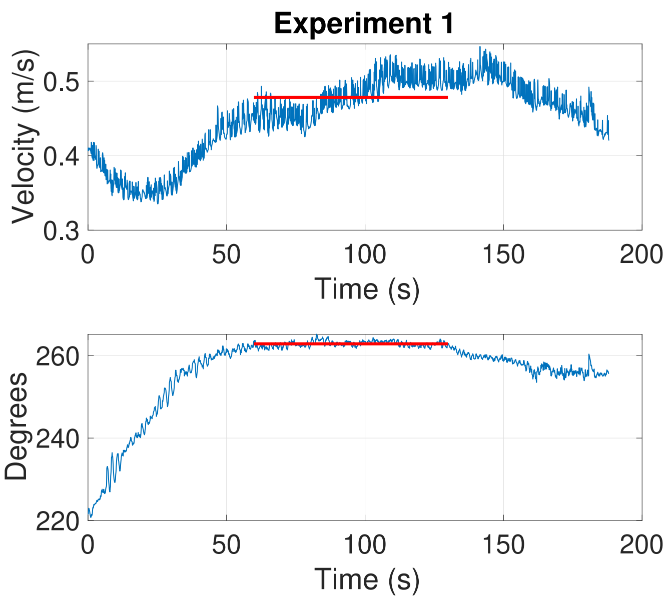

Figure 5.2: Velocity and direction of MilliAmpère when it is floating freely during experiment 1, together with the intervals for the mean values, indicated by the red line

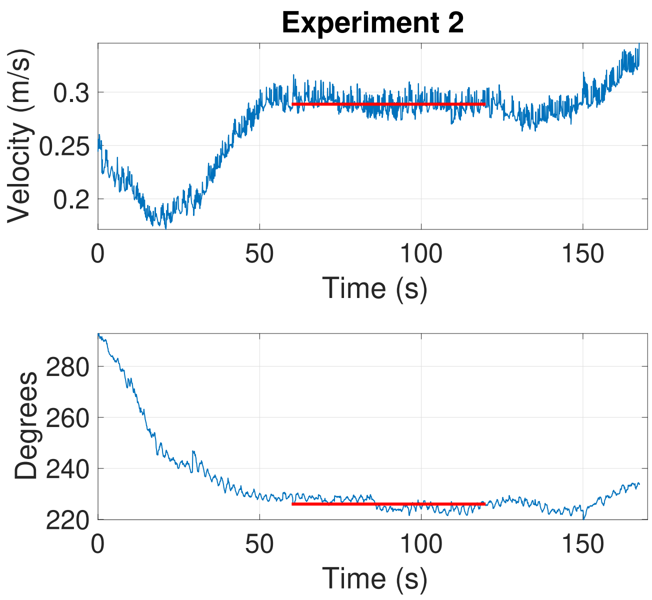

Figure 5.3: Velocity and direction of MilliAmpère when it is floating freely during experiment 2, together with the intervals for the mean values, indicated by the red line

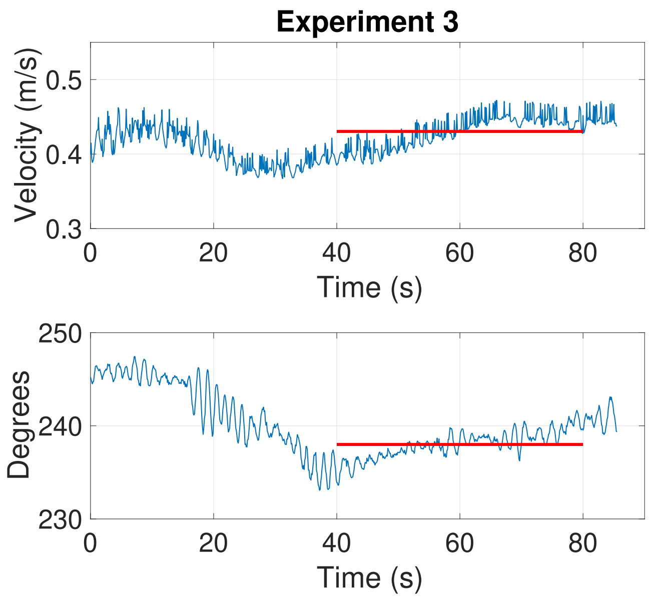

Figure 5.4: Velocity and direction of MilliAmpère when it is floating freely during experiment 3, together with the intervals for the mean values, indicated by the red line

5.1 Specifications of the Current Estimation Methods

In this section, a brief overview of the methods used to perform the current estimation is given. Current estimates are performed on data recorded when MilliAmpère was floating freely, was in DP, and was moving with constant velocity.

5.1.1 Estimation of Current Using Machine Learning

Two different types of machine learning models are used to estimate the current, and consist of deep densely connected neural networks and radial basis function networks. The models used are the ones trained on the theoretical data as presented in Section 4.2, except that the RBFNs with centers initialized by K-means are not presented since they showed no improvement on the theoretical test data. The machine learning models are developed for stationary cases, and approximately stationary intervals of the test cases are used.

5.1.2 Estimation of Current Using the EKF

The extended Kalman filter is implemented using the alterations given in Section 3.3.2. The heading used in the rotation matrices is the measured heading. The gains of the $\boldsymbol{R}$ matrix was initially found by performing high-pass filtering of the measurements and calculating the variance. This proved unsatisfactory, and the $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrices were manually tuned further. The final gains used are:

$$\boldsymbol{Q} = \text{diag}(1.0, 1.0, 0.01, 1.0, 1.0, 0.01, 10, 10, 0.0, 1.0 \cdot 10^2, 1.0 \cdot 10^2, 2.0 \cdot 10^2)$$
$$\boldsymbol{R} = \text{diag}(3.0 \cdot 10^{-2}, 3.0 \cdot 10^{-2}, 9.0 \cdot 10^{-4}, 4.0 \cdot 10^{-2}, 4.0 \cdot 10^{-2}, 1.0 \cdot 10^{-3}).$$

5.2 Estimation of Current when MilliAmpère is Floating Freely

This section presents the results obtained using machine learning models and the EKF to estimate the current using the same recordings used to determine the true values of the current, that is when MilliAmpère is floating freely. The estimation of the current is only done in the approximately stationary area.

5.2.1 Estimation of Current Using Machine Learning

The mean predicted current velocities and directions are presented for the deep densely connected neural networks (DDNNs) in Table 5.4, and the mean predicted current velocities and directions for the radial basis function networks (RBFNs) are presented in Table 5.5. The models indicated by green in the tables are the presumably best predicting models.

5.2.2 Estimation of Current Using the EKF

Since MilliAmpère is floating freely, there are no control forces. The input $\boldsymbol{\tau}$ is set to zero, and $\boldsymbol{I}_{3\times3}$ in the bottom right of the $\boldsymbol{E}$-matrix in Equation (3.11g) is replaced with $\mathbf{0}_{3\times3}$, so that this state will have no impact on the rest of the Kalman filter. The estimated current velocity and direction are shown in Figures 5.5 to 5.7. The predictions from two machine learning models are also shown, one deep densely connected neural network (DDNN) with three layers of 500 neurons and dropout, and one RBFN with 100 neurons and varying widths across the features. The EKF is assumed to stabilize after approximately 20 seconds, shown as the green line in the figures. The mean estimated values for the current velocity and direction obtained using the EKF, together with the MAE between the estimated and the assumed true values are presented in Table 5.6. These values are calculated on the estimations after the filter is assumed to have stabilized. In Appendix C.1,C.2, and C.3 the estimates for the position and velocities of MilliAmpère are presented for the three tests.

Table 5.4: Mean predicted values for current velocity and direction using deep dense neural network (DDNN) models when MilliAmpère is floating freely. The presumed best networks are indicated in green.

DDNN Model	Experiment 1 Floating 0.48 m/s 262.9°	Experiment 2 Floating 0.29 m/s 226.0°	Experiment 3 Floating 0.43 m/s 238.0°
1×100	0.43 m/s 264.4°	0.24 m/s 228.9°	0.36 m/s 237.9°
3×100	0.48 m/s 262.2°	0.27 m/s 224.9°	0.43 m/s 239.2°
3×100 with dropout	0.47 m/s 258.6°	0.25 m/s 221.0°	0.41 m/s 241.0°
3×100 w/weight regularization	0.33 m/s 265.7°	0.19 m/s 232.1°	0.30 m/s 238.7°
3×500	0.48 m/s 263.5°	0.26 m/s 224.9°	0.42 m/s 239.2°
3×500 with dropout	0.47 m/s 260.7°	0.26 m/s 222.2 °	0.46 m/s 236.4°
3×500 w/weight regularization	0.35 m/s 269.5°	0.19 m/s 236.4°	0.30 m/s 239.9 °

Table 5.5: Mean predicted values for current velocity and direction using RBFN models when MilliAmpère is floating freely. The presumed best networks are indicated in green.

RBFN Model	Experiment 1 Floating 0.48 m/s 262.9°	Experiment 2 Floating 0.29 m/s 226.0°	Experiment 3 Floating 0.43 m/s 238.0°
50 neurons random c_{init} $\boldsymbol{\beta} = \mathbf{1}_{50\times1}$	0.33 m/s 262.9°	0.23 m/s 229.2°	0.27 m/s 235.5°
100 neurons random c_{init} $\boldsymbol{\beta} = \mathbf{1}_{100\times1}$	0.33 m/s 280.3°	0.20 m/s 250.9°	0.21 m/s 252.6°
200 neurons random c_{init} $\boldsymbol{\beta} = \mathbf{1}_{200\times1}$	0.42 m/s 267.4°	0.25 m/s 236.8°	0.37 m/s 243.7°
50 neurons random c_{init} $\boldsymbol{r} = \mathbf{1}_{50\times6}$	0.46 m/s 261.6°	0.26 m/s 227.1°	0.45 m/s 233.9°
100 neurons random c_{init} $\boldsymbol{r} = \mathbf{1}_{100\times6}$	0.46 m/s 266.7°	0.25 m/s 231.3°	0.40 m/s 239.5°
200 neurons random c_{init} $\boldsymbol{r} = \mathbf{1}_{200\times6}$	0.46 m/s 266.8°	0.25 m/s 228.3°	0.40 m/s 239.5°

Table 5.6: Mean estimated values for the current velocity and direction and MAE between the true and the estimated velocities and directions

Experiment	1 0.48 m/s 262.9°	2 0.29 m/s 226.0°	3 0.43 m/s 238.0°
Mean Velocity (m/s)	0.47	0.24	0.48
Mean direction (degrees)	246.0	195.9	246.4
MAE Velocity (m/s)	0.174	0.060	0.064
MAE Direction (degrees)	16.96	31.18	11.97

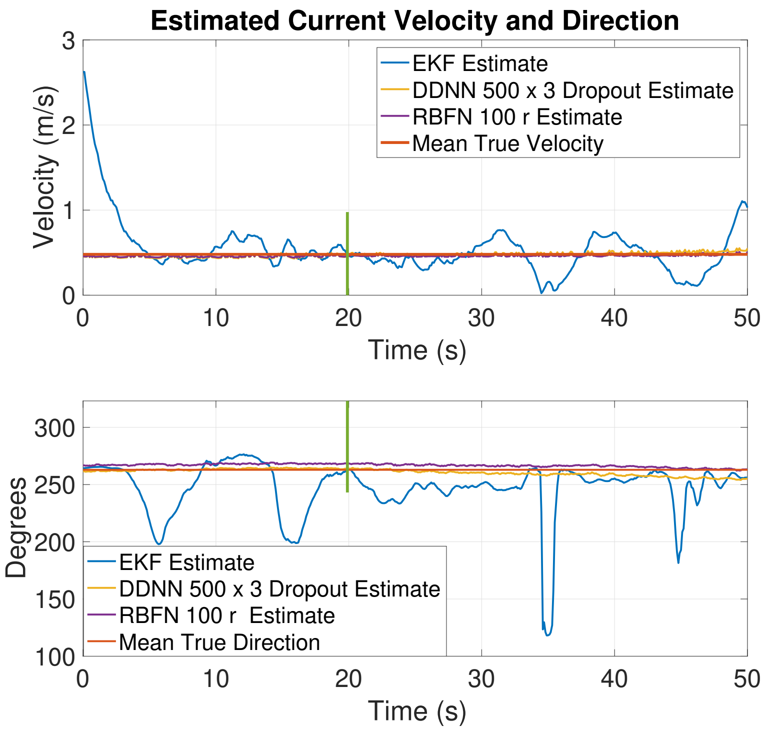

Figure 5.5: Estimated velocity (top) and direction (bottom) of the current during experiment 1 when MilliAmpère is floating freely. The estimates are obtained using the EKF and machine learning models.

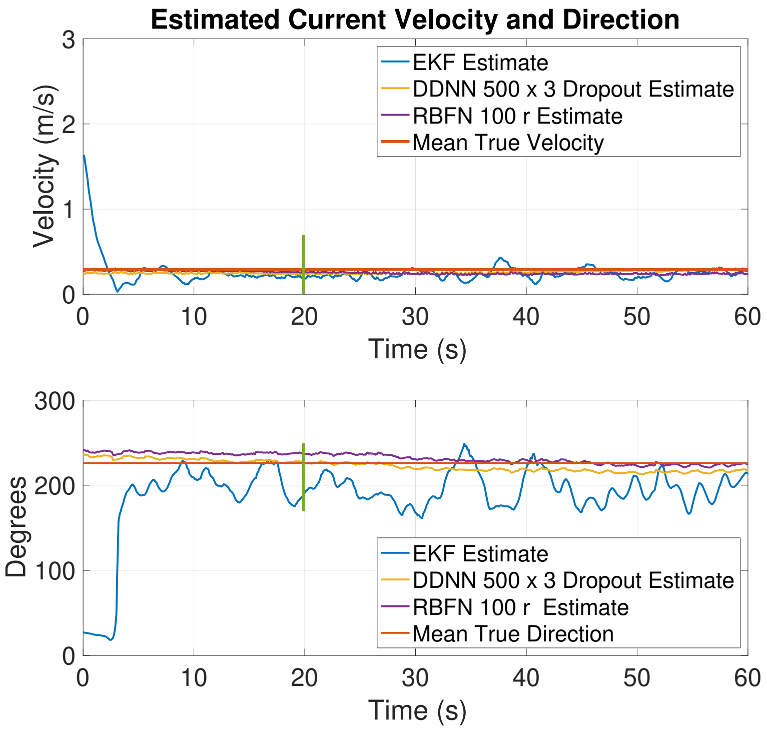

Figure 5.6: Estimated velocity (top) and direction (bottom) of the current during experiment 2 when MilliAmpère is floating freely. The estimates are obtained using the EKF and machine learning models.

5.3 Estimation of Current when MilliAmpère is in DP or Moving with Constant Velocity

The DP system implemented on MilliAmpère is used to keep it at a specified location and with a given heading, or move with constant velocity and a given heading. Different headings were tested for each experiment, and the desired heading for each test is shown in Table 5.7. Four different tests are presented, three where MilliAmpère is in DP, and one where MilliAmpère is moving with constant velocity and constant heading.

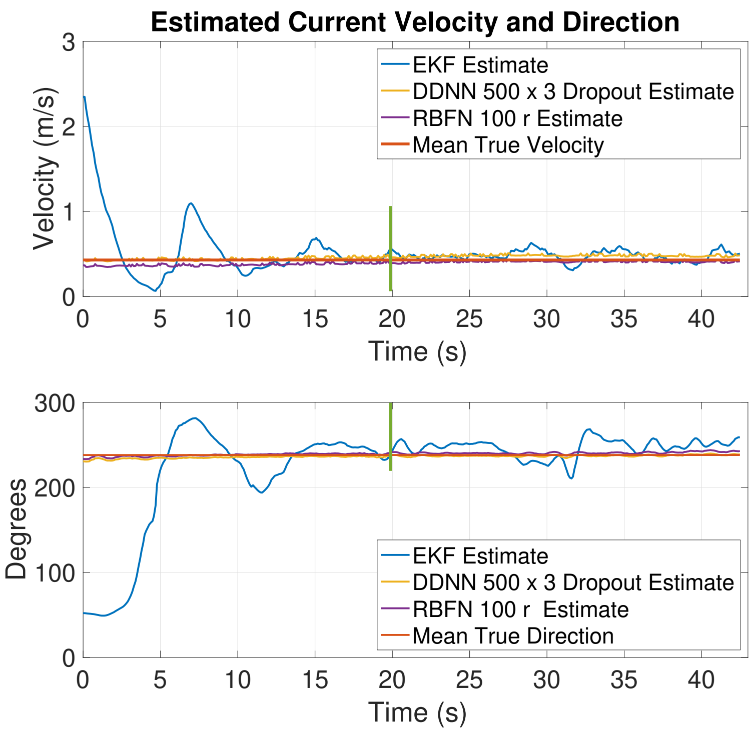

Figure 5.7: Estimated velocity (top) and direction (bottom) of the current during experiment 3 when MilliAmpère is floating freely. The estimates are obtained using the EKF and machine learning models.

Table 5.7: The desired headings for each of the tests performed during experiments 1, 2, and 3

Experiment	1 DP	2 DP	3 DP	3 constant velocity
Desired Heading	0.5 rad 28.6 °	1.0 rad 57.3 °	1.57 rad 90.0 °	1.57 rad 90.0 °

5.3.1 Estimation of Current Using Machine Learning

In Table 5.8 the mean predicted current velocities and directions are shown for the deep densely connected neural networks, and in Table 5.9 the mean predicted current velocities and directions are shown for the radial basis function networks

(RBFNs). The models with the presumed best estimates are marked with green.

5.3.2 Estimation of Current Using the EKF

The results from applying the EKF to the tests presented in Table 5.7 are presented in this section. The estimates for the current velocity and direction for experiment 1 is given in Figure 5.8, experiment 2 is given in Figure 5.9, experiment 3 in DP is given in Figure 5.10, and experiment 3 when MilliAmpère is moving with constant velocity is shown in Figure 5.11. The results from the predictions obtained using the two machine learning models, as presented for the tests where MilliAmpère was floating freely, are also shown in the figures. The control input given to the machine learning models is here the first-order low-pass filtered control input calculated by the EKF. The mean values for the estimates from the EKF together with the MAE are presented in Table 5.3. These values are from the estimates created after approximately 20 seconds, indicated by the green lines in the figures. The position and velocity estimates for these four cases are presented in Appendices C.4 to C.7.

Table 5.8: Mean predicted values for current velocity and direction using deep dense neural network (DDNN) models when MilliAmpère is in DP or moving with constant velocity. The presumably best network is indicated in green.

DDNN Model	Experiment 1 DP 0.48 m/s 262.9°	Experiment 2 DP 0.29 m/s 226.0°	Experiment 3 DP 0.43 m/s 238.0°	Experiment 3 constant velocity 0.43 m/s 238.0°
1×100	0.39 m/s 239.9°	0.27 m/s 236.2°	0.38 m/s 264.7°	0.29 m/s 233.6°
3×100	0.56 m/s 249.2°	0.37 m/s 234.6°	0.51 m/s 261.43°	0.43 m/s 230.9°
3×100 with dropout	0.51 m/s 244.2°	0.32 m/s 233.1°	0.46 m/s 263.4°	0.37 m/s 239.4°
3×100 w/weight regularization	0.24 m/s 239.8°	0.18 m/s 238.6°	0.26 m/s 267.1°	0.15 m/s 228.6°
3×500	0.6 m/s 252.3°	0.38 m/s 236.1°	0.52 m/s 260.0°	0.44 m/s 230.2°
3×500 with dropout	0.58 m/s 248.8°	0.36 m/s 235.7°	0.51 m/s 261.9°	0.44 m/s 232.7°
3×500 w/weight regularization	0.22 m/s 230.1°	0.18 m/s 234.4°	0.25 m/s 266.1°	0.12 m/s 230.7°

Table 5.9: Mean predicted values for current velocity and direction using RBFN models when MilliAmpère is in DP or moving with constant velocity. The presumably best network is indicated in green.

RBFN Model	Experiment 1 DP 0.48 m/s 262.9°	Experiment 2 DP 0.29 m/s 226.0°	Experiment 3 DP 0.43 m/s 238.0°	Experiment 3 constant velocity 0.43 m/s 238.0°
50 neurons random c_{init} $\boldsymbol{\beta} = \mathbf{1}_{50\times1}$	0.20 m/s 236.8°	0.17 m/s 253.7°	0.24 m/s 280.7°	0.11 m/s 249.6°
100 neurons random c_{init} $\boldsymbol{\beta} = \mathbf{1}_{100\times1}$	0.33 m/s 220.2°	0.35 m/s 223.0°	0.46 m/s 257.8°	0.40 m/s 230.3°
200 neurons random c_{init} $\boldsymbol{\beta} = \mathbf{1}_{200\times1}$	0.29 m/s 251.6°	0.19 m/s 237.5°	0.30 m/s 262.2°	0.28 m/s 229.1°
50 neurons random c_{init} $\boldsymbol{r} = \mathbf{1}_{50\times6}$	0.52 m/s 245.1°	0.33 m/s 240.8°	0.45 m/s 268.4°	0.37 m/s 239.7°
100 neurons random c_{init} $\boldsymbol{r} = \mathbf{1}_{100\times6}$	0.57 m/s 248.8°	0.38 m/s 234.2°	0.50 m/s 262.6°	0.42 m/s 237.0°
200 neurons random c_{init} $\boldsymbol{r} = \mathbf{1}_{200\times6}$	0.60 m/s 251.5°	0.39 m/s 236.6°	0.52 m/s 262.9°	0.43 m/s 232.6°

Table 5.10: Mean estimated values for the current velocity and direction using the EKF and MAE between the true and the estimated velocities and directions

Experiment	1 DP 0.48 m/s 262.9°	2 DP 0.29 m/s 226.0°	3 DP 0.43 m/s 238.0°	3 constant velocity 0.43 m/s 238.0°
Mean Velocity (m/s)	0.71	0.41	0.58	0.41
Mean direction (degrees)	250.9	242.7	240.8	223.9
MAE Velocity (m/s)	0.23	0.13	0.17	0.08
MAE Direction (degrees)	14.5	20.3	5.1	15.4

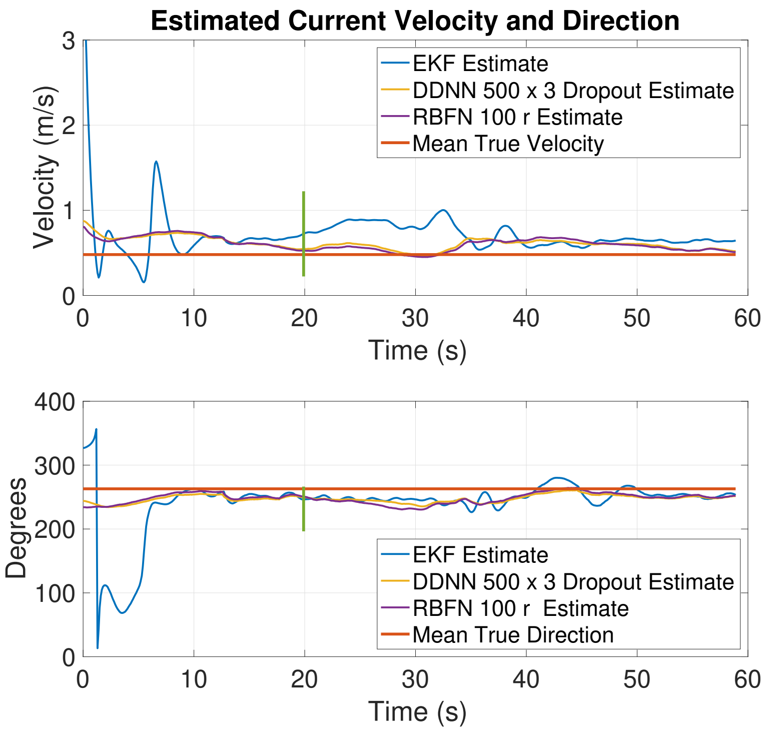

Figure 5.8: Estimated velocity (top) and direction (bottom) of the current during experiment 1 when MilliAmpère is in DP, desired heading is 28.6°. The estimates are obtained using the EKF and machine learning models.

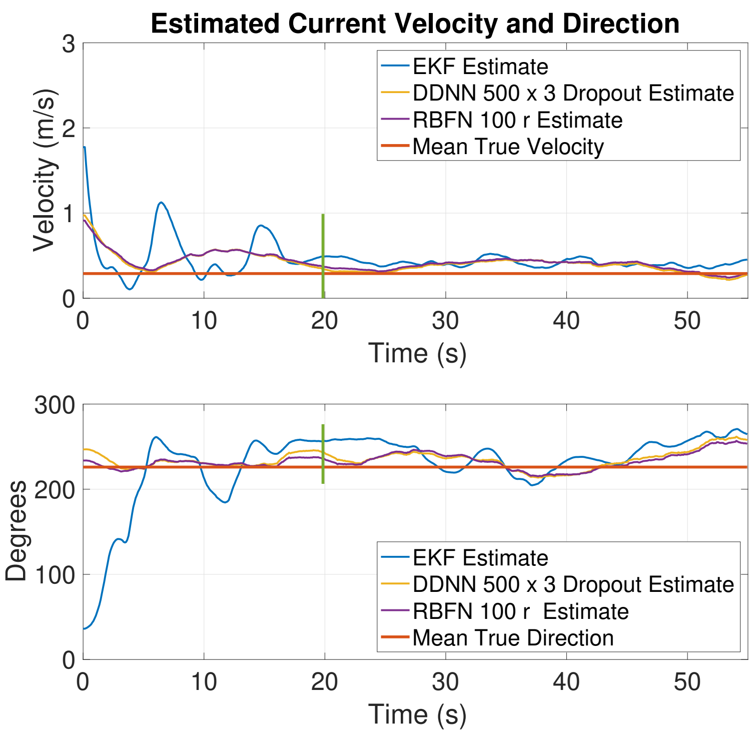

Figure 5.9: Estimated velocity (top) and direction (bottom) of the current during experiment 2 when MilliAmpère is in DP, desired heading is $57.3°$. The estimates are obtained using the EKF and machine learning models.

5.4 Current Estimates from Machine Learning Models and the EKF for a Difficult Case

For some tests, the estimates for the position and velocities of MilliAmpère obtained from the EKF were not satisfactory, and the gains were adjusted. The position and velocity estimates for one of these cases are presented in Figures 5.12 and 5.13, where MilliAmpère has the desired heading of $-0.5\text{rad} = 331.4°$. This case was performed during experiment 3, and the estimates denoted 1 are obtained using the original gains for the $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrices, and the ones denoted 2 has the gains

$$\boldsymbol{Q} = \text{diag}(1.0, 1.0, 0.01, 1.0, 1.0, 0.01, 10, 10, 0.0, 1.0 \cdot 10^4, 1.0 \cdot 10^4, 2.0 \cdot 10^4)$$
$$\boldsymbol{R} = \text{diag}(3.0 \cdot 10^{-2}, 3.0 \cdot 10^{-2}, 3.0 \cdot 10^{-4}, 4.0 \cdot 10^{-3}, 4.0 \cdot 10^{-2}, 4.0 \cdot 10^{-4}).$$

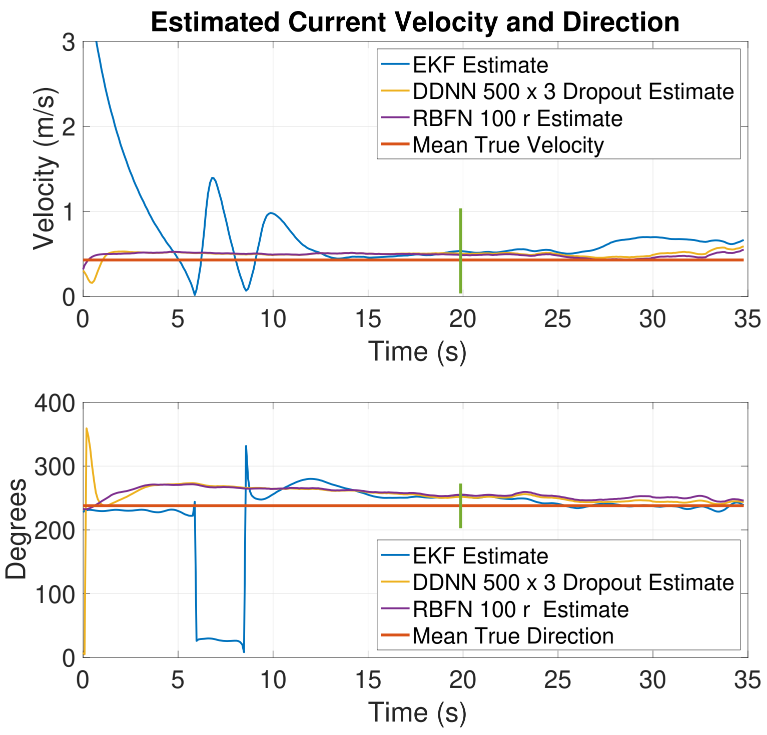

Figure 5.10: Estimated velocity (top) and direction (bottom) of the current during experiment 3 when MilliAmpère is in DP, desired heading is $90.0°$. The estimates are obtained using the EKF and machine learning models.

In Figure 5.14 the current estimates from the EKF for the two types of gains are shown, together with the estimates from the two machine learning models used for the other tests. The input to these machine learning models is also here the filtered control forces obtained from the EKF, and will therefore vary depending on the gains used by the EKF. The mean predictions for the current velocity and direction for this test data made by the several machine learning models, using the original control forces as input, are shown in Table 5.11.

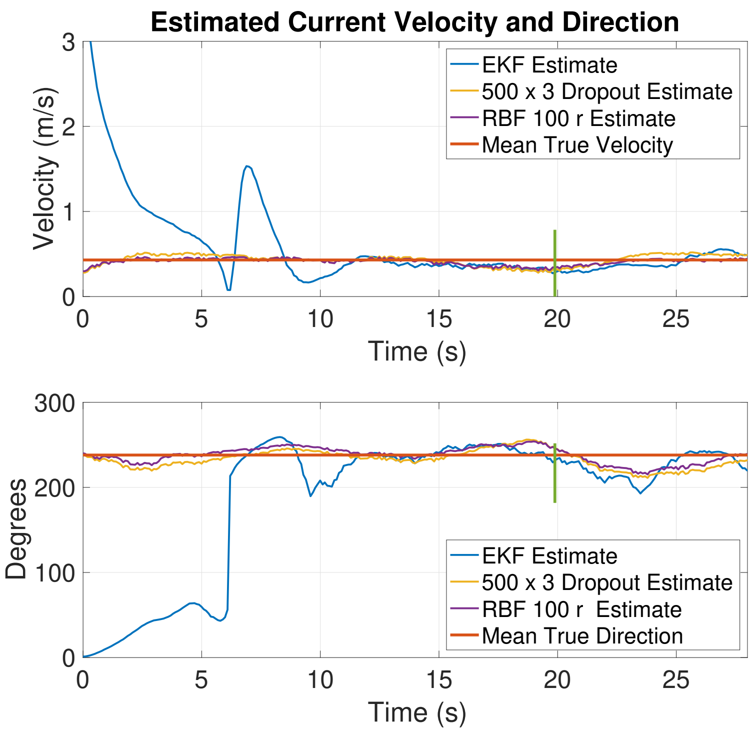

Figure 5.11: Estimated velocity (top) and direction (bottom) of the current during experiment 3 when MilliAmpère is moving with constant velocity, desired heading is $90.0°$. The estimates are obtained using the EKF and machine learning models.

5.5 Discussion

One issue with these experiments is that the exact true value of the current is unknown. The test cases used to estimate the true values for the current velocities and directions at each experiment may have been influenced by other environmental effects, such as the wind. The experiments were performed in a river but were also affected by tides. When experiments 1 and 2 were performed the tides were going from low to high tide, which means that tide might have been pushing the water upstreams, creating a more complex current situation. However, the estimates obtained from measuring the velocity when MilliAmpère was floating freely, is assumed to be the true value.

For the deep neural networks, it is clear that the models with weight regularization predict a significantly smaller value for the current velocity than its true value. The

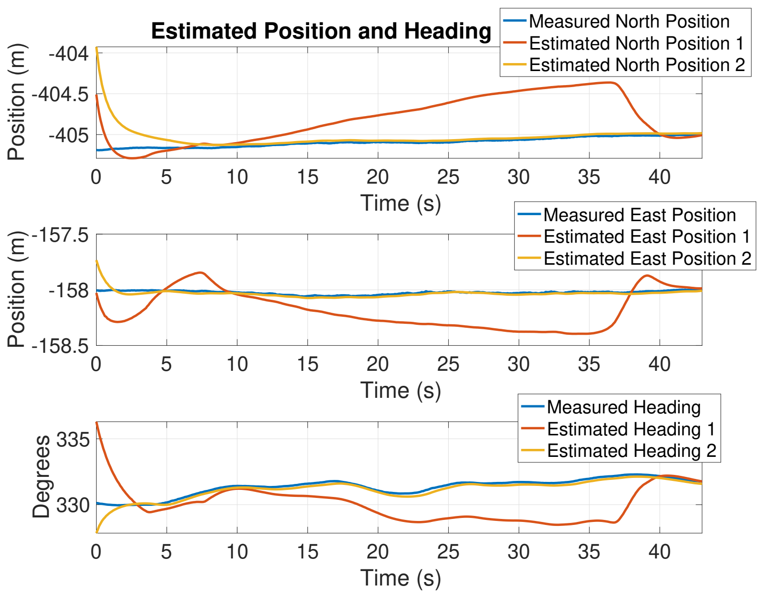

Figure 5.12: Estimated position and heading from the EKF during experiment 3 when MilliAmpère is in DP, desired heading is $331.4°$

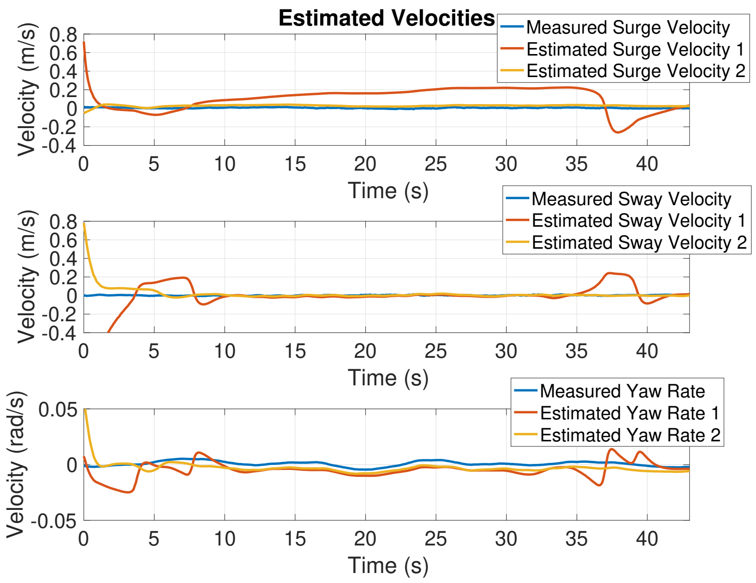

Figure 5.13: Estimated velocities from the EKF during experiment 3 when MilliAmpère is in DP, desired heading is $331.4°$

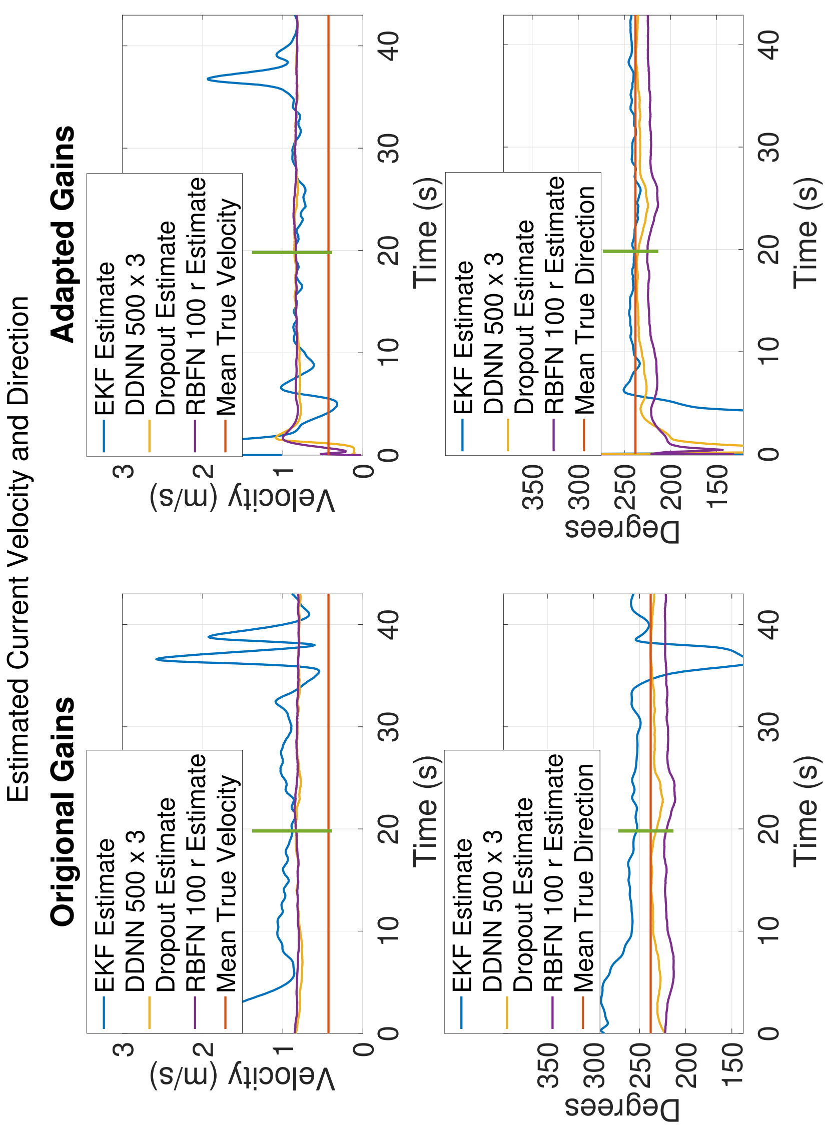

Figure 5.14: Estimated velocity and direction of the current during experiment 3 when MilliAmpère is in DP, desired heading is 331.4°. The estimates are obtained using the EKF and machine learning models.

Table 5.11: Mean predicted values for current velocity and direction for experiment 3 with desired heading of $331.4°$ using machine learning models

DDNN Models	3×100	3×100 with dropout	3×500	3×500 with dropout
Velocity Direction	0.83 m/s 219.5°	0.77 m/s 227.6°	0.76 m/s 223.7°	0.80 m/s 232.8°
RBFN Models	100 neurons random c_{init} $\boldsymbol{\beta} = \mathbf{1}_{100\times1}$	50 neurons random c_{init} $\boldsymbol{r} = \mathbf{1}_{50\times6}$	100 neurons random c_{init} $\boldsymbol{r} = \mathbf{1}_{1000\times6}$	200 neurons random c_{init} $\boldsymbol{r} = \mathbf{1}_{200\times6}$
Velocity Direction	0.58 m/s 209.0°	0.89 m/s 226.0°	0.81 m/s 218.6°	0.82 m/s 219.1°

model with only one layer with 100 neurons is also predicting a slightly smaller current velocity. The trend for the models with and without dropout is that the models with dropout predict a slightly smaller current velocity than the models without dropout, though the gap between the two models is larger for the models with three layers of 100 neurons than for the ones with three layers of 500 neurons. The RBFNs model with constant widths across the features is also predicting a smaller current velocity than the assumed true velocity. The models with varying widths across the features and only 50 neurons results in the best predictions when MilliAmpère is in DP, but this model predicts too small values for the current velocity when MilliAmpère is moving. The two models with 100 and 200 neurons, however, predicts a larger current velocity for the DP cases, and a more accurate current velocity for the moving case. The predicted direction of the current is approximately within 15 degrees off for all the models, except for experiment 3 when MilliAmpère is in DP, which resulted in a significantly larger offset for both the deep neural network models and the RBFN models. From these results, it would seem that the best neural networks are either the deep densely connected neural network with three layers of 500 neurons and dropout between the layers or the RBFN with 100 neurons and varying widths across the features, but there are no large differences between many of the models. Comparing these two, the RBFN consists of fewer parameters, which makes it faster to train, but all of these networks are relatively small networks.

The EKF performs quite well when MilliAmpère is floating freely, although there is some oscillation in the estimates, especially in experiment 1. From Figure 5.2, we see that the velocity of MilliAmpère is not constant, and this change in velocity may have contributed to the oscillating estimates. When MilliAmpère is in DP, it can be seen that the estimates from the EKF results in a higher current velocity estimate for all the three experiments, while the estimate is more accurate when MilliAmpère is moving with constant velocity. A problem with the EKF is the adjustable gains in the $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrices. The gains were adjusted a lot by trial and error, and might be far from the optimal gains.

All the methods, both the EKF and the various neural networks, which resulted in good estimates on both the current velocity and direction when MilliAmpère was moving with constant velocity, resulted in a too high estimate when MilliAmpère is in DP. This may suggest that thruster forces are not sufficiently precise compared to the commanded thruster forces from the DP system, and that this error is less significant when moving with constant speed, and the thruster forces are larger. In the EKF, this was first improved by implementing $\boldsymbol{\tau}$ as a state, given as the first-order low-pass filter between the commanded and the actual thrust, which also helped to make the signal smoother. The gains for the control forces in the covariance matrix was also set quite large, and which in terms means that the EKF has less trust in this state.

However, as shown in Figures 5.12 and 5.13, this was not always sufficient, and we see that the position and velocity estimates drift away from the measured states. The new adjusted gains have even larger values for the control forces, as well as smaller gains in the $\boldsymbol{R}$ matrix for the heading, surge, and yaw rate measurements, as a way to push the filter towards the states where it had an offset with the original gains. This resulted in better estimates for the measured states, as well as the direction of the estimated current, as seen in Figure 5.14. From this figure, we also see that the estimate for the current velocity stabilizes with a significant offset from the true value, and this is also true for the estimates from the machine learning models.

The reason for the poor estimates given by both the EKF and the machine learning models may be that the commanded thrust was not the thrust given by the propellers. The thruster dynamics of MilliAmpère have been specified by a performing a bollard pull test. This means that there was no velocity in the water surrounding the propeller when this test was performed, and the propellers may result in a different thrust when exposed to surrounding current velocities. It is also worth mentioning that there might have been particularly strong wind or other environmental effects not accounted for at the time when this test was performed, which may also have affected how the DP system calculated the desired thruster forces. The estimates for the current direction are not far off for both the adjusted EKF and the machine learning methods. This indicates that the direction of the control forces, calculated by the DP system, is able to adjust for the current forces quite well. Having an offset in the current velocity estimates, but quite accurate estimates of the direction would still be a valuable input for a path planning and collision avoidance algorithm, as it provides information about which direction the current is pushing the vessel towards.

Even though the propellers in the simulator were not physically exposed to a current, the results shown in Section 4.1 express how the accuracy of the current estimates degrades the larger the offset is between the actual control forces affecting the vessel and the control forces the system believes are affecting the vessel. This was also true if the Coriolis, centripetal, and damping matrices were not known precisely. When performing the full-scale tests it is therefore difficult to know how close the simulated models, used when developing both the EKF and the machine learning methods, are to the true dynamics of the physical system. However, since both the methods predict a large offset in the current velocity, for some cases, there is a strong indication that the model of MilliAmpère may not be accurate enough. The values for the simulator were, as mentioned, obtained through curve fitting of experimental data, which means that the current velocities present when these experiments were performed, may in fact have impacted the model parameters.

These difficulties are well known and is why instead of predicting the current velocity, a bias term is often predicted in traditional DP systems. This bias term captures all the unmodeled dynamics, which in some ways are ideal since then these unmodeled dynamics will not be assigned to a state one assumes represents only the true current. However, in terms of improving the situation awareness of an autonomous vessel, and thereby making more information available to perform improved path planning and collision avoidance, having some unmodeled dynamics in the current estimates may not be problematic. As seen in the difficult case, these methods seem to estimate the direction of the current quite accurately, and the estimate for the velocity is greater than the true value. In terms of path planning, this would still give an indication as to where the vessel will float in the case of for example engine failure, and measures can be taken to calculate the safest path possible.

The methods presented here are demonstrated and tested on MilliAmpère but is applicable to other vessels as well. The main problem when it comes to implementing the methods on other types of vessels may be that velocity measurements are not common, although they are becoming more and more common through the increased accuracy in the position measurements. An effort of implementing the EKF with only position and heading measurements was done, but it proved difficult to find gains resulting in a good current estimate, and the current velocity estimates would sometimes blow up. Nonetheless, there may exist gains resulting in good estimates. The machine learning methods are entirely based on the body-fixed velocity measurements, and will not be possible without them.

Comparing the machine learning and the EKF, machine learning has a great advantage in terms that the performance is not dependent on tuneable parameters. There is no need for the machine learning algorithms to stabilize either, and the offset seen at the beginning of some of the plots is due to the offset in the control forces from the EKF. These offsets are present at the beginning of the EKF estimates before the filter has stabilized. One advantage of the EKF is, however, that it works with non-stationary data, while the machine learning models are developed for an assumed stationary case.

Chapter 6

Conclusions and Recommendations for Further Work

The research questions of this thesis were:

- *How can an estimate of the current velocity have an impact on the path planning and collision avoidance of a highly autonomous vessel?*
- *Is it possible to obtain decent current estimates onboard a real ship knowing its mathematical model?*
- *How does noise in the mathematical ship model affect the accuracy of the current estimates?*

This chapter concludes on the work done in this thesis and presents some suggestions as to further work, that can provide useful insight on the topic.

6.1 Conclusions

The focus of this thesis has been to investigate how an estimate of the current can have an impact on the path planning algorithm of a highly autonomous marine vessel, and how one can obtain current estimates. A framework for taking the current estimates into account has been proposed, which presented the significance of the current in terms of the path planning and collision avoidance algorithm. The architecture also explained how other elements, such as the propulsion energy of the vessel and the obstacle detection, would have different impacts on the path planning and collision avoidance algorithm when considered together with the current.

Two methods for performing current estimation were presented, one based on the extended Kalman filter and one using machine learning models. Both methods were capable of predicting the current in a satisfying manner, which was presented in the form of simulation results and full-scale experimental results. The extended Kalman filter proved difficult to tune, and it was presented how the gains in the $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrices could result in good estimates for one test-file from the experiments, and in quite poor estimates for another.

The machine learning models proved simpler to both implement and apply to different cases, due to their lack of tuning, but had the disadvantage of only being applicable for stationary conditions. Different types of machine learning models, with different sizes, were presented and tested. Two models showed the best current estimation capabilities. One was a deep densely connected neural network, consisting of three hidden layers of 500 neurons and dropout layers with a dropout rate of 0.2 between these layers. The other one was a radial basis function network with 100 neurons, where the widths belong to each of the centers of the radial basis functions varied across the features. The advantage of the radial basis function network was that it contained fewer parameters, making it faster to train, but both the proposed networks can be considered quite small.

The accuracy of the current estimates when the model of the vessel contained noise was investigated through simulation. It showed that noise in the Coriolis, centripetal, and damping matrices and the control forces significantly affected the estimates. Through experimental tests, where it was unknown how different the mathematical model was from the real ship, it was shown that it was possible to obtain good estimates of the current using both the extended Kalman filter and the machine learning models. It was also shown that the models sometimes resulted in quite significant offsets, especially in the estimates of the current velocity. The main assumption for this offset was that the desired control forces given by the dynamic positioning system were not the forces generated by the propellers.

To conclude, the proposed methods are applicable to provide current estimation onboard a real ship. These estimates will further enable an improvement in the path planning and collision avoidance algorithm, through the presented framework.

6.2 Recommendations for Further Work

In Section 4.1 it was shown how the estimate of the current was influenced if the EKF did not have access to the correct control forces $\boldsymbol{\tau}$. The same was found looking at the results from the machine learning methods presented in Section 4.2. Poor knowledge of $\boldsymbol{\tau}$ is also assumed to be a cause for the offset in the current estimates of the experimental data. Better knowledge regarding the thrust created by the propellers, and especially the efficiency of the propellers in the presence of currents could, therefore, significantly improve the current estimates. This can be further improved if the thruster dynamics in the presence of currents are known.

It would be interesting to investigate further if it is possible to obtain good estimates for the current using the EKF with only position and heading measurements. This would make the method more generalizable in terms of what type of vessel the method will be suitable for.

The machine learning methods performed quite well on the experimental data, but there is a good chance that the simulation model used to generate the data might not sufficiently express the dynamics of the physical model. One great advantage of machine learning models is that new data can be appended to an already trained model. By performing experiments in a more controlled environment, such as an ocean basin, where the current velocity is known, new training data can be obtained. This data can further be appended to the neural networks making the current estimates onboard the real vessel even better.

The models implemented using machine learning in this thesis is restricted to stationary cases and does not have an aspect of time. To expand the machine learning methods to work for non-stationary cases, other types of networks, such as recurrent neural networks (RNNs) might be suitable. These networks make it possible to have previous outputs as inputs, and the aspect of time can, therefore, be included in the model.

More environmental forces should also be taken into account in order to create a more complex model. This includes the wind and the wave forces. When second-order wave forces are present, drift forces will further complicate the situation around the current velocity. The wind forces will also have a greater impact on vessels with a larger superstructure and should be taken into account to make the method more suitable for numerous types of vessels.

Bibliography

Borup, K. T. (2018). *Air data estimation for small unmanned aircraft*. Vol. 2018:297. Doktoravhandlinger ved NTNU (trykt utg.) Trondheim: Norwegian University of Science, Technology, Faculty of Information Technology, and Electrical Engineering, Department of Engeneering Cybernetics. ISBN: 978-82-326-3384-5.

Chollet, F. (2018). *Deep learning with Python*. Shelter Island, New York: Manning Publications Co. ISBN: 978-1-61729-443-3.

Dabbura, I. (Apr. 29, 2020). *K-means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks*. Medium. Library Catalog: towardsdatascience.com. URL: `https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a` (visited on 05/04/2020).

DNV GL (Jan. 31, 2018). *Autonome skip: Hvorfor skal vi fjerne mannskapet?* Tu.no. URL: `https://www.tu.no/storylabs/autonomi/annonse-forsker-pa-fjernstyrte-skip/414265` (visited on 10/27/2019).

Du, K. L. and Swamy, M. N. S. (2006). “Radial basis function networks”. In: *Neural networks in a softcomputing framework*. Publisher: Springer, pp. 251–294.

Endsley, M. R. (May 1988). “Situation awareness global assessment technique (SAGAT)”. In: *Proceedings of the IEEE 1988 National Aerospace and Electronics Conference*, 789–795 vol.3. DOI: `10.1109/NAECON.1988.195097`.

Endsley, M. R. and Garland, D. J. (2000). *Situation Awareness Analysis and Measurement*. CRC Press. ISBN: 978-0-8058-2133-8. URL: `http://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=44637&site=ehost-live`.

Faris, H., Aljarah, I., and Mirjalili, S. (2017). “Chapter 28 - Evolving Radial Basis Function Networks Using Moth–Flame Optimizer”. In: *Handbook of Neural Computation*. Ed. by P. Samui, S. Sekhar, and V. E. Balas. Academic Press, pp. 537–550. ISBN: 978-0-12-811318-9. DOI: `10.1016/B978-0-12-`

811318-9.00028-4. URL: http://www.sciencedirect.com/science/article/pii/B9780128113189000284.

Fossen, T. I. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control*. Chichester, UK: John Wiley & Sons, Ltd. ISBN: 978-1-119-99149-6.

Goff, D., Thomas, S., Jones, R., and Massey, C. (2000). “A neural network approach to predicting airspeed in helicopters”. In: *Neural Computing & Applications* 9.2. Publisher: Springer, pp. 73–82.

Hassani, V., Sørensen, A. J., Pascoal, A. M., and Aguiar, A. P. (2012). “Multiple model adaptive wave filtering for dynamic positioning of marine vessels”. In: *2012 American Control Conference (ACC)*, pp. 6222–6228.

Hegrenæs, O., Hallingstad, O., and Jalving, B. (2007). “A framework for obtaining steady-state maneuvering characteristics of underwater vehicles using sea-trial data”. In: *2007 Mediterranean Conference on Control Automation*, pp. 1–6.

Hoem, Å., Rødseth, Ø., and Fjørtoft, K. (2019). “Addressing the Accidental Risks of Maritime Transportation: Could Autonomous Shipping Technology Improve the Statistics?” In: *TransNav, the International Journal on Marine Navigation and Safety of Sea Transportation* 13. DOI: 10.12716/1001.13.03.01.

IMO (2019). *Autonomous shipping*. URL: http://www.imo.org/en/MediaCentre/HotTopics/Pages/Autonomous-shipping.aspx (visited on 11/19/2019).

Jokioinen, E., Poikonen, J., Jalonen, R., and Saarni, J. (2016). “Remote and autonomous ships-the next steps”. In: *AAWA Position Paper, Rolls Royce plc, London*.

Keras (2020). *Keras: the Python deep learning API*. URL: https://keras.io/ (visited on 05/14/2020).

Kim, E., Fan, S., and Bose, N. (2018). “Estimating Water Current Velocities by Using a Model-Based High-Gain Observer for an Autonomous Underwater Vehicle”. In: *IEEE Access* 6, pp. 70259–70271.

KONGSBERG (2019). *Autonomous ship project, key facts about YARA Birkeland - Kongsberg Maritime*. URL: https://www.kongsberg.com/maritime/support/themes/autonomous-ship-project-key-facts-about-yara-birkeland/ (visited on 11/13/2019).

Lee, K.-B. and Blaabjerg, F. (2007). “Robust and stable disturbance observer of servo system for low-speed operation”. In: *IEEE Transactions on Industry Applications* 43.3. Publisher: IEEE, pp. 627–635.

Lekkas, A. M. and Fossen, T. I. (2014). “Trajectory tracking and ocean current estimation for marine underactuated vehicles”. In: *2014 IEEE Conference on Control Applications (CCA)*, pp. 905–910.

Li, J., Li, S., and Li, S. (2014). “The design of robust MIMO neural network disturbance observer for multi-variable system”. In: *Proceedings of the 33rd Chinese Control Conference*. IEEE, pp. 2379–2383.

Lorentzen, M. (2018). *Kongsberg Maritime skal helautomatisere Bastøfergen*. Library Catalog: e24.no. URL: `https://e24.no/i/3jadBP` (visited on 04/09/2020).

Martinez, A., Hernandez, L., Sahli, H., Valeriano-Medina, Y., Orozco-Monteagudo, M., and Garcia-Garcia, D. (2015). “Model-Aided Navigation with Sea Current Estimation for an Autonomous Underwater Vehicle”. In: *International Journal of Advanced Robotic Systems* 12.7. _eprint: https://doi.org/10.5772/60415, p. 103. DOI: `10.5772/60415`. URL: `https://doi.org/10.5772/60415`.

Mastro, M. (2013). *Financial derivative and energy market valuation*. Wiley Online Library.

Paliotta, C. and Pettersen, K. Y. (2016). “Geometric path following with ocean current estimation for ASVs and AUVs”. In: *2016 American Control Conference (ACC)*, pp. 7261–7268.

Patrón, P., Miguelanez, E., Cartwright, J., and Petillot, Y. R. (2008). “Semantic knowledge-based representation for improving situation awareness in service oriented agents of autonomous underwater vehicles”. In: *OCEANS 2008*. IEEE, pp. 1–9.

Pedersen, A. A. (2019). “Optimization Based System Identification for the milliampere Ferry”. Master’s Thesis. Trondheim: Norwegian University of Science and Technology (NTNU). URL: `https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2625699`.

Perera, L. P. (2018). “Autonomous Ship Navigation under Deep Learning and the challenges in COLREGs”. In: *ASME 2018 37th International Conference on Ocean, Offshore and Arctic Engineering*. American Society of Mechanical Engineers Digital Collection.

Refsnes, J. E., Sørensen, A. J., and Pettersen, K. Y. (2007). “Output feedback control of slender body underwater vehicles with current estimation”. In: *International Journal of Control* 80.7. Publisher: Taylor & Francis, pp. 1136–1150.

Rødseth, Ø. J. (2019). “Defining Ship Autonomy by Characteristic Factors”. In: *Proceedings of the 1st International Conference on Maritime Autonomous Surface Ships*. SINTEF Academic Press.

Rosbach, M. (2018). *Fjord1 med stor bestilling på nyvinning fra Rolls-Royce*. smp.no. Library Catalog: www.smp.no. URL: `https://www.smp.no/naeringsliv/2018/04/17/Fjord1-med-stor-bestilling-p%C3%A5-nyvinning-fra-Rolls-Royce-16499548.ece` (visited on 04/09/2020).

Ruder, S. (Jan. 19, 2016). *An overview of gradient descent optimization algorithms*. Sebastian Ruder. Library Catalog: ruder.io. URL: `https://ruder.io/optimizing-gradient-descent/` (visited on 05/04/2020).

Sharma, A., Nazir, S., and Ernstsen, J. (2019). "Situation awareness information requirements for maritime navigation: A goal directed task analysis". In: *Safety Science* 120, pp. 745–752. ISSN: 09257535.

SNAME (1950). "Nomenclature for treating the motion of a submerged body through a fluid". In: *The Society of Naval Architects and Marine Engineers, Technical and Research Bulletin No*, pp. 1–5.

Sørensen, A. J. (2013). *Marine control systems : propulsion and motion control of ships and ocean structures*. Vol. UK-2013-76. Kompendium (Norges teknisk-naturvitenskapelige universitet. Institutt for marin teknikk). Trondheim: Department of Marine Technology. Norwegian University of Science and Technology.

— (2018). *Marine Cybernetics: Towards Autonomous Marine Operations and Systems*. Vol. UK-18-76. Lecture Nores. Trondheim: Department of Marine Technology. Norwegian University of Science and Technology.

Stensvold, T. (2018). *Nå er MF Folgefonn verdens første fartøy med automatisk dokking*. Tu.no. Library Catalog: www.tu.no. URL: `https://www.tu.no/artikler/na-er-mf-folgefonn-verdens-forste-fartoy-med-automatisk-dokking/435810` (visited on 04/09/2020).

Strand, J. P. (1999). *Nonlinear position control systems design for marine vessels*. Vol. 1999:50. Doktor ingeniøravhandling (Trondheim : trykt utg.) Trondheim: Department of Engineering Cybernetics, Norwegian University of Science and Technology. ISBN: 82-471-0418-0.

TensorFlow (2020). *TensorFlow*. URL: `https://www.tensorflow.org/` (visited on 05/14/2020).

Trana, K. (Dec. 12, 2019). *Håper små og førerløse elferger kan bli nytt industrieventyr*. NRK. Library Catalog: www.nrk.no. URL: `https://www.nrk.no/trondelag/haper-sma-og-forerlose-elferger-kan-bli-nytt-industrieventyr-1.14813382` (visited on 04/09/2020).

Vidnerova, P. (2019). *RBF-Keras: an RBF Layer for Keras Library*. original-date: 2017-10-02T11:38:09Z. URL: `https://github.com/PetraVidnerova/rbf_keras` (visited on 05/14/2020).

Vidnerova, P., Slusny, S., and Neruda, R. (2008). "Evolutionary trained radial basis function networks for robot control". In: *2008 10th International Conference on Control, Automation, Robotics and Vision*, pp. 833–838.

Appendices

Appendix A

Calculation of the Jacobi Matrix

$$\boldsymbol{J} = \frac{\partial \mathbf{f}(\boldsymbol{x}, \boldsymbol{u})}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial \mathbf{f}_1}{\partial \boldsymbol{\eta}} & \frac{\partial \mathbf{f}_1}{\partial \boldsymbol{\nu}} & \frac{\partial \mathbf{f}_1}{\partial \boldsymbol{V}_c} \\ \frac{\partial \mathbf{f}_2}{\partial \boldsymbol{\eta}} & \frac{\partial \mathbf{f}_2}{\partial \boldsymbol{\nu}} & \frac{\partial \mathbf{f}_2}{\partial \boldsymbol{V}_c} \\ \frac{\partial \mathbf{f}_3}{\partial \boldsymbol{\eta}} & \frac{\partial \mathbf{f}_3}{\partial \boldsymbol{\nu}} & \frac{\partial \mathbf{f}_3}{\partial \boldsymbol{V}_c} \end{bmatrix} \tag{A.1}$$

$$\frac{\partial \mathbf{f}_1}{\partial \boldsymbol{\eta}} = \begin{bmatrix} 0 & 0 & \\ 0 & 0 & \frac{\partial \mathbf{R}(\psi)}{\partial \psi}\boldsymbol{\nu} \\ 0 & 0 & \end{bmatrix} \tag{A.2}$$

$$\frac{\partial \mathbf{f}_1}{\partial \boldsymbol{\nu}} = \mathbf{R}(\psi) \tag{A.3}$$

$$\begin{aligned} \frac{\partial \mathbf{f}_2}{\partial \boldsymbol{\eta}} = -\mathbf{M}^{-1} & \begin{bmatrix} 0 & 0 & \\ 0 & 0 & \frac{\partial \mathbf{N}(\boldsymbol{\nu}_r)}{\partial \psi}(\boldsymbol{\nu} - \mathbf{R}^T(\psi)\boldsymbol{V}_c) \\ 0 & 0 & \end{bmatrix} \\ & + \mathbf{M}^{-1}\mathbf{N}(\boldsymbol{\nu}_r) \begin{bmatrix} 0 & 0 & \\ 0 & 0 & \frac{\partial \mathbf{R}^\top(\psi)}{\partial \psi}\boldsymbol{V}_c \\ 0 & 0 & \end{bmatrix} \end{aligned} \tag{A.4}$$

$$\frac{\partial \mathbf{f}_2}{\partial \boldsymbol{\nu}} = -\mathbf{M}^{-1} \left(\mathbf{N}(\boldsymbol{\nu}_r) + \begin{bmatrix} \vdots & \vdots & \vdots \\ \frac{\partial \mathbf{N}(\boldsymbol{\nu}_r)}{\partial u}\boldsymbol{\nu}_r & \frac{\partial \mathbf{N}(\boldsymbol{\nu}_r)}{\partial v}\boldsymbol{\nu}_r & \frac{\partial \mathbf{N}(\boldsymbol{\nu}_r)}{\partial r}\boldsymbol{\nu}_r \\ \vdots & \vdots & \vdots \end{bmatrix} \right) \tag{A.5}$$

$$\frac{\partial \mathbf{f}_2}{\partial \boldsymbol{V}_c} = -\mathbf{M}^{-1}\begin{bmatrix} \frac{\partial \mathbf{N}(\boldsymbol{\nu}_r)}{\partial V_{c,x}}\boldsymbol{\nu}_r & \frac{\partial \mathbf{N}(\boldsymbol{\nu}_r)}{\partial V_{c,y}}\boldsymbol{\nu}_r & \begin{matrix}0\\0\\0\end{matrix} \end{bmatrix} + \mathbf{M}^{-1}\mathbf{N}(\boldsymbol{\nu}_r)\mathbf{R}^T(\psi)\begin{bmatrix} \frac{\partial \boldsymbol{V}_c}{\partial V_{c,x}} & 0 & 0 \\ 0 & \frac{\partial \boldsymbol{V}_c}{\partial V_{c,y}} & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{A.6}$$

$$\frac{\partial \mathbf{f}_1}{\partial \boldsymbol{V}_c} = \frac{\partial \mathbf{f}_3}{\partial \boldsymbol{\eta}} = \frac{\partial \mathbf{f}_3}{\partial \boldsymbol{\nu}} = \frac{\partial \mathbf{f}_3}{\partial \boldsymbol{V}_c} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{A.7}$$

Appendix B

Source Code

The source code attached to this thesis is divided in two parts, the source code for the EKF and the source code for the machine learning algorithms.

B.1 EKF Source Code

- `env_estimation_sim.py` is the source code given for the extended Kalman filter used together with the simulator.
- `env_estimation.py` is the updated source code for the extended Kalman filter used for the physical model, this includes an extra state for the control forces $\boldsymbol{\tau}$.
- `models.py` is a help function used for both the Kalman-filter functions. This includes the parameters of MilliAmpère and the Coriolis, centripetal, and damping matrices which is a part of the original simulator, along with some of the matrices differentiated versions used to calculate the Jacobi matrix.

B.2 Machine Learning Source Code

- `DeepLearning.ipynb` contains the notebook for generating deep neural networks.

- `DeepLearningNoiseVerification.ipynb` contains the notebook showing an example of how a saved model is evaluated on test data with noise.
- `RBFN_constant_widths.ipynb` contains the notebook for developing RBFNs where the widths of the centers is constant across the features.
- `RBFN_feature_wise_widths.ipynb` contains the notebook for developing RBFNs where the widths of the centers is changing across the features.

Appendix C

Estimates Using The EKF During Full-Scale Experiments

C.1 Experiment 1 - MilliAmpère is Floating Freely

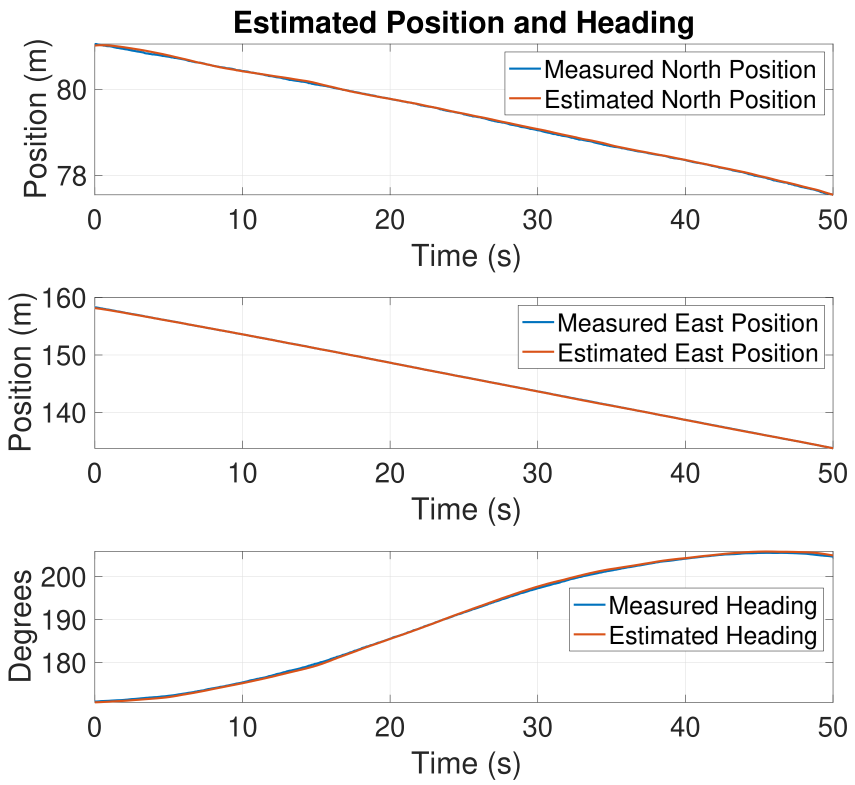

Figure C.1: Estimated position and heading of MilliAmpère during experiment 1, MilliAmpère is floating freely

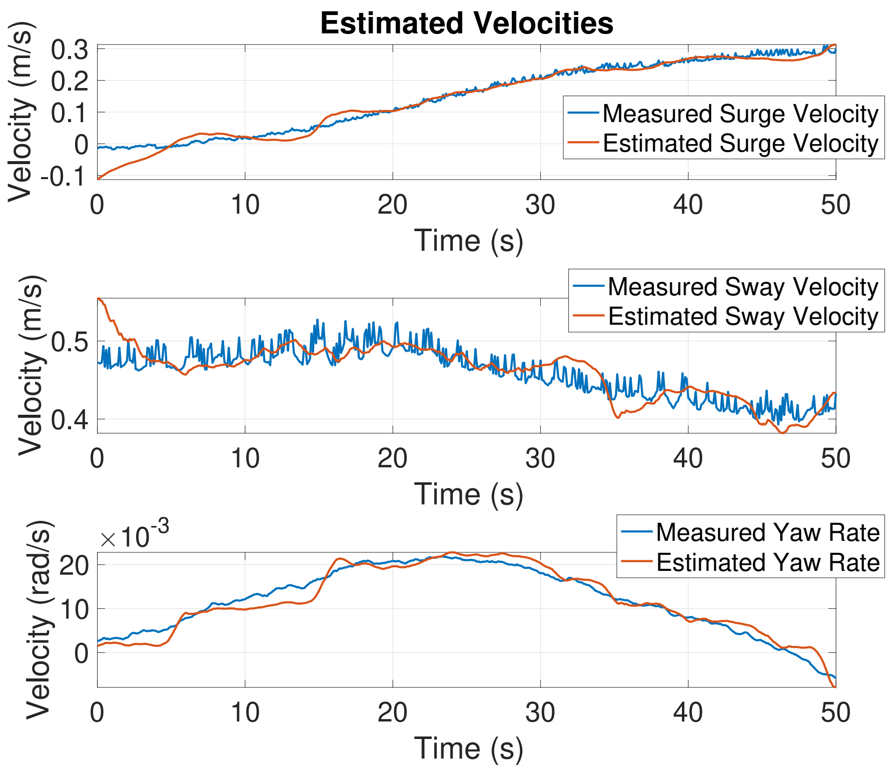

Figure C.2: Estimated velocities of MilliAmpère during experiment 1, MilliAmpère is floating freely

C.2 Experiment 2 - MilliAmpère is Floating Freely

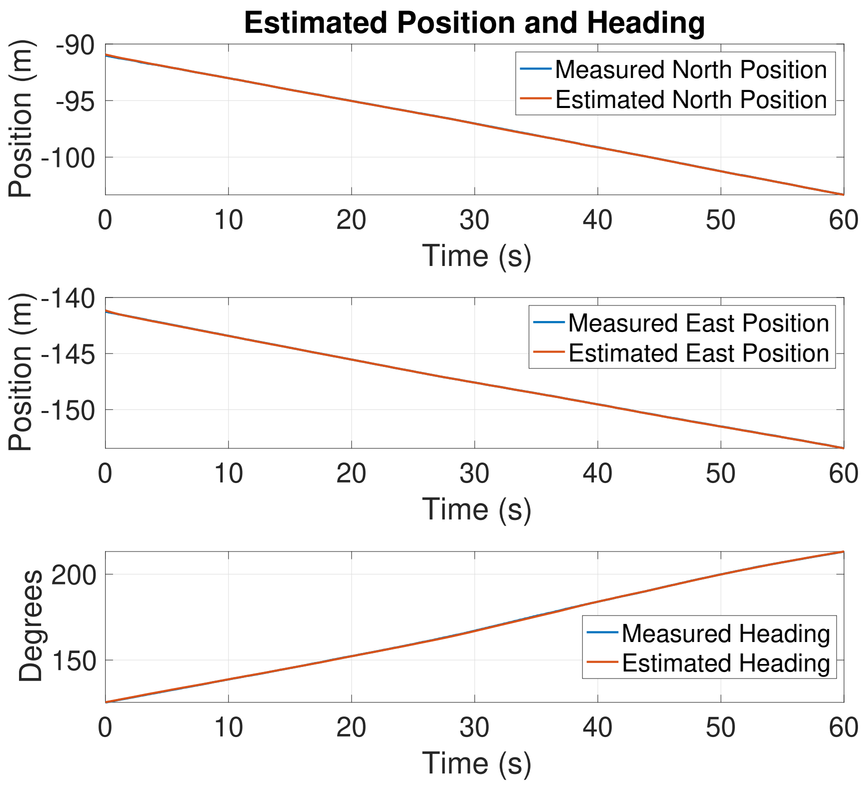

Figure C.3: Estimated position and heading of MilliAmpère during experiment 2, MilliAmpère is floating freely

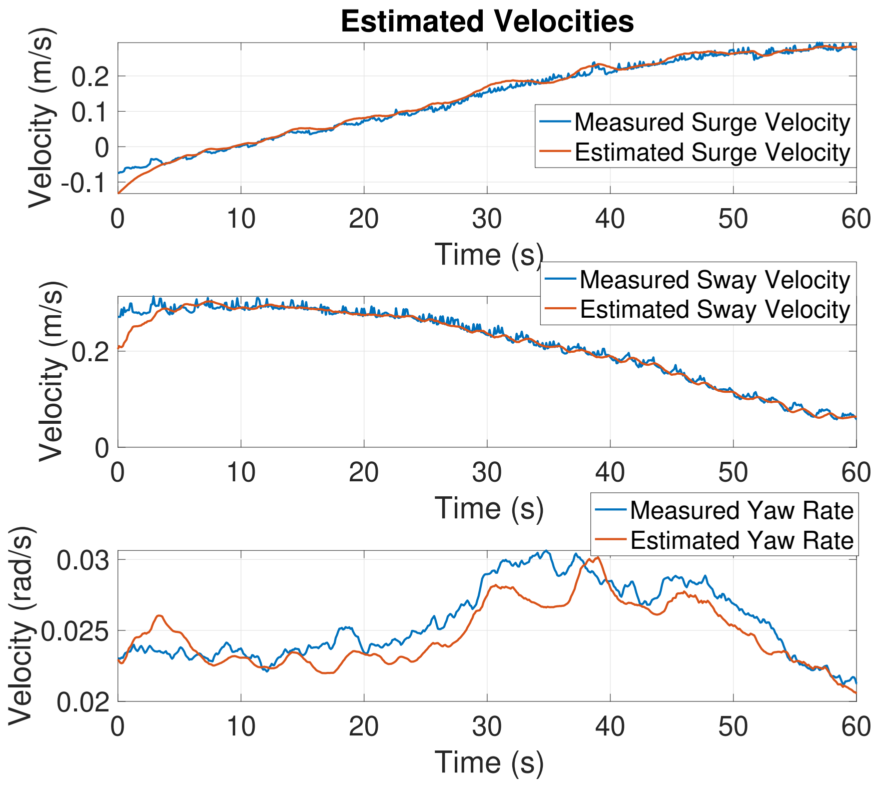

Figure C.4: Estimated velocities of MilliAmpère during experiment 2, MilliAmpère is floating freely

C.3 Experiment 3 - MilliAmpère is Floating Freely

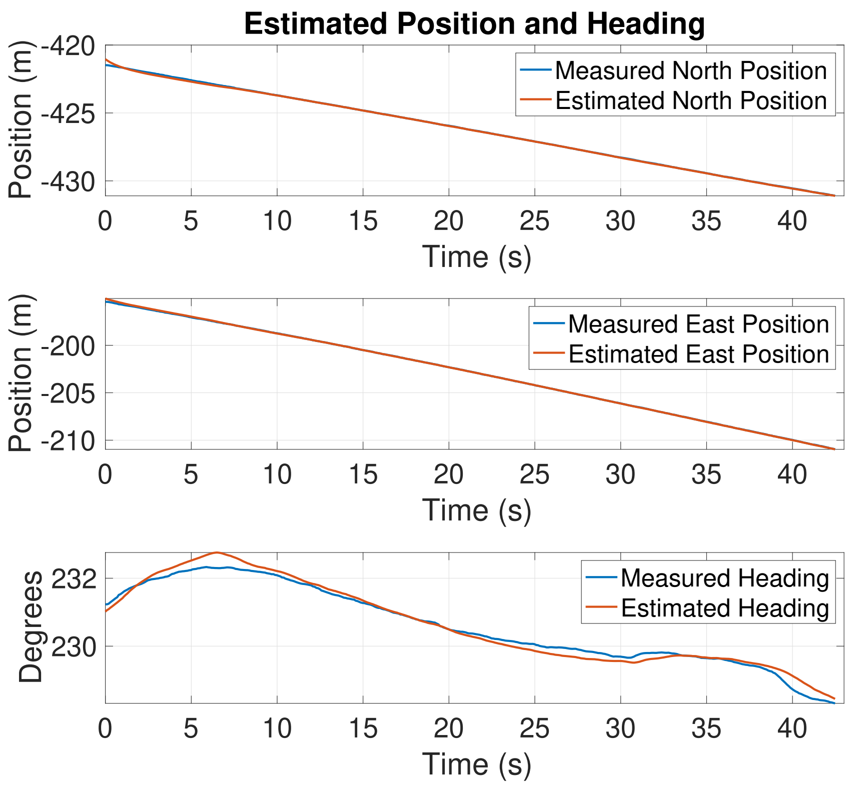

Figure C.5: Estimated position and heading of MilliAmpère during experiment 3, MilliAmpère is floating freely

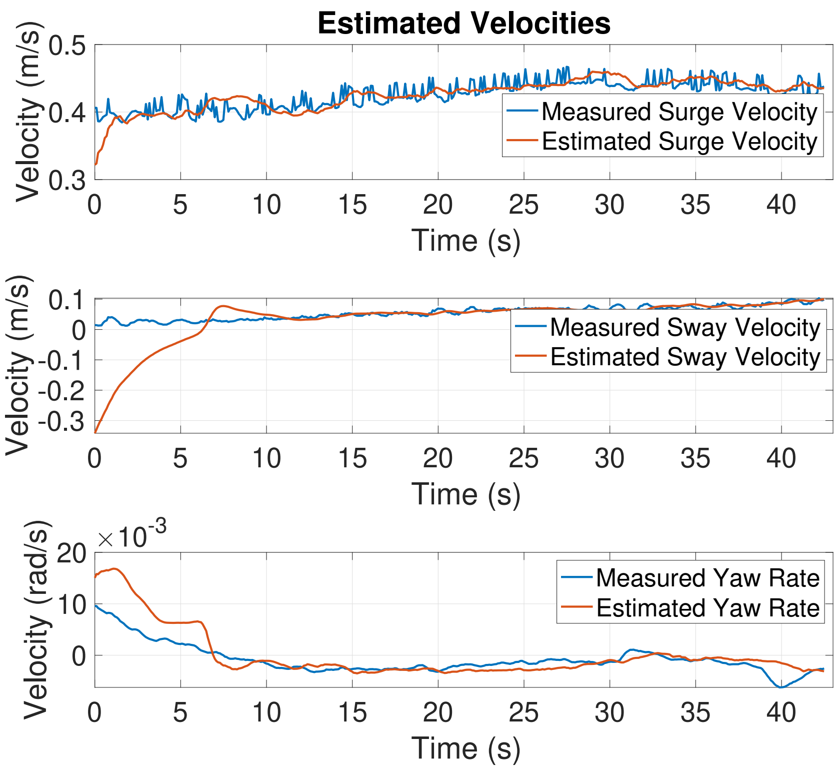

Figure C.6: Estimated velocities of MilliAmpère during experiment 3, MilliAmpère is floating freely

C.4 Experiment 1 - MilliAmpère is in DP

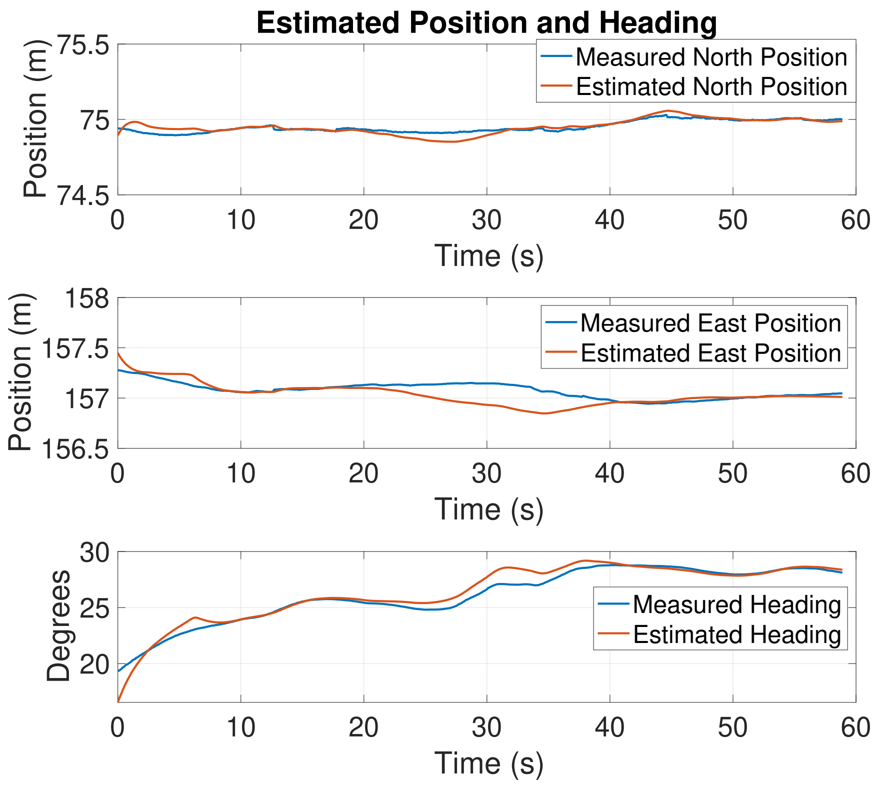

Figure C.7: Estimated position and heading of MilliAmpère during experiment 1, MilliAmpère is in DP with desired heading = 28.6°

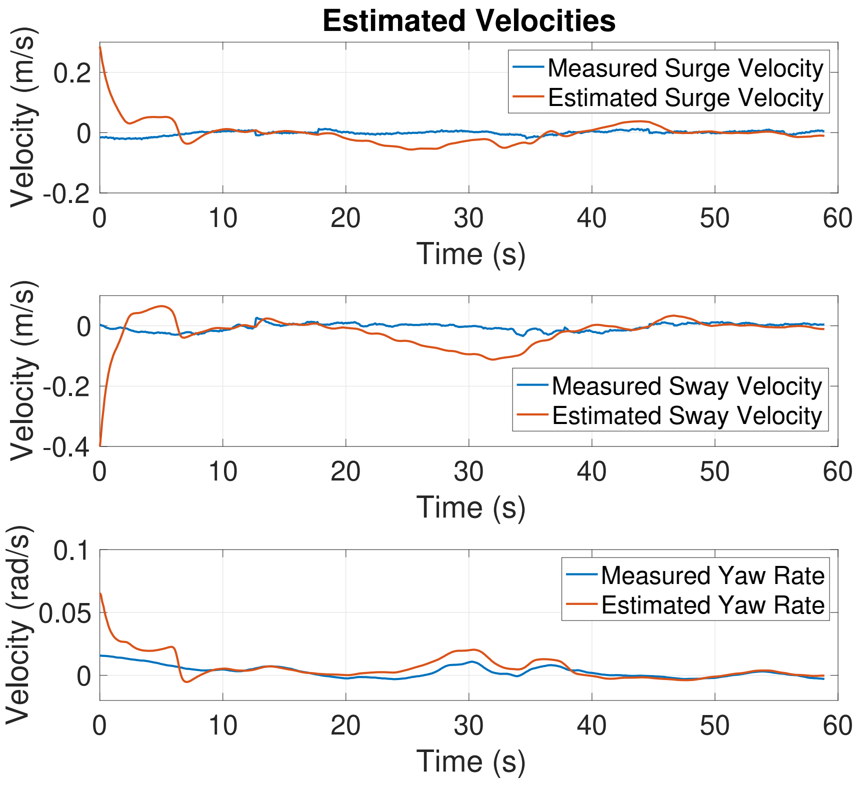

Figure C.8: Estimated velocities of MilliAmpère during experiment 1, MilliAmpère is in DP with desired heading = $28.6°$

C.5 Experiment 2 - MilliAmpère is in DP

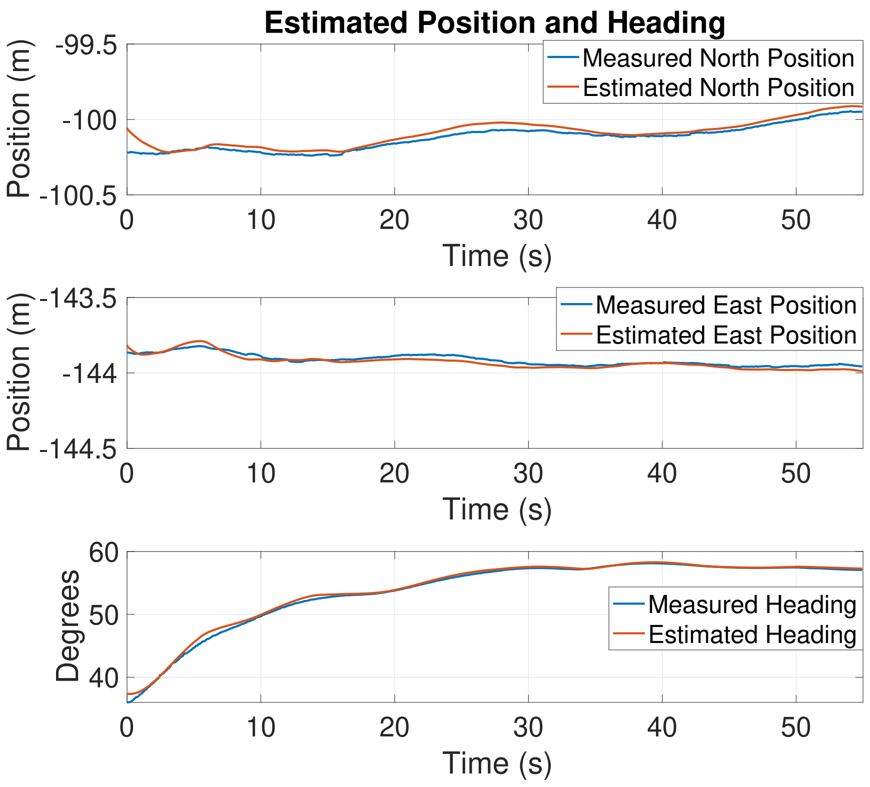

Figure C.9: Estimated position and heading of MilliAmpère during experiment 2, MilliAmpère is in DP with desired heading = 57.3°

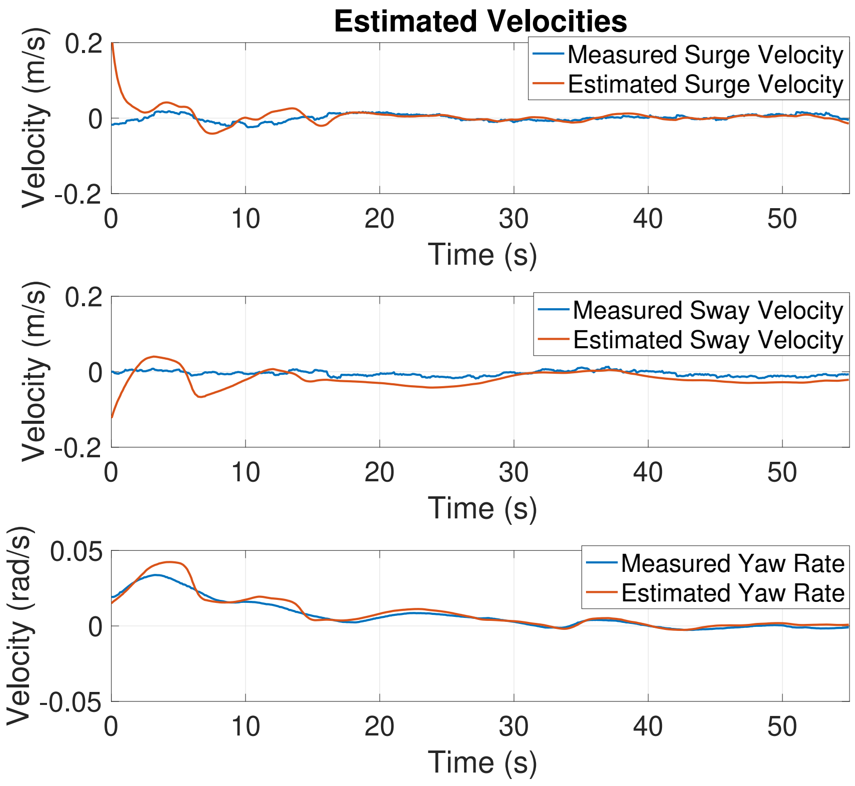

Figure C.10: Estimated velocities of MilliAmpère during experiment 2, MilliAmpère is in DP with desired heading = $57.3°$

C.6 Experiment 3 - MilliAmpère is in DP

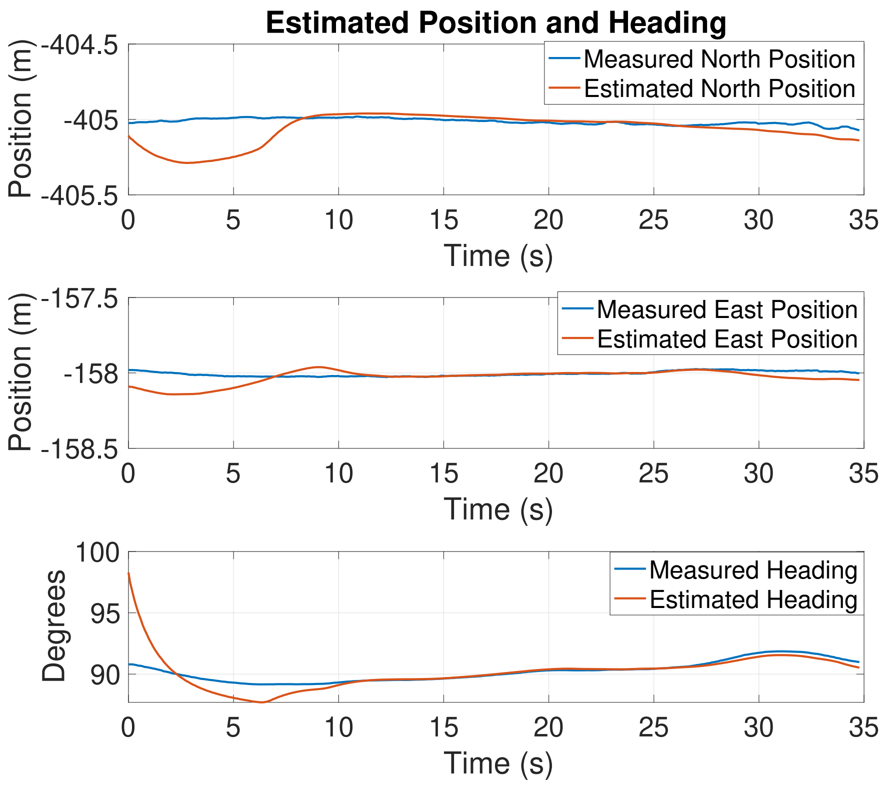

Figure C.11: Estimated position and heading of MilliAmpère during experiment 3, MilliAmpère is in DP with desired heading = $90.0°$

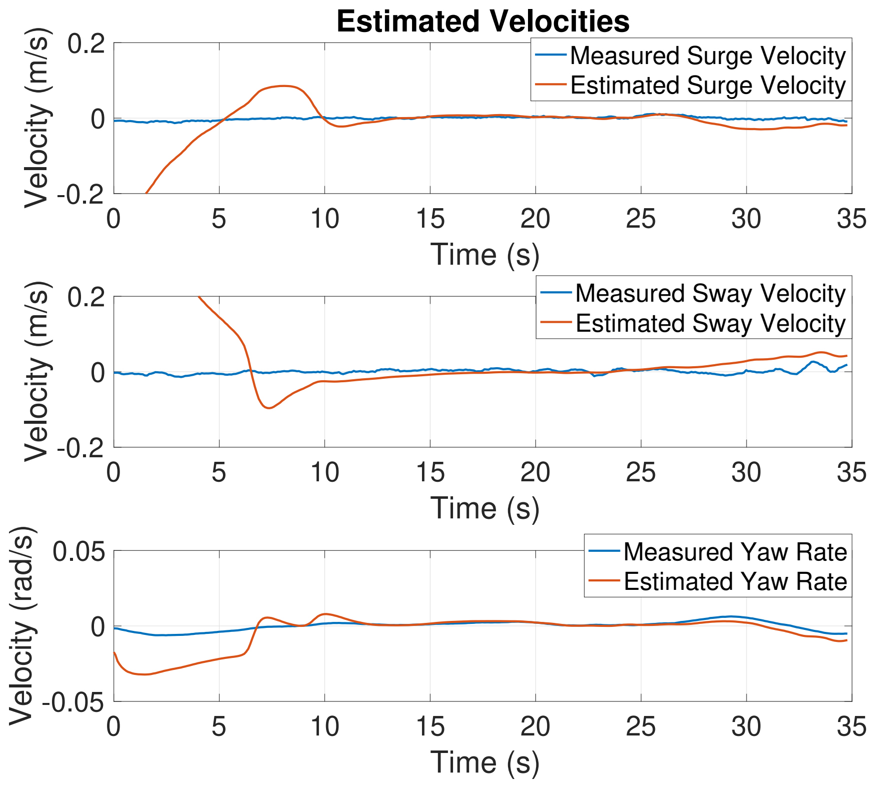

Figure C.12: Estimated velocities of MilliAmpère during experiment 3, MilliAmpère is in DP with desired heading = 90.0°

C.7 Experiment 3 - MilliAmpère is Moving with Constant Velocity

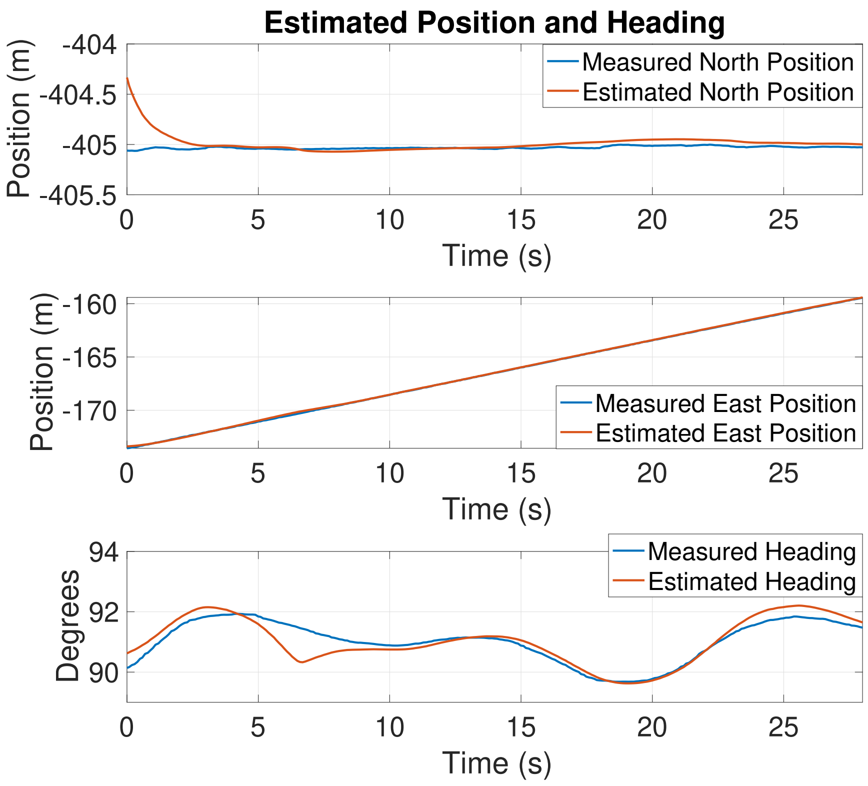

Figure C.13: Estimated position and heading of MilliAmpère during experiment 3, MilliAmpère is moving with constant velocity with desired heading = 90.0°

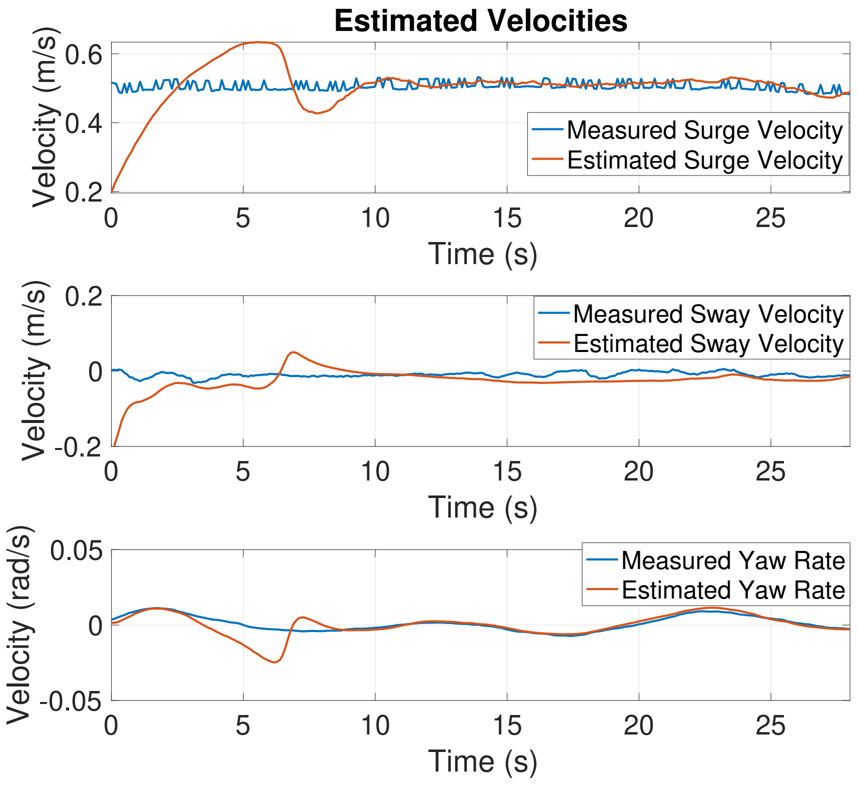

Figure C.14: Estimated velocities of MilliAmpère during experiment 3, MilliAmpère is moving with constant velocity with desired heading = 90.0°