Master's thesis

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

Ørjan Gjernes Riise

Current Estimation for Autonomous Underwater Vehicle using Supervised Learning

Master's thesis in Marine Cybernetics
Supervisor: Dong Trong Nguyen
June 2021

NTNU
Norwegian University of
Science and Technology

Ørjan Gjernes Riise

Current Estimation for Autonomous Underwater Vehicle using Supervised Learning

Master's thesis in Marine Cybernetics
Supervisor: Dong Trong Nguyen
June 2021

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

NTNU
Norwegian University of
Science and Technology

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

PROJECT DESCRIPTION SHEET

Name of the candidate: Ørjan Gjernes Riise

Field of study: Marine Cybernetics

Thesis title (Norwegian): Estimering av havstrøm for autonome undervanns farkoster ved hjelp av overvåket læring.

Thesis title (English): Current Estimation for Autonomous Underwater Vehicle using Supervised Learning.

Background:
NTNU has established a research center in the underwater robot field; Applied Underwater Robotics Laboratory (AUR-Lab). The AUR-Lab has a cross technology discipline including cybernetics, control techniques, marine biology, underwater technique, etc. Underwater vehicles are used to perform research for the AUR-Lab. Among these underwater vehicles, Remote Operated Vehicle (ROV) and Autonomous Underwater Vehicle (AUV) are the most common. Several studies have been performed on underwater vehicles by, among others, Msc-, PhD-students and professors. Simulation models in Matlab/Simulink are forwarded from students and researchers to continuously do research on the vehicles. The purpose of this master thesis is to develop a machine learning algorithm by using Supervised Learning method to estimate the ocean current speed and direction. Another aspect is to further develop the simulation model in Simulink by including a depth controller.

1. **Work description** Perform a background and literature review to provide information and relevant references on:
   - Previous work on underwater vehicles
   - Machine learning in marine application
   - Mathematical modeling of AUVs
   - AUV in subsea operations

2. Propose a mathematical model of REMUS 100.
3. Develop machine learning algorithm for predicting current speed and direction for REMUS 100.
4. Develop a depth controller to handle vertical current.
5. Run simulations to verify and compare proposed algorithms and controller.
6. Conclude and propose ideas for further work

Specifications
The scope of work may prove to be larger than initially anticipated. By the approval from the supervisor, described topics may be deleted or reduced in extent without consequences with regard to grading.

The candidate shall present personal contribution to the resolution of problems within the scope of work. Theories and conclusions should be based on mathematical derivations and logic reasoning identifying the various steps in the deduction.

The report shall be organized in a logical structure to give a clear exposition of background, results, assessments, and conclusions. The text should be brief and to the point, with a clear language. Rigorous mathematical deductions and illustrating figures are preferred over lengthy textual descriptions. The report shall have font size 11 pts. and not be longer than 80 A4 pages, from introduction to conclusion, unless otherwise agreed upon. It shall be written in English (preferably US) and contain the following elements: Title page, abstract, project specification, list of symbols and acronyms, table of contents, introduction and background, problem formulations, scope and delimitations, main body with derivations/developments and results, conclusions with recommendations for further work, references, and optional appendices. All figures, tables, and equations shall be numerated. The original contribution of the candidate and material taken from

other sources shall be clearly identified. Work from other sources shall be properly acknowledged using quotations and a Harvard citation style (e.g. *natbib* Latex package). The work is expected to be conducted in an honest and ethical manner, without any sort of plagiarism and misconduct. Such practice is taken very seriously by the university and will have consequences. NTNU can use the results freely in research and teaching by proper referencing, unless otherwise agreed upon.

The thesis shall be submitted with a printed and electronic copy to the main supervisor, with the printed copy signed by the candidate. The final revised version of this thesis description must be included after title page. The report must be submitted according to NTNU procedures. Computer code, pictures, videos, data series, and a PDF version of the report shall be included electronically with all submitted versions.

Start date: 15 January 2021 **Due date:** 21.06.2021

Supervisor: Dong Trong Nguyen

Trondheim, _____

_____

Dong Trong Nguyen
Supervisor

Preface

This thesis presents work done during TMR4930 Marine Technology - Master's Thesis at NTNU (30 ECTS), and represents the final delivery for a Master of Science within Marine Cybernetics. The work is done from January to June 2021. The thesis has been written entirely by Ørjan Gjernes Riise.

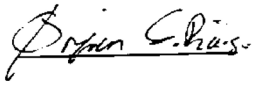
The work was motivated by improving estimation of ocean current for autonomous underwater vehicle REMUS 100 by utilizing Supervised Learning. The aim was to make a regression and classification algorithm capable of predicting the current speed and direction, respectively.

Further, the work was motivated by improving the existing simulator of REMUS 100. The aim was to design a depth controller which is capable of withstand unknown vertical current. Integral Line of Sight algorithm was combined with an existing PID controller and implemented into the simulator.

The reader is expected to have some basic knowledge of marine engineering and control systems, including the development and design of motion controllers. It is also beneficial to have basic knowledge of machine learning. A good intuition from Calculus notation is also beneficial.

Acknowledgments

This master thesis is part of the Master of Science in Marine Technology at the Norwegian University of Science and Technology (NTNU) with specialization in marine cybernetics. I would like to thank my supervisor Dong Trong Nguyen for consulting me in this thesis. Lastly i want to thank my fellow co students for good motivation and discussions through the working process.



ØGR
June 21, 2021

Abstract

Navigation techniques for underwater vehicle depends on reliable velocity estimates. The main uncertainty for velocity estimation is current speed, which also constitutes to the main external disturbance. Doppler Velocity Log (DVL) is, as of today, the leading technology for calculating current speed. When operating low-cost and small Autonomous Underwater Vehicles (AUV), DVL can be unnecessary large, expensive and contribute to high energy consumption. Therefore, this thesis will investigate the potential machine learning has to predict current speed and direction by using Supervised Learning in form of regression and classification algorithms.

Datasets are developed in AUVsim, a simulation model developed by Petter Norgen in Matlab/Simulink. Different variables, such as operating speed, current speed and direction, are altered. The output parameters from the internal controller are saved in $.csv$-files and combined into larger datasets with variables from multiple operating condition. Dataset is used to train each algorithm, and evaluation of the performance is measured by Root Mean Squared Error (RMSE) and Accuracy score (ACC) for regression and classification algorithms, respectively. When algorithms are trained, holdout validation is used due to significant size of dataset.

Comparison between Polynomial, Multiple Linear, Fine Decision Tree, Medium Decision Tree, and Coarse Decision Tree Regression were conducted in a case study. Polynomial Regression model provided best results for numeric estimation of current speed. Weighted and Fine k-Nearest Neighbor (KNN) were compared to Fine and Medium Decision Tree Classification in another case study, where Fine Decision Tree model performed on the highest level.

AUVs must be able to operate in various environmental conditions. An important aspect of the control scheme is precise depth control. The main difficulties evolving precise depth control are due to underactuated system of the AUV, and vertical current caused by upwelling and downwelling. This thesis has therefore further investigated the potential a depth controller has to withstand an unknown vertical current component. The controller scheme includes a vertical Integral Line of Sight (ILOS) algorithm together with an existing PID controller. Simulations were conducted to compare the designed depth controller with the original controller used in the simulator. The presented results are promising as the designed controller was able to reach the desired depth point in both fixed and level depth flight.

The thesis concludes that Supervised Learning method has large potential. The Polynomial Regression model was able to predict the current speed in steady state condition with perfect precision in all cases. The corresponding Fine Decision Tree Classification model followed the same tendency in steady state condition and was able to predict the

current direction in 3/4 cases with perfect precision. The developed depth controller was able to counteract the vertical current component and reach the desired depth point. This indicates that machine learning has great potential to predict current speed and direction by using Supervised Learning in form of regression and classification algorithms. Although, the process is quite time consuming to achieve good results. This is due to the need of high-quality data and significantly training time when dataset is large.

Sammendrag

Navigasjonsteknikker for undervannsfartøy avhenger av pålitelige hastighetsestimater. Den største usikkerheten for estimatene er havstrømninger, som utgjør den største eksterne forstyrrelsen. DVL er per i dag den ledende teknologien for beregning av havstrømninger. Ved små, lav budsjetterte AUV-er, kan DVL være unødvendig stort, dyr i drift og bidra til høyt energiforbruk. Derfor vil denne oppgaven undersøke potensialet maskinlæring har får å estimere hastighet og retning på havstrømninger ved bruk av overvåket opplæring i form av regresjons- og klassifiseringsalgoritmer.

Datasett er utviklet i AUVsim, en simuleringsmodell utviklet av Petter Norgen i Matlab/Simulink. Ulike variabler, som driftshastighet, havstrømning og retning, endres. Utgangsparameterne fra den interne kontrolleren lagres i $.csv$ -filer og kombineres i større datasett med variabler fra flere driftsforhold. Datasett brukes til å trene hver algoritme, og evaluering av ytelsen måles ved henholdsvis RMSE og ACC for regresjons- og klassifiseringsalgoritmer. Når algoritmer blir trent, brukes tilbake-holdning av datasettet som valideringsmetoden, grunnet betydelig størrelse på datasettet.

Sammenligning mellom polynom, multivariat lineær, fin-, middels- og grov-beslutningstre regresjon ble utført i en casestudie. Polynomial regresjonsmodell ga de beste resultatene for numerisk estimering av havstrømning. Vektet og fin KNN ble sammenlignet med fin- og middels-beslutningstre klassifisering i en annen casestudie, hvor fin-beslutningstre klassifisering oppnådde best resultat.

AUV-er må kunne operere under varierende miljøforhold, hvor presis dybdekontroll er en avgjørende faktor. Vanskeligheter med å utvikle presis dybdekontroll skyldes AUV-enes underaktiverte system og vertikal strøm forårsaket av opp- og nedtrekk. Av den grunn, har denne oppgaven videre undersøkt potensialet en dybdekontroller har for å tåle en ukjent vertikal strømkomponent. Kontrolleren inkluderer en ILOS-algoritme sammen med en eksisterende PID-kontroller. Simuleringer ble gjennomført for å sammenligne den foreslåtte dybdekontrolleren med den opprinnelige kontrolleren originalt brukt i simulatoren. Simuleringer indikerer lovende resultater hvor den foreslåtte kontrollere er i stand til å nå ønsket dybdepunkt, både ved fast og varierende dybde.

Oppgaven konkluderer med at metoden for overvåket opplæring har stort potensiale. Polynomial regresjonsmodellen var i stand til å forutsi gjeldende hastighet i stabil tilstand med perfekt presisjon i alle tilfeller. Den tilsvarende fin-beslutningstre-modellen fulgte den samme trenden i stabil tilstand og var i stand til å forutsi gjeldende retning i 3/4 tilfeller, med perfekt presisjon. Dybdekontrolleren foreslått i denne oppgaven var i stand til å motvirke den vertikale strømkomponenten, og dermed nådde ønsket dybdepunkt. Dette indikerer at maskinlæring har stort potensiale for å estimere hastighet og retning på havstrømning ved bruk av overvåket opplæring i form av regresjons- og klassifiser-

ingsalgoritmer. For å oppnå gode resultater, er prosessen imidlertid tidkrevende. Dette skyldes behovet for data av høy kvalitet og betydelig opplæringstid når datasettet er stort.

Table of Contents

List of Tables

List of Figures

Abbreviations

ACC	=	Accuracy
ADCP	=	Acoustic Doppler Current Profilers
AOI	=	Area Of Interest
APR	=	Average Precision
AUG	=	Autonomous Underwater Gliders
AUR-Lab	=	Applied Underwater Robotics Laboratory
AUV	=	Autonomous Underwater Vehicle
BEP	=	Breakeven Point
CACLA	=	Continuous Actor Critic Learning Automaton
CB	=	Centre of Buoyancy
CG	=	Center of Gravity
CO	=	Center of Origin
CNN	=	Convolutional Neural Network
EKF	=	Extended Kalman Filter
DOF	=	Degrees of Freedom
DP	=	Dynamic Positioning
DPSS	=	Differential Pressure Sensor Speedometer
DUNE	=	Unified Navigational Environment
DUSBL	=	Digital Ultra-short Baseline
DVL	=	Doppler Velocity Log
GNSS	=	Global Navigation Satellite System
GUI	=	Graphical User Ineterface
HGO	=	High Gain Observer
HiPAP	=	High Precision Acoustic Positioning
HMD	=	Head-Mounted-Display
HMI	=	Human-Machine-Interface
IMU	=	Inertial Measurement Unit
KNN	=	k-Nearest Neighbor
LBL	=	Long Baseline
LMI	=	Linear Matrix Inequality
LOS	=	Line Of Sight
ILOS	=	Integral Line-Of-Sight
LQG	=	Linear Quadratic Gaussian
LQR	=	Linear Quadratic Regulator
INS	=	Inertial Navigation System
MA-INS	=	Model Aided Inertial Navigation System
MR-EKF	=	Multi-Rate Extended Kalman Filter
NED	=	North-East-Down
NN	=	Neural Network
OOI	=	Object Of Interest

PID	=	Proportional Integral Derivative
RBF-NN	=	Radial Basis Function Neural Network
RMSE	=	Root Mean Squared Error
RNN	=	Recurrent Neural Network
ROV	=	Remote Operated Vehicle
SARSA	=	State–Action–Reward–State–Action
SLAM	=	Simultaneous Localization and Mapping
SMC	=	Sliding Mode Control
SNAME	=	Society of Naval Architects and Marine Engineers
SSS	=	Side Scan Sonar
SVM	=	Support Vector Machine
DUNE	=	Unified Navigation Environment

Chapter 1

Introduction

1.1 Background

NTNU has established a research center for underwater robot field; Applied Underwater Robotics Laboratory (AUR-Lab). The AUR-Lab has a cross technology discipline including cybernetics, control techniques, marine biology, underwater technique, etc. Underwater vehicles are used to perform research for the AUR-Lab. Among these underwater vehicles, Remote Operated Vehicle (ROV) and Autonomous Underwater Vehicle (AUV) are the most common. Several studies have been performed on underwater vehicles by, among others, master students, Doctor of Philosophy (PhD) and professors. Simulation models in Matlab/Simulink are forwarded from students and researchers to continuously do research on the vehicles.

Underwater vehicles can be divided into unmanned and human occupied vehicles as illustrated in Figure 1.1. Unmanned underwater vehicles can further be divided into three main categories: ROVs, AUVs and Autonomous Underwater Gliders (AUGs). These categories play an important role in monitoring the marine environment where they operated in different regimes. ROVs are characterized by remote operation and presence of a tether cable. They are typical used for underwater operations unreachable for human operators such as sampling of biological, chemical, geological objects of interests, deep-water archaeology, pipeline survey, and sub-sea structure inspection, maintenance and repair (Candeloro et al., 2012). AUVs are characterized by their autonomous behavior in absence of a tether cable (Souza and Maruyama, 2007). They are propulsion driven torpedo shaped vehicles programmed to execute a specific mission without human interference. AUG is a special type of AUV with the advantage of long spatial coverage. The glider can change its buoyancy with a hydraulic pump and utilize lift from the wing to generate forward motion (Cruz, 2011).

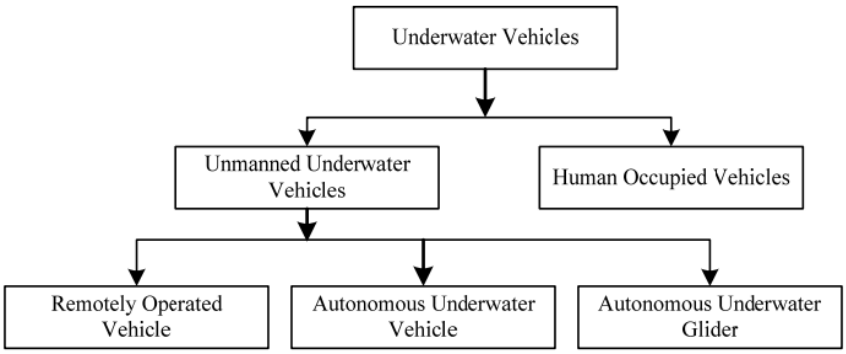


Figure 1.1: Different categories of underwater vehicles divided into unmanned underwater vehicles and human occupied vehicles (Cruz, 2011).

Figure 1.2 illustrate the performance of ROV, AUV and AUG with respect to maneuverability and endurance. As the spatial coverage becomes larger, the spatial resolution becomes smaller. The AUG has low maneuverability which results in poor spatial resolution of an area due to its inability to perform a fixed depth or level flight (Cruz, 2011). On the other hand, ROV has high maneuverability and low spatial coverage. The performance of the AUV fit in between the AUG and ROV.

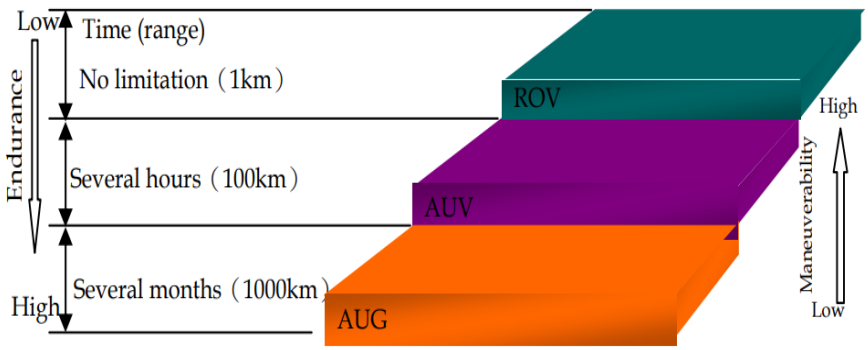


Figure 1.2: Performance of ROV, AUV and AUG with respect to maneuverability and endurance (Cruz, 2011).

1.1.1 Level of Autonomy

Due to autonomous system and operations being associated with unmanned systems, it is important to distinguish between unmanned system and autonomy. The level of autonomy is, according to Utne et al. (2017), divided into four levels: automatic operation (remote control), management by consents, semi-autonomous operation or management by exception, and highly autonomous operation. The following paragraph explains the four levels (Utne et al., 2017):

- Automatic operation (remote control): The system operates automatically, where the human operator directs and controls the high-level mission planning functions. The mission is often programmed before executing the operation. The operator is directly involved in the operation and is presented system states, environmental conditions through a Human-Machine-Interface (HMI).

- Management by consents: The system automatically makes recommendations for mission or process actions related to specific functions. The operator is given notification when important decisions need to me made. The system has often a limitation in communication bandwidth.

- Semi-autonomous operation: The system will execute mission-related functions when response times from operator is too short. The system could be override or channeled by operator within certain time frame.

- Highly autonomous operation: The system is highly sophisticated and could plan and replan the mission process where the human may be informed by the progress. This level of the system is classified as independent, and human is out if the loop.

1.1.2 REMUS 100 AUV

The REMUS 100 is a medium sized AUV driven by a single propeller, illustrated in Figure 1.3. REMUS 100 has two control surfaces consisting of two horizontal and two vertical fins to control pitch and heading, respectively. The fins are unable to be controlled individually, thus moves as one unit.

Figure 1.3: Illustration of the AUV REMUS 100 (Hydroid, 2012).

REMUS 100 takes part in several different operations, such as bathymetric mapping, substrate identification, biomass assessments in the water column, water column characterization and environmental monitoring. Each of the operations require different payload sensors and a sophisticated control system. The specifications and sensor configurations are described in detail in Section 2.4.

1.1.3 AUVSim

All simulations in this thesis is performed using AUVSim. AUVSim is a simulator for REMUS 100 created by previous PhD candidate Petter Norgren (Norgren and Skjetne, 2015). The simulator is implemented in Matlab and Simulink.

1.2 Motivation

The ocean covers more than 70% of the earth and contains a large amount of resources (National Oceanic and Atmospheric Administration). The oil and gas industry are one of the largest spenders of the ocean, both over and under water. Underwater vehicles play an important role in mapping, monitoring, and repairing of structures and environment under water. Since there is a constantly interest to investigate new and sustainable technologies for extracting resources from the ocean, further technology on underwater vehicles is needed.

Mainly, the ROVs are used for inspection and maintenance which requires remote-control operator and an umbilical tether from a support vessel (Zagatti et al., 2018). Thus, making pipeline inceptions to an expensive operation. The costs can be significantly reduced, and weather window extended, by replacing the ROVs with AUVs. AUVs are essential when it comes to underwater mapping and inspections, but to be highly reliable they must be able to handle various operating conditions. Therefore, the AUVs require a well develop control system to be able to follow a predetermined path with the influence of external disturbances. This motivates for a robust control system.

Ocean currents are one of the main challenges for underwater vehicles. One of the most established navigation techniques is to integrate the velocity and acceleration to obtain the position. The velocity is often obtained with a water speed sensor. The problem with this technique, is that the current velocity is not captured and the current profile adds a velocity component (Leonard et al., 1998). Accurate prediction of the current is therefore necessary to establish trustworthy estimates.

1.3 Literature Review

This chapter firstly provides previous work on underwater vehicles in Section 1.3.1 and machine learning in Section 1.3.2. Some of the presented studies are developed for ROVs, but parallels can be drawn to AUVs. Various methods for current estimation are further presented in Section 1.3.3, and a general description of control system in Section 1.3.4. Some of the literature review is based on a project thesis from 2020 (Riise, 2020).

1.3.1 Previous Work on Underwater Vehicles

Within recent years, several studies have contributed to the AUR-Lab project. One of the challenges regarding underwater navigation, is the lack of Global Navigation Satellite System (GNSS). In 2010, Dukan (2014) started the work on ROV Dynamic Positioning

(DP) system where his PhD thesis focused on guidance and navigation, where a motion control system was developed. A model based Kalman filter, both linearized and extended, was developed and tested. The use of sensor-based state estimation was inspired by the need of an observer that works during manipulation work and other uncertainties. An explicit complementary filter was adopted and modified to use as an altitude estimator. For guidance module, a joystick in closed loop was made with reference models consisting of straight-line movements from point A to B, and path tracking. In addition, altitude control and terrain following were developed.

Candeloro (2016) studied various operation regime of the ROV, where a multi-objective observer based on the nonlinear passive filter was developed. To the guidance aspect, an ILOS algorithm was implemented and tested where the user was a part of the control chain. An innovative Human-Machine-Interface (HMI), based on the Head-Mounted-Display (HMD) technology, was developed and implemented with the purpose of improving the state of the art of the ROV interface. The information from the camera technology was used to make a prediction for the optimal direction of the vehicle. The idea was to move towards the area with greater density of Object Of Interest (OOI).

Nornes (2018) further developed various aspect of the system after Dukan (2014) and Candeloro (2016). Nornes considered development of the methods for motion control and mapping systems for marine robotic with different level of autonomy. The thesis focus on increasing the level of autonomy of the systems to reduce cost and the need for human interaction. An automated relative motion control strategy was developed for mapping underwater structures using a ROV. This method involves the use of a Doppler Velocity Logger (DVL) in the direction of the camera, to be able to keep a constant distance to the Area Of Interest (AOI). Through full scale experiments, the strategy showed that the ROV was able to record high quality images of a challenging structure.

A study by Hegrenæs and Hallingstad (2011) evaluated the state of the art of Model Aided Inertial Navigation System (MA-INS) for underwater vehicle. Hegrenæs and Hallingstad (2011) stated that for all scenarios, the MA-INS is considerably more robust compared to the system not including model aiding. Further, experimental results verified that the MA-INS solution was superior to that obtained with the conventional Inertial Navigation System (INS) when DVL measurements are unavailable.

A study considering the modelling, design and control of Kaxan ROV was performed by García-Valdovinos et al. (2014). The following aspect was done: a complete 6 Degrees of Freedom (DOF), nonlinear hydrodynamic model with its parameters, the Kaxan hardware/software architecture, numerical simulations in Matlab/Simulink platform of a model-free second order sliding mode control along with ocean currents as disturbances and thruster dynamics, a virtual environment to visualize the motion of the Kaxan ROV, and experimental results of a 1 DOF underwater system. The study showed excellent result for the proposed sliding mode controller.

Norgren (2018) studied AUVs in arctic marine operations with the aim of using the ve-

hicle as a sensor platform when monitoring ice operations. Iceberg mapping was the main focus with AUV used to generate trajectory models, as well as decision support in iceberg management. AUVsim was developed in Matlab/Simulink to represent the real AUV. For estimation of the relative position and velocity between the iceberg and AUV, an Extended Kalman filter (EKF) was implemented with Simultaneous Localization and Mapping (SLAM) states as input.

Holsen (2015) implemented Unified Navigational Environment (DUNE) for developing control system on REMUS 100. DUNE is an open-source framework implemented as an interface with REMUS 100 to control the heading and altitude of the AUV. Simulations and field tests showed that DUNE was suitable to control REMUS 100. For further research, DUNE can be used as a development platform.

To prolong the operation regime of AUVs and reduce the hands on intervention of human before and after mission, Ruud (2016) studied and developed an autonomous home docking algorithm. Simulations showed promising result together with Digital Ultra-short Baseline (DUSBL) sensor. A particle filter with range-only data from Long Baseline (LBL) sensor showed that docking was successfully.

The control aspect of AUVs is often divided into two separate systems; horizontally and vertically control. Wang et al. (2011) developed a path following controller in the vertical plane, where a backstepping method based on feedback gains was used. To compensate for dynamics of the AUV, an adaptive Neural Network (NN) was introduced. The network weight adaptation law was derived from the Lyapunov stability analysis. Validation of the presented controller and Radial Basis Function NN (RBF-NN) was done through simulations.

Line Of Sight (LOS) is a common guidance algorithm used for path following. If the vehicle is exposed to external disturbances, such as current, integral action could be introduced to the guidance level to counteract the unknown disturbance. Caharija et al. (2012) modified three-dimensional LOS guidance with integral action (ILOS) together with three adaptive feedback controllers to perform a horizontal path following in presence of vertical irrotational ocean current. Through simulation the guidance law was tested, and the closed loop dynamics gave explicit condition to guarantee asymptotic path following. Ye et al. (2018) combined backstepping technique and ILOS guidance law to develop a diving control for underactuated AUVs.

Østeby (2017) proposed a cable detection algorithm that uses the Side Scan Sonar (SSS) on REMUS 100. A Multi-Rate Extended Kalman Filter (MR-EKF) with a cable model was used to fusing measurements from multiple sensors with different sampling rates. The cable position was estimated from the magnetometer and SSS. The state estimations from MR-EKF are sent to the guidance law, which produce the wanted heading by a LOS algorithm. A more detailed description can be found in Østeby (2017). To be able to follow the seabed and do pipeline inspection, precise depth control of AUV is essential. Difficulties involving depth control is mainly due to influence of vertical ocean currents

caused by upwelling and downwelling phenomena, and the fact that the system is underactuated (Ye et al., 2018).

Various control schemes have been developed to handle different operating conditions. To guarantee a robust and trustworthy system, non-linear control is often chosen for AUVs due to highly non-linear dynamics. Johansen (2020) proposed a Sliding Mode Control (SMC) dealing with the highly coupled and non-linear dynamics in 6 DOF. Simulations showed that the SMC was a suitable control scheme for underwater vehicles.

1.3.2 Examples of Machine Learning

Machine learning is a up growing field dealing with various aspects of algorithms, trying to learn the relations between variables. People are exposed to machine learning every day in the form of virtual personal assistants used in our smartphone, social media service used for personalizing your news feed to better ads targeting, email spam and online customer support, to mention some.

In marine applications, the use of machine learning has been implemented to identify the unique sea state conditions and their impact on vessels, see Bailey et al. (2019) for more information. Sclavounos and Ma (2018) introduced machine learning to study complex potential and viscous flow problems in marine hydrodynamics with the use of Support Vector Machine algorithm (SVM).

A study by Mak and Düz (2019) estimated the sea state characteristic from an in-service ship motion by using data from a sampling period of 2 years. The input data needed to be able to capture the relation between input channels and time dependence. Here, Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) were used.

Sea state estimation with use of quadratic discriminant and partial least square regression was conducted by Arneson et al. (2019). The study used machine learning algorithms instead of the vessel transfer function, and simply relied on estimating the sea state based on a combination of parameters calculated using the vessel response in all its DOF. The trained algorithm showed promising result when estimating wave directions. Estimation of significant wave height and peak wave period provided the best result for lower sea states.

One of the many operations AUV performers are pipeline inceptions. Fjerdingen et al. (2010) studied the use of reinforcement learning techniques for continues state and action spaces to pipeline following. The study looked at continuous space State Action Reward State Action (SARSA) and Continuous Actor Critic Learning Automaton (CACLA). These was extended into a supervised reinforcement learning architecture, and the result was validated with simulations. The supervised CACLA was the best candidate as it showed the ability to generalize the learned pipeline following strategy to new and unknown pipe geometries. Fjerdingen et al. (2010) stated that reinforcement learning was well suited to optimize pipeline following behavior for an AUV.

1.3.3 Current Estimation

Navigation of AUVs depends highly on the quality of the state estimates. Ocean current contributes to the main uncertainty in navigation of AUVs. Reliable estimates of the velocity are therefore important to provided trustworthy state estimations. State of the art velocity estimation relies on expensive acoustic sensors with considerable energy requirements and a large form factor, such as DVL and Acoustic Doppler Current Profilers (ADCP) (Meurer et al., 2020).

Within resent years, several studies have been performed to optimize velocity estimations. The sensor system for navigation is restricted by size, energy requirements, budgets and operating conditions (Meurer et al., 2019). A study by Meurer et al. (2019) was conducted on a Differential Pressure Sensor Speedometer (DPSS). The following points were obtained during filed tests:

- DPSS and DVL experienced similarly effect by environmental disturbances.
- In open sea, with higher environmental disturbance, the DPSS showed a robust performance. For lower velocities, the impact on disturbance was more significant for the DPSS compared to the DVL.
- Hydrostatic correction algorithm produced a more accurate velocity estimation for the DPSS in the beginning of filed trails, but further investigation is suggested regarding hydrodynamic effects in roll and pitch.
- Based on the two filed tests, Meurer et al. (2019) suggested that an autocalibration to procedure individual offset should be done before each trail. This would lead to an increased performance of the DPSS.
- In terrains with rough or unstructured surfaces, or in bottom lock, the DPSS showed superior performance over the DVL.

In a study by Meurer et al. (2020), a low cost DPSSv2 sensor was introduced to estimate the relative velocity of fluid in two-dimensional based on differential pressure. The sensor was validated with field test in the Trondheim Fjord. There was conducted 14 trails where the AUV followed a straight line with constant depth. Operating speed was set to minimum 1.25 m/s for 6 trails, and 1.5 m/s for the remaining trails. Both cases executed half of its trails against the main direction of the anticipated current, and the remaining in the opposite direction. Through filed trails, the DPSSv2 sensors managed to estimate tidal currents in situ with comparable accuracy to a DVL. The filed test also established that in most of the cases, the DVL-WL and DPSSv2 estimates the speed with lower variance then the ADCP. Meurer et al. (2020) stated that the DPSSv2 has the potential to reduce the power requirements to 0.244W versus 1.3 W for the DVL.

Kim et al. (2020) studied the path following problem for AUV under nonuniform current to estimate the three-dimensional current velocity along the AUV. A High Gain Observer (HGO) was used and chosen as a nonlinear estimation algorithm. By solving the estimation errors dynamics through a Linear Matrix Inequality (LMI), the observer gain

was computed. The current velocities were determined by calculating the differences between the measured absolute velocities of the vehicle and the estimated relative velocities of the vehicle, predicted by the observer. To verify the HGO, numerical simulations were performed with current compensation. The result showed that the AUV converged to the desired path.

Different methods to compensate for the bias load in DP marine vessel was presented in an article by Værnø et al. (2019). Four different methods were investigated: bias estimates from an observer tuned to estimate the position and velocity, a wave-filtered version of the bias load, one separated observer that was tuned specific for the bias load, and lastly, a basic integral action on the tracking error. Værnø et al. (2019) stated that the best method to compensate for bias load is using the bias estimated from the separated observer.

1.3.4 Control System

Control systems consist of several modules with different tasks. Figure 1.4 illustrate how the modules are connected to each other. The modules will be presented in the following subsections, with the intention to give the reader a brief overview.

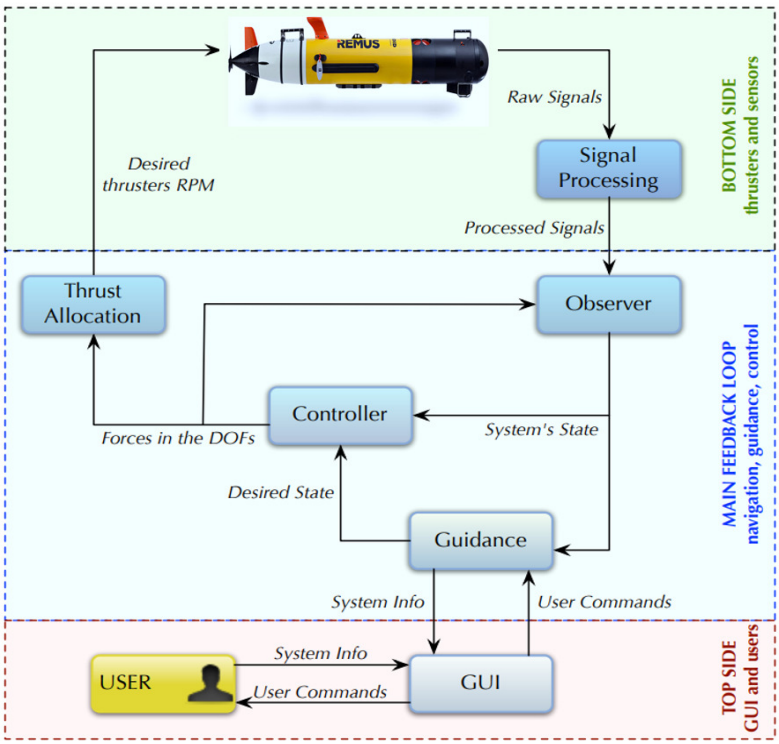


Figure 1.4: Overview of the modules for control system and there connections with each other (Candeloro, 2016).

Controller

The controller takes the system state as input and calculates the desired force in surge, sway and heave, and moments in roll, pitch and yaw. The input to the controller consists of the estimated and desired state from the observer and guidance module, respectively. There exists a various type of controllers which can be divided into two main groups; linear and nonlinear controllers. Linear controllers include Proportional Integral Derivative (PID), Linear Quadratic Regulator (LQR) and linear Quadratic Gaussian (LQG) (Kokegei et al., 2011; Reshmi and Priya, 2016; Naeem et al., 2003). Nonlinear include sliding mode, backstepping, adaptive control, predictive control, and fuzzy logic control (Johansen, 2020; Ye et al., 2018; Liu et al., 2012). The controllers output is given in Equation 1.1.

$$\tau = [\tau_x, \tau_y, \tau_z, \tau_\phi, \tau_\theta, \tau_\psi]^T \tag{1.1}$$

Thrust Allocation

Thrust allocation module computes the desire force vector, τ, from the controller to the corresponding force and direction commands, to each thruster device. The low level thrust controller will control the propeller pitch, torque, speed and power, to satisfy the desired thrust demands (Sørensen, 2018). An important task for this module is to optimize the algorithm to minimize fuel and energy consumption.

Signal Processing

The signal processing module takes in raw signals for evaluation, before sending the signals to the observer. To detect failures, it is important to evaluate each signal. Different methods to evaluate signals are (Sørensen, 2018):

- Signal range testing: If the signal is outside of the defined range, the signal is rejected.
- Variance testing: This gives an indication of the variations in amplitude and frequency. For instance, high variance can indicate high level of process noise.
- Wild point testing: Wild points are indicated as a value that varies substantively from the previously sample, and the measurement should therefore be rejected for one sample.

For a redundant sensor or position reference system configuration, the signal process module can do voting between the sensors. If two position reference system are available, the module can detect drifting between the two sensors. Weighting between sensors is done to decide which sensor gives the “best” measurement. More information can be found in Sørensen (2018).

Observer

The main objective for the observer is to filter out unwanted noise and estimate various states. Various states are needed to be estimated if there is a loss of signal, such that the

predicted estimates are used in the control loop. This is called dead reckoning. Another feature of the observer is estimating the unmodelled and unmeasured slowly varying forces and moments, mainly due to ocean current (Sørensen, 2018).

Guidance System

The guidance system gives the desired states as input to the controller. This could for example be a constant set-point for the vessel in DP-mode. For an operation where the desired state changes, it is important to include a reference model to get a smooth transition to avoid jerk on the vessel (Johansen, 2020). A sophisticated guidance system involves way-point tracking, path planning and weather routing. The guidance system could also be interfaced to electric map system (Sørensen, 2018).

Graphical User Interface

Graphical User Interface (GUI) provide a graphical presentation of the computer program. It enables communication between a person and a computer, and therefore makes connection between the system and the operator. The communication works through symbols, visual metaphors and pointing device. GUI provide commands to the guidance system and vice versa.

1.3.5 Sensor Systems

The aim of an AUV is to gather data of interest in underwater locations. This can include geological data of surroundings, water column inspection, subsea operations, pipeline inspections, etc. This information could be of interest to academic research, commercial players, private players, military or policy sectors. It is common to divide the sensors into two groups: payload sensors and navigation sensors (Sørensen and Ludvigsen, 2015).

Payload senors generally involves the collection of data. An AUV can have different sensor configuration depending on the mission. Navigation sensor measure the state of the vehicle and are essential for the AUV such that it is able to perform a given task. The information required for navigation is the direction, speed, and position of the vehicle (Nebot, 1999). Table 1.1 shows the main payload and navigation sensors for an AUV. Detailed description on each sensor can be found in Ruud (2016).

Payload sensors	Navigation sensors
Acoustic Doppler Current Profilers	Acoustic Baseline Sensors:
	- Long Baseline
Conductivity Temperature Depth sensors	- Short Baseline
	- Super-Short Baseline
Syntetic Aperture Sonar	Doppler Velocity Log
Side Scan Sonar	Heading and Inertial sensors
Environmental Characterization Optics	
Multiparameter Sonde	

Table 1.1: Main payload- and navigation sensors for AUV.

1.4 Objectives

The aim of this master thesis is to investigate the potential machine learning has to estimate ocean current for NTNUs Autonomous Underwater Vehicle (AUV) REMUS 100. Estimation of ocean current speed and direction is done by using Supervised Learning in form of regression and classification algorithms, which is validated with simulations in Matlab/Simulink. Further an Integral Line-Of-Sight (ILOS) algorithm designed to produce the desired theta angle, is combined with an existing PID controller to counteract vertical ocean current. The field of machine learning is complex and consist of a large number of algorithms. Therefore, a background research is necessary before the developing process begins. The goals and research questions are comprised into the following objectives:

- Review a wide variety of literature regarding previous work on underwater vehicles, machine learning in marine applications and current estimation. State relevant theory regarding mathematical modelling of underwater vehicles and machine learning.
- Design a machine learning algorithm for predicting speed and current direction of REMUS 100 by evaluating various regression and classification methods.
- Develop a depth controller that can handle vertical current.
- Run simulations in Simulink to verify and compare proposed algorithms and controller.

1.5 Contribution

The main contribution is to estimate the current speed and direction for the AUV REMUS 100. Supervised learning is used due to the datasets being labeled. The aim is to predict the numeric value of the current speed based on thrust force, fin and rudder angle, and the surge speed of the AUV. Therefore, the regression method is chosen. The algorithms are developed in Python by using the programming platform Spyder, and Matlabs build

in app, Regression learner. Matlab is selected to investigate the potential with direct implementation from Matlab to Simulink. To categorize the current direction, classification algorithms are developed using the same input as for the regression. Several algorithms, both regression and classification, are tested and compared to investigate the potential of machine learning performing current estimation. Lastly, a depth controller is development to counteract vertical current, which involves a ILOS algorithm in combination with an existing PID controller.

1.6 Organization of Project

Chapter 2 presents a general mathematical modeling for underwater vehicles inspired from marine vessel. Further, a basic introduction of machine learning with different methods and evaluation of algorithms, are presented. In **Chapter 3** the simulation environment is presented along with the method for generating the machine learning algorithms, and vertical ILOS algorithm in the depth controller. The result from the algorithms and depth controller are presented in **Chapter 4** along with a discussion regarding the findings. Finally, the conclusion and further work are stated in **Chapter 5**.

Chapter 2

Theory

This chapter presents relevant theory about mathematical modelling of underwater vehicles, control and guidance system for REMUS 100 and machine learning. The theory in this Section 2.1 - 2.4 is based on previously PhDs, Master thesis, and the *Handbook Of Marine Craft Hydrodynamics And Motion Cotrol* by Fossen (2011). There is therefore none new contributions to the mathematical models and equations in these sections. In Section 2.5, a brief introduction to machine learning are presented. Some of the theory is based on a project thesis from 2020 (Riise, 2020).

2.1 Mathematical Modelling of Underwater Vehicle

An underwater vehicle usually consists of 6 Degrees of Freedom (DOF). When the body has an accelerated motion, it refers to the dynamics. The dynamics is highly nonlinear due to ridged body coupling and hydrodynamic forces on the vehicle. The mathematical model is obtained through two models: the dynamic- and kinematic model (Sabiha and Pushkin, 2018).

Dynamic model: The dynamic model uses Newton's law to obtain the equation of translation and rotation. The model allows for the actual forces causing the motion and the dynamical properties to be accounted for.

Kinematic model: The kinematic model does not take the force and the dynamical properties into account when computing the equation of motion. This model allows decoupling of the vehicle dynamics from its movement.

The 6 DOF standard definition of Society of Naval Architects and Marine Engineers (SNAME) for marine vessel is used, presented in Table 2.1.

DOF		Forces and moments	Linear and angular velocities	Positions and Euler angles
1	surge	X	u	x^n
2	sway	Y	v	y^n
3	heave	Z	w	z^n
4	roll	K	p	ϕ
5	pitch	M	q	θ
6	yaw	N	r	ψ

Table 2.1: The notation of SNAME for marine craft (Fossen, 2021).

2.1.1 Kinematics

When analyzing motion of AUV, two reference frames are used: the North-East-Down (NED) frame and the body frame. NED frame is denoted as $\{n\} = (x_n, y_n, z_n)$, where x_n axis's point towards true north, and y_n axis's points towards east. z_n axis's points downwards, normal to earth's surface. The body frame is denoted as $\{b\} = (x_b, y_b, z_b)$. The origin is a moving coordinate frame that is fixed to the AUV. The position and orientation of the AUV is described relative to the inertial reference frame. The linear and angular velocities are expressed in the body frame (Fossen, 2021). Equation 2.1a and 2.1b presents the position and velocity vector, respectively.

$$\boldsymbol{\eta} = [x, y, z, \phi, \theta, \psi]^T \tag{2.1a}$$

$$\boldsymbol{\nu} = [u, v, w, p, q, r]^T \tag{2.1b}$$

The generalized position, velocity and force vector is given is Equation 2.2, respectively. The relation between NED and body frame are given in the kinematic Equation 2.3, and the rotation matrices are given in Equation 2.4 and 2.5 (Fossen, 2021).

$$\boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{p}_{nb}^n \\ \boldsymbol{\Theta}_{nb} \end{bmatrix}, \quad \boldsymbol{\nu} = \begin{bmatrix} \boldsymbol{v}_{nb}^b \\ \boldsymbol{\omega}_{nb}^b \end{bmatrix}, \quad \boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{f}_b^b \\ \boldsymbol{m}_b^b \end{bmatrix} \tag{2.2}$$

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}_{\boldsymbol{\Theta}}(\boldsymbol{\eta})\boldsymbol{\nu} \tag{2.3a}$$

$$\begin{bmatrix} \dot{\boldsymbol{p}}_{nb}^n \\ \dot{\Theta}_{nb} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}(\boldsymbol{\Theta}_{nb}) & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{T}(\boldsymbol{\Theta}_{nb}) \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_{nb}^b \\ \boldsymbol{\omega}_{nb}^b \end{bmatrix} \tag{2.3b}$$

$\boldsymbol{R}_b^n$ denotes the rotation matrix from body frame to NED frame, where $c\cdot = \cos(\cdot)$, $s\cdot = \sin(\cdot)$, and $t\cdot = \tan(\cdot)$

$$\boldsymbol{R}(\Theta_{nb}) = \begin{bmatrix} c\psi c\theta & -\mathrm{s}\psi\mathrm{c}\phi + \mathrm{c}\psi\mathrm{s}\theta\mathrm{s}\phi & \mathrm{s}\psi\mathrm{s}\phi + \mathrm{c}\psi\mathrm{c}\phi\mathrm{s}\theta \\ \mathrm{s}\psi\mathrm{c}\theta & \mathrm{c}\psi\mathrm{c}\phi + \mathrm{s}\phi\mathrm{s}\theta\mathrm{s}\psi & -\mathrm{c}\psi\mathrm{s}\phi + \mathrm{s}\theta\mathrm{s}\psi\mathrm{c}\phi \\ -\mathrm{s}\theta & \mathrm{c}\theta\mathrm{s}\phi & \mathrm{c}\theta\mathrm{c}\phi \end{bmatrix} \tag{2.4}$$

$\boldsymbol{T}(\boldsymbol{\Theta}_{nb})$ is given by:

$$\boldsymbol{T}\left(\boldsymbol{\Theta}_{nb}\right) = \begin{bmatrix} 1 & \mathrm{s}\phi\mathrm{t}\theta & \mathrm{c}\phi\mathrm{t}\theta \\ 0 & \mathrm{c}\phi & -\mathrm{s}\phi \\ 0 & \mathrm{s}\phi/\mathrm{c}\theta & \mathrm{c}\phi/\mathrm{c}\theta \end{bmatrix} \tag{2.5}$$

2.1.2 Kinetics

Process plant model is a high-fidelity model which simulates as close as possible the real physical aspect of the system dynamics, including process disturbance, sensor outputs and control inputs. This type of model is for numerical analysis of the stability and performance of the closed-loop system (Dong, 2005). The nonlinear 6 DOF body-fixed coupled equation of the low frequency (LF) motions is given in Equation 2.6 (Sørensen, 2018).

$$M\dot{\nu} + C_{RB}(\nu)\nu + C_A\left(\nu_r\right)\nu_r + D\left(\kappa, \nu_r\right) + g(\eta) = \tau + \tau_{wind} + \tau_{wave} \tag{2.6}$$

$M = M_{RB} + M_A \in \mathbb{R}^{6x6}$, where M_{RB} and M_A denotes the ridged body inertia matrix and the added mass matrix, respectively. $C_{RB}(\nu)\nu \in \mathbb{R}^{6x6}$ denotes the ridged body Coriolis matrix, and the added mass Coriolis matrix is given as $C_A(\nu_r) \in \mathbb{R}^{6x6}$. The damping matrix consist of a linear and nonlinear part given as $D\left(\kappa, \nu_r\right) = D_L\left(\kappa, \nu_r\right) + D_{NL}\left(\nu_r, \gamma_r\right) \in \mathbb{R}^{6x6}$. $g(\eta) \in \mathbb{R}^{6x6}$ is the restoring matrix, $\nu \in \mathbb{R}^6$. $\nu_r = \nu - \nu_c \in \mathbb{R}^6$ where ν_c is the velocity vector of the fluid. The velocity and relative velocity vector are given in body frame. $\tau \in \mathbb{R}^6$ are the forces and moments produced by the thruster, rudder and fin system. $\tau_{wind} + \tau_{wave}$ is the wind and wave forces. The current effects are included in the relative velocity vector in the nonlinear damping term. Further description can found in Sørensen (2018).

Riged-body Dynamics

M_{RB} and C_{RB} are the rigid body mass and Coriolis matrix. These are expressed in Equation 2.7 (Dukan, 2014).

$$\boldsymbol{M}_{RB}^{CG} = \begin{bmatrix} m\boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_g \end{bmatrix} \tag{2.7}$$

m denotes the mass of the AUV and $\boldsymbol{I_g} \in \mathbb{R}^{3x3}$ the inertia matrix about Center of Gravity (CG) as expressed in Equation 2.8.

$$\boldsymbol{I}_g = \begin{bmatrix} \boldsymbol{I}_x & -\boldsymbol{I}_{xy} & -\boldsymbol{I}_{xz} \\ -\boldsymbol{I}_{yx} & \boldsymbol{I}_y & -\boldsymbol{I}_{yz} \\ -\boldsymbol{I}_{zx} & -\boldsymbol{I}_{zy} & \boldsymbol{I}_z \end{bmatrix} \tag{2.8}$$

The diagonal in Equation 2.8 represent the moment about x_b, y_b and z_b axis. Off diagonal are the product of inertia defined as (Fossen, 2021):

$$\begin{array}{ll} I_x = \int_V \left(y^2 + z^2\right)\rho_m \,\mathrm{d}V; & I_{xy} = \int_V xy\rho_m \,\mathrm{d}V = \int_V yx\rho_m \,\mathrm{d}V = I_{yx} \\ I_y = \int_V \left(x^2 + z^2\right)\rho_m \,\mathrm{d}V; & I_{xz} = \int_V xz\rho_m \,\mathrm{d}V = \int_V zx\rho_m \,\mathrm{d}V = I_{zx} \\ I_z = \int_V \left(x^2 + y^2\right)\rho_m \,\mathrm{d}V; & I_{yz} = \int_V yz\rho_m \,\mathrm{d}V = \int_V zy\rho_m \,\mathrm{d}V = I_{zy} \end{array} \tag{2.9}$$

The rigid body Coriolis and centripetal matrix in CG is expressed in Equation 2.10.

$$\boldsymbol{C}_{RB}^{CG} = \begin{bmatrix} m\boldsymbol{S}\left(\omega_{b/n}^{b}\right) & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & -\boldsymbol{S}\left(\boldsymbol{I}_g\omega_{b/n}^{b}\right) \end{bmatrix} \tag{2.10}$$

To transform the mass and Coriolis matrix to Center of Origin (CO), the transformation matrix $H(r_g^b)$ with respect to the vector from CG to CO is used, as expressed in Equation 2.11 (Fossen, 2021).

$$\boldsymbol{H}\left(\boldsymbol{r}_g^b\right) := \begin{bmatrix} \boldsymbol{I}_3 & \boldsymbol{S}^\top\left(\boldsymbol{r}_g^b\right) \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_3 \end{bmatrix}, \quad \boldsymbol{H}^\top\left(\boldsymbol{r}_g^b\right) = \begin{bmatrix} \boldsymbol{I}_3 & \boldsymbol{0}_{3\times3} \\ \boldsymbol{S}\left(\boldsymbol{r}_g^b\right) & \boldsymbol{I}_3 \end{bmatrix} \tag{2.11}$$

With the given Equation 2.11, M_{RB} and C_{RB} can be transformed to the CO as (Fossen, 2021):

$$\begin{aligned} \boldsymbol{M}_{RB} &:= \boldsymbol{H}^\top\left(\boldsymbol{r}_{bg}^b\right)\boldsymbol{M}_{RB}^{CG}\boldsymbol{H}\left(\boldsymbol{r}_{bg}^b\right) \\ \boldsymbol{C}_{RB} &:= \boldsymbol{H}^\top\left(\boldsymbol{r}_{bg}^b\right)\boldsymbol{C}_{RB}^{CG}\boldsymbol{H}\left(\boldsymbol{r}_{bg}^b\right) \end{aligned} \tag{2.12}$$

Hydrodynamic

The general 6 DOF hydrodynamic added mass matrix is given as (Fossen, 2021):

$$\boldsymbol{M}_A = -\begin{bmatrix} X_{\dot{u}} & X_{\dot{v}} & X_{\dot{w}} & X_{\dot{p}} & X_{\dot{q}} & X_{\dot{r}} \\ Y_{\dot{u}} & Y_{\dot{v}} & Y_{\dot{w}} & Y_{\dot{p}} & Y_{\dot{q}} & Y_{\dot{r}} \\ Z_{\dot{u}} & Z_{\dot{v}} & Z_{\dot{w}} & Z_{\dot{p}} & Z_{\dot{q}} & Z_{\dot{r}} \\ K_{\dot{u}} & K_{\dot{v}} & K_{\dot{w}} & K_{\dot{p}} & K_{\dot{q}} & K_{\dot{r}} \\ M_{\dot{u}} & M_{\dot{v}} & M_{\dot{w}} & M_{\dot{p}} & M_{\dot{q}} & M_{\dot{r}} \\ N_{\dot{u}} & N_{\dot{v}} & N_{\dot{w}} & N_{\dot{p}} & N_{\dot{q}} & N_{\dot{r}} \end{bmatrix} \tag{2.13}$$

The top-bottom and port-starboard symmetry for REMUS 100 imply that the added mass matrix can be reduced to (Prestero, 2001):

$$M_{\mathrm{A}} = \begin{bmatrix} X_{\dot{\mathrm{u}}} & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{\dot{\mathrm{v}}} & 0 & 0 & 0 & Y_{\mathrm{r}} \\ 0 & 0 & Z_{\dot{\mathrm{w}}} & 0 & Z_{\dot{\mathrm{q}}} & 0 \\ 0 & 0 & 0 & K_{\dot{\mathrm{p}}} & 0 & 0 \\ 0 & 0 & M_{\dot{\mathrm{w}}} & 0 & M_{\dot{\mathrm{q}}} & 0 \\ 0 & N_{\dot{\mathrm{v}}} & 0 & 0 & 0 & N_{\mathrm{r}} \end{bmatrix} \tag{2.14}$$

Coriolis and Centripetal Forces

From (Fossen, 2021): *"The hydrodynamic Coriolis and centripetal matrix $\boldsymbol{C_A}$ for a rigid body matrix moving through an ideal fluid can always be parametrized to be skew-symmetric"*:

$$\boldsymbol{C}_A(\boldsymbol{\nu}) = -\boldsymbol{C}_A^T(\boldsymbol{\nu}), \quad \forall \boldsymbol{\nu} \in \mathbb{R}^{6\times1} \tag{2.15}$$

$$\boldsymbol{C}_A(\boldsymbol{\nu}) = \begin{bmatrix} \boldsymbol{0}_{3\times3} & -\boldsymbol{S}\left(\boldsymbol{A}_{11}\boldsymbol{\nu}_1 + \boldsymbol{A}_{12}\boldsymbol{\nu}_2\right) \\ -\boldsymbol{S}\left(\boldsymbol{A}_{11}\boldsymbol{\nu}_1 + \boldsymbol{A}_{12}\boldsymbol{\nu}_2\right) & -\boldsymbol{S}\left(\boldsymbol{A}_{21}\boldsymbol{\nu}_1 + \boldsymbol{A}_{22}\boldsymbol{\nu}_2\right) \end{bmatrix} \tag{2.16}$$

Where A_{ij} is defined as:

$$\boldsymbol{M}_A := \begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{bmatrix} \tag{2.17}$$

Damping

For underwater vehicle, all wave related damping effects are neglected and the main contribution to the damping are vortex shedding and skin friction (Dukan, 2014). The total hydrodynamical damping is the sum of linear and nonlinear damping:

$$\boldsymbol{D}\left(\boldsymbol{\nu}_r\right) = \boldsymbol{D} + \boldsymbol{D}_n\left(\boldsymbol{\nu}_r\right) \tag{2.18}$$

For underwater vehicles moving at high speed, the effect of damping is highly nonlinear (Norgren, 2018). The nonlinear and linear damping matrix is expressed in Equation 2.19 and 2.20, respectively.

$$\boldsymbol{D_n}\left(\boldsymbol{\nu}_\mathbf{r}\right) = -\begin{bmatrix} X_{|\mathrm{u}|\mathrm{u}}\left|u_\mathrm{r}\right| & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{|\mathrm{v}|\mathrm{v}}\left|v_\mathrm{r}\right| & 0 & 0 & 0 & Y_{|\mathrm{r}|\mathrm{r}}|r| \\ 0 & 0 & Z_{|\mathrm{w}|\mathrm{w}}\left|w_\mathrm{r}\right| & 0 & Z_{|\mathrm{q}|\mathrm{q}}|q| & 0 \\ 0 & 0 & 0 & K_{|\mathrm{p}|\mathrm{p}}|p| & 0 & 0 \\ 0 & 0 & M_{|\mathrm{w}|\mathrm{w}}\left|w_\mathrm{r}\right| & 0 & M_{|\mathrm{q}|\mathrm{q}}|q| & 0 \\ 0 & N_{|\mathrm{v}|\mathrm{v}}\left|v_\mathrm{r}\right| & 0 & 0 & 0 & N_{|\mathrm{r}|\mathrm{r}}|r| \end{bmatrix} \tag{2.19}$$

$$\boldsymbol{D} = -\begin{bmatrix} X_u & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_v & 0 & Y_p & 0 & Y_r \\ 0 & 0 & Z_w & 0 & Z_q & 0 \\ 0 & K_v & 0 & K_p & 0 & K_r \\ 0 & 0 & M_w & 0 & M_q & 0 \\ 0 & N_v & 0 & N_p & 0 & N_r \end{bmatrix} \tag{2.20}$$

Hydrostatic and Restoring Force

The forces from the gravitation and buoyancy are refereed to as restoring force. The submerged weight and buoyancy force are presented in Equation 2.21, respectively. The gravitational force acts in CG, while the buoyancy force acts in Centre of Buoyancy (CB).

$$W = mg, \quad B = \rho g \nabla \tag{2.21}$$

The restoring force vector ($\boldsymbol{\eta}$) is expressed in body frame and presented in Equation 2.22 (Fossen, 2021).

$$\boldsymbol{g}(\boldsymbol{\eta}) = \begin{bmatrix} (W-B)\sin(\theta) \\ -(W-B)\cos(\theta)\sin(\phi) \\ -(W-B)\cos(\theta)\cos(\phi) \\ -(y_g W - y_b B)\cos(\theta)\cos(\phi) + (z_g W - z_b B)\cos(\theta)\sin(\phi) \\ (z_g W - z_b B)\sin(\theta) + (x_g W - x_b B)cos(\theta)cos(\psi) \\ -(x_g W - x_b B)\cos(\theta)\sin(\phi) - (x_g W - x_b B)\cos(\theta)\cos(\phi) \end{bmatrix} \tag{2.22}$$

2.1.3 Control Plant Model

The control plant model is a simplification of the process plant model. It is used for controller design and analytical study of stability, such as in the sense of Lyapunov. Different control plant models are necessary for different control objectives and operational regimes of the vehicle. AUVs normally operates below the wave zone, therefore the wave excitation and wind forces can be neglected. The shape of a typical AUV can often be assumed symmetric around the port/starboard and top/bottom. The nonlinear 6 DOF in Equation 2.6 can therefore be reduced to (Fossen, 2021):

$$M\dot{\nu} + C(\nu)\nu + D(\nu)\nu + g(\eta) = \tau \tag{2.23}$$

where M is the mass matrix. Coriolis matrix $C(\nu)$ is computed from M, and the higher order of damping in $D(\nu)$ is neglected. The generalized force vector is given as τ.

2.2 Generalized Forces

The generalized forces that act on underwater vehicle are thruster forces, ocean current and the force from control surfaces. The model for these forces is presented in the following subsections.

2.2.1 Environmental Force

A large proportion of the AUVs mission is submerged, therefore wind and waves are not taking into consideration. The main external environmental force is the ocean current which is considered to be constant and irrotational. The ocean current is expressed in the body frame. The resulting model from Norgren and Skjetne (2015) is presented in Equation 2.24.

$$\begin{aligned} \nu_{\mathrm{c}} &= \begin{bmatrix} u_{\mathrm{c}} & v_{\mathrm{c}} & 0 & 0 & 0 & 0 \end{bmatrix}^{\top} \\ &= \begin{bmatrix} V_{\mathrm{c}}\cos(\psi_{\mathrm{c}}) & V_{\mathrm{c}}\sin(\psi_{\mathrm{c}}) & 0 & 0 & 0 & 0 \end{bmatrix}^{\top} \end{aligned} \tag{2.24}$$

2.2.2 Rudder and Fin

To control the attitude, steering fin and rudder are located at the stern to control the depth and turning, respectively. These are controlled by a feedback control system to ensure the rudder and fin angle becomes as close as possible to the commanded angles. The steering machine containing the rudder angle has two important physical limitations: the maximum rudder angle and rudder rate (Fossen, 2021).

In computer simulation, Van Amerongen (1984) suggested that the maximum rudder angle and rudder rate was specified with certain values. The simplification is based on the assumption that the steering machine dynamics is much faster than the saturated turning rate commands generated by the autopilot. The maximum rudder angle and angle rate for REMUS 100 are:

$$\delta_{R\,\max} = 30(\text{deg}); \quad 10(\text{deg/s}) \leq \dot{\delta}_{R\,\max}$$

The steering fin follow the same assumption and the maximum fin angle and angle rate are:

$$\delta_{S\,\max} = 13.6(\text{deg}); \quad 10(\text{deg/s}) \leq \dot{\delta}_{S\,\max}$$

2.2.3 Control Surfaces

REMUS 100 has two control surfaces: two horizontal fins to control the pitch, and two vertical fins to control the heading. Each surface is controlled as one unit and therefore cannot be controlled individually. The generalized force and moment generated by the rudder and stern fins are presented in Equation 2.25 derived by Prestero (2001). Propeller coefficients for REMUS 100 is estimated by Allen et al. (2000).

$$\begin{aligned}
Y_r &= \tfrac{1}{2}\rho c_{L\alpha} S_{fin}\left[u^2\delta_r - uv - x_{fin}(ur)\right] \\
Z_s &= -\tfrac{1}{2}\rho c_{L\alpha} S_{fin}\left[u^2\delta_s - uw - x_{fin}(uq)\right] \\
M_s &= \tfrac{1}{2}\rho c_{L\alpha} S_{fin} x_{fin}\left[u^2\delta_s - uw - x_{fin}(uq)\right] \\
N_r &= \tfrac{1}{2}\rho c_{L\alpha} S_{fin} x_{fin}\left[u^2\delta_r - uv - x_{fin}(ur)\right]
\end{aligned} \tag{2.25}$$

ρ is the density of the fluid the vehicle is moving in. $c_{L\alpha}$ is the lift coefficient for the fin, S_{fin} is the fin area, and x_{fin} is the distance from the body-fixed origin to the fin position. δ_s and δ_r are the stern fin and rudder angle, respectively, relative to the body's x-axis.

2.2.4 Thrust Force

The force and moment produced by the propeller is given in Equation 2.26 derived by Carlton (2007).

$$\begin{aligned}
X_p &= K_T \rho D^4 n^2 \\
K_p &= K_Q \rho D^5 n^2
\end{aligned} \tag{2.26}$$

Trust and torque coefficients are given as K_T and K_Q, respectively. D is the diameter of the propeller, and n is the revolution per second. Concluding the given information, the generalized force and moments vector is given in Equation 2.27.

$$\boldsymbol{\tau} = \begin{bmatrix} X_p & Y_r & Z_s & M_s & N_r & K_p \end{bmatrix}^T \tag{2.27}$$

2.3 Control System for REMUS 100

2.3.1 Low Level Control System

The control system for REMUS 100 consists of a path planner that uses waypoints to determine the path of the AUV, and a low level control system taking the set-points as input to generate the desired speed, rudder and fin angle. Input to the AUV model is given as three signals: speed, pitch, and heading. The control system is therefore divided into three controllers.

The default control algorithm in the motion control system is a speed, depth and heading controller. The speed controller consists of one PI controller which gives the desired RPM. The depth of the AUV is changed by altering the pitch angle. The depth controller consists of two loops where the first loop generates the desired theta angle from a PI controller taking the depth error as input. The second loop is a PID controller that controls the stern planes, further refereed to as pitch controller. The heading is regulated by a PID controller that controls the rudder angle where the desired heading is produced by ILOS guidance. The general equation for PID controller is shown in Equation 2.28.

$$\boldsymbol{\tau} = -(\boldsymbol{K_P}\tilde{\boldsymbol{\eta}} + \boldsymbol{K_I}\int \tilde{\boldsymbol{\eta}}dt + \boldsymbol{K_D}\dot{\boldsymbol{\eta}}) \tag{2.28}$$

2.3.2 Guidance System

One of the main mission planner for underwater vehicle consists of a set with three-dimensional waypoints, $WP_k = \begin{bmatrix} x_k & y_k & z_k \end{bmatrix}^T$. The heading set-point for the vehicle can be produced with a LOS guidance scheme. This is done by taking the intersection point between the path and a look ahead distance, Δ. The desired course angle would be the sum of path-tangential angle, χ_p, and the path-relative angle, χ_r, presented in Equation 2.29. Figure 2.1 illustrate the LOS guidance scheme (Norgren, 2018).

$$\chi_{\mathrm{d}}(e) = \chi_{\mathrm{p}} + \chi_{\mathrm{r}}(e) \tag{2.29}$$

Path-tangential angle is given in Equation 2.30.

$$\chi_{\mathrm{p}} = \alpha_{\mathrm{k}} = \operatorname{atan2}\left(y_{\mathrm{k+1}} - y_{\mathrm{k}}, x_{\mathrm{k+1}} - x_{\mathrm{k}}\right) \tag{2.30}$$

The last and next waypoint is given as $(x_k,\ y_k)$ and $(x_{k+1},\ y_{k+1})$, respectively. The cross-track error, e, is computed from the current position of the vehicle, $(x(t),\ y(t))$, as

given in Equation 2.31.

$$e(t) = -\left[x(t) - x_{\mathrm{k}}\right] \sin\left(\alpha_{\mathrm{k}}\right) + \left[y(t) - y_{\mathrm{k}}\right] \cos\left(\alpha_{\mathrm{k}}\right) \tag{2.31}$$

The path-relative angle is produced by including tuning parameters, K_p and K_i, to ensure that the vehicle converges to the path.

$$\chi_{\mathrm{r}}(e) = \arctan\left(-K_{\mathrm{p}} e - K_{\mathrm{i}} \int_0^{\mathrm{t}} e(\tau) d\tau\right) \tag{2.32}$$

The tuning parameters represent the same parameters as for a PI-controller. Integral action is used to cancel ocean current. When tuning K_i, care must been taken to avoid overshoot (Fossen, 2011). Set-point for the depth is set to the active waypoint. The speed set-point is chosen by operator and is constant for the whole simulation. When switching to the next waypoint, circle of acceptance, R_{acc}, is often used. When the vehicle enters this circle, the next waypoint is selected. Formula is given in Equation 2.33.

$$\left[x_{\mathrm{k+1}} - x(t)\right]^2 + \left[y_{\mathrm{k+1}} - y(t)\right]^2 \leq R_{\mathrm{acc}}^2 \tag{2.33}$$

R_{acc} should not be chosen to small due to the chance of the vehicle not enter the circle of acceptance, thereby resulting in an unstable guidance scheme (Fossen, 2011).

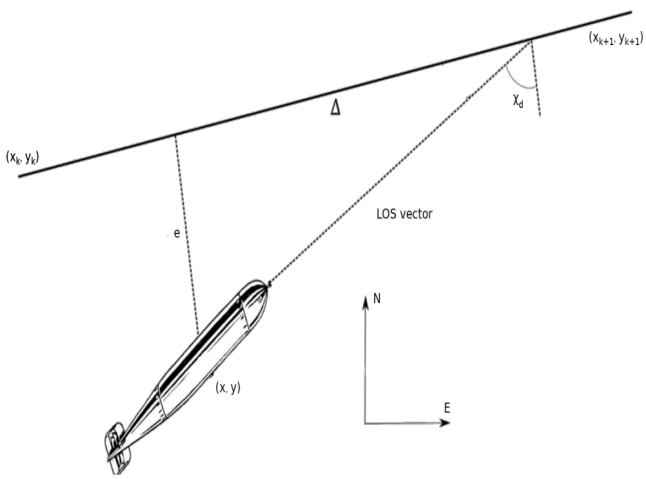


Figure 2.1: Illustration of the LOS guidance scheme (Norgren, 2018).

2.4 Specifications of REMUS 100

Remus 100 is equipped with different sensors. A High Precision Acoustic Positioning (HiPAP) are connected with a LBL transponder mounted on the AUV. The transponder gives the north and east coordinate (x, y) given in the NED frame. A depth sensor measures

the pressure transformed to the depth values, z. The heading (ψ) is measured by the Inertial Measurement Unit (IMU). The DVL measures the surge and sway velocity (u, v) relative to the body frame. r is the yaw rate provided by the IMU. The full list of sensors are given in Table 2.2.

Sensors
Oxygen Optode Sensor (Aanderaa 4831)
Neil Brown G-CTD Sensor (NBOSI)
ECO Puck (WetLabs Triplet)
LBL High Frequency Transducer
IMU (Honeywell HG1700AG58 with NavP)
ADCP/DVL (TD Explorer R100)
Sidescan Sonar (MSTL SF 900 kHz)
GPS
Iridium modem
Wi-Fi capabilities

Table 2.2: Sensors for REMUS 100 (Hydroid, 2012).

The specifications and physical characteristic for REMUS 100 are given in Table 2.3.

Physical/functional characteristics	
Vehicle Diameter	19 cm
Weight in air	31 kg
Operating Depth Range	3 m to 100 m
Speed Range	0.25 m/s to 2.57 m/s
Maximum Operating Water Current	1 m/s
Typical Endurance	4 hours @ 4 knots
	5 hours @ 3 knots

Table 2.3: REMUS 100 specifications (Hydroid, 2012).

2.5 Introduction to Machine Learning

Over the last years, the field of machine learning have received increased interest. Modern processors have become more powerful and the density to performance ratio has improved dramatically. The cost of storing and managing large amounts of data has decreased, and the ability to distribute compute processing across cluster of computers, are some of the main reasons for the enormous interest for machine learning (Sørensen, 2020).

2.5.1 Different Methods of Machine Learning

Within the field of machine learning there is multiple classes or methods. These are presented in Figure 2.2.

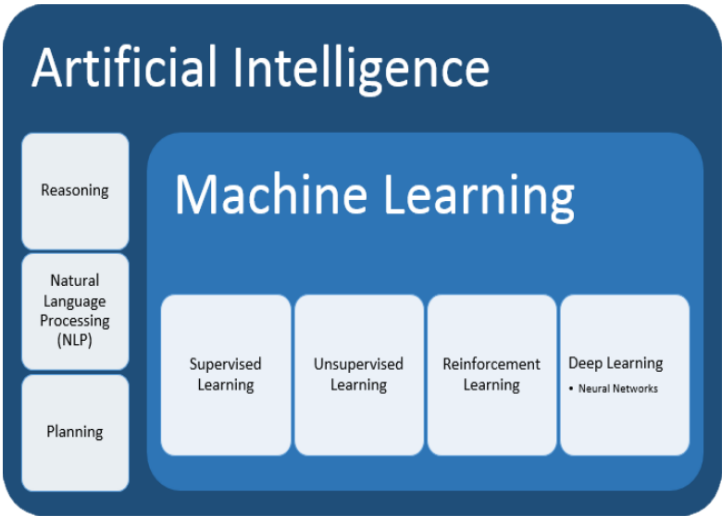


Figure 2.2: Artificial intelligence and different methods of machine learning (IBM Cloud Education, 2020).

A brief description of different methods is listed below.

- Supervised Learning: Uses a labeled dataset and needs to be supervised with labeled data.
- Unsupervised Learning: Does not need any supervisory, is independent and works on its own.
- Reinforcement Learning: Learns through trails and errors, and there is no labeled dataset.
- Deep Learning: Based on neural networks where it uses an iterative method to learn. This method is useful when you have an unstructured dataset.

For this thesis, the method of Supervised Learning is used. Supervised Learning can be divided into classification and regression. Regression is used when estimating the numeric value of the current speed, and classification when assigning a class to the estimated current direction.

2.5.2 Regression

Regression predicts a numeric value of the dependent variable with respect to the independent variables. Regression can again be divided into several algorithms. The main algorithms concluding supervised regression will be described in the following subsection.

Linear Regression

For Linear Regression, the aim is to predict a relationship between a dependent variable (y) and one single independent variable (x). Linear Regression is given in Equation 2.34. β_0 is the intercept constant value where intersection between y-axis and the predicted line takes place. β_1 is the corresponding coefficient to the independent variable x.

$$y_n = \beta_0 + \beta_1 x_1 \tag{2.34}$$

Multiple Linear Regression

Multiple Linear Regression is an extension of the Linear Regression. It contains more than one independent variable as illustrated in Equation 2.35, where β_n correspond to the independent variable x_n. Because it is assumed that the dependent variable is directly related to a linear combination of the independent variables, is it thereby called Multiple Linear Regression (Tranmer et al., 2020).

$$y_n = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n \tag{2.35}$$

Polynomial Regression

The difference between Polynomial Regression and Multiple Linear Regression is the independent variable (x) which is in the n'th power. In other words, it will fit when there are nonlinear relations between the dependent and independent variable. Equation 2.36 present the Polynomial Regression algorithm.

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \cdots + \beta_n x^n \tag{2.36}$$

Support Vector Regression

Support Vector Regression (SVR) is an algorithm where the aim is to find a line that best fits the data. The best fitted line is a hyperplane that contains most data points. A kernel is used to find the hyperplane. An advantage of SVR is that dimensionality of the input space does not affect the computational complexity. Further, SVR show excellent generalization capability with high prediction accuracy (Awad and Khanna, 2015).

Linear Support Vector Regression is based on insensitive tube. The algorithm disregard the error inside the tube while focusing on minimizing the distance of the slack variable outside the tube. This is illustrated in Figure 2.3 (Eremenko et al., 2015).

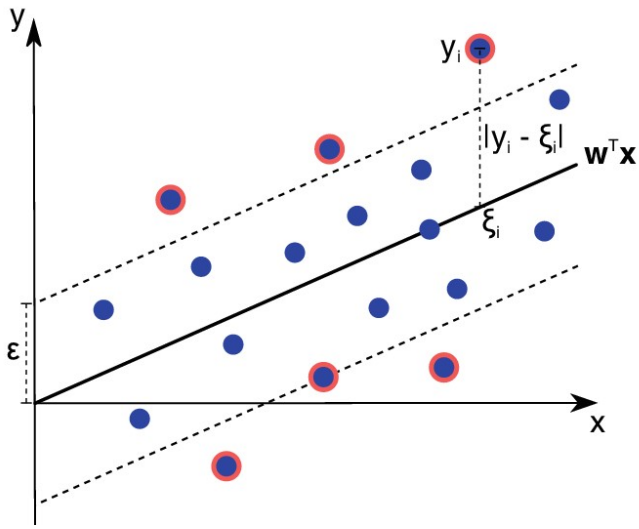


Figure 2.3: Linear Support Vector Regression (Rosenbaum et al., 2013).

Linear one-dimension SVR can be written as stated in Equation 2.37.

$$y = f(x) =< w, x > +b = \sum_{j=1}^{M} w_j x_j + b, y, b \in \mathbb{R}, x, w \in \mathbb{R}^M \tag{2.37}$$

By extending multidimensional data, x is augmented by one. By including b in the w vector to simply the mathematical notation, the Multivariate Regression is obtained (Awad and Khanna, 2015) presented in Equation 2.38.

$$f(x) = \begin{bmatrix} w \\ b \end{bmatrix}^T \begin{bmatrix} x \\ 1 \end{bmatrix} = w^T x + b \quad x, w \in \mathbb{R}^{M+1} \tag{2.38}$$

Decision Tree Regression

Decision Tree Regression builds algorithms in the structure of a tree. It estimates by asking several questions where the answer is either True or False. The order of questions and their content is determined by the algorithm. An example of the structure in a pseudo-code manner is illustrated in Figure 2.4.

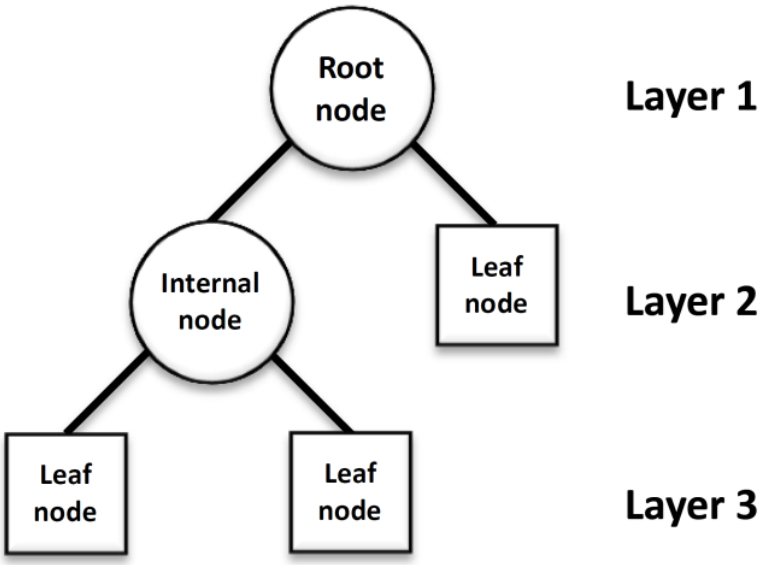


Figure 2.4: Illustration of the decision tree structure given as a pseudo-code (Chiu et al., 2016).

Random Forest Regression

Random Forest Regression is an extension of the Decision Tree Regression. It contains several decisions trees, where each are trained by different datasets with own predictions. A single output is then produced by the average of all predictions (Mwiti, 2020).

2.5.3 Classification

Classification assigned a label to the dependent variable. The labels are defined by the training set. An example of classification is to give the algorithm several pictures where it should tell if the picture contains a car or a train. Classification cloud be divided into several algorithms, where the main ones are described in the following subsections. Table 2.4 presents the algorithms and user defined parameters.

Algorithm	**User-defined parameters**
k-Nearest neighbour	Number of neighbours considered (k)
Support Vector Machine (SVM)	Cost or slack parameter (C)
Kernel SVM	Kernel type Kernel-dependent parameters: Polynomial: polynomial order (p) Radial basis: gamma (γ)
Naive Bayes	Priors(n_classes,): Prior probabilities of the classes. Var_smoothingfloat (default=1e-9): Portion of the largest variance of all features that is added to variances for calculation stability.
Decision Tree Classification	Pruning parameters (cp)
Random Forest Classification	Number of trees (n) Number of variables randomly sampled as candidates at each split (m)

Table 2.4: Classification algorithms and user defined parameters

k-Nearest Neighbor

The k-Nearest Neighbor (KNN) does not produce an algorithm, it simply compares the unknown data to the original training data. The unknown data is assigned to the k training sample that are nearest in the feature space to the unknown sample (Maxwell et al., 2018). The complexity of the boundary layer depends on the k number, where a higher number result in greater generalization, and a lower number will produce a complex decision boundary.

Support Vector Machine

Support Vector Machine (SVM) aims to find the optimal boundary between the support vectors. SVM are binary and distinguish between two classes. For classes that are inherently not separable, soft margin could be introduced. This will lead to training points being allowed to be on the inside of the boundary line. The feature is regulated with the parameter C. Higher C values will result in a more complex decision boundary, and less generalization (Maxwell et al., 2018).

Kernel Support Vector Machine

At first, SVM was designed to identify class boundary. Under the assumption that a linear boundary may exists in a higher dimensional feature space, the kernel trick could be used to project to a higher dimensional feature space. There exists several kernels where the most common is exponential and radial basis function kernels (Maxwell et al., 2018).

Naive Bayes

The Naive Bayes Classification is based on the Bayesian theorem. It calculates the probability for future events based on previously result. Naive Bayes Classification is well suited when the dimensionality of the inputs are high (Islam et al., 2007). Bayes theorem are given in Equation 2.39.

$$P(h \mid D) = \frac{P(D \mid h)P(h))}{P(D)} \tag{2.39}$$

Where
$P(h)$: Prior probability of hypothesis h- Prior
$P(D)$: Prior probability of training data D-Evidence
$P(D|h)$: Probability of D given h- Likelihood
$P(h|D)$: Probability of h given D- Posterior probability

The Naive Bayes Classification assumes that observation of different events are the product of the probability of each individual events (Rish, 2001).

Decision Tree Classification

Decision Tree Classification is based on the same concept as described in Decision Tree Regression (2.5.2). A classification tree represents classes, while the regression tree represents numerical values. This method gives an intuitive understanding on how the algorithm split the data. The classification is computational fast due to low mathematical complexity. Problems with decision tree include the possibility of generating a non-optimal solution and overfitting. To avoid overfitting, one possibility is to remove one layer of branches to reduce the accuracy of the classifier (Maxwell et al., 2018).

Random Tree Classification

Random Tree Classification is an ensemble classifier where it uses multiple decision trees to compensate for the weakness of one single tree. The majority "vote" of all trees is used to assign a final class for each unknown (Maxwell et al., 2018). When reducing the training data and variables, each individually tree will be less accurate. However, when combining the trees, it will become more reliable.

2.6 How to Perform Machine Learning

If a machine learning algorithm is to perform in the best way, the input data must be of high quality. This means that the data source needs to be accurate and meaningful when combined (dependent and independent variable) with each other, such that the model becomes accurate and trustworthy (Hurwitz and Kirsch, 2018). The dataset also needs to be cleaned and transformed in such way that the machine learning algorithm can understand it.

The machine learning cycle is a continues process and the algorithm must be updated once a while. The cycle of machine learning are presented in Figure 2.5 starting from identifying the data. For more information see Hurwitz and Kirsch (2018).

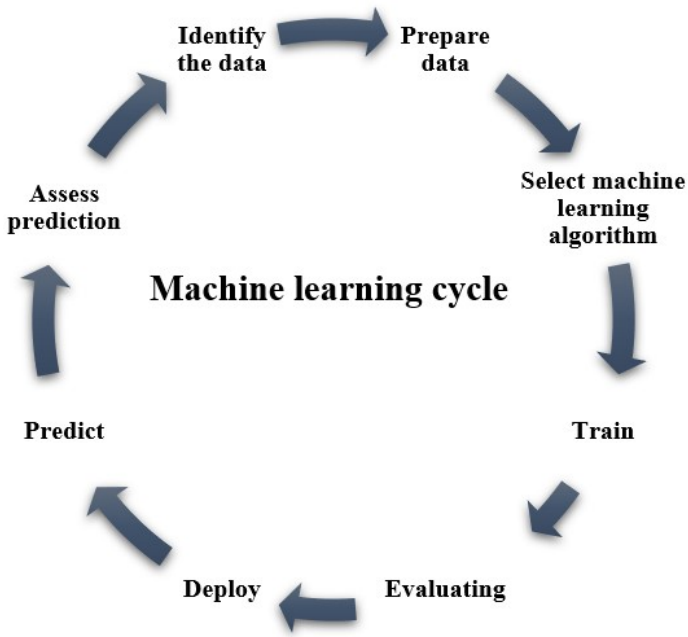


Figure 2.5: The machine learning cycle starting from identifying the data.

When training a machine learning algorithm, the process can be split into three steps (Hurwitz and Kirsch, 2018):

- Representation: To achieve the desired results, the algorithm creates a model to transform the data. The algorithm will begin to learn the relationship between the raw data and which data points are strong predictors for the desired outcome, as the learning algorithm is exposed to more data.
- Evaluation: When multiple models are created, the human or the algorithm needs to evaluate and score the models. The evaluation is done based on which model

produces the most accurate predictions. After the model is operationalized, it will be exposed to unknown data. Therefore, it is important to make sure the model is generalized and not overfitted to the training data.

- Optimization: The best performing algorithm created, is selected. Further, as the algorithm is exposed to diverse sets of input data, the most generalized model is selected.

2.6.1 Validation of Data

It is important that the data used in machine learning is verified with regard to accuracy and context to make the model accurate and trustworthy. This is a part of the machine learning cycle of preparing the data. The human need to make sure that the dataset is making sense when independent and dependent variables are combined. Sometimes datasets are given with letters or names, but the machine learning algorithm needs to have the dataset in a manner of numbers. Therefore, the algorithm uses a method called data preprocessing. This is done by implementation of sickit-learn library in Python. A general way of data preprocessing can be divided into three methods: replacing missing data, encoding and feature scaling. If the dataset has missing data, the data can be replaced by the average, the median or the most frequent data. The most commonly used, is to replace the missing data with the average (Eremenko et al., 2015).

In Figure 2.6a an example of a dataset with animals and age is given as the independent variables and raw food is given as the dependent variable. This could represent a case where a raw food production wants to develop a machine learning algorithm to predict if a given animal eats raw food by stating the type of animal and age. To implement this dataset into a machine learning algorithm, one hot encoding and binary encoding has to be done. The animal category uses one hot encoding to turn them into vectors and the dependent variable, raw food, has to be encoded to a binary, see formulation below:

$$Dog = [1\ 0\ 0] \quad Cat = [0\ 1\ 0] \quad Lion = [0\ 0\ 1]$$

$$Yes = 1 \quad No = 0$$

Figure 2.6 illustrates an example of dataset given as strings (Figure 2.6a) manipulated into dataset of integer (Figure 2.6b). In this way, the machine learning algorithm is able to use the data.

Animal	Age	Raw food
Dog	10	Yes
Cat	4	No
Dog	2	No
Lion	22	No
Dog	14	Yes

(a) Dataset with strings

Animal	Age	Raw food
[1 0 0]	10	1
[0 1 0]	4	0
[1 0 0]	2	0
[0 0 1]	22	0
[1 0 0]	14	1

(b) Dataset with integer

Figure 2.6: Example of dataset given as strings (a) manipulated into dataset given as integer (b).

The last method for data preprocessing is feature scaling, which scales the features in the same scale. This prevents one feature to dominate others. If this is not implemented, the algorithm could weight the higher values more than the lower values.

2.6.2 Evaluation of Models

When the model is trained, it is important to evaluate the performance before deploying it. Caruana and Niculescu-Mizil (2006) describes eight evaluation matrices divided into three subgroups: threshold metrics, ordering/rank metrics and probability metrics. Threshold matrices consist of accuracy (ACC), F-score and lift. Ordering/rank matrices consists of area under the ROC curve (ROC), average precision (APR), and precision/recall breakeven point (BEP). The probability matrices consist of squared error (RMS) and cross-entropy (MXE). See Caruana and Niculescu-Mizil (2006) and Caruana and Niculescu-Mizil (2004) for more information on evaluation matrices.

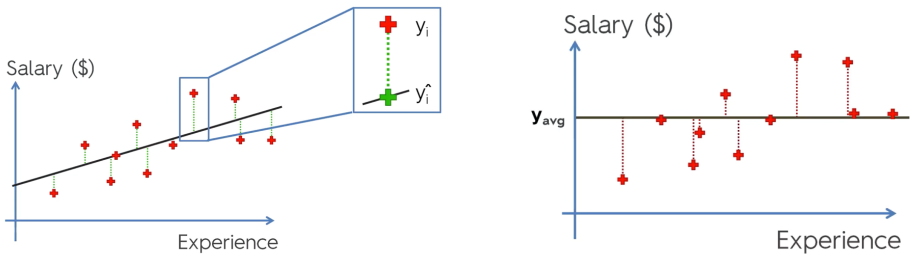
A validation method called R-squared, sates the proportion of the variance in the dependent variable that is predictable from the independent variable (Eremenko et al., 2015). R-squared produce a value between 0-1. The closer the method is to 1, the better the estimated line is. This means, if the R-squared equals one, the model accounts for 100% of the variance by going through all the training points and perfectly explains the observed data points. If R-squared equals zero, the model accounts for 0% of the variance by going through none of the training points and therefore explains precisely nothing about the observed data. The mathematical formulation of the R-squared method is given in Equation 2.40 as a function of the residual sum of squared errors (SS_{red}) and the total sum of squared errors (SS_{tot}).

The sum of residual is presented in Equation 2.41. Figure 2.7a illustrate an example of a linear regression model that visualize the training points in red and the estimated line in black. The total sum of squared error is presented in Equation 2.42. Figure 2.7b illustrates the total sum of squared error by using the same example as in Figure 2.7a. The black line is the average value of the training set and are noted as y_{avg}.

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{2.40}$$

$$SS_{res} = (y_i - \hat{y}_i)^2 \tag{2.41}$$

$$SS_{tot} = (y_i - y_{avg})^2 \tag{2.42}$$



(a) Sum of residual, SS_{res}.

(b) Total sum of squared error, SS_{tot}.

Figure 2.7: Illustration example of R-squared method as a function of (a) the sum of residual SS_{res} and (b) the total sum of squared error SS_{tot} (Eremenko et al., 2015).

Another validation method is the Root Mean Squared Error (RMSE). It takes the square root of the second power of the error between predicted and observed variable of interest. RMSE is showed in Equation 2.43. A perfect fit would give an RMSE score of 0.

$$\sqrt{mean((y - \hat{y})^2)} \tag{2.43}$$

To visualize the performance of classification models, confusion matrices can be used (Figure 2.8). The matrices compare the wrongly classified data with the correct ones. This is a nXn matrix where n is the number of classes defined for the dataset (Choudhary and Gianey, 2017).

		Predicted	
		0	1
Actual	0	**TN**	**FP**
	1	**FN**	**TP**

Figure 2.8: Example of confusion matrix.

From the confusion matrix in Figure 2.8, the corresponding rates of each cell can be calculated as (Choudhary and Gianey, 2017):

True negative (TN): The proportion of the negative cases.

$$\frac{TN}{TN + FP} \tag{2.44}$$

False Positive (FP): The proportion of negative cases that were falsely classified as positive.

$$\frac{FP}{TN + FP} \tag{2.45}$$

False Negative (FN): The proportion of positive cases that were falsely classified as negative.

$$\frac{FN}{FN + TP} \tag{2.46}$$

True Positive (TP): The proportion of positive cases that were correctly classified.

$$\frac{TP}{FN + TP} \tag{2.47}$$

Given the confusion matrix, the accuracy score (ACC) can be found. ACC score denotes proportion of correct predictions calculated as:

$$ACC = \frac{TN + TP}{TN + FP + FN + TP} \tag{2.48}$$

2.6.3 Overfitting and Underfitting

Over- and underfitting are phenomenons that the algorithm needs to account for. Overfitting occur when the model is trained to well on a specific training dataset. When the model makes new predictions, the estimation is poorly. An example of overfitting is when predictions of the training dataset has a high accuracy (e.g. 99%) and the test set gives a lower accuracy (e.g. 55%). Underfitting occur when the model cannot predict the training dataset, or make new predictions on unseen data. When choosing the right model, a good balance between over- and underfitting is needed, a task that is hard to perform.

There are several methods to find the sweet spot between over- and underfitting. For instance, cross-validation is a commonly used method. It splits the dataset into multiple training and testsets. The model will be trained on each individual trainingset, and validated on the corresponding testset unseen for the model (Hurwitz and Kirsch, 2018). Since cross-validation uses multiple trainingsets, it require more computational power compared to for example holdout method. The holdout method simply splits the dataset into a trainingset and a testset. For the holdout method, the split is often set to 80%, which correspond to a testset of 20%. As this method requires less computational power, the method is useful when large dataset is present.

Chapter 3

Method

This chapter presents and describes the methods used to produce the results in Chapter 4. Firstly, the simulation environment is described in Section 3.1. Further, a description of the vertical ILOS algorithm developed in depth controller follows in Section 3.2. Lastly, the method for producing the current estimation algorithms is described in Section 3.3.

3.1 AUVsim

AUVsim is a simulator developed by Norgren and Skjetne (2015) to replicate as close as possible the real system of REMUS 100. The simulator is made in Matlab/Simulink. Simulink provide graphical modulation and practical simulation, where the result can be extracted to Matlab for further analysis. The main modules are guidance, controller and AUV model. These has been modified to best fit the scope of this thesis. The original model was made in discrete time. This has been altered to make the model run in time-continues. The main modules are presented in Figure 3.1.

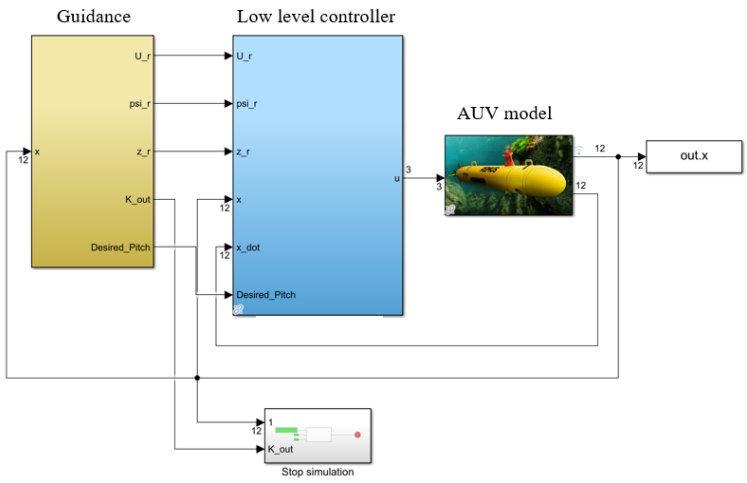


Figure 3.1: Structure of AUVsim consisting of AUV model, low level controller and guidance layer, extracted from Simulink.

The AUV model describes the dynamics and kinetics of REMUS 100 which can be mathematically written as:

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}_{\boldsymbol{\Theta}}(\boldsymbol{\eta})\boldsymbol{\nu} \tag{3.1}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}_{\boldsymbol{RB}}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{C}_{\boldsymbol{A}}(\boldsymbol{\nu}_{\boldsymbol{r}})\boldsymbol{\nu}_{\boldsymbol{r}} + \boldsymbol{D}(\boldsymbol{\nu}_{\boldsymbol{r}})\boldsymbol{\nu}_{\boldsymbol{r}} + \boldsymbol{g}(\boldsymbol{\eta}) = \boldsymbol{\tau} \tag{3.2}$$

The shape of REMUS 100 is assumed to be symmetric around top/bottom and port/starboard. This leads to the following mass inertia matrix:

$$\boldsymbol{M} = \begin{bmatrix} 31.4091 & 0 & 0 & 0 & .5974 & 0 \\ 0 & 65.9791 & 0 & -0.5974 & 0 & -1.9300 \\ 0 & 0 & 65.9791 & 0 & 1.9300 & 0 \\ 0 & -0.5974 & 0 & 0.2591 & 0 & 0 \\ 0.5974 & 0 & 1.9300 & 0 & 8.3417 & 0 \\ 0 & -1.9300 & 0 & 0 & 0 & 8.3300 \end{bmatrix} kg \tag{3.3}$$

Added mass Coriolis matrix is simplified to a skew-symmetric matrix, due to low speed assumption and symmetric properties. The coefficients are calculated by Prestero (2001) and can be found in Appendix A. Equation 3.4 presents the simplified added mass Coriolis matrix.

$$\boldsymbol{C}_{\boldsymbol{A}}(\boldsymbol{\nu}_{\boldsymbol{r}}) = \begin{bmatrix} 0 & 0 & 0 & 0 & -Z_{\dot{w}}w & Y_{\dot{v}}v_r \\ 0 & 0 & 0 & Z_{\dot{w}}w & 0 & -X_{\dot{u}}u_r \\ 0 & 0 & 0 & -Y_{\dot{v}}v_v & X_{\dot{u}}u_r & 0 \\ 0 & -Z_{\dot{w}}w & Y_{\dot{v}}v_r & 0 & -N_{\dot{r}}r & M_{\dot{q}}q \\ Z_{\dot{w}}w & 0 & -X_{\dot{u}}u_r & N_{\dot{r}}r & 0 & -K_{\dot{p}}p \\ -Y_{\dot{v}}v_r & X_{\dot{u}}u_r & 0 & -M_{\dot{q}}q & K_{\dot{p}}p & 0 \end{bmatrix} \tag{3.4}$$

When modeling the damping matrices, linear second order (or higher) and coupled damping terms, are neglected. For underwater vehicles moving at high speed, the effect of damping is highly nonlinear (Norgren, 2018). The resulting damping matrix is expressed in Equation 3.5.

$$D(\nu_{\mathrm{r}}) = -\begin{bmatrix} X_{|\mathrm{u}|\mathrm{u}}|u_{\mathrm{r}}| & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{|\mathrm{v}|\mathrm{v}}|v_{\mathrm{r}}| & 0 & 0 & 0 & Y_{|\mathrm{r}|\mathrm{r}}|r| \\ 0 & 0 & Z_{|\mathrm{w}|\mathrm{w}}|w_{\mathrm{r}}| & 0 & Z_{|\mathrm{q}|\mathrm{q}}|q| & 0 \\ 0 & 0 & 0 & K_{|\mathrm{p}|\mathrm{p}}|p| & 0 & 0 \\ 0 & 0 & M_{|\mathrm{w}|\mathrm{w}}|w_{\mathrm{r}}| & 0 & M_{|\mathrm{q}|\mathrm{q}}|q| & 0 \\ 0 & N_{|\mathrm{v}|\mathrm{v}}|v_{\mathrm{r}}| & 0 & 0 & 0 & N_{|\mathrm{r}|\mathrm{r}}|r| \end{bmatrix} \tag{3.5}$$

Equation 3.6 describes the CG and CB, respectively.

$$\boldsymbol{r}_g^b = \begin{bmatrix} 0 \\ 0 \\ 0.0196 \end{bmatrix}, \quad \boldsymbol{r}_b^b = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.6}$$

By taking the distance between CG and CB, and calculating the restoring force from equation 2.22, the $\boldsymbol{g}(\boldsymbol{\eta})$ becomes:

$$\boldsymbol{g}(\boldsymbol{\eta}) = \begin{bmatrix} (W-B)\sin(\theta) \\ -(W-B)\cos(\theta)\sin(\phi) \\ -(W-B)\cos(\theta)\cos(\phi) \\ z_g W\cos(\theta)\sin(\phi) \\ z_g W\sin(\theta) \\ 0 \end{bmatrix} \tag{3.7}$$

The resulting environmental model from Norgren and Skjetne (2015) presented in Equation 2.24, has been modified to include a vertical component that acts perpendicular to the surface. The ocean current velocity vector is than given as:

$$\begin{aligned} \nu_{\mathrm{c}} &= \begin{bmatrix} u_{\mathrm{c}} & v_{\mathrm{c}} & w_{\mathrm{c}} & 0 & 0 & 0 \end{bmatrix}^\top \\ &= \begin{bmatrix} V_{\mathrm{c}}\cos(\psi_{\mathrm{c}}) & V_{\mathrm{c}}\sin(\psi_{\mathrm{c}}) & w_{\mathrm{c}} & 0 & 0 & 0 \end{bmatrix}^\top \end{aligned} \tag{3.8}$$

Equations 3.3 - 3.8 are implemented into the AUV model in Simulink, presented in Figure 3.2. The generalized fore and moment vector is calculated as stated in Equation 2.25 and 2.26 described in Section 2.2. The coefficients can be found in Appendix A.

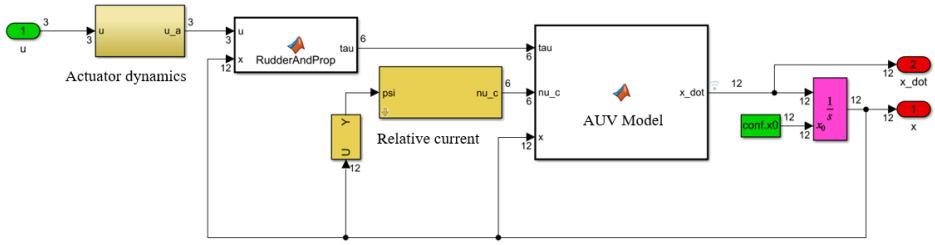


Figure 3.2: Structure of the AUV model extracted from Simulink.

3.1.1 Low Level Control

Low level control system for REMUS 100 is previous described in Section 2.3.1. The tuning of each controller are found by trial and error, and the resulting gains are visualized in Table 3.1.

	Controller			
	Pitch	Depth	Heading	Shaft Speed
P	0.8	0.05	0.9	200
I	0.1	0.003	0.05	60
D	0.9	0	1	0

Table 3.1: Gains for different controllers

The main controllers are presented in Figure 3.3, where the produced outputs are speed in RPM, and heading and pitch angle in radian.

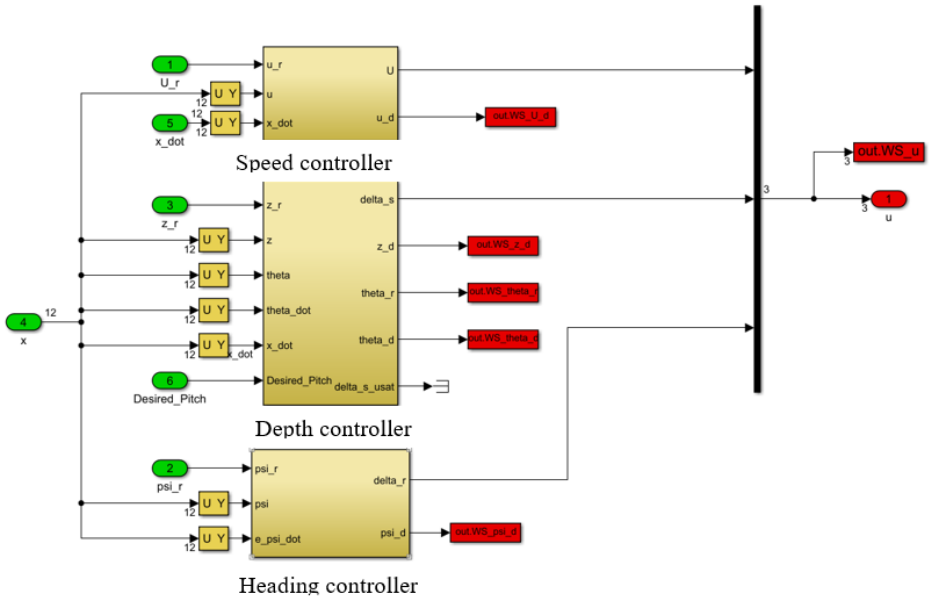


Figure 3.3: Structure of Low Level Control extracted from Simulink.

3.1.2 Guidance System

The guidance system takes the predefined waypoints consisting of north, east and depth coordinates, as input. These are defined in *conf.wplist* and can be found in the attached folder. Further, the guidance system calculates the desired heading (ψ) and theta (θ) with respect to the current state $\boldsymbol{x}$. Figure 3.4 presents the implementation in Simulink.

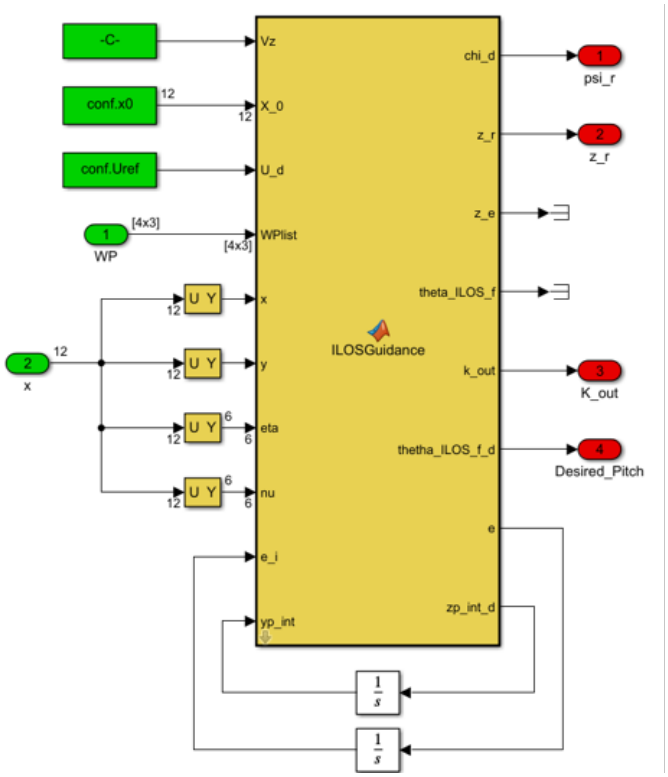


Figure 3.4: Structure of guidance system extracted from Simulink.

To make the guidance system stable when switching between points in *wplist*, the circle of acceptance, previously described in Section 2.3.2, is chosen as $R_v = 15$ and $R_z = 5$. This represents the accepted horizontal and vertical error, respectively.

3.2 Depth Controller

The aim for this depth controller is to counteract vertical current speed. This is done by combining a vertical ILOS algorithm with the existing pitch controller in AUVsim. The ILOS algorithm produce the desired theta angle (θ_d), which is send as the desired set-point to the pitch controller. The ILOS algorithm for the vertical plane is an extension of the horizontal LOS scheme described in Fossen (2011). Figure 3.5 illustrates the concept of both vertical and horizontal LOS scheme.

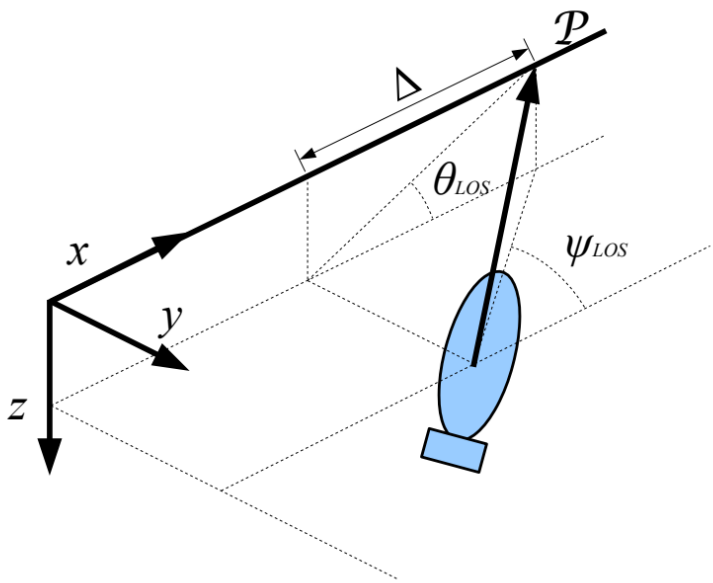


Figure 3.5: Line of sight (LOS) guidance scheme (Caharija et al., 2012).

The vertical cross-track error, z_p, is calculated from the current position of the AUV, $(x(t),\ z(t))$, with respect to the current waypoints $(x_k,\ z_k)$ given as:

$$z_e(t) = -\left[x(t) - x_{\mathrm{k}}\right]\sin\left(\alpha_{\mathrm{z}}\right) + \left[z(t) - z_{\mathrm{k}}\right]\cos\left(\alpha_{\mathrm{z}}\right). \tag{3.9}$$

where the path tangential angle is given as:

$$\alpha_{\mathrm{z}} = \operatorname{atan}2\left(-(z_{\mathrm{k+1}} - z_{\mathrm{k}}), (x_{\mathrm{k+1}} - x_{\mathrm{k}})\right) \tag{3.10}$$

and the resulting desired theta is fund by:

$$\theta_d = \alpha_{\mathrm{z}} + \tan^{-1}\left(K_p z_e^p + K_i z_{\mathrm{int}}^p\right) \tag{3.11}$$

By integrating Equation 3.12, z_{int}^p is found.

$$\dot{z}_{\mathrm{int}}^p = \frac{\Delta z_e^p}{\sqrt{\Delta^2 + \left(z_e^p + \kappa z_{\mathrm{int}}^p\right)^2}} \tag{3.12}$$

Kp and Ki is found by altering the lookahead distance and scaling parameter k according to Fossen (2011). $\Delta = 12$ and $k = 0.115$ is chosen, and the gains are calculated as:

$$K_p = \frac{1}{\Delta},\ K_i = k \cdot K_p \tag{3.13}$$

3.3 Current Estimation

In this section the method for current estimation is presented. Supervised Learning methods need to be trained on datasets to be able to predict values. The generation of the trainingset and validation of the method are described in subsection 3.3.1 and 3.3.2, respectively. Further, the development and training of regression and classification models are respectively described in subsection 3.3.3 and 3.3.4.

3.3.1 Dataset

The result of machine learning models depends heavily on the trainingdata. For instance, if the dataset contains wildpoints, too little datapoints, or the structure of data contain values outside the operating regime, the model will lead to poor prediction. When generating the dataset, care must be taken to avoid producing to large trainingsets resulting in long training time. The simulation is automatically stopped when the last waypoint is reached, to avoid drift in trainingdata. In addition, the first 1000 datapoints is removed to avoid spikes and data that are not correlated to operating state of the AUV.

Two dataset, one for estimating the current speed and one for estimating the current direction, needs to be produced to be able to train the regression and classification models. These will be generated through Matlab by running a for-loop with various simulation parameters, such as operating speed of AUV (Uref), current speed and direction. For the complete code see the attached *Make_csv_file.m*.

The dataset for current speed is generated by saving output parameters of the simulation in multiple *csv* files consisting of RPM, rudder and fin angle, actual speed of the AUV, and current speed. In this set, the current speed is the dependent variable while the remaining are independent variables. The operating speed for REMUS 100 is set to between 1.0 to 2.4 m/s with increment of 0.1. The current speed is set between 0.1 to 0.8 m/s, with increment of 0.02.

The dataset for current direction includes the same independent variables as for current speed. The dependent variable in this dataset is, however, current direction. The procedure is the same as described in the previous paragraph for current speed. Simulations are run by altering the operating speed between 1.0 to 2.4 m/s with increment of 0.2, current speed between 0.1 to 0.8 m/s with increment of 0.1, and current direction from 0 to 360 with increment of 15 degrees.

3.3.2 Validation

R-squared and RMSE method described in Section 2.6.2 are used to evaluate regression models developed in both Python and Matlab. When the model is trained in Matlabs classification app, the standard validation is given as an Accuracy as stated in Equation 2.48.

3.3.3 Regression

When the dataset is generated, the data can be deployed to each algorithm and training can begin. Various platforms with different programming languages can be used to develop algorithms. One of the most used languages for machine learning, is Python, which has a library specific developed for machine learning called scikitlearn. The Regression models aim is to predict the current speed. Therefore, dataset with current speed as the dependent variable is used. The equation-based method is developed in Python where the regression coefficient and intercept values are extracted and implemented into Matalab. See attached *current_dir_estimation.m* folder for the implementation of Polynomial and Multiple Linear Regression equations. For more sophisticated regression models, such as Decision Three, Matlabs regression app are used. Holdout validation was chosen where 80% was used as trainingset. An overview of the algorithms with RMSE score are presented in Table 3.2 sorted from highest to lowest values.

Regression Models	RMSE
Linear regression	Not valid
Support vector regression	0.017595
Multiple linear regression	0.014561
Coarse tree regression	0.011655
Medium tree regression	0.011532
Fine tree regression	0.011495
Polynomial regression	0.000724

Table 3.2: Overview of tested regression models with RMSE score. A perfect fit would give an RMSE score of 0.

When training phase is done, models need to be exported so prediction on new data can take place. Models generated in Matlab are saved as $.mat$ files (see attached folder) where model parameters are stored. Python code for Polynomial and Multiple Linear Regression can be found in Appendix B.

3.3.4 Classification

Classification algorithms are developed in Matlabs classification app. The trainingdata consist of current direction as dependent variable. The aim for the classification algorithm is to predict the current direction and categorize it to a label specified by trainingdata.

Before starting the training session in the app, protection for overfitting must be chosen. Here, holdout validation is chosen to reduce the training time due to the size of the datasets. The trainingset was chosen to 80%. Table 3.3 presents an overview of tested classification models with accuracy score sorted from lowest to highest values.

Classification Models	Accuracy
Gausian Naive Bayes Classification	17.0%
Medium Tree Classification	17.6%
Fine Tree Classification	23.9%
Weighted KNN Classification	87.9%
Fine KNN Classification	88.5%

Table 3.3: Overview of tested Classification models with Accuracy score.

When training phase is done, models need to be exported so prediction on new data can take place. Models generated in Matlab are saved as $.mat$ files (see attached folder) where model parameters are stored.

3.4 Simulation Cases

Table 3.4 presents simulation cases with various regression and classification models. Each case number has been categorized by R or C corresponding to simulation of regression and classification, respectively. R1, R2 and R3 corresponds to different current speed, and C1, C2, C3 and C4 corresponds to different current directions. $Uref$ denotes desired speed of REMUS 100.

Regression			
Case nr.	Current speed [m/s]	Current Direction[°]	Uref [m/s]
R1	0.25	0	1.6
R2	0.5	0	1.6
R3	0.75	0	1.6
Classification			
Case nr.	Current speed [m/s]	Current Direction[°]	Uref [m/s]
C1	0.25	0	1.6
C2	0.25	80	1.6
C3	0.25	180	1.6
C4	0.25	270	1.6

Table 3.4: Overview of simulation cases with corresponding parameters for current speed and direction, and desired speed of REMUS 100 (Uref). R and C corresponds to regression and classification simulations, respectively. Numeration corresponds to various current speed and direction.

Chapter 4

Result

This chapter firstly presents limitations regarding machine learning in Section 4.1. The result from regression and classification models are presented and discussed in Section 4.2. A comparison of the original controller and designed depth controller for vertical plane is presented in Section 4.3 along with discussion. All result are generated in AUVsim, and the figures are processed and plotted in Matlab. All files that are used to produced the result, along with unused developed algorithms in Python, can be found in the attached folder.

4.1 Limitations

Computer power is one of the main limitations evolving machine learning and simulation. When dataset for current speed and direction was develop, problem with Random Access Memory (RAM) was occasionally experienced. To overcome this problem, the datasets had to be reduced. Regression models was therefore trained on dataset consisting of various operating speed and head current speed. The first 1000 data points of the simulation was removed, and the amount of simulation files were reduced. Prediction of current speed when the AUV experienced current direction, was therefore neglected.

The classification models classifies the current direction according to the labels extracted from the dependent variable in the training data. The prediction done by the classification models are therefore only able to categorize the direction in classes of 15 degrees. Experiments with more precise prediction was made, but lead to significantly longer training time, and the process of the computer was not able to complete the training phase.

All classification and regression models was originally developed in Python. Due to changes in project description, only two of these models could be used. Therefore, the remaining models needed to be developed in Matlab.

4.2 Machine Learning Result and Discussion

The following result present regression and classification estimates of current speed and current direction, respectively. The results are divided into subsections, and presented according to case numbers given in Table 3.4. The AUV is set to follow a straight-line with constant depth. The waypoint-list used to conduct the test is presented in Table 4.1, with the initial position of the AUV given as $x_0 = [0, 0, 0, 0, 0, 0]$.

Waypoint-list		
x	y	z
100	0	30
200	0	30
300	0	30
400	0	30

Table 4.1: Waypoints for straight line and constant depth.

4.2.1 Regression

Case R1 - Current speed 0.25 m/s

Figure 4.1 illustrates the estimated current performed by Polynomial and Multiple Linear Regression model. The polynomial model makes almost a perfect fit after 100 seconds. This is due to the AUV having reached its steady state condition and the variance of independent variables are approximately zero. The same tendency is illustrated for the multiple linear regression when steady state is reached. The Multiple Linear regression model, however, overestimates the current speed to $V_c = 0.32\ m/s$.

By looking into the Multiple Linear model, the intercept coefficient $\beta_0 = 0.3175$. This indicates that the Multiple Linear Regressing model is best suited for current estimation of higher magnitude then what is tested in this case. See $current\_dir\_estimation.m$ for full implementation of the model.

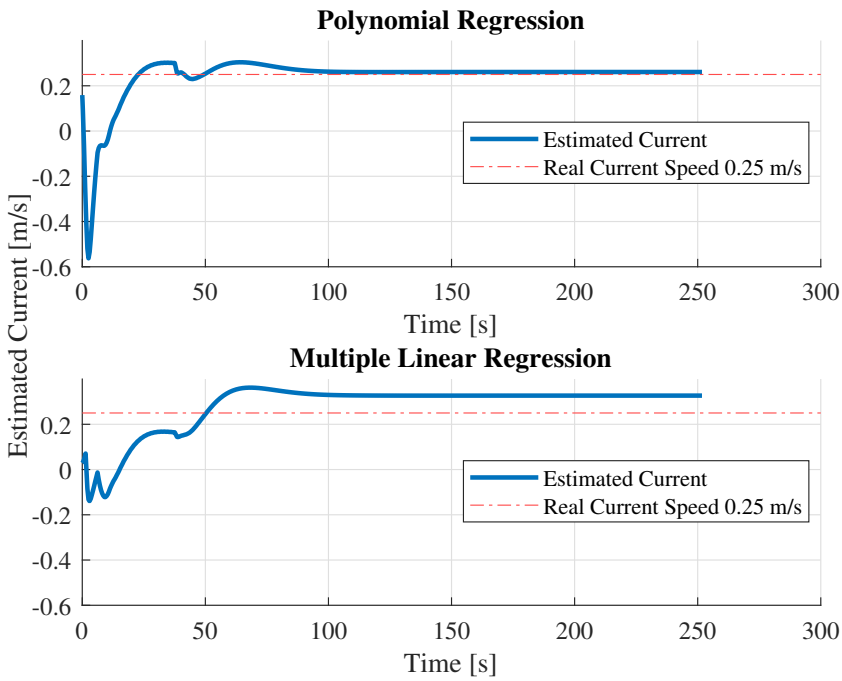


Figure 4.1: Polynomial and Multiple Linear Regression for Case R1 with real current speed of 0.25 m/s. The Polynomial Regression model makes almost a perfect fit, while the Multiple Linear Regression overestimates the current speed to $V_c = 0.32\ m/s$.

Figure 4.2 compare the Fine, Medium and Coarse Decision Tree Regression models. All models perform significantly worse than the Polynomial and Multiple Linear Regression models presented in Figure 4.1. Before steady state condition is reach, there is a consid-

erable amount of noise in the estimations. This is due to variations in the independent variables.

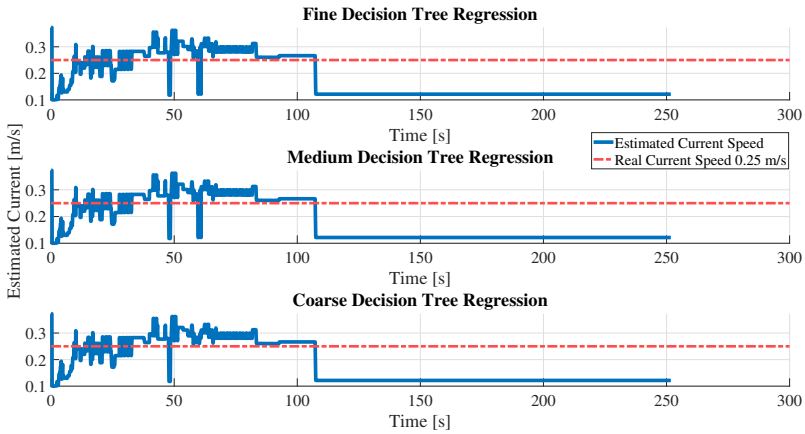


Figure 4.2: Fine, Medium and Coarse Decision Tree Regression for Case R1 with real current speed of 0.25 m/s. Compared to Polynomial and Multiple Linear Regression (Figure 4.1), these models perform significantly worse due to variations in the independent variable and thereby considerably amount of noises in the estimates.

Case R2 - Current speed 0.50 m/s

Figure 4.3 illustrates the estimated current performed by Polynomial and Multiple Linear Regression. As the figure illustrates, the same tendency appears for Case R2 as for Case R1 (Figure 4.1). The Polynomial Regression model makes a perfect fit after 100 seconds when steady state condition is reached. Comparing to Case 1, the Multiple Linear regression performs better for Case R2 as the overestimate for current speed is smaller, $V_c = 0.53\ m/s$.

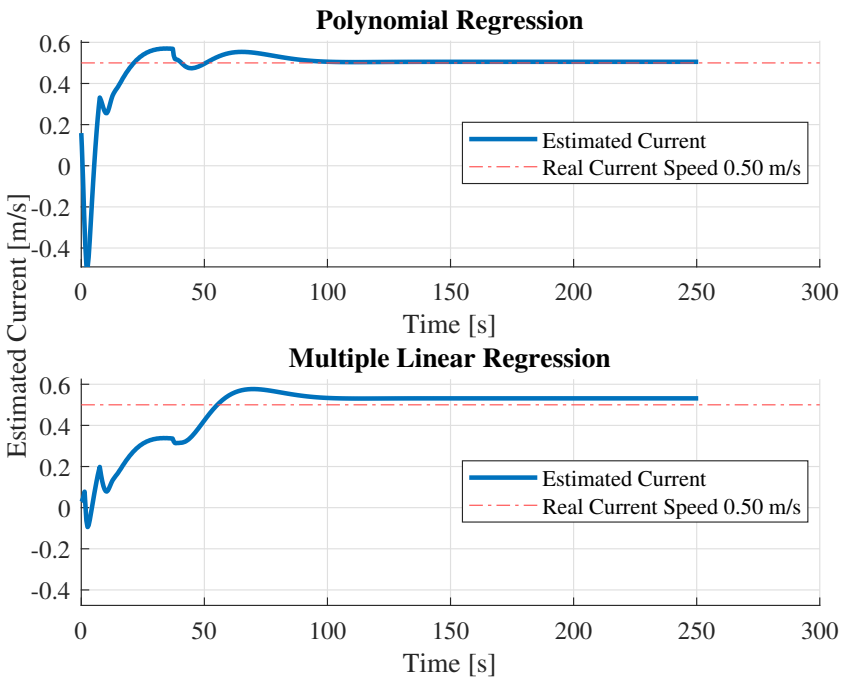


Figure 4.3: Polynomial and Multiple Linear Regression for Case R2 with real current speed of 0.50 m/s. The Polynomial Regression model makes a perfect fit in accordance with Case R1 (Figure 4.1). The Multiple Linear Regression has a smaller overestimate for current speed compared to Case R1.

Figure 4.4 compare the Fine, Medium and Coarse Decision Tree Regression models for Case R2. The predictions made in steady state illustrate improvements compared to Case R1 (Figure 4.2) with almost perfect estimates. Before steady state condition is reached, the same tendency appear with noisy predictions.

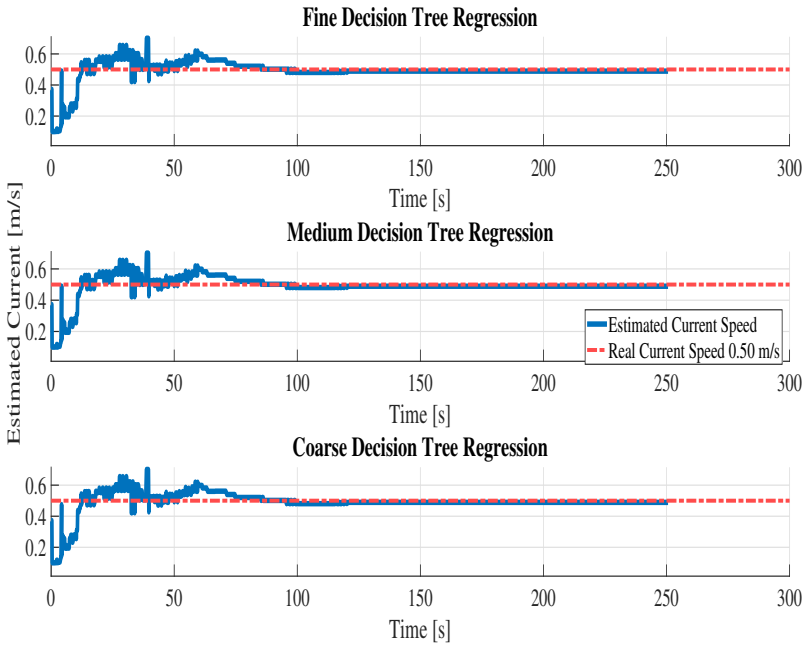


Figure 4.4: Fine, Medium and Coarse Decision Tree Regression for Case R2 with real current speed of 0.5 m/s. Compared to results presented for Case R1 (Figure 4.2), the results for Case R2 illustrates an almost perfect fit with the same tendency with noisy prediction before steady state condition is reached.

Case R3 - Current speed 0.75 m/s

Figure 4.5 illustrates the estimated current performed by Polynomial and Multiple Linear Regression. Both models perform well in steady state condition as the current speed of 0.75 m/s is reached after approximately 100 seconds. The Multiple Linear Regression model however, performs better before steady state condition compared to the Polynomial Regression model, due to smaller variation in estimated current speed.

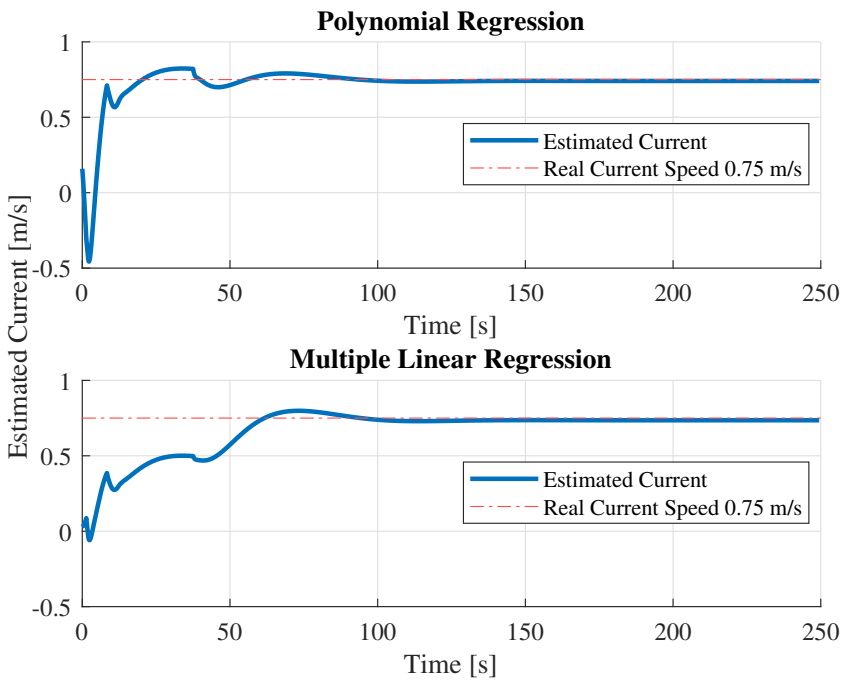


Figure 4.5: Polynomial and Multiple Linear Regression for Case R3 with real current speed of 0.75 m/s. Both models perform well in steady state conditions, while the Multiple Linear Regressions is superior before steady state condition is reached.

Figure 4.6 compare the Fine, Medium and Coarse Decision Tree Regression models. The same tendency with noisy predictions before steady state appearing in Case R1 (Figure 4.2) and Case R2 (Figure 4.4), is still present for Case R3. The estimated current speed in steady state however, illustrates perfect predictions for both models.

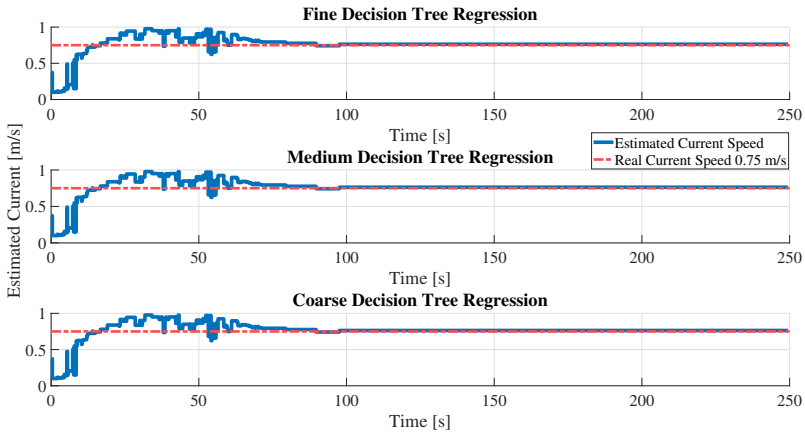


Figure 4.6: Fine, Medium and Coarse Decision Tree Regression for Case R3 with real current speed of 0.75 m/s. Compared to Case R1 (Figure 4.2) and Case R2 (Figure 4.4), the noisy prediction still occurs before steady state condition is reached, while in steady state the models illustrates perfect predictions.

The presented regression results are performed with head current corresponding speed of 0.25, 0.50 and 0.75 m/s. Polynomial and Multiple Linear Regression, and Fine, Medium and Coarse Decision Tree Regression models are compared. All models are within an acceptable range with respect to RMSE (Table 3.2), where the Polynomial Regression models scored the highest. This is consistent with result presented for Case R1, R2 and R3 which illustrates that the Polynomial Regression model overall performs on a higher level compared to remaining models. This is due to less noise and perfect predictions in steady state condition for all cases. The Polynomial Regression model can easily be implemented in the real system due to the model consisting of equation with estimated coefficient. This lead to high predicting speed, and can be deployed for real time analysis.

To produce more complex and larger dataset, a computer with better specifications is needed. Alternatively, to get a more general regression model, smaller simulations with less data could be run together with current direction at one specific operating speed. For example, $Uref = 1.6\ m/s$ which is a normal operating speed for REMUS 100. Another alternative is to produce more specific machine learning algorithms that operates at a certain range. For example, current speed from 0 to 0.3 m/s and 0.3 to 0.6 m/s, and so on. Further, these could be combined into a hybrid system with some form of switching mechanism. See Xia et al. (2009) and Sang et al. (2018) for development of hybrid controller for underwater vehicle.

Results from regression models when current direction is present, can be found in Appendix C. These are not included in the thesis due to results generated outside of the training range of the models. Thereby, the models performs poorly.

4.2.2 Classification

Case C1 - Current direction 0°

Estimation of current direction performed by Fine and Weighted KNN models are presented in Figure 4.7. The current direction from 0 to 25 seconds contains noisy estimates for both methods. When steady state is reached, the Weighted KNN classifies the current as 0°, while Fine KNN as 360°. This will in practice be the same direction, and will therefore not affect the result. In steady state condition, both models performs well.

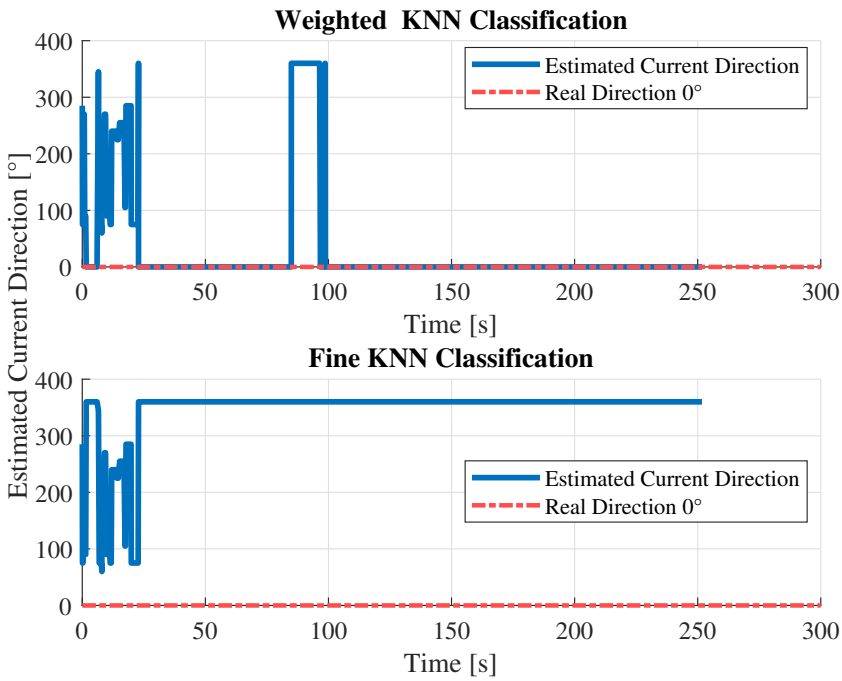


Figure 4.7: Fine and Weighted KNN Classification for Case C1 with real current direction of 0°. Both models illustrates noisy estimates before steady state condition, while they performs well when steady state is reached. 0° and 360° equals in practice the same direction.

Figure 4.8 illustrates the prediction for Fine and Medium Decision Tree Classification. The models are identical and classifies the current at 360°. Compared to the KNN methods in Figure 4.7, the Fine and Medium Decision Tree Classification contains virtually no noise before steady state is reached.

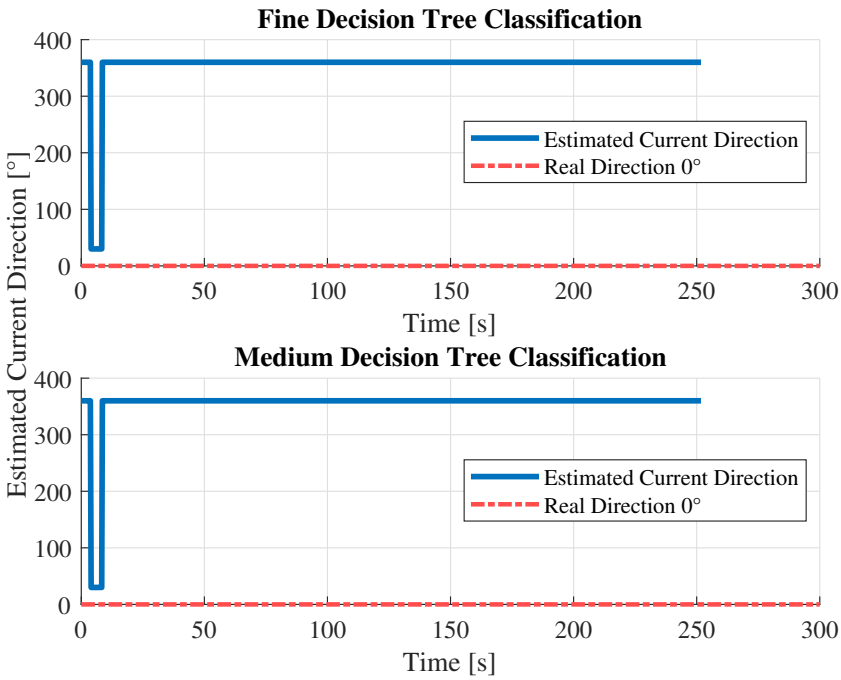


Figure 4.8: Fine and Medium Decision Tree Classification for Case C1 with real current direction of 0°. Both models performs well and equally, with virtually no noise compared to Case C1 (Figure 4.7).

Case C2 - Current direction 90°

Figure 4.9 presents estimated results for Fine and Weighted Classification models. Compared to Case C1 (Figure 4.7), the amount of noise across the whole simulation period is significantly higher. However, the two models presented i Figure 4.9 illustrates similar results.

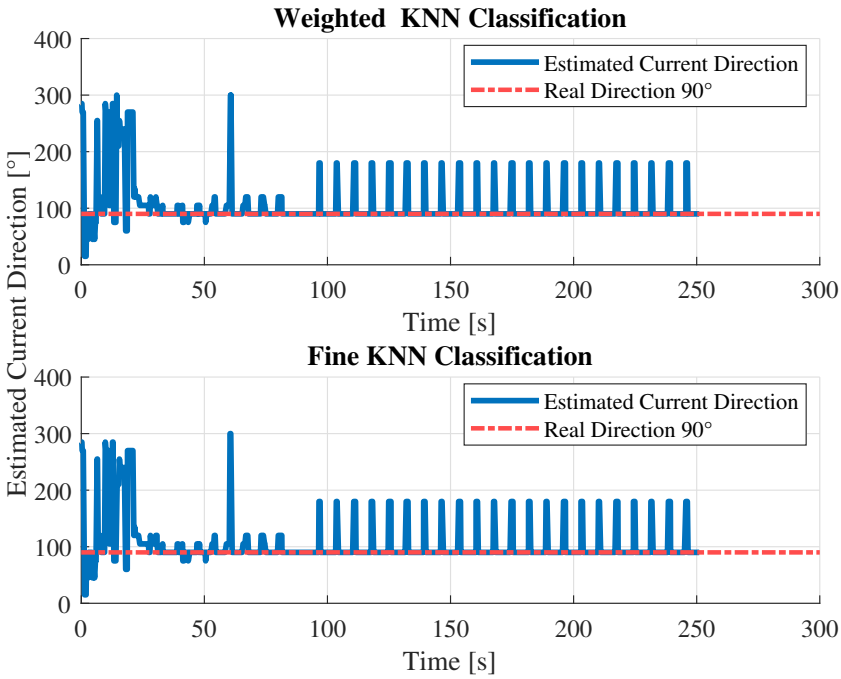


Figure 4.9: Fine and Weighted KNN Classification for Case C2 with real current direction of 90°. Both models illustrate noise across the whole simulation period.

The Fine and Medium Decision Tree Classification models, presented in Figure 4.10, illustrates good prediction in steady state. The Fine Tree model makes a perfect prediction, while the Medium Tree model overestimates by 15°. Overall, Fine and Medium Tree models produce less noise then KNN models for current direction presented in Case C1 (Figure 4.8).

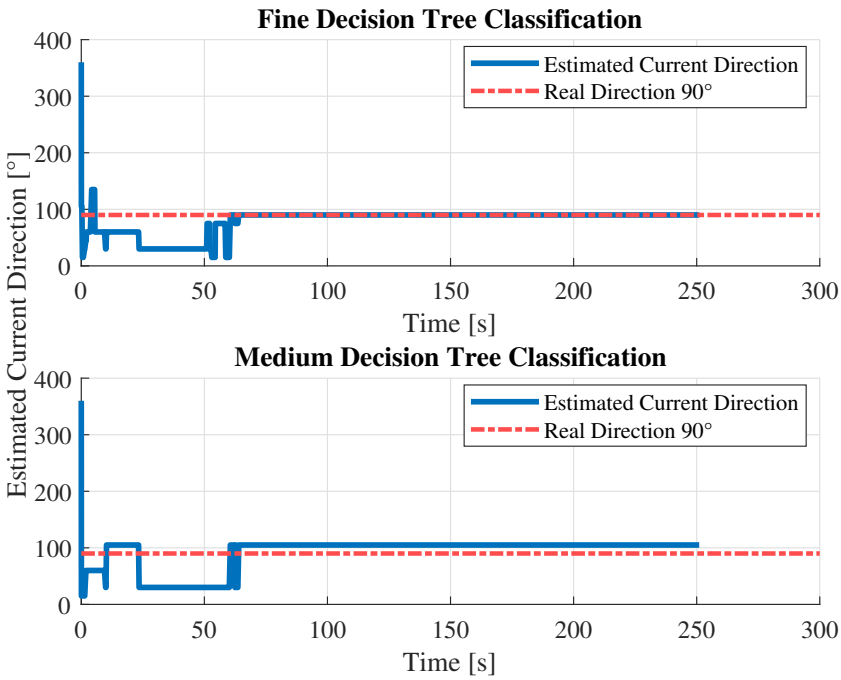


Figure 4.10: Fine and Medium Decision Tree Classification for Case C2 with real current direction of 90°. Fine Tree Classification model illustrates a perfect prediction, while the Medium Tree Classification model overestimates by 15°.

Case C3 - Current direction 180°

Figure 4.11 illustrates the estimates from Fine and Weighted KNN Classification models. The estimation is noisy and the models preformed poorly when the AUV follow the selected current direction.

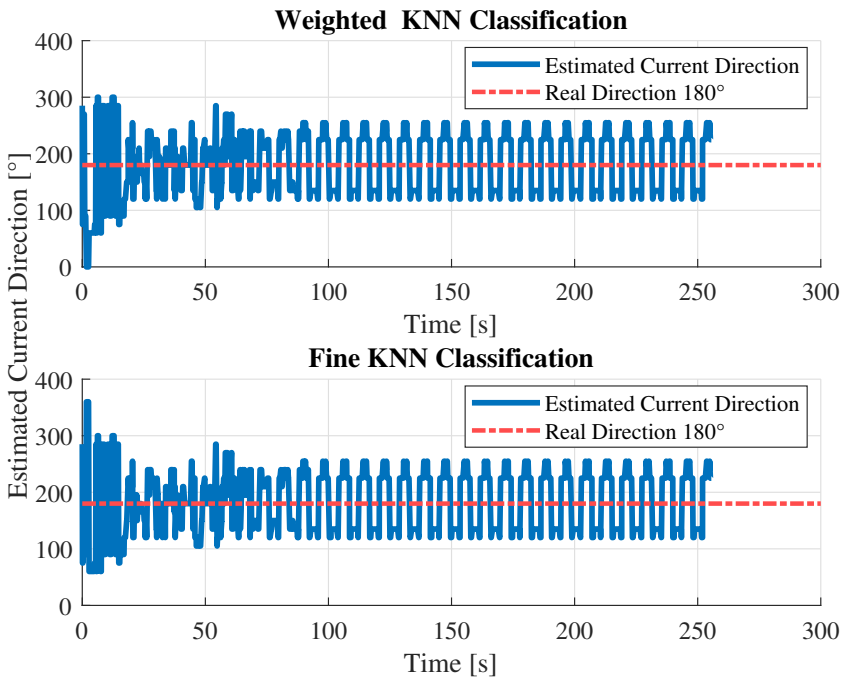


Figure 4.11: Fine and Weighted KNN Classification models for Case C3 with real current direction of 180°. Both models illustrates poorly predictions of current direction.

The Fine and Medium Decision Tree Classification models produce more or less noisy predictions over the whole simulation period as illustrated in Figure 4.12. Compared to previous Decision Tree Classification models for Case C1 and Case C2, Case C3 illustrates significantly amount of noise and are not able to preform prediction with selected current direction.

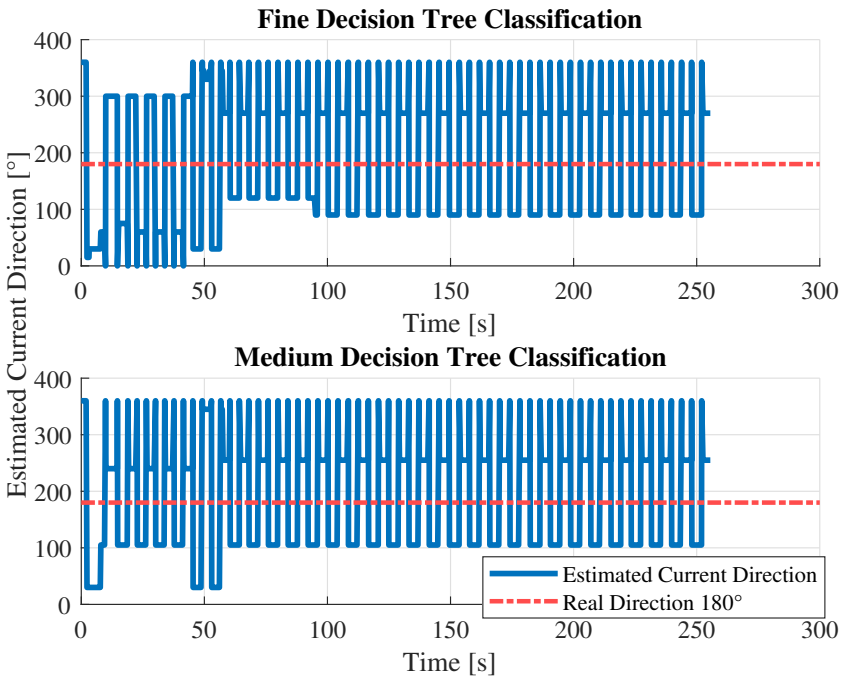


Figure 4.12: Fine and Medium Decision Tree Classification for Case C3 with real current direction of 180°. Both models illustrates poorly predictions with selected current direction.

Case C4 - Current direction 270 °

Figure 4.13 illustrates the predictions for Weighted and Fine KNN Classification for current direction of 270°. Compared to Case C2 with current direction of 90° (Figure 4.9), Case C4 illustrates improved performance as the estimation is less noisy. In steady state condition, after 100 seconds, there is some small spikes of under- and overestimation of 15° for both models.

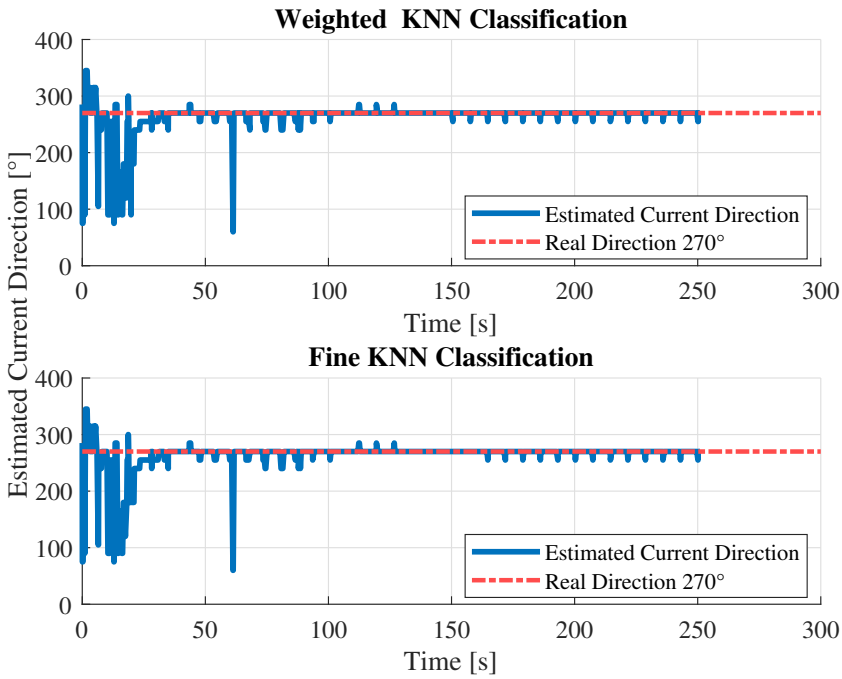


Figure 4.13: Fine and Weighted KNN Classification for Case C4 with real current direction of 270°. Compared to Case C2, these models illustrates improved performance as the estimates are less noisy. Some small spikes of under- and overestimation of 15° appears in steady state condition.

Fine and Medium Decision Tree Classification models for Case C4, Figure 4.14, predicts more consistent estimates, and contains less noise compared to KNN models in Figure 4.13. Medium Decision Tree Classification underestimates the current direction by 15 degrees, while Fine Decision Tree model makes a perfect prediction as the estimates reaches a current direction of 270° in steady state condition.

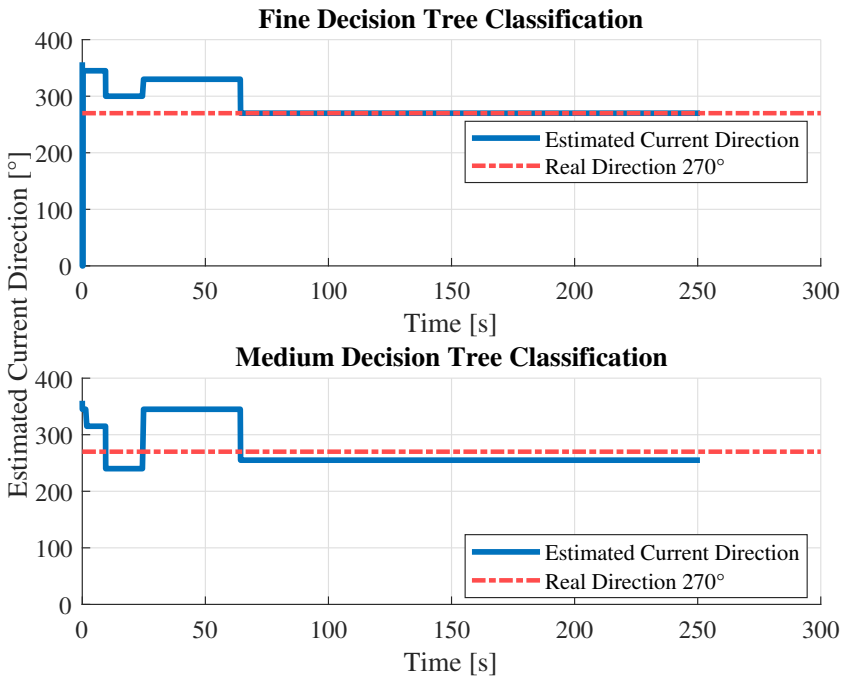


Figure 4.14: Fine and Medium Decision Tree Classification models for Case C4 with real current direction of 270°. The figure illustrates more consistent estimates and less noise compared to KNN models in Figure 4.13.

Classification results are presented for Weighted and Fine KNN, and Fine and Medium Decision Tree models. The current direction for Case C1, C2, C3 and C4 corresponds to 0°, 90°, 180° and 270°, respectively. As the presented result illustrates, the classification models are not able to predict the current direction for Case C3 when the AUV is exposed to a real current direction of 180°. This leads to noisy estimates and poorly predicted directions for both the KNN models, and Fine and Medium Decision Tree models. The unpleasant results for Case C3 could be due to fluctuation in the independent variables. Better tuning could possibly result in more stability among independent variables, and thereby result in better predictions.

Fine and Medium Decision Tree Classification models scored an accuracy of 23.9 % and 17.0 % receptively, while the Fine and Weighted KNN models scored an accuracy of 88.5% and 87.9%, respectively (Table 3.3). These results are inconsistent with the presented graphs as the KNN Classification models overall produces more noisy estimates compared to Fine and Medium Decision Tree Classification models. Fine Decision Tree Classification model predicted the exact current direction in Case C1, C2 and C4, and overall performed higher then the remaining models. This illustrates that one should not simply trust the accuracy score from the training phase, and that deployment and testing

of each models are necessary before any conclusion can be drawn. Appendix E presents confusion matrices for the trained classification models.

4.3 Depth Controller Result and Discussion

ILOS algorithm was developed to produce the desired theta angle in present of vertical ocean current. The method is described in Section 3.2. The depth controller is tested with comparison to the original controller in AUVsim described in Section 2.3.1. The result is presented in this section. All simulations are done in head current, where current speed is $V_c = 0.2\ m/s$. The speed of the AUV is chosen as a constant of $U_{ref} = 1.6\ m/s$, while the initial position of the AUV is chosen as $x_0 = [0, 0, 0, 0, 0, 0]$. The vertical current speed component is chosen as $w_c = -0.3\ m/s$.

Although the vertical current speed is the main focus, horizontal current speed has also been included. This is due to the horizontal current is almost always present. In theory, the horizontal current speed should not affect the results for vertical current speed. However, the controller developed in this thesis showed improved performance when both horizontal and vertical current speed was included. Therefore, the author has chosen to include this variable in the simulation. Better tuning of the horizontal controller would lead to improved performance of the proposed depth controller, and could thereby eliminate the need for horizontal current.

4.3.1 Fixed Depth Path

The AUV is set to follow a straight-line with constant depth. The waypoint-list used to conduct the simulation is presented in Table 4.2.

Waypoint-list		
x	y	z
100	0	30
200	0	30
300	0	30
400	0	30

Table 4.2: Waypoints for straight line and constant depth.

Figure 4.15 illustrates the North-East position of the AUV. The red crosses correspond to the horizontal waypoints (x-y) presented in Table 4.2. As the figure illustrates, the AVU is able to follow a straight line with minimal variations. With a distance of 100 meters towards north, the variations is $\pm$ 0.25 m in east direction.

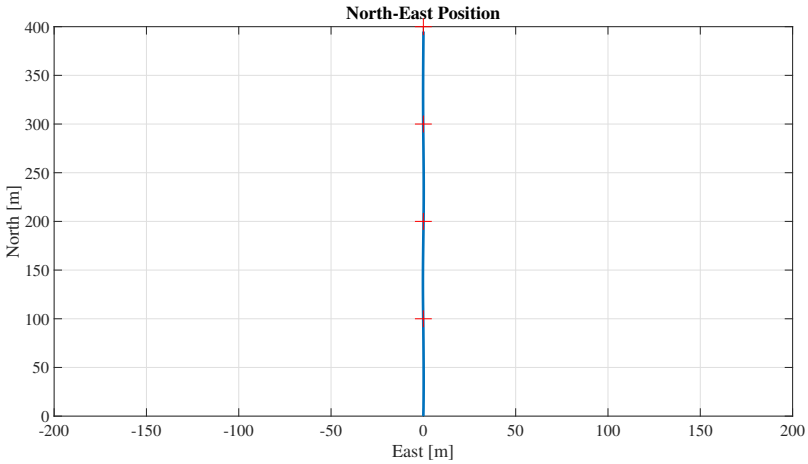


Figure 4.15: The North-East position illustrates that the AVU are following a straight line with minimal variations. Red crosses illustrated the waypoints presented in Table 4.2. With a distance of 100 meters towards north, the variations is $\pm$ 0.25 m in east direction.

Figure 4.16 illustrates the difference in depth position for REMUS 100 by using the original controller (blue line) and the designed depth controller (red line). The desired depth is set to $z =$ 30 m. As the figure illustrates, the original controller are not able to counteract the vertical current and therefore not able to reach the desired depth of 30 m.

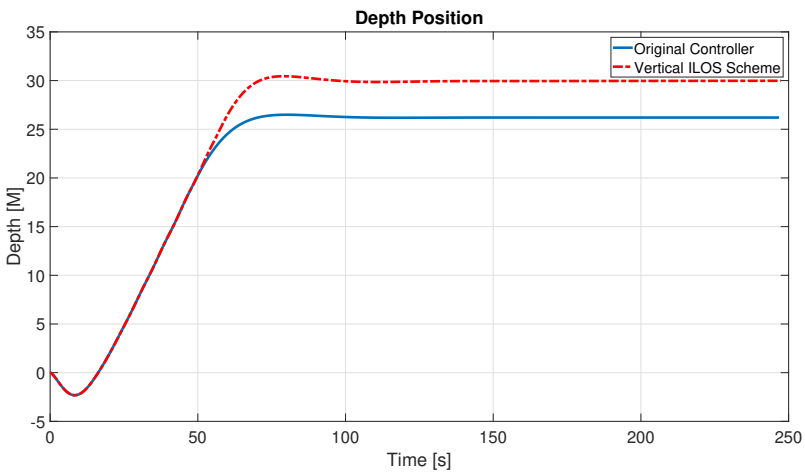


Figure 4.16: Comparison of original controller and designed depth controller with the vertical ILOS algorithm conducting a fixed level flight. The original controller are not able to counteract the current and therefore are not able to reach the desired depth of 30 m.

4.3.2 Various Depth Path

The following result is generated by running simulation with the waypoints shown in Table 4.3.

Waypoint-list		
x	y	z
100	0	30
200	0	30
300	0	30
450	50	45
500	100	45
600	200	10
700	300	10
800	400	20
900	500	20

Table 4.3: Waypoints for straight lines and various depth.

Figure 4.17 illustrates the North-East position of the AUV where the red crosses correspond to the horizontal waypoints presented in Table 4.3.

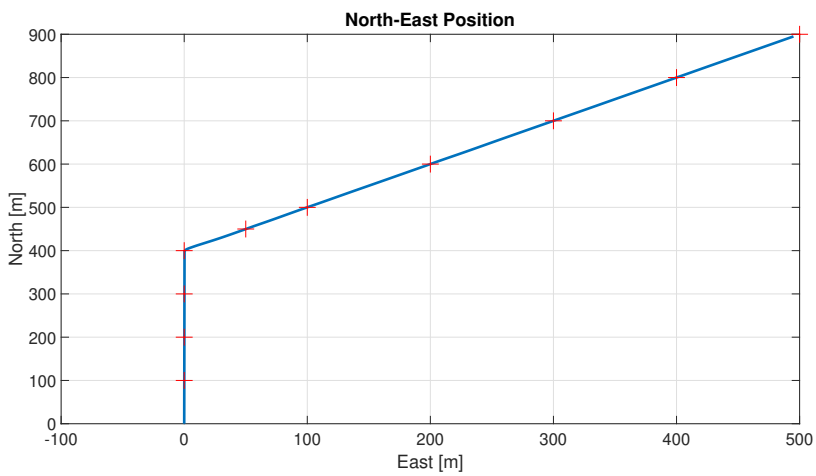


Figure 4.17: North-East position av AUV in present of ocean vertical current of 0.2 m/s with direction of 0°.

Figure 4.18 illustrates the difference in depth position of REMUS 100 by using the original controller (blue line) and the designed depth controller (red line) for various depth.

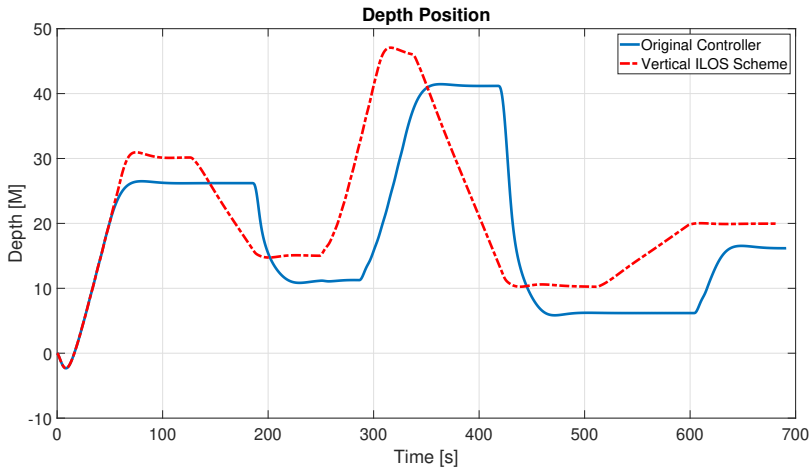


Figure 4.18: Comparison with original controller and designed depth controller with the vertical ILOS algorithm with various depth.

Figure 4.16 compares the original and designed controller by executing a fixed level flight. As the graphs illustrates, the original controller is not able to reach the desired depth, while the designed controller is able to counteract the vertical current component and reaches desired set-point of 30 m. Figure 4.18 compares the original and designed controller in various desired depth points. The graphs illustrates that the original controller has a quicker response compared to the designed controller, but is not able to reach the desired set-point. A more aggressive tuning of pitch controller and ILOS parameters would possibly make a faster response of the designed controller.

Chapter 5

Conclusion

The aim of this thesis has been to investigate current estimation of autonomous underwater vehicles by using the machine learning method, Supervised Learning. Chapter 1 presented an introduction to the thesis by stating previous studies on underwater vehicles, machine learning and current estimation. Chapter 2 presented the mathematical modeling for underwater vehicles inspired from marine vessel along with a basic introduction to machine learning. Chapter 3 described the methods used to produce the result in Chapter 4.

Different regression and classification models were developed to estimate the current speed and -direction for AUV REMUS 100. A considerably amount of time has been dedicated to produce optimal dataset for both regression and classification models. An optimal dataset consisting of the whole operating regime of the AUV was difficult to achieve due to limitations in computer power. Therefore, the regression models were only trained in head current.

Comparison between Polynomial, Multiple Linear, Fine Decision Tree, Medium Decision Tree, and Coarse Decision Tree Regression models was conducted in a case study. Polynomial Regression model was best suited for numeric estimation for current speed. Weighted and Fine KNN Classification models were compared to Fine and Medium Decision Tree Classification in another case study. The Fine Decision Tree model performed on the highest level.

In addition to presenting Supervised Learning methods, a depth controller consisting of an ILOS algorithm combined with a PID controller was designed. The ILOS algorithm produce the desired theta angle which the PID controller regulates the stern fins according to. The controller was tested exposing the AUV to vertical current. The presented result was promising as the controller was able to reach the desired depth point in both fixed and various depth flight. A more aggressive tuning of the pitch controller and ILSO scheme would possibly make the AUV converge to the desired set-point faster.

The thesis concludes that the machine learning method, Supervised Learning, has large potential to estimate current speed and direction. Although, the process is quite time consuming to achieve good results. This is due to the need of high-quality data and significantly training time when dataset is large. Polynomial Regression model was able to predict the current speed in steady state condition with perfect precision in all cases. The corresponding Fine Decision Tree Classification model followed the same tendency in steady state condition and was able to predict the current direction in 3/4 cases with perfect precision. The designed depth controller was able to counteract the vertical current component and reach the desired depth point. This will provide the AUV the options to work in conditions with unknown vertical current, and potentially reduce the need of expensive sensors.

5.1 Further Work

This thesis presents estimation of current speed and direction, and a depth controller for the autonomous underwater vehicle REMUS 100. The following paragraphs present the authors proposal for further work:

- Regression models developed in this thesis suffers from lack of training data. For further work on the machine learning aspect, the author propose to include larger datasets by involving current direction for improving the estimation of the models.
- To produce an optimal dataset, the internal controller should be adjusted and tuned so the output is stable through the whole operating regime.
- Further investigation on how the regression and classification models perform with other controllers.
- To make a wider comparison between different models, the author propose to include other regression and classification models.
- Tuning and optimization of the presented models is needed to investigate if higher performance are achievable.
- Filed tests of REMUS 100 is necessary to investigate how the regression and classification models along with designed depth controller performs under real circumstances. Real datasets needs to be generated through field tests, and further compared to the trained algorithms.
- To investigate the potential machine learning has on predicting the main external forces acting on underwater vehicles, a comparison with other current estimation methods, like ADCP, observer estimation and DPSS should be conducted.

Bibliography

Allen, B., Vorus, W., Prestero, T., 2000. Propulsion system performance enhancements on remus auvs. IEEE Xplore 3, 1869–1873. doi:10.1109/OCEANS.2000.882209.

Arneson, I.B., Brodtkorb, A.H., Sørensen, A.J., 2019. Sea state estimation using quadratic discriminant analysis and partial least squares regression. IFAC-PapersOnLine 52, 72–77. doi:10.1016/j.ifacol.2019.12.285.

Awad, M., Khanna, R., 2015. Support vector regression, in: Efficient Learning Machines, p. 67–80. doi:10.1007/978-1-4302-5990-9_4.

Bailey, J., Milne, I., Someren, M., 2019. Seastate detection from vessels at motion using learning algorithms doi:https://doi.org/10.1016/j.oceaneng.2017.08.047.

Caharija, W., Pettersen, K.Y., Gravdahl, J.T., Børhaug, E., 2012. Integral los guidance for horizontal path following of underactuated autonomous underwater vehicles in the presence of vertical ocean currents, in: 2012 American Control Conference (ACC), pp. 5427–5434. doi:10.1109/ACC.2012.6315607.

Candeloro, M., 2016. Tools and Methods for Autonomous Operations on Seabed and Water Column using Underwater Vehicles. Ph.D. thesis. Norwegian University of Science and Technology.

Candeloro, M., Sørensen, A.J., Longhi, S., Dukan, F., 2012. Observers for dynamic positioning of rovs with experimental results. IFAC Proceedings Volumes 45, 85–90. URL: https://www.sciencedirect.com/science/article/pii/S1474667016312095?via%3Dihub, doi:10.3182/20120919-3-IT-2046.00015.

Carlton, J., 2007. 11-propeller–ship interaction, in: Marine Propellers and Propulsion. 2nd ed.. Butterworth-Heinemann, Oxford, pp. 264–283. doi:https://doi.org/10.1016/B978-075068150-6/50013-9.

Caruana, R., Niculescu-Mizil, A., 2004. An empirical analysis of supervised learning performance criteria URL: https://www.cs.cornell.edu/~caruana/perfs.kdd04.revised.rev1.pdf.

Caruana, R., Niculescu-Mizil, A., 2006. An empirical comparison of supervised learning algorithms. Proceedings of the 23rd international conference on Machine learning, ICML 06 doi:10.1145/1143844.1143865.

Chiu, M.H., Yu, Y.R., Liaw, H., Hao, L., 2016. The use of facial micro-expression state and tree-forest model for predicting conceptual-conflict based conceptual change URL: https://www.researchgate.net/publication/295860754_THE_USE_OF_FACIAL_MICRO-EXPRESSION_STATE_AND_TREE-FOREST_MODEL_FOR_PREDICTING_CONCEPTUAL-CONFLICT_BASED_CONCEPTUAL_CHANGE.

Choudhary, R., Gianey, H.K., 2017. Comprehensive review on supervised machine learning algorithms, in: 2017 International Conference on Machine Learning and Data Science (MLDS), IEEE. pp. 37–43.

Cruz, N.A., 2011. Autonomous Underwater Vehicles. URL: https://www.intechopen.com/books/autonomous-underwater-vehicles.

Dong, N.T., 2005. Design Of Hybrid Marine Control Systems For Dynamic Positioning. Ph.D. thesis. Department Of Civil Engineering National University Of Singapore.

Dukan, F., 2014. ROV Motion Control Systems. Ph.D. thesis. Norwegian University of Science and Technology. URL: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/239258.

Eremenko, K., de Ponteves, H., Team, S., Support, S., 2015. Machine learning aztm: Hands-on python & r in data science. URL: https://www.udemy.com/course/machinelearning/.

Fjerdingen, S.A., Kyrkjeboe, E., Transeth, A.A., 2010. Auv pipeline following using reinforcement learning, in: ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics), pp. 1–8.

Fossen, T.I., 2011. Handbook of Marine Craft Hydrodynamics and Motion Control. 1st ed., John Wiley & Sons, Ltd. doi:10.1002/9781119994138.

Fossen, T.I., 2021. Handbook of Marine Craft Hydrodynamics and Motion Control. 2nd ed., John Wiley & Sons, Ltd.

García-Valdovinos, L.G., Salgado-Jiménez, T., Bandala-Sánchez, M., Nava-Balanzar, L., Hernández-Alvarado, R., Cruz-Ledesma, J.A., 2014. Modelling, design and robust control of a remotely operated underwater vehicle. International Journal of Advanced Robotic Systems 11, 1. doi:10.5772/56810.

Hegrenæs, O., Hallingstad, O., 2011. Model-aided ins with sea current estimation for robust underwater navigation. Journal of Oceanic Engineering 36, 316–337. doi:10.1109/JOE.2010.2100470.

Holsen, S.A., 2015. DUNE: Unified Navigation Environment for the REMUS 100 AUV - Implementation, Simulator Development, and Field Experiments. Master's thesis. Norwegian University of Science and Technology. URL: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2350792?locale-attribute=en.

Hurwitz, J., Kirsch, D., 2018. Machine Learning for dummies. John Wiley & Sons Inc.

Hydroid, 2012. Our new generation man-portable auv remus 100 autonomous underwater vehicle. URL: https://www.kongsberg.com/globalassets/maritime/km-products/product-documents/remus-100-autonomous-underwater-vehicle.

IBM Cloud Education, 2020. What is machine learning? URL: https://www.ibm.com/cloud/learn/machine-learning.

Islam, M.J., Wu, Q.M.J., Ahmadi, M., Sid-Ahmed, M.A., 2007. Investigating the performance of naive- bayes classifiers and k- nearest neighbor classifiers, in: 2007 International Conference on Convergence Information Technology (ICCIT 2007), pp. 1541–1546. doi:10.1109/ICCIT.2007.148.

Johansen, J.H., 2020. Non-linear control and digital twin modeling of the REMUS 100 AUV. Master's thesis. Norwegian University of Science and Technology.

Kim, E., Fan, S., Bose, N., Nguyen, H., 2020. Current estimation and path following for an autonomous underwater vehicle (auv) by using a high-gain observer based on an auv dynamic model. International Journal of Control, Automation and Systems doi:10.1007/s12555-019-0673-5.

Kokegei, M., He, F., Sammut, K., 2011. Fully coupled 6 degree-of-freedom control of an overactuated autonomous underwater vehicle. Autonomous Underwater Vehicles , 147–170.

Leonard, J.J., Bennett, A.A., Smith, C.M., Jacob, H., Feder, S., 1998. Autonomous underwater vehicle navigation, in: MIT Marine Robotics Laboratory Technical Memorandum.

Liu, S., Wei, Y., Gao, Y., 2012. 3d path planning for auv using fuzzy logic , 599–603doi:10.1109/CSIP.2012.6308925.

Mak, B., Düz, B., 2019. Ship As a Wave Buoy: Estimating Relative Wave Direction From In-Service Ship Motion Measurements Using Machine Learning 9. doi:10.1115/OMAE2019-96201.

Maxwell, A.E., Warner, T.A., Fang, F., 2018. Implementation of machine-learning classification in remote sensing: an applied review. International Journal of Remote Sensing 39, 2784–2817. doi:10.1080/01431161.2018.1433343.

Meurer, C., Fuentes, J., Palomeras, N., Carreras, M., Kruusmaa, M., 2019. Differential pressure sensor speedometer for autonomous underwater vehicle velocity estimation. IEEE Journal of Oceanic Engineering , 1–33doi:10.1109/JOE.2019.2907822.

Meurer, C., Fuentes-Pérez, J.F., Schwarzwälder, K., Ludvigsen, M., Sørensen, A.J., Kruusmaa, M., 2020. 2d estimation of velocity relative to water and tidal currents based on differential pressure for autonomous underwater vehicles. IEEE Robotics and Automation Letters 5, 3444–3451. doi:10.1109/LRA.2020.2976318.

Mwiti, D., 2020. Random forest regression: When does it fail and why? URL: https://neptune.ai/blog/random-forest-regression-when-does-it-fail-and-why.

Naeem, W., Sutton, R., Ahmad, S., 2003. Lqg/ltr control of an autonomous underwater vehicle using a hybrid guidance law. IFAC Proceedings Volumes 36, 31 – 36. URL: http://www.sciencedirect.com/science/article/pii/S1474667017366533, doi:https://doi.org/10.1016/S1474-6670(17)36653-3. iFAC Workshop on Guidance and Control of Underwater Vehicles 2003, Newport, South Wales, UK, 9-11 April 2003.

National Oceanic and Atmospheric Administration, . How much water is in the ocean? URL: https://oceanservice.noaa.gov/facts/oceanwater.html. accessed: 2020-11-11.

Nebot, E., 1999. Sensors used for autonomous navigation, in: Advances in Intelligent Autonomous Systems. Springer, pp. 135–156.

Norgren, P., 2018. Autonomous underwater vehicles in arctic marine operations: Arctic marine research and ice monitoring .

Norgren, P., Skjetne, R., 2015. Line-of-sight iceberg edge-following using an auv equipped with multibeam sonar. IFAC-PapersOnLine 48, 81–88. doi:https://doi.org/10.1016/j.ifacol.2015.10.262.

Nornes, S.M., 2018. Guidance and Control of Marine Robotics for Ocean Mapping and Monitoring. Ph.D. thesis. Norwegian University of Science and Technology.

Prestero, T., 2001. Verification of a six-degree of freedom simulation model for the remus autonomous underwater vehicle. URL: https://darchive.mblwhoilibrary.org/handle/1912/3040.

Reshmi, K.R.G., Priya, P.S., 2016. Design and control of autonomous unerwater vehicle for depth control using lqr controller. International Journal of Science and Research 5, 1432 – 1436. URL: https://www.ijsr.net/search_index_results_paperid.php?id=ART2016447.

Riise, Ø., G., 2020. Guidance and control of underwater vehicle. Project thesis. Norwegian University of Science and Technology. Not published.

Rish, I., 2001. An empirical study of the naive bayes classifier. IJCAI 2001 workshop on empirical methods in artificial intelligence 3, 41–46.

Rosenbaum, L., Dörr, A., Bauer, M., Boeckler, F., Zell, A., 2013. Inferring multi-target qsar models with taxonomy-based multi-task learning. Journal of cheminformatics 5, 33. doi:10.1186/1758-2946-5-33.

Ruud, F.J., 2016. Autonomous Homing and Docking of AUV REMUS 100 - Homing and Docking Guidance Algorithm and Relative Localization. Master's thesis. Norwegian University of Science and Technology. URL: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2410753.

Sabiha, W., Pushkin, K., 2018. Autonomous underwater vehicles modeling, control design, and simulation. Tyler & Francis Group.

Sang, H., Zhou, Y., Sun, X., Yang, S., 2018. Heading tracking control with an adaptive hybrid control for under actuated underwater glider. ISA Transactions 80, 554–563. doi:10.1016/j.isatra.2018.06.012.

Sclavounos, P.D., Ma, Y., 2018. Artificial intelligence machine learning in marine hydrodynamics. 37th International Conference on Ocean, Offshore and Arctic Engineering doi:10.1115/omae2018-77599.

Souza, E.C.D., Maruyama, N., 2007. Intelligent uuvs: Some issues on rov dynamic positioning. IEEE Transactions on Aerospace and Electronic Systems 43, 214 – 226.

Sørensen, A.J., 2018. Towards Autonomous Marine Operations and Systems. Department of Marine Technology, NTNU.

Sørensen, A.J., 2020. Brief overview machine learning. University Lecture notes (TMR16).

Sørensen, A.J., Ludvigsen, M., 2015. Towards integrated autonomous underwater operations. IFAC-PapersOnLine 48, 107–118. doi:https://doi.org/10.1016/j.ifacol.2015.06.018.

Tranmer, M., Murphy, J., Elliot, M., Pampaka, M., 2020. Multiple Linear Regression. 2nd ed., Cathie Marsh Institute Working Paper. URL: http://hummedia.manchester.ac.uk/institutes/cmist/archive-publications/working-papers/2020/multiple-linear-regression.pdf.

Utne, I.B., Sørensen, A.J., Schjølberg, I., 2017. Risk management of autonomous marine systems and operations URL: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2468206.

Van Amerongen, J., 1984. Adaptive steering of ships—a model reference approach. Automatica 20, 3–14. doi:https://doi.org/10.1016/0005-1098(84)90060-8.

Værnø, S., Brodtkorb, A., Skjetne, R., 2019. Compensation of bias loads in dynamic positioning of marine surface vessels. Ocean Engineering 178, 484–492. doi:10.1016/j.oceaneng.2019.03.010.

Wang, H., Chen, Z., Jia, H., Chen, X., 2011. Nn-backstepping for diving control of an underactuated auv , 1–6doi:10.23919/OCEANS.2011.6107239.

Xia, G., Tang, L., Guo, F., Chen, Q., Leng, J., 2009. Design of a hybrid controller for heading control of an autonomous underwater vehicle , 1–5doi:10.1109/ICIT.2009.4939629.

Ye, H., Xue, W., Yang, X., 2018. Backstepping-based diving control design for underactuated auvs combined with ilos method. IEEE , 703–708doi:10.23919/ChiCC.2018.8483805.

Zagatti, R., Juliano, D.R., Doak, R., Souza, G.M., Nardy, L.d.P., Lepikson, H.A., Gaudig, C., Kirchner, F., 2018. Flatfish resident auv: Leading the autonomy era for subsea oil and gas operations doi:https://doi.org/10.4043/28881-MS.

Østeby, E., 2017. Subsea Cable Tracking using Sensor Fusion on an Autonomous Underwater Vehicle. Master's thesis. Norwegian University of Science and Technology. URL: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2456601.

Appendix A

Coefficients

Parameter	Value	Unit
$X_{\dot{x}}$	-0.93	kg
$Y_{\dot{v}}$	-35.5	kg
$Z_{\dot{w}}$	-35.5	kg
$K_{\dot{p}}$	-0.0704	$\mathrm{kg} \cdot m^2/\mathrm{rad}$
$M_{\dot{q}}$	-4.88	$\mathrm{kg} \cdot m^2/\mathrm{rad}$
$N_{\dot{r}}$	-4.88	$\mathrm{kg} \cdot m^2/\mathrm{rad}$
X_{uu}	-1.62	kg/m
X_{vv}	-1310	kg/m
X_{ww}	-2.86	kg/m
K_{pp}	-0.13	$\mathrm{kg} \cdot m^2/\mathrm{rad}^2$
M_{qq}	-188	$\mathrm{kg} \cdot m^2/\mathrm{rad}^2$
N_{rr}	-94	$\mathrm{kg} \cdot m^2/rad^2$
Y_{rr}	0.632	$\mathrm{kg} \cdot m/\mathrm{rad}^2$
Z_{qq}	-0.632	$\mathrm{kg} \cdot m/rad^2$
M_{ww}	3.18	kg
N_{vv}	-3.18	kg

Parameter	Value	Unit
K_T	2.5075	$\cdot$
K_Q	0.32033	$\cdot$
D_{prop}	0.1397	m

Appendix B

Python files

```python
# Polynomial Regression

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd


# Importing the dataset
dataset = pd.read_csv('file.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
    0.20, random_state = 1)

# Training the Polynomial Regression model on the Training set
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
poly_reg = PolynomialFeatures(degree = 2)
X_poly = poly_reg.fit_transform(X_train)
regressor = LinearRegression()
regressor.fit(X_poly, y_train)

# Predicting the Test set results
y_pred = regressor.predict(poly_reg.transform(X_test))
np.set_printoptions(precision=3)
np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test)
    ,1)),1)


# Evaluating the Model Performance
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
```

```
print(r2_score(y_test, y_pred))

from sklearn.metrics import mean_squared_error
print(mean_squared_error(y_test, y_pred))

print(regressor.coef_)
print(regressor.intercept_)

# Visualising the Polynomial Regression results
#plt.scatter(X, y, color = 'red')
#plt.plot(y_train, regressor.predict(poly_reg.fit_transform(X_train)),
    color = 'blue')
#plt.title('Polynomial regression')
#plt.xlabel('variables')
#plt.ylabel('Estimated Current [m/s]')
#plt.show()
```

```
# Multiple Linear Regression

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('file.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
     random_state = 0)

# Training the Multiple Linear Regression model on the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Predicting the Test set results
y_pred = regressor.predict(X_test)
np.set_printoptions(precision=8)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(
    y_test),1)),1))



# Evaluating the Model Performance with R2 and RMSE
from sklearn.metrics import r2_score
print(r2_score(y_test, y_pred))


from sklearn.metrics import mean_squared_error
print(mean_squared_error(y_test, y_pred))

print(regressor.coef_)
print(regressor.intercept_)
```

```
# Visualising the Test set results
#plt.scatter(X_test, y_test, color = 'red')
#plt.plot(X_train, regressor.predict(X_train) , color = 'blue')
#plt.title(' Current estimation')
#plt.xlabel('variables')
#plt.ylabel('Current speed [m/s]')
#plt.show()
```

Appendix C

Regression result

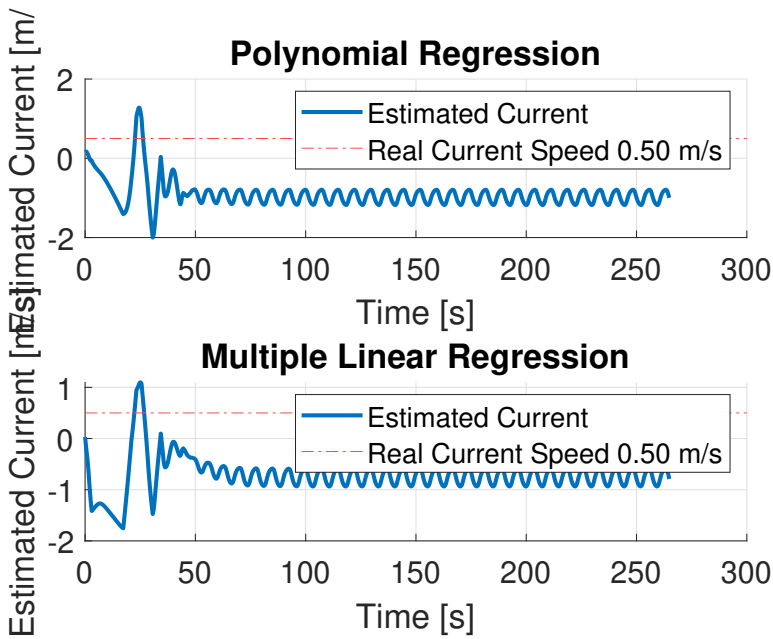


Figure C.1: Polynomial and Multiple Linear Regression estimation in 90° current direction

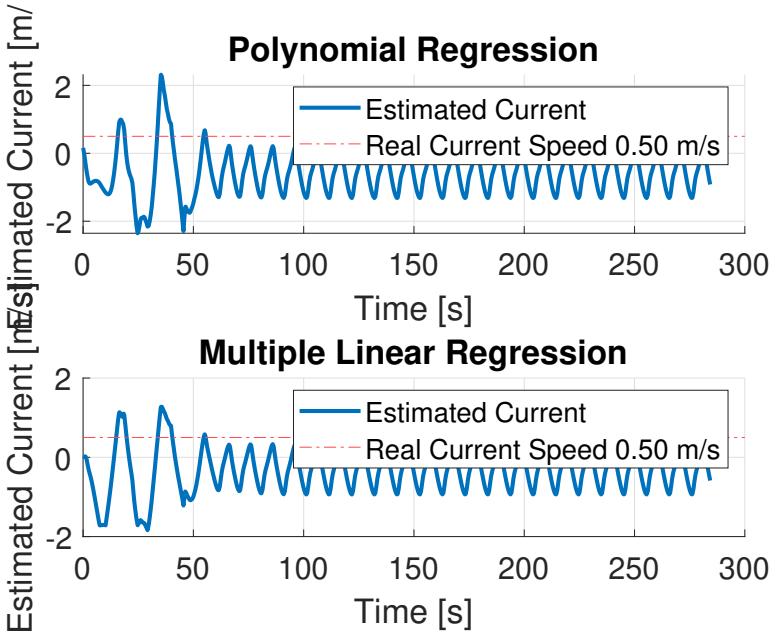


Figure C.2: Polynomial and Multiple Linear Regression estimation in 180° current direction

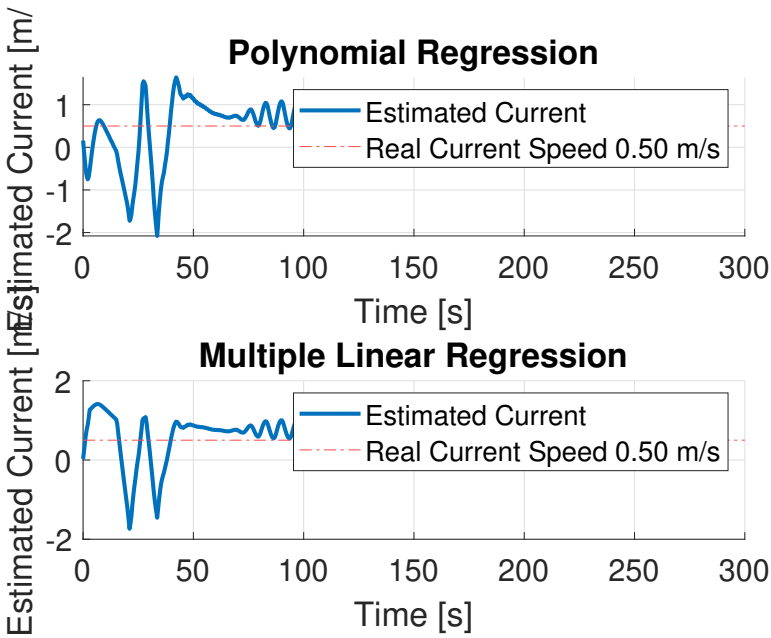


Figure C.3: Polynomial and Multiple Linear Regression estimation in 270° current direction

Appendix D

Confusion Matrix

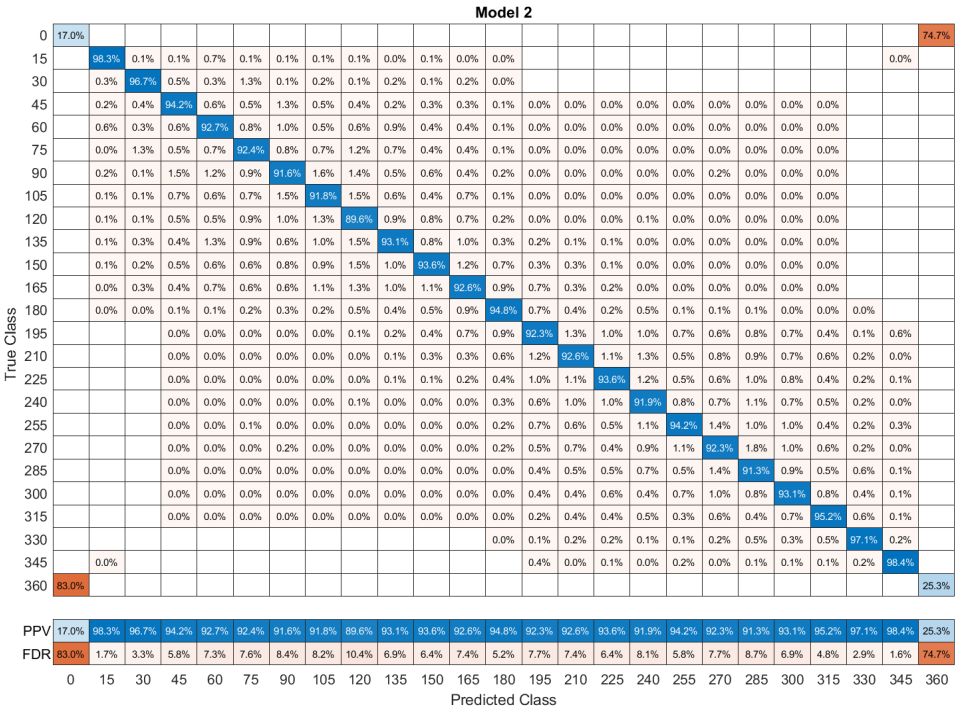


Figure D.1: Confusion matrix for Fine KNN Classification model

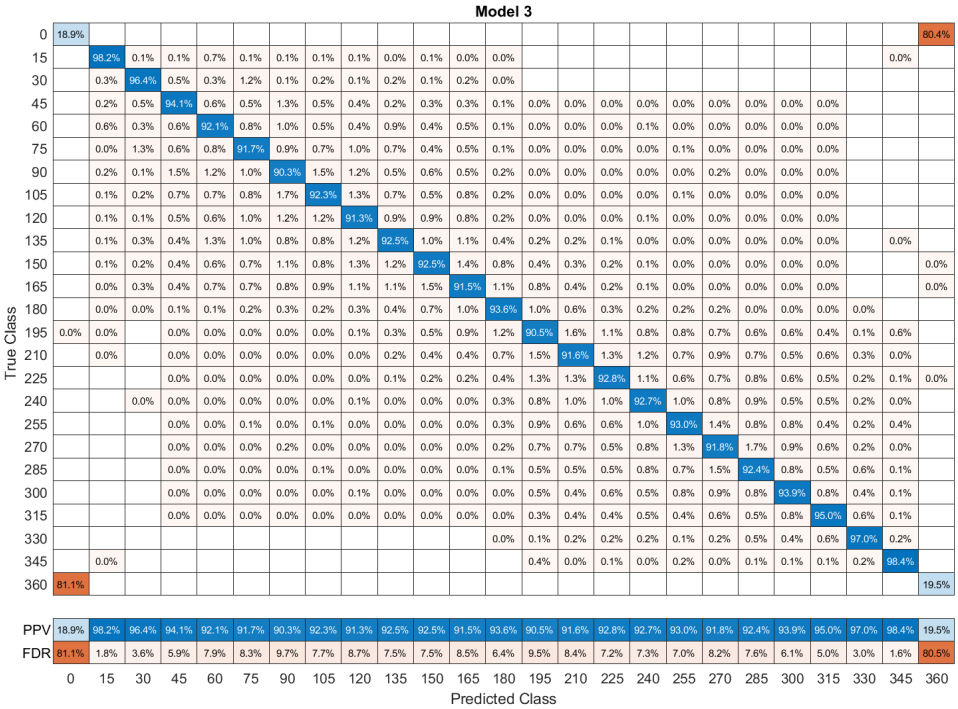


Figure D.2: Confusion matrix for Weighted KNN Classification model

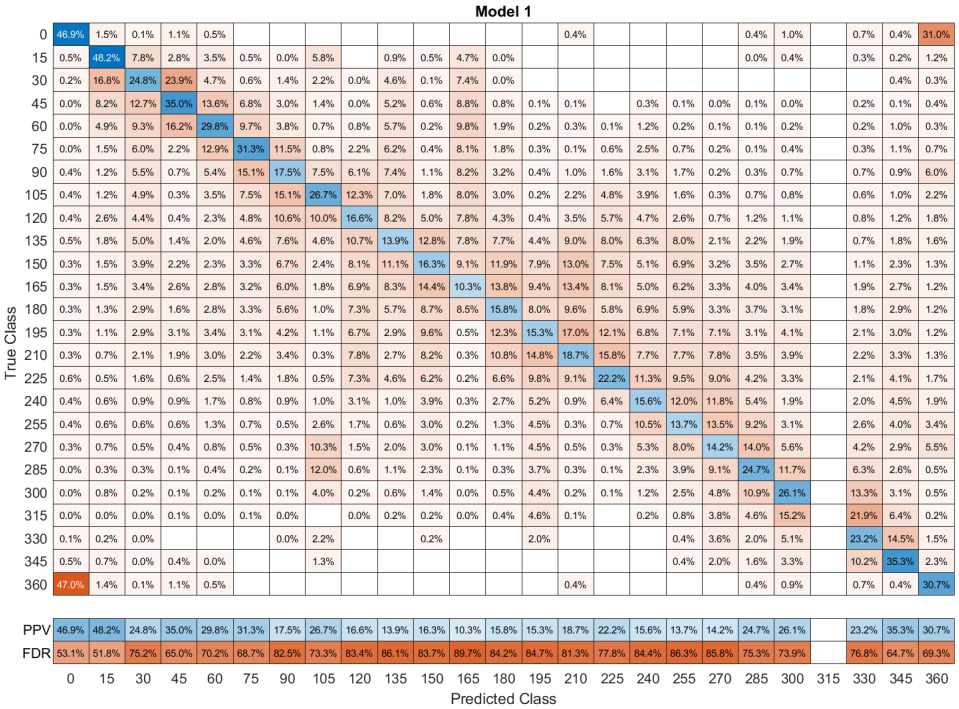


Figure D.3: Confusion matrix for Fine Decision Tree Classification model

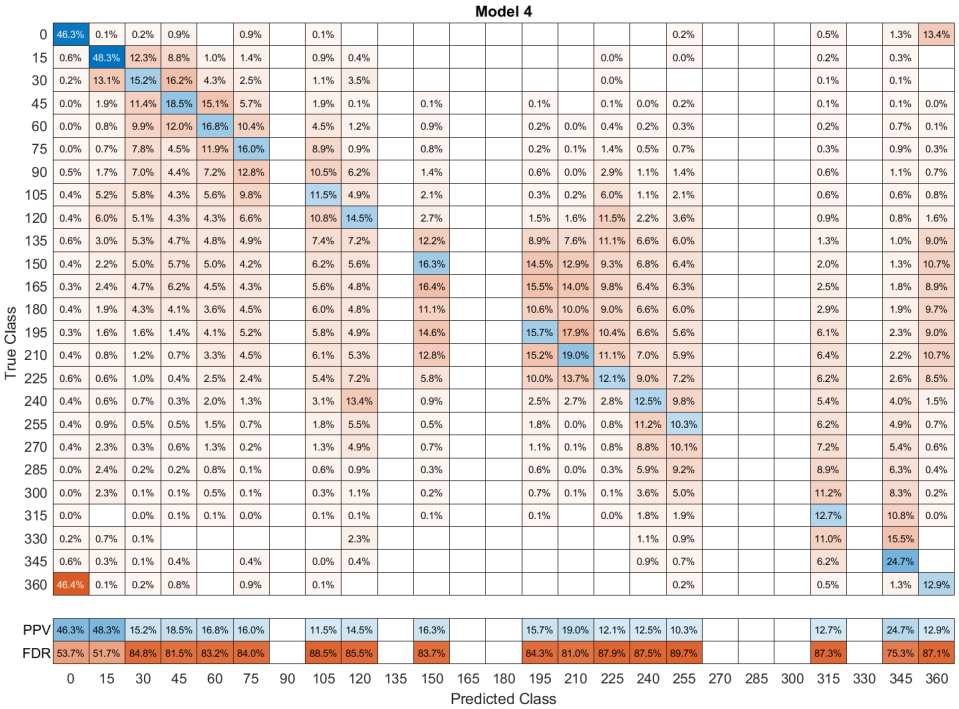


Figure D.4: Confusion matrix for Medium Decision Tree Classification model

Appendix E

Attachments

The attached zip-file, Code_Deliver, include the following:

- **Generated dataset**, which include developed dataset
- **Mat files**, all exported regression and classification models
- **Predict functions**, generated functions to make predictions on exported models
- **Python files** , multiple regression and classification algorithms
- **Simulator**, initialization files

To run the AUVsim follow these steps: (must have Matalb version 2020b or newer):

1. open *run_auv_simulator.m*
2. Include all the folder described above to the path
3. Run *run_auv_simulator.m*
4. Open Simulink simulator *AUVsim_.slx*. and run this file
5. Run *current_dir_estimation.m* to make predictions with the developed regression and classification models.

Ørjan Gjernes Riise

Current Estimation for Autonomous Underwater Vehicle using Supervised Learning

NTNU

Norwegian University of
Science and Technology