Master's thesis

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

Brynjar Abrahamsen

Fault Tolerant Dynamic Positioning for the Autonomous Test Platform ReVolt

Master's thesis in Marine Technology
Supervisor: Roger Skjetne
June 2019

NTNU
Kunnskap for en bedre verden

Brynjar Abrahamsen

Fault Tolerant Dynamic Positioning for the Autonomous Test Platform ReVolt

Master's thesis in Marine Technology
Supervisor: Roger Skjetne
June 2019

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

NTNU
Norwegian University of
Science and Technology

Abstract

A fault tolerant dynamic positioning system for the DNV GL model ship ReVolt has been developed, aiming to diagnose and accommodate sensor and actuator faults that may occur. Parts of the implementation seeks to explore non-traditional solutions, such as actuator fault diagnosis and signal outlier detection systems utilizing machine learning.

A module for signal fault tolerance has been implemented, to continually evaluate the health of navigational sensors and if necessary disable them should the behaviour match any known failure mode characteristics. In addition to hindering invalid sensor measurements from propagating through the control system, the module also serves as an alert and debugging tool for the operator. For this module, an navigational Extended Kalman Filter has been developed and tuned, allowing amongst other things dead-reckoning capabilities and residual evaluation for fault detection.

An actuator fault tolerance module powered by a recurrent neural network has been implemented. This component aims to diagnose and isolate malfunctioning thrusters, and disable them before the performance degradation jeopardizes the operation.

The performance of the fault tolerance modules are tested in different scenarios, featuring varying forms of faults and environmental disturbances. Finally, a qualitative evaluation of the system and its strengths and shortcomings is presented. Ideas for improving the system and proposals for further work are also mentioned.

Samandrag

Eit feiltolerant kontrollsystem for dynamisk posisjonering av DNV GLs modellskip ReVolt har vorte utvikla, med mål om å kunne diagnostisere og handtere sensor- og aktuatorfeil som kan oppstå. Delar av implementasjonen forsøker å utforske utradisjonelle løysingar, som t.d. bruk av maskinlæring for aktuator feildiagnostisering og deteksjon av ville observasjonar i signal.

Ein modul for signal feiltoleranse har vorte implementert, for å kunne kontinuerleg evaluere helsa til navigasjonsrelaterte sensorar og om naudsynt deaktivere sensorar om det liknar karakteristikken til kjende feiltypar. I tillegg til å hindre ugyldige sensormålingar å forplante seg utover resten av kontrollsystemet, vil modulen også fungere som eit alarm- og feilsøkingsverkty for operatøren. Eit navigasjonsbasert Extended Kalman Filter har vorte utvikla og kalibrert, for å muliggjere blant anna dead-reckoning og residual generering for feildeteksjon.

For å gjere aktuatorsystemet feiltolerant er det implementert ein modul som nyttar eit tilbakevendande nevralt nettverk, som er kopla til ein kontrollerbank med ulike thrustkonfigurasjonar. Hensikta til denne komponenten er å diagnostisere og isolere feil på thrusterar, samt deaktivering av slike aktuatorar før kapabiliteten til fartøyet å utføre dynamisk posisjonering vert uakseptabelt dårleg.

Kapabilitetane til det feiltolerante kontrollsystemet har vorte undersøkt i fleire ulike scenario, med variande grad av feil og miljømessige forstyrringar. Ei kvalitativ evaluering av systemet og konklusjon, med styrker og svakheiter, vert til slutt presentert. Også idear til forbetring av det endelege systemet vert foreslått, samt forslag til framtideg arbeid.

Preface

This thesis concludes my 5-year integrated Master's degree in Marine Technology at Norwegian University of Science and Technology (NTNU). The experience has been exciting and rewarding, with many unforeseen challenges arising along the way.

I would like to thank my supervisors, Roger Skjetne and Tom Arne Pedersen, for offering their valuable guidance throughout my thesis. The discussions and input from supervisors along the way has given me both motivation for further work and insight in the problems encountered. Furthermore, I also want to express my heartfelt thanks to family and friends who have supported me throughout the semester.

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

MSC THESIS DESCRIPTION SHEET

Name of the candidate: Abrahamsen, Brynjar

Field of study: Marine control engineering

Thesis title (Norwegian): Feil-tolerant dynamisk posisjonering for den autonome testplattformen ReVolt

Thesis title (English): Fault Tolerant Dynamic Positioning for the Autonomous Test Platform ReVolt

Background

Ability to demonstrate adequate safety and risk-management properties is paramount in order for autonomous vessels to become widely adopted. Faults at sea can be devastating if not handled properly, and this becomes all the more critical for autonomous ships with the absence of on-site technical personnel and human operators. In case of severe failure, additional risk could be averted by making the vessel hold position and counteract environmental forces acting upon it. Thus, dynamic positioning (DP) could serve as a fail-safe mode for autonomous vessels, provided the number of healthy actuators allows for it.

By implementing fault-tolerant control for DP, a vessel could show satisfactory stationkeeping performance despite developing faults onboard. This thesis will develop a fault tolerant DP control scheme for the autonomous demonstration platform ReVolt by DNV-GL. The scope considered in this thesis will be limited to signal and actuator faults. The developed system will include FTC modules for fault detection, fault diagnosis and isolation, and fault-handling. The final closed-loop FTC system should be able to operate resiliently when subject to relevant failure modes and environmental conditions.

The capabilities of the fault-tolerant control scheme shall be assessed through simulation using a digital twin version of the ReVolt.

Work description

1. Perform a background and literature review to provide information and relevant references on:
    - ReVolt and CyberSea simulator.
    - Fault-tolerant control for DP:
        - Signal fault-tolerance for surface vessels.
        - Actuator fault-tolerance for fully-actuated surface vessels.
    - Navigation filter/observer for position, velocity, and attitude (PVA).
    - Machine learning in fault-tolerant control.

    Write a list with abbreviations and definitions of terms, explaining relevant concepts related to the literature study and project assignment.

2. Design and tune an EKF model-based observer for position, velocity, and attitude (PVA) to be used in the signal fault-tolerance module.

3. Develop a signal fault-tolerant module for DP, with components for signal quality checking, voting, and averaging. The module should offer robustness with regards to:
    - Wildpoints / outliers.
    - Loss of signal/signal outage and signal freeze.
    - High variance noise.
    - Sudden signal drift and bias.

4. Implement a diagnosis and fault control scheme for the actuators/thrusters, which enables stationkeeping capabilities despite certain thruster faults by utilizing controller reconfiguration.

5. Simulate the fault tolerant DP system under different scenarios, with varying fault modes and environmental disturbances, and verify intended performance. Both single and multiple faults should be tested and discussed.

6. Compare with the DP system without FTC modules. Analyze the performance and discuss the results. Propose ideas for future work to improve upon.

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 it is not expected to be longer than 70 A4-pages, 100 B5-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, acknowledgements, thesis specification, list of symbols and acronyms, table of contents, introduction with objective, background, and scope and delimitations, main body with problem formulations, 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 an electronic copy to the main supervisor and department according to NTNU administrative procedures. The final revised version of this thesis description shall be included after the title page. Computer code, pictures, videos, dataseries, etc., shall be included electronically with the report.

Start date: January, 2019 **Due date:** As specified by the administration.

Supervisor: Roger Skjetne
Co-advisor(s): Tom Arne Pedersen (DNV-GL)

Trondheim, 10.04.2019

Roger Skjetne
Supervisor

Table of Contents

List of Tables

List of Figures

Abbreviations

AI	=	Artifical Intelligence
ANN	=	Artificial Neural Network
BN	=	Bayesian Network
CPU	=	Central Processing Unit
CUSUM	=	Cumulative Sum
DC	=	Direct Current
DNV GL	=	Det Norske Veritas Germanischer Lloyd
DP	=	Dynamic Positioning
DOF	=	Degrees Of Freedom
ECEF	=	Earth-Fixed, Earth-Centered
FD	=	Fault Detection
FDD	=	Fault Detection & Diagnosis
FT	=	Fault Tolerant
FTC	=	Fault Tolerant Control
GNSS	=	Global Navigation Satellite System
GPS	=	Global Positioning System
HDOP	=	Horizontal Dilution of Precision
IEEE	=	Institute of Electrical and Electronics Engineers
IMO	=	International Maritime Organization
IP	=	Internet Protocol
I/O	=	Input Output
IMU	=	Inertial Measurement Unit
LLA	=	Latitude, Longitude and Altitude
LSTM	=	Long Short-Term Memory
NED	=	North-East-Down (frame)
OCSVM	=	One Class Support Vector Machine
PID	=	Propotional Integral Derivative (controller)
PVA	=	Position, Velocity & Attitude
RANSAC	=	Random Sample Consensus
RNN	=	Recurrent Neural Network
ROS	=	Robot Operating System
SNAME	=	Society of Naval Architects and Marine Engineers
SVM	=	Support Vector Machine
USB	=	Universal Serial Bus
VDOP	=	Vertical Dilution of Precision

Nomenclature

$\{\mathrm{b}\}$	– Body Frame
x, y, z	– Body coordinates from the BODY frame
u, v, w	– Body-fixed linear velocities in the x, y and z axes
ϕ, θ, ψ	– Euler angles around the x, y and z axes
p, q, r	– Body-fixed angular velocities around the x, y and z axes
$\{\mathrm{n}\}$	– North-East-Down Frame
N, E, D	– North/East/Down position in the NED frame
$\{\mathrm{e}\}$	– Earth Centered, Earth Fixed Frame
l, μ	– Longitude, Latitude
$\boldsymbol{\eta}$	– Pose and attitude vector
$\boldsymbol{\tau}$	– Force and moment vector
$\mathbf{M}$	– Inertia Matrix
$\mathbf{M}_{\mathrm{RB}}$	– Rigid Body Inertia Matrix
$\mathbf{M}_{\mathrm{A}}$	– Added Mass Matrix
$\mathbf{C}$	– Coriolis & Centripetal Matrix
$\mathbf{C}_{\mathrm{RB}}$	– Rigid Body Coriolis & Centripetal Matrix
$\mathbf{C}_{\mathrm{A}}$	– Added Mass Coriolis & Centripetal Matrix
$\mathbf{D}$	– Damping Matrix
$\mathbf{D}_L$	– Linear Damping Matrix
$\mathbf{D}_{NL}$	– Nonlinear Damping Matrix
$\boldsymbol{\nu}$	– Velocity vector (linear and angular)
$\mathbf{p}$	– Position vector in the NED-frame
$\mathbf{b}$	– Bias vector
$\mathbf{q}$	– Quaternion vector
η	– Real component of a quaternion
$\boldsymbol{\epsilon}$	– Imaginary components of a quaternion
ΔT	– Time step
$\mathbf{I}$	– Identity matrix
$\mathbf{R}(\mathbf{q})/\mathbf{R}(\psi)$	– Rotation matrix
$\mathbf{r}$	– Residual vector
$\mathbf{F}$	– Thrust force vector
$\boldsymbol{\alpha}$	– Thruster angle vector
$\boldsymbol{S}()$	– Skew Symmetric Matrix
$\mathcal{N}$	– Normal (Gaussian) Distribution
$\mathbf{R}_{\mathrm{kalman}}$	– Noise Covariance Matrix
$\mathbf{Q}_{\mathrm{kalman}}$	– Process Noise Covariance Matrix
$\mathbf{H}_{\mathrm{k}}$	– Observation Matrix
$\mathbf{F}_{\mathrm{k}}$	– State Transition Matrix

Chapter 1

Introduction

This section will provide a background for the problem that shall be addressed, a description and scope of the thesis, specific contributions and a outline of the structure of the report.

1.1 Background

Dynamic positioning enables many forms of marine operations, such as subsea installation, drilling and diving. Customers, operators and class societies have high demands to reliability and fault tolerance of dynamic positioning systems, especially so in operations in close proximity to other vessels or structures like oil rigs.

Capability to conduct dynamic positioning is also interesting in the context of autonomous vessels. Depending on the minimal risk condition of the situation, additional risk could be averted by making a vessel in peril hold position and counteract the environmental forces acting upon it. Thus, dynamic positioning could in certain situations serve as a fail-safe mode for compatible autonomous vessels, which may prove to be very relevant in the future. The ability to demonstrate adequate safety and risk-management properties is absolutely paramount in order for autonomous vessels to become widely adopted. Faults at sea can be devastating if not handled properly, and this becomes all the more critical for autonomous ships in the absence of on-site technical personnel and human operators.

By implementing fault-tolerant control for dynamic positioning, a vessel could show a minimum of stationkeeping performance despite developing faults onboard – averting an out of control situation. Projects to improve the reliability and fault tolerance have been ongoing in the industry and academia since the birth of the

dynamic positioning technology. This thesis seeks to implement such a system on a real demonstration platform, and explore new approaches to actuator and sensor fault tolerance.

1.2 Scope of Thesis

The thesis will develop a fault tolerant DP control scheme for the autonomous demonstration platform ReVolt by DNV GL. The scope considered in this thesis will be limited to signal and actuator faults. The actuator faults have been limited to a drive-off scenario due to time and technical constraints. The final scheme should be able to operate when subject to different failure modes and environmental conditions. The capabilities of the fault-tolerant control scheme shall be assessed through simulation using a digital twin version of the ReVolt. Sea trials has not been conducted in addition to the simulations, as the physical model does not yet support the level of redundancy necessary for fault tolerant control.

1.3 Contributions

The thesis' main contribution is the development of a fault-tolerant dynamic positioning mode for the ReVolt autonomous demonstration platform. The implementation has had a holistic focus on fault tolerance, encompassing both the actuator and sensory dimension of the dynamic positioning problem, and the control system as a whole. Parts of the implementation seeks to explore non-traditional solutions, such as actuator fault diagnosis and signal outlier detection systems utilizing machine learning.
The final system is implemented in a ship control system using the Robotic Operating System, and simulated in various fault and environmental scenarios.

1.4 Outline

The report is formatted in the following order:

- Ch. 2 – Theoretical basis, with the backgrounds of methodologies used in the thesis. Includes a literature review of papers and dissertations relevant to the problem.
- Ch. 3 – Details of simulation set-up and experimental platform, along with previous work
- Ch. 4 – Description of implementation and methods, including control system design and software development
- Ch. 5 – Results from simulations and subsequent performance analysis
- Ch. 6 – Conclusion and recommendations for further work

A complete list of references is attached at the end of the report, along with appendices containing supplementary information. Relevant code has been submitted with the thesis.

Chapter 2

Theory and Methods

2.1 Fault Tolerant Control

Before delving into the field of fault tolerant control, it is important to distinguish between two terms, which will be used extensively throughout the thesis. A '*fault*' is defined by Skjetne and Egeland (2006) as a '*A defect in a system or component; e.g. a software bug or a short circuit in a component'* – note that it does not necessarily prohibit the overall system from working as intended. The same paper defines a 'failure' as '*The inability of a system or component to perform its required functions within specified performance requirements*'. In other words, an improperly handled fault may develop into a failure, which may have grave consequences for the system in question. A way to mitigate malfunction risks in control systems is the implementation of fault tolerant control schemes, which purpose is defined by Blanke et al. (2006) as to '*prevent a fault from causing a failure at the system level*'.

Fault tolerant control may further be divided into three unique substeps, as illustrated in the block diagram of Figure 2.1. Fault detection is most often regarded as the first step in a holistic system for fault tolerant control system design, where a fault diagnosis system subsequently identifies the size, kind and location of a fault based upon the information from the fault detection block. What happens next differ from system to system, as some may incorporate an open-loop procedure where human operators are alarmed about a fault including its diagnosis, and are ought to rectify it manually. However, in a complete fault-tolerant system, a fault-handling system will act upon the information from the fault diagnosis block to ensure adequate performance by exploiting controller redesign and physical redundancies without human intervention (Blanke et al., 2006).

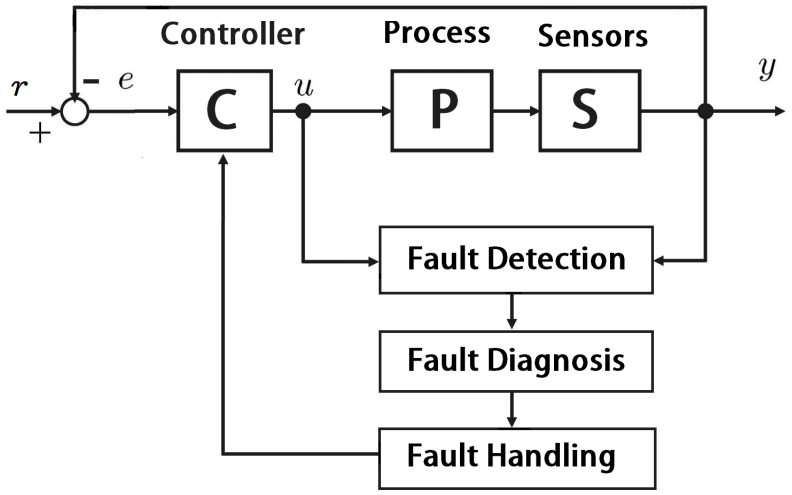


Figure 2.1: Illustration of the individual parts of a simplified fault-tolerant control system.

Fault Detection

The purpose of the fault detection component is to affirm that a fault has happened, and liaison this information to the diagnosis block along with a timestamp of detection. The block does not seek to find the cause nor exact location of the fault, as those are objectives of subsequent modules.

Several differing approaches have been proposed for fault detection, most which can be grouped either in the family of model-based or data-driven methods (Alauddin et al., 2018). Model-based fault detection has been and is still widely used, especially in situations where the system dynamics are well-understood and relatively predictable. However, model-based approaches may fall short in highly complex systems, where the dynamics can be more uncertain or hard to model rigorously. An alternative methodology for such cases is using a data-driven approach, which does not take into account the internal system dynamics - but rather relies purely on sensory data output (Freeman et al., 2013). Combinations of these two also exist, exploiting the advantages of both methodologies.

Fault Diagnosis

Isolating the fault is a crucial step of the diagnosis block, as pinpointing the location of the fault is important to be able to rectify or alternatively deactivating the faulty component. Identifying the type of fault is another goal of the fault diagnosis

block, along with estimating the magnitude of the fault. The latter is important, as the severity can affect how the fault should be handled, for instance could fault magnitude mean the difference between partial and full disablement of a thruster in the topic of dynamic positioning.

Fault diagnosis can be split into two separate families of methodologies, namely classification and inference methods (lsermann, 2006). Classification is a stochastic approach to diagnosis, using statistical classification and pattern recognition techniques, including machine learning and decision trees. Inference methods uses a more deterministic approach, where diagnosis is based on logical reasoning and utilizes if-then rules extensively in a similar fashion to an expert system.

Fault Handling

Fault handling is the final step in a fault-tolerant control system, where the fault will be mitigated in the best way possible to prevent performance degradation of the overall system. Controller redesign is a common procedure that aims to change the behaviour of the system by either switching to a different control law or setting new controller parameters.

Accommodation of faults relies on redundancies already present in the system, whether these are physical (hardware) redundancies or analytical redundancies (lsermann, 2006). The latter may be provided by virtual sensors, observers, or reconstruction of the state through process models and different measurements. A fault handling system should exploit the tools available through the system architecture, whether it offers physical or analytical redundancy. Provided the system architecture allows it, exploiting physical redundancies is often the easiest way to handle a fault, e.g. activating a stand-by engine in event of an engine fault. Sensor faults may often be mitigated by an analytical redundancy, through producing state estimates through an accurate model or a filter running in dead-reckoning mode.

2.1.1 Model-Based Methods

Model-based methods are based on running a mathematical model of the system in parallel with the physical system, enabling measured and model-derived signals to be compared (Blanke et al., 2006). Among the state-of-the-art model-based fault detection schemes today is residual generation and structural analysis.

Residual Generation

By constructing an accurate model of the plant and feeding it the same control input u and measurements y from the real-life system, an estimate of the state in each time instant may be produced. A residual vector $\mathbf{r}$ is generated when subtracting the actual measured values from estimated measurements, as in Equation 2.1.

$$\mathbf{r} = \mathbf{y} - \hat{\mathbf{y}} = \mathbf{y} - \mathbf{C}\hat{\mathbf{x}} \tag{2.1}$$

where $\mathbf{C}$ is the output matrix for transforming state estimates $\hat{\mathbf{x}}$ to measurement estimates $\hat{\mathbf{y}}$. A residual value reaching a significant value and not vanishing within a reasonable time can be indicative of a fault – given that the model and input signals are scrupulous. The latter presumption is also the weakness of the method, as modelling errors and sensor malfunctions may result in false alarms, or false negatives which can be arguably worse.

Nevertheless, the method of residual generation for fault detection is potent and well-established, especially for systems where accurate plant models are available. Various techniques for generation residuals includes using state observers, parity equations or parametric estimation.

Blanke et al. (2006) postulates two desired properties for a linear time-invariant filter to be used as a residual generator:

- The sequence of output residual values $r(k)$, $k = 1, 2.., n$ is a zero mean white noise vector sequence which is not affected by control input u nor disturbances d, once the transient due to initial conditions has vanished.

- In the presence of a fault ($f(k) \neq 0 \; \forall \; k \geq k_0$) the mean of $r(k)$ is different from zero for at least some $k \geq k_0$.

Zhang et al. (2008) proposed using an Extended Kalman Filter to produce credible estimates of the states of a propulsion system, and feeding the output into a threshold based hypothesis test to decide whether or not a fault had occurred. Hassani et al. (2018) proposes a fault detection approach for position-mooring systems using a similar scheme. A bank of different Kalman filters provides online model parameter estimates, which in turn are evaluated by a dynamic hypothesis testing algorithm to detect changes in plant dynamics indicating mooring breakages.

Structural Analysis

Structural analysis is a comprehensive methodology that can not only detect faults, but also identify and with sufficient redundancy handle the fault (Blanke et al., 2006). It is based on graph theory, and thus circumvents the detailed modelling needed in a residual-based approach. Structural analysis can yield lists of detectable faults, and a subset of isolatable faults. Combining residual generation and structural analysis is popular, as conducting structural analysis may generate suggestions of residual generators (Blanke and Staroswiecki, 2007).

2.1.2 Data-Driven Methods

In recent years there has been an increasing interest in fusing fault detection with modern data analysis tools, in fields such as wind energy (Stetco et al., 2018). The advantage of a purely data-driven methodology is that it does not need a model to detect faulty situations, and the algorithms used may detect unknown faults that is not explicitly covered by a plant model. Additionally it has been observed that data-driven approaches may detect anomalies which may go unnoticed or unmodeled by conventional designs (Freeman et al., 2013).
Among the publications on data-driven methodologies for fault detection, there has been an increasing interest in applying Bayesian Networks (BN), Artificial Neural Networks (ANN), and Support Vector Machines (SVM) algorithms over the last decade (Alauddin et al., 2018). The two latter are supervised machine learning algorithms, that are based on creating learning models using historical datasets which have been binary labelled as either faulty and non-faulty.

In Bo et al. (2018) the authors explored the potential of machine learning in estimation of hydrodynamical thruster losses, in order to prevent severe thrust degradations during dynamic positioning.

2.2 Signal Fault Tolerance

A signal may be subject to a multitude of different failure modes, with varying consequences for the overall system reliant on the signals. Creating a fault tolerant signal processing system is thus important to avoid having signal malfunctions propagate throughout the control system and potentially jeopardize the operation of the vehicle.
The following section will characterize important failure modes and ways that detect, diagnose and potentially rectify them:

Signal Dropout

Signal dropout refers to both momentary and permanent loss of communication, when the system would otherwise expect transmission of a signal.

By timestamping signals it is possible to detect a signal drop out by calculating the difference between the timestamp and current time according to the main computer unit. If this discrepancy grows larger than a reasonable threshold, it may be indicative of a signal dropout. Note that the actual cause of the signal dropout can not be directly deduced from this fact alone, which may be related to instrumentation fault or wire breakage for instance.

Bias

Bias is a constant additive offset that skews the true value of the underlying signal. This phenomenon may occur in uncalibrated sensors and signals with constant DC bias. The bias will not impact the variance of the stochastic model, as it is a constant. However, the mean value of the distribution will be affected by the bias. The characteristic signal displaying bias may be modelled as having a Gaussian distribution with a normal variance, but with a constant additive mean value – as shown in Equation 2.2.

$$z \sim \mathcal{N}(z_{\text{true}} + z_{\text{bias}},\, \boldsymbol{\sigma}^2) \tag{2.2}$$

The cumulative sum (CUSUM) algorithm is a statistical method for detecting an underlying change in some property, for instance a change in the mean value of an underlying distribution (Blanke et al., 2006). This is helpful in detecting suddenly occuring bias as it manifests itself as a change in the mean of the signal distribution. Expected measurement values from a model is required to conduct CUSUM testing, such as the residuals from a Kalman Filter.

Signal Drift

A signal displaying drift behaviour has a similar characteristic as a signal with bias, as there is an additive constant while the noise spectral density remains constant. However, a signal with drift will have time-varying bias. The drifting may have origins in a deterministic or stochastic process, such as a Wiener process. A stochastic model of signal drift is shown in Equation 2.3.

$$z \sim \mathcal{N}(z_{\text{true}} + z_{\text{drift}}(t),\, \boldsymbol{\sigma}^2) \tag{2.3}$$

As with the bias failure mode, CUSUM testing may be utilized to detect signal drift scenarios. As the stochastic model of the failure mode suggests, there is a change in the mean of the distribution, which the algorithm may be able to detect.

Frozen Signal

A signal freeze is characterized by continuous transmission of the exact same measurement, hence the signal will have zero variance. There will always be some form of measurement noise present in a sensor, while a frozen signal will not exhibit a changing measurement noise – as modelled in Equation 2.4. Thus the signal freeze failure mode can be inferred by observing zero variance over some window of consecutive samples.

$$z \sim \mathcal{N}(z_{k-1}^{\text{true}},\, 0) \tag{2.4}$$

Special precaution has to be made when checking for freeze behaviour in signals that have been converted from analog to digital. This is because quantization errors may result in slight changes in the resulting digital signal, contraindicative of a freeze despite the analogous signal being frozen.

Outliers

An outlier is a spurious signal the deviates considerably from the expectation and previous measurements. Characterized by a standard deviation exceeding the normal value, it can be modelled as in Equation 2.5.

$$z \sim \mathcal{N}(z^{\text{true}},\, \boldsymbol{\sigma}_{outlier}^2) \tag{2.5}$$

Differentiating between outliers and authentic large jumps in signals may be difficult, and several differing procedures has been proposed for the problem. The Random Sample Consensus (RANSAC) algorithm which was spawned out of the image processing community, has been shown to be a versatile way of detecting outliers in other applications. Using machine learning algorithms for outlier detection has also been proposed, such as the One-Class Support Vector Machine (OCSVM) method (Scholkopf et al., 2000).

High Variance

High variance is a special failure mode where the noise spectral density is increased substantially for some period of time. In contrast to outliers where spurious measurements are the odd ones out among true signals, the failure mode may result in many noisy measurements subsequently over a period of time. The failure mode can be modelled as in Equation 2.6.

$$z \sim \mathcal{N}(z^{\text{true}},\, \boldsymbol{\sigma}_{high}^2) \tag{2.6}$$

An example of the origin of this failure mode can be electromagnetic interference, where components and machinery close to inadequately isolated cables may corrupt the signals being transmitted. There may also be legitimate causes for high variances, not indicative of a fault, such as high dynamics due to heavy seas – and this should be reflected in the implementation (Sørensen, 2013).

Summary

The types of failure modes reviewed can be seen in Table 2.1, along with their respective stochastic models and proposed rectification.

Table 2.1: Signal Failure Mode Characterizations

Failure Mode	Proposed Remedy	Statistical Model
Dropout	Timestamp Checks	$\mathcal{N}(NaN,\, 0)$
Bias	CUSUM	$\mathcal{N}(z^{\text{true}} + z^{\text{bias}},\, \boldsymbol{\sigma}^2)$
Drift	CUSUM	$\mathcal{N}(z^{\text{true}} + z^{\text{drift}}(t),\, \boldsymbol{\sigma}^2)$
Freeze	Variance Testing	$\mathcal{N}(z^{\text{true}}_{k-1},\, 0)$
Outlier	OCSVM	$\mathcal{N}(z^{true},\, \boldsymbol{\sigma}^2_{outlier})$
High Variance	Variance Testing	$\mathcal{N}(z^{\text{true}},\, \boldsymbol{\sigma}^2_{high})$

2.3 Kinematics and Modelleing

Three geographical reference frames will be used extensively in the report, namely the Earth-Fixed Earth-Centered (ECEF) frame, North-East-Down frame (NED) and the BODY frame. The three frames are illustrated in Figure 2.2. The ECEF

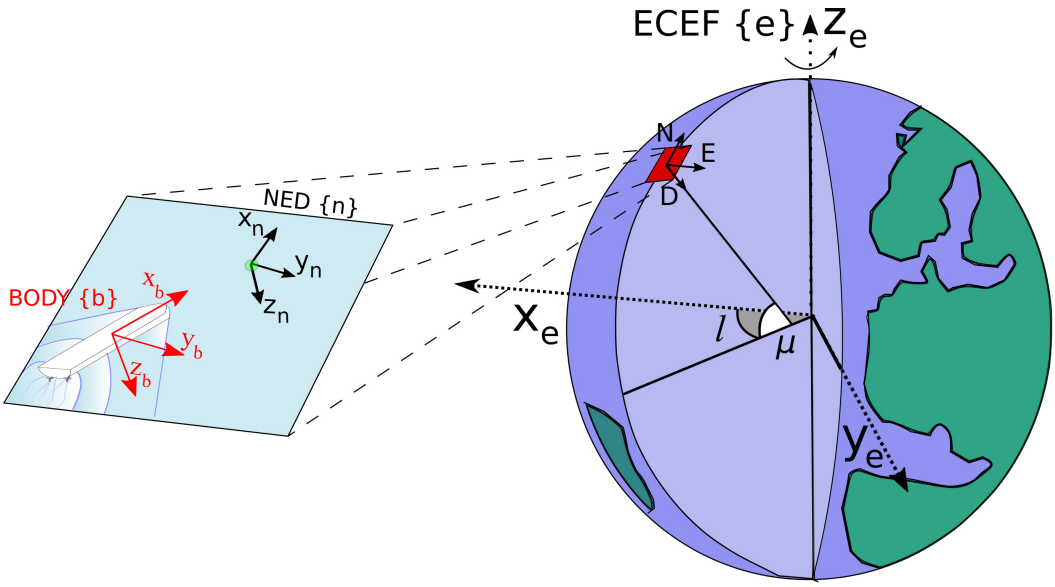


Figure 2.2: Illustration of different reference frames.

frame rotates with the Earth and has its origin in the centre of the planet, and the frame can easily be transformed to geodetic coordinates (latitude and latitude). The NED frame is a geographic reference frame, where the x-axis points to the true North, y-axis points East and z-axis points normal to the surface of the Earth. The frame can be defined as a local tangent plane situated at the surface of the ellipsoid that is Earth. The BODY frame is a moving reference frame that follows the vessel. Neither the NED or BODY frame are inertial frames, as they do not account for the angular rotation of the Earth. In some control systems, this has to be compensated by introduction of fictitious centripetal and Coriolis forces in the equation of motion. However, for stationkeeping purposes it may be safe to omit this compensation, as the geographical area of operation is relatively small (Fossen, 2011).

2.3.1 Horizontal plane model for ships

The kinetical notations in this thesis follow the 1950 SNAME convention, shown in Table 2.2, and their vectorial counterparts defined by Fossen (2011) in Equation 2.7.

Name	*Forces & Moments*	*Linear Velocities & Angular Rates*	*Pose & Attitude*
Surge	X	u	x
Sway	X	v	y
Heave	Z	w	z
Roll	K	p	ϕ
Pitch	M	q	θ
Yaw	N	r	ψ

Table 2.2: Overview of the 1950 SNAME convention of kinetics

$$
\begin{array}{llll}
\text{ECEF position} & \boldsymbol{p}^{e}_{b/e} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} & \text{Longitude and latitude} & \boldsymbol{\Theta}_{en} = \begin{bmatrix} l \\ \mu \end{bmatrix} \\
\text{NED position} & \boldsymbol{p}^{n}_{b/n} = \begin{bmatrix} N \\ E \\ D \end{bmatrix} & \text{Attitude (Euler angles)} & \boldsymbol{\Theta}_{nb} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \\
\text{Body-fixed linear velocity} & \boldsymbol{v}^{b}_{b/n} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} & \text{Body-fixed angular velocity} & \boldsymbol{\omega}^{b}_{b/n} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \\
\text{Body-fixed force} & \boldsymbol{f}^{b}_{b} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} & \text{Body-fixed moment} & \boldsymbol{m}^{b}_{b} = \begin{bmatrix} K \\ M \\ N \end{bmatrix}
\end{array}
\tag{2.7}
$$

Following the complete maneuvering model for surface vessels, a total of six coordinates are needed to describe attitude and position. The six degrees-of-freedom (DOF) motion of a vessel can be established by the vectors shown in Equation 2.8

(Fossen, 2011):

$$\boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{p}^n_{b/\prime} \\ \boldsymbol{\Theta}_{nb} \end{bmatrix} \qquad \boldsymbol{\nu} = \begin{bmatrix} \boldsymbol{v}^b_{b/n} \\ \boldsymbol{\omega}^b_{b/n} \end{bmatrix} \qquad \boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{f}^b_b \\ \boldsymbol{m}^b_b \end{bmatrix} \tag{2.8}$$

where η is the position and attitude vector, ν is the velocity vector containing linear and angular velocities, and τ is a kinetic vector containing forces and moments. The 6-DOF axes in the body frame are shown and labelled in Figure 2.3.

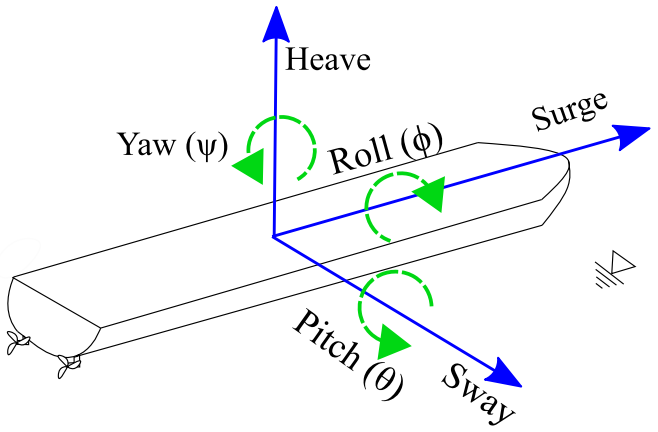


Figure 2.3: A vessel displaying the six degrees of freedom of the full maneuvering model

However, the configuration space can be reduced for most surface ships by using the horizontal plane model. Under this assumption, the vessel can be modelled with only three degrees of freedom – surge, sway and yaw (Fossen, 2011). This simplification does not introduce substantial errors for displacement vessels in low-speed applications such as dynamic positioning. The 3-DOF equations of motion can be written in vectorial form as in Equation 2.10.

$$\dot{\boldsymbol{\eta}} = \mathbf{R}(\psi)\boldsymbol{\nu} \tag{2.9}$$

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau}_{\text{actuators}} + \boldsymbol{\tau}_{\text{environment}} \tag{2.10}$$

where $\boldsymbol{M}$ is the mass matrix, composed of rigid-body mass and inertia matrix $\boldsymbol{M}_{RB}$ and the added-mass matrix $\boldsymbol{M}_A$, as in Equation 2.11. The added mass terms stem from hydrodynamics, due to the vessel interacting with a non-massless body of water in vicinity of the hull.

$$\boldsymbol{M}_{RB} + \boldsymbol{M}_A = \begin{bmatrix} m & 0 & 0 \\ 0 & m & x_g m \\ 0 & x_g m & I_z \end{bmatrix} + \begin{bmatrix} -X_{\dot{u}} & 0 & 0 \\ 0 & -Y_{\dot{v}} & -Y_{\dot{r}} \\ 0 & -Y_{\dot{r}} & -N_{\dot{r}} \end{bmatrix} \tag{2.11}$$

The $\boldsymbol{C}$ matrix accounts for the rotational effects attributed to Coriolis and centripetal fictitious forces, stemming from non-inertial property of the NED frame.

$$\boldsymbol{C}(\boldsymbol{\nu})_{RB} + \boldsymbol{C}(\boldsymbol{\nu})_A = \begin{bmatrix} 0 & 0 & -m(rx_g + v) \\ 0 & 0 & mu \\ m(rx_g + v) & -mu & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & Y_{\dot{v}}v + Y_{\dot{r}}r \\ 0 & 0 & -X_{\dot{u}}u \\ -Y_{\dot{v}}v - Y_{\dot{r}}r & X_{\dot{u}}u & 0 \end{bmatrix}$$

The damping terms are contained in the $\boldsymbol{D}$ matrix, and are composed of a linear damping term $\boldsymbol{D}_L$ and non-linear term $\boldsymbol{D}(\boldsymbol{\nu})_{NL}$. Dynamic positioning is characterized by low speed (meaning $\boldsymbol{v} \approx 0$), which makes the nonlinear damping term negligible as the linear term dominates.

$$\boldsymbol{D}_L + \boldsymbol{D}(\boldsymbol{\nu})_{NL} = \begin{bmatrix} -X_u & 0 & 0 \\ 0 & -Y_v & -Y_r \\ 0 & -N_v & -N_r \end{bmatrix} + \begin{bmatrix} -X_{|u|u}|u| & 0 & 0 \\ 0 & -Y_{|v|v}|v| - Y_{|r|v}|r| & -Y_{|v|r}|v| - Y_{|r|r}|r| \\ 0 & -N_{|v|v}|v| - N_{|r|v}|r| & -N_{|v|r}|v| - N_{|r|r}|r| \end{bmatrix}$$

2.4 Dynamic Positioning

Control systems for simultaneous control of the 3 DOF motions (surge, sway and yaw motion) onboard vessels and rigs are known as dynamic positioning systems (Sørensen, 2013). An example of a dynamic positioning control system is shown in Figure 2.4.

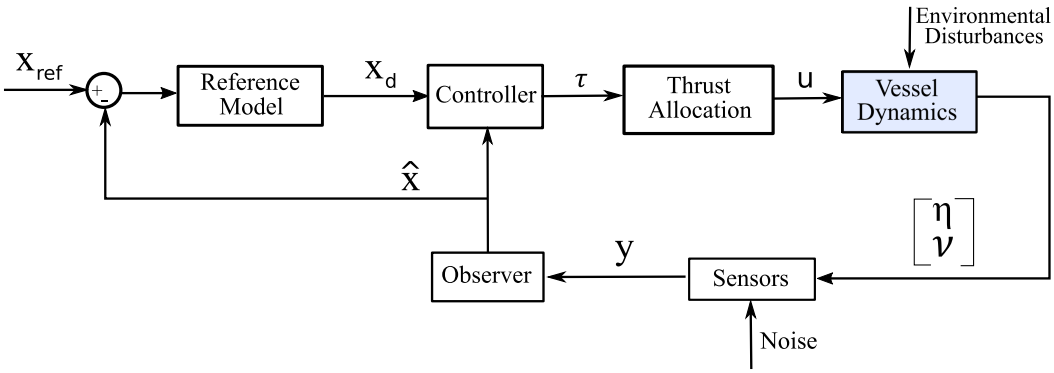


Figure 2.4: Block diagram of a rudimentary dynamic positioning system

An operator or a high-level control system gives a reference pose and attitude for the vessel to attain. The aim of the reference model is to generate a well-fitting

trajectory for the vessel to follow, without sudden setpoint changes resulting in step inputs to the control system. The observer may have several functions, some which will be explained in the section on the Extended Kalman Filter. However, the primary role of the observer in a DP system is to provide state estimates from noisy measurements. The controller compares the desired state from the reference model and the estimated state from the observer, and outputs a vector containing the wanted forces and moments to reach the setpoint.
One of the potentially most complex blocks is the thrust allocation, described in the next section.

2.4.1 Thrust Allocation

The aim of the thrust allocation is to arrive at a distributed control input vector $\mathbf{u}$ for the actuators by considering the $\boldsymbol{\tau}$ vector containing the desired forces and moments. There can be multiple different types of actuators on a vessel, ranging from control surfaces such as fins and rudders to dedicated propulsion units like azimuth thrusters and main propellers. Regardless of type, the actuators all share a common a goal in providing the forces and moments desired by the controller. The thrust produced by an actuator may be modelled as a linear relation between a force coefficient and control input, resulting in Equation 2.12: (Fossen, 2011).

$$\mathbf{f} = \mathbf{K}\mathbf{u} \tag{2.12}$$

where $\mathbf{u}$ is the control input matrix, and $\mathbf{K}$ is a diagonal matrix containing force coefficients. The resulting $\mathbf{f}$ matrix contains forces produced by each actuator. Equation 2.12 does not provide a relation between the generalized force vector $\boldsymbol{\tau}$ and control inputs $\mathbf{u}$. This is mended by introducing a matrix containing possible thruster angles $\boldsymbol{\alpha}$ and torque arms $\mathbf{l}$, as seen in Equation 2.13.

$$\boldsymbol{\tau} = \mathbf{T}(\boldsymbol{\alpha})\mathbf{f} = \mathbf{T}(\boldsymbol{\alpha})\mathbf{K}\mathbf{u} \tag{2.13}$$

The previously introduced thrust configuration matrix is named $\mathbf{T}$, with dimension $\mathbb{R}^{n\times r}$ where n is equal to the degrees of freedom for the vessel and r is equal to the number of actuators. The elements of $\mathbf{T}$ corresponding to rotational thrusters are dependant on the angular position $\boldsymbol{\alpha}$ of the thruster. Control inputs and possibly thruster angles may now be computed by utilizing the given generalized desired forces vector $\boldsymbol{\tau}$ and Equation 2.13.

In vessels designed for DP use, the amount of actuators often exceeds the degrees of a freedom considered, which results in multiple to potentially infinite solutions to the allocation problem. For actuators capable of rotation the amount of possible

configurations increases further and thus also the number of valid solutions. The thrust allocation problem thus becomes a model-based optimization problem, where the designer may for instance aim to minimize fuel consumption while fulfilling a set of constraints posed by the operation and actuators involved. The main constraint is posed by the generalized forces and moment vector τ, while additional constraints can be shaft speed limits, rotational rate limits and/or forbidden angles to prevent thruster-thruster interaction among others.

2.4.2 Fault-Tolerance in Dynamic Positioning

There is a significant industrial interest in improving the fault-tolerance of dynamic positioning systems in ships and floating rigs. Classification societies have strict requirements to redundancy and survivability of crucial components used in DP

Hansen (2011) reviews different means to enhance the dependability of DP systems, and classifies fault severity as a function of time spent without DP capability. The report classifies 10 seconds as the maximum time limit of sustained failure before reaching an irreversible and hazardous state, not unlikely to result in damages and cost related to off-hire, operation delays and examination from authorities and so forth. The value is approximate and would in reality take variables like environmental conditions, vessel and operation type into account. Nevertheless, it gives an impression of the high reliability demands of vessels conducting DP and the motivation behind the stringent redundancy requirements of the class societies.

During drift-off, the thruster signals either freeze or approach zero, resulting in the vessel drifting away from the area. In a drive-off situation, the vessel erroneously commands an excessive or even maximal thrust, resulting in the ship leaving the operations area unwillingly. Chen and Moan (2004) lists causes of drive-off situations in shuttle tankers, and mentions amongst them DP control system bugs and software freezes as ones of the culprits behind actual accidents and near misses.

Blanke (2005) examined a specific case of fault-tolerant control applied to a ship stationkeeping scenario, where the vessel would suffer both actuator and sensor faults. The final system was shown to perform stationkeeping adequately, even in an under-actuated case. It should be noted that the author defines a stationkeeping operation as a low precision variant of dynamic position, with correspondingly lower expectations towards setpoint deviations.

2.5 Machine Learning

Kohavi and Provost (1998) defines machine learning as "the field of scientific study that concentrates on induction algorithms and on other algorithms that can be said to *learn*", where the learning refers to the ability to improve performance gradually as the amount of data is increased. An induction algorithm is an algorithm that produces a model that is generalizable outside of the training data that it has been inputted (Kohavi and Provost, 1998).

To construct a machine learning model, a dataset is usually split in three independent sets – the largest being a *training set*, which the model will try to fit, a *validation set* which is used subsequently to evaluate and select the model with the lowest error. Finally, the proposed model based on the training and validation set is evaluated with regards to the *test set*, which is also independent and used as a performance indicator for how generalizeable the final model is. This last step is to decrease the risk of overfitting, which is a caveat of all machine learning algorithms. A model that is overfitted can predict the training set close to perfectly, but is not versatile enough to perform well evaluating data sets that differ from the training set. This may occur if the data lacks the diversity or size required for truly generalizeable. Thus a model that reports an extraordinary accuracy is in most cases too good to be true. Another weakness of machine learning algorithms is its reliance on vast amounts of data, which may sometimes prove difficult to produce and time consuming, especially so in the case of supervised machine learning.

The field is traditionally divided in three categories, namely supervised learning, unsupervised learning, and reinforcement learning. Supervised algorithms requires that the training data is labelled, meaning the input variables needs to form pairs with a corresponding output value. The supervised methods can be used for either regression or classification problems, where the former gives a continuous output and the latter gives a discrete output belonging to a set of predefined categories (Pedregosa, 2011). Unsupervised algorithms does not require labelled training data, and will rely on identifying the existence or absence of commonalities in data instead of responding to feedback.

The topic of fault classification in the most primal form is treated as a binary classification problem, where the datapoints are classified as either faulty or non-faulty. A popular performance metric for these types of problems is the F_1 score, which is an assessment of the accuracy of a model. As seen in Equation 2.14 it is calculated on the basis of the *recall* and *precision* values. The precision value is calculated by dividing the number of true positives by the sum of both true

and false positives, e.g. in a fault detection and diagnosis model this would be the number of correctly identified faults divided by the total number of presumed faults. The recall value is calculated by dividing the true positives by the number of both true positives and false negatives, in other words for a FDD system it would be beneficial to have a recall value close or equal to one, to ensure all faulty situations are correctly identified. The F_1 measure thus combines both the recall and precision measures.

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.14}$$

2.5.1 Support Vector Machine

The SVM algorithm works by creating a set of hyperplanes, whose purpose is to divide the datapoints into the correct classes and keep the distinct classes as far away from each other as possible. Future datapoints added to the model are then mapped into the feature space, and classified accordingly to what side of the hyperplane they lie on.

The *support vectors* themselves are the datapoints that lie closest to the dividing hyperplane. The objective of the SVM algorithm is to maximize the distance from the hyperplane to the closest datapoint, which is called the margin. There may be multiple hyperplanes that separate the different classes of the training data perfectly. However, this does not mean they have the same qualities – as the margin may be different for each hyperplane (Kulkarni and Harman, 2011). By making the margin as large as possible, the model will be more generalizable and resistant to overfitting, thus more likely to correctly classify future datapoints outside the training set.
Mathematically a hyperplane can be expressed as in Equation 2.15, where $\mathbf{w}$ is a weight vector, $\mathbf{x_i}$ is a datapoint, and b a bias, that is

$$\mathbf{w^T x_i} + b = 0 \tag{2.15}$$

The concept of classifying data using dividing hyperplanes is depicted in Figure 2.5:

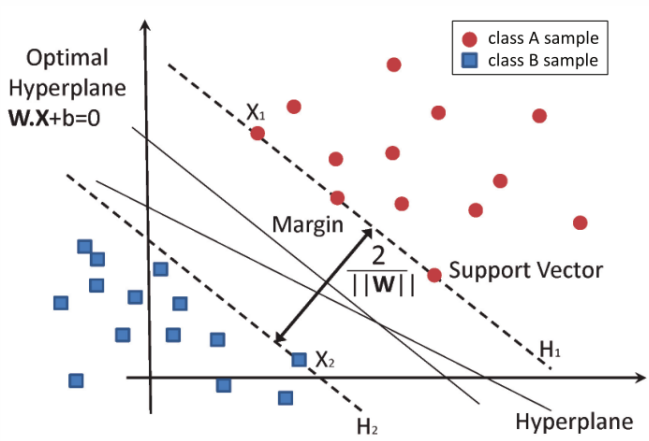


Figure 2.5: A set of binary classified datapoints, separated by two hyperplanes. The optimal choice here is the plane orthogonal to the support vectors, as it maximizes the margin $\frac{2}{||w||}$. From: García-Gonzalo et al. (2016)

However, the case of Figure 2.5 is an ideal situation, as the training datapoints will often not initially be linearly separable. The solution requires a transformation to a higher dimensional space, where separation might be possible – this is one of the key ideas of the SVM algorithm, called the *kernel trick*. By mapping the feature vectors using a nonlinear function to a higher dimension, an ordinary linear classifier may be used. When transformed back to the original feature space, the formerly linear classifier used in the higher dimensional feature space will become nonlinear.
An example of a kernel trick is shown in Figure 2.6. Here, the transformation shown in Equation 2.16 has been used, which yields an additional z-dimension to the feature space.

$$\phi(x_1, x_2) = (x_1, x_2, x_1^2 + x_2^2) \tag{2.16}$$

An ordinary linear hyperplane may now be used to classify the datapoints.

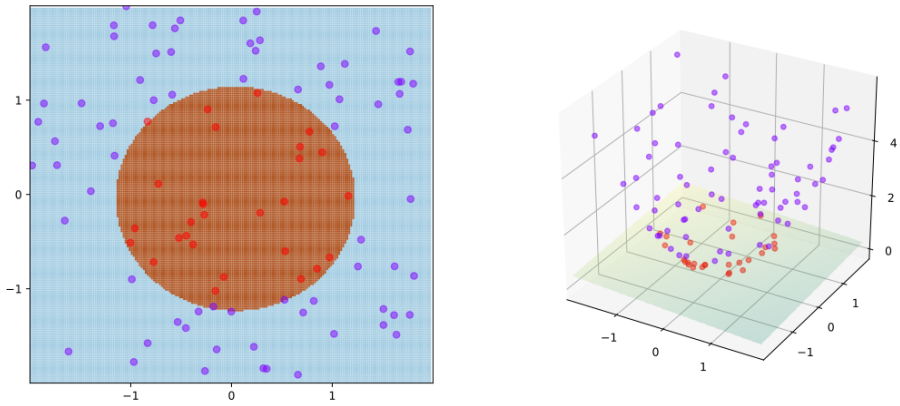


Figure 2.6: Illustration of a kernel trick, where a two-dimensional feature space (left) is transformed to a three-dimensional one (right). From: Shiyu (2017)

.

One-class Support Vector Machine

The One-Class Support Vector Machine (OCSVM) is a special case of the SVM algorithm, which considers only one single class in contrast with the usual classification problems concerning multiple different classes. Additionally it does not rely on supervision during training, meaning the dataset(s) used for training does not need to be labelled for the algorithm to construct an classification model. A relevant use case for the one-class implementation is outlier detection, where the objective is to establish whether a data point is part of the class of valid ”inlier” points or not.

While the algorithm is unsupervised, it still requires some a priori estimate of the amount of outliers in form of the ν parameter (Scholkopf et al., 2000). This value may be between 0 to 1, and is understood as the upper bound on the fraction of outliers – hence a large ν would correspond to data containing a large amount of outliers, and a model using a lower ν would be conservative in labelling a new datapoint as an outlier.

2.5.2 Artificial Neural Networks

One of the most widely used forms of machine learning algorithms is the family of artificial neural networks. They draw inspiration from biological neural networks, although they differ from contemporary understanding of how the human brain

works.

Akin to the brain, the network is formed by interconnected neurons, which form the backbone of the algorithm. The neurons are sorted in layers, and the neurons may transmit signals from one layer to the next through what is called 'edges'. There are three kinds of layers in a conventional neural network setup, namely input, output and hidden layers. Only the input and output layers are tangible to the user, which must specify their dimensions and choose what features to use as input. The purpose of the hidden layers is to transform the inputted data to an output value, for instance the probability that a given datapoint is part of some class X. An example of a neural network is illustrated in Fig 2.7, where there are two hidden layers – making it a deep learning algorithm by definition, as there is more than one hidden layer.

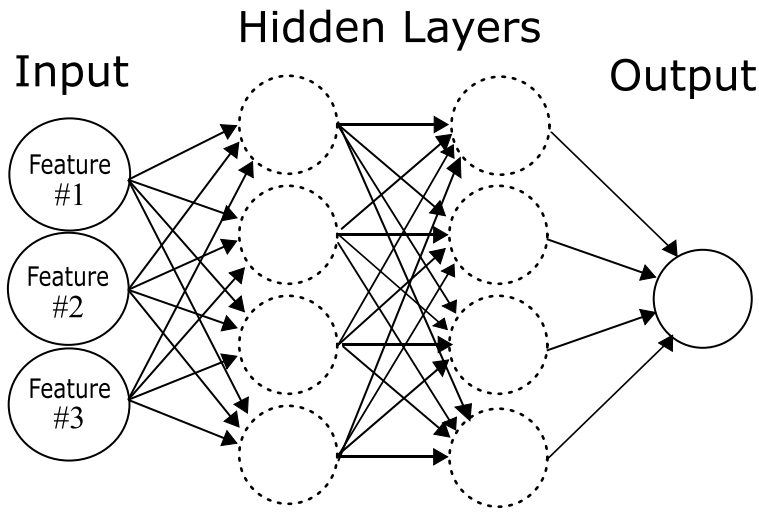


Figure 2.7: Simplified illustration of a feedforward neural network setup

Each edge in the network have weights w_i, which are scalars which can be thought of as gain values. Additionally, the neurons may have an individual bias b_i which is added in the summation of the weighted inputs. As a single neuron may have multiple inputs x_i, the calculation of the weighted sum y is best represented by vectors, as seen in Equation 2.17.

$$y = \vec{w} \cdot \vec{x} + b \tag{2.17}$$

Going back to the biological analogy, the weights are the strength of the neural connections, while the bias impacts the threshold that must be met or exceeded

for the neuron to fire. Whether the weighted sum will actually be transmitted from a neuron to the next layer depends on the activation function. The function decides whether the y value is propagated further and if so, to what degree – as the final output may be lower, equal or higher than the input depending on the chosen function. The mathematical model of one single neuron is illustrated in Figure 2.8.

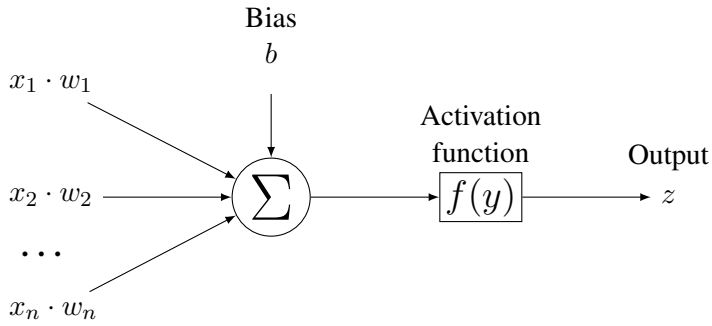


Figure 2.8: Illustration a single neuron, with weighted inputs from a previous layer on the left. The final output z is dependant on the output of the activation function $f(y)$.

.

One of the desired features of the activation function is that it should be continuous, so that gradient-based methods can be used in the optimization of the cost function. It should also be nonlinear, as given a sufficient amount of neurons in the hidden layer, a two-layer ANN should in theory be able to approximate any function (Ketkar, 2017). A popular choice is the sigmoid function, as shown in Equation 2.18, which fulfills both the criteria of non-linearity and continuity as seen by its derivative.

$$f(y) = \frac{e^y}{e^y + 1} \quad \frac{df(y)}{dy} = \frac{e^y}{e^y + 1} \cdot \left(1 - \frac{e^y}{e^y + 1}\right) \tag{2.18}$$

The great diversity of activation functions, biases, ways of optimizing weights, and other parameters makes the family of neural networks variations large. The user is thus left to pick a good fit for each application on a case to case basis.

Long Short-Term Memory Neural Networks

The feedforward neural network described in the previous section considers only the spatial information of the datapoints, and ignores the potentially valuable temporal information contained in time series data. To exploit the information contained in

a ordered dataset with a temporal dimension, the subclass of Recurrent Neural Networks (RNN) may be utilized. A major difference between a RNN and a conventional feedforward neural network is the memory capacity of the former, as each neuron may have an internal state that is carried over from each iteration. A popular choice among RNN is the Long Short-Term Memory (LTSM) neural network, invented by Schmidhuber and Hochreiter (1997). The memory capability of the LTSM network stems from memory cells, as shown in Figure 2.9.

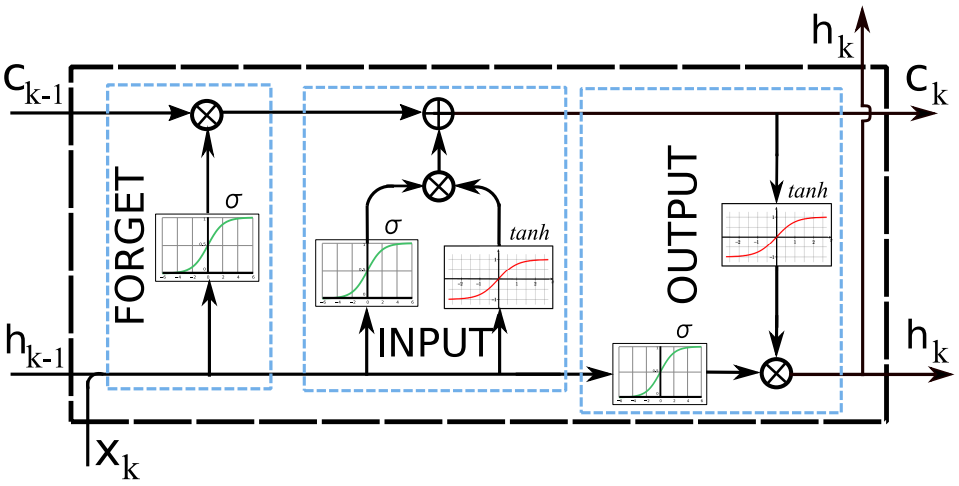


Figure 2.9: Individual cell of a Long Short-Term Memory layer.

Each memory cell contains three input vectors, all which have been altered by weights $\mathbf{W_{gate}}$ as in a conventional ANN. The previous memory cell values are denoted by the cell state vector $\mathbf{c}_{k-1}$, while $\mathbf{x}_k$ are the input values from the previous layer at the current time step k. Finally, there is the output value vector $\mathbf{h}_{k-1}$ also originating from the previous memory cell. The $\oplus$ symbol is a element-wise addition operation, and $\otimes$ represents element-wise multiplication.

In the original cell architecture, which is still widely used, each cell contains three gates with individual functions to regulate the information flow through the cell. To achieve this the cell uses two mathematical functions, the sigmoid function (σ) as described by Equation 2.18 and the hyperbolic tangent function ($tanh$).
The forget gate $\mathbf{f}$ decides to what extent the previous cell states should be carried over to the next cell, and may completely wipe the memory clean if deemed necessary. The input gate $\mathbf{i}$ serves as a gatekeeper and controls to what extent the input values $\mathbf{x}_k$ should be included in the cell state vector $\mathbf{c}_k$. The output gate $\mathbf{o}$ is the last step of the process, and decides the values of the output vector $\mathbf{h}_k$

which will be passed on both to the next cell and the next layer. The equations for each gate and outputs are summed up in Equations 2.19-2.23 (Schmidhuber and Hochreiter, 1997).

$$\mathbf{f_k} = \sigma\left(\mathbf{W_f}\mathbf{x}_k + \mathbf{W_f}\mathbf{h}_{k-1}\right) \tag{2.19}$$

$$\mathbf{i_k} = \sigma\left(\mathbf{W_i}\mathbf{x}_k + \mathbf{W_i}\mathbf{h}_{k-1}\right) \tag{2.20}$$

$$\mathbf{o_k} = \sigma\left(\mathbf{W_o}\mathbf{x}_k + \mathbf{W_o}\mathbf{h}_{k-1}\right) \tag{2.21}$$

$$\mathbf{c_k} = \mathbf{f_k} \circ \mathbf{c_{k-1}} + \mathbf{i_k} \circ \sigma\left(\mathbf{W_i}\mathbf{x}_k + \mathbf{W_i}\mathbf{h}_{k-1}\right) \tag{2.22}$$

$$\mathbf{h_k} = \mathbf{o_k} \circ \sigma(\mathbf{c_k}) \tag{2.23}$$

Where $\circ$ is the element-wise product. Note that LSTM cells may also contain bias values, like in ordinary ANN neurons.

Some drawbacks of using LSTM versus a conventional feedforward ANN are the increased risks of overfitting due to the extra parameters involved, and the increased computational and memory overhead attributed with the architecture.

2.6 Extended Kalman Filter

In contrast with an ordinary Kalman Filter, which is linear, the Extended Kalman Filter is made to handle nonlinearities. This is important because nonlinearities will unvariably emerge in both seakeeping and maneuvering applications for vessels, most importantly due to rotation matrices and damping terms. The filter handles this by linearizing about the current mean and covariance, as shown in Table 2.3 (Fossen, 2011):

Design matrices	$\mathbf{Q}_{kalman} = \mathbf{Q}_{kalman}^T > 0$ $\mathbf{R}_{kalman} = \mathbf{R}_{kalman}^T > 0$
Initial conditions	$\bar{\mathbf{x}}_{k=0} = \mathbf{x}_0$ $\bar{\mathbf{P}}_{k=0} = E\big[(\mathbf{x}(0) - \hat{\mathbf{x}}(0))(\mathbf{x}(0) - \hat{\mathbf{x}}^T)\big] = \mathbf{P}_0$
Correction	$\mathbf{K}_k = \bar{\mathbf{P}}_k \mathbf{H}^T \big[\mathbf{H}\bar{\mathbf{P}}_k\mathbf{H}^T + \mathbf{R}\big]^{-1}$ $\hat{\mathbf{P}}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\bar{\mathbf{P}}_k(\mathbf{I} - \mathbf{K}_k\mathbf{H})^T + \mathbf{K}_k\mathbf{R}\mathbf{K}_k^T$ $\hat{\mathbf{x}} = \bar{\mathbf{x}}_k + \mathbf{K}_k(\mathbf{y}_k - \mathbf{H}\bar{\mathbf{x}}_k)$
Prediction	$\bar{\mathbf{x}}_{k+1} = \mathcal{F}[k, \hat{\mathbf{x}}(k), u(k)]$ $\bar{\mathbf{P}}_{k+1} = \mathbf{F}(k)\hat{\mathbf{P}}_k\mathbf{F}(k)^T + \mathbf{Q}^T$

Table 2.3: Discrete-time Kalman filter

The matrix $\mathbf{F}_k$ is defined as the state transition matrix, which is a Jacobian found by partial differentiation of the original equations – shown in Equation 2.24. The observation matrix is denoted as $\mathbf{H}_k$ and is found in a similar fashion, by partial differentiation of the measurement prediction – shown in Equation 2.25.

$$\mathbf{F}_k = \frac{\delta \mathbf{f}(k, \mathbf{x}, \mathbf{u})}{\delta \mathbf{x}}\Big|_{\substack{\mathbf{x}=\hat{\mathbf{x}}(k+1) \\ \mathbf{u}=\mathbf{u}(k+1)}} \quad (2.24) \qquad \mathbf{H}_{k+1} = \frac{\delta \mathbf{h}(k+1, \mathbf{x})}{\delta \mathbf{x}}\Big|_{\mathbf{x}=\hat{\mathbf{x}}(k+1)} \quad (2.25)$$

The residuals are generated by the innovation step of the EKF algorithm, as shown in Equation 2.26. These may be used for fault detection, as non-vanishing residuals with considerable magnitudes can indicate a fault (Blanke et al., 2006).

$$\mathbf{r} = (\mathbf{y}_k - \mathbf{H}\bar{\mathbf{x}}_k) \quad (2.26)$$

The purpose of the filter examined in this thesis is to provide estimation of position, velocity and attitude of the a vessel. The attitude may be measured in quaternions

instead of Euler angles, which complicates the algorithm. Among the benefits of the attitude measurement being in quaternions is the absence of gimbal lock and singularities, and increased computational efficiency (Kraft, 2003).

2.6.1 Quaternion Representation in EKF

The quaternion product between two quaternions are expressed with the $\otimes$ symbol, and shown in Equation 2.27:

$$q_1 \otimes q_2 = \begin{bmatrix} \eta_1 \\ \boldsymbol{\epsilon}_1 \end{bmatrix} \otimes \begin{bmatrix} \eta_2 \\ \boldsymbol{\epsilon}_2 \end{bmatrix} = \begin{bmatrix} \eta_1\eta_2 - \epsilon_1^{\mathrm{T}}\epsilon_2 \\ \eta_1\epsilon_2 + \eta_2\epsilon_1 + S(\epsilon_1)\epsilon_2 \end{bmatrix} \tag{2.27}$$

where $S(\epsilon_1)$ is the skew-symmetric matrix, defined as:

$$S(\boldsymbol{\epsilon}) = \begin{bmatrix} 0 & -\epsilon_3 & \epsilon_2 \\ \epsilon_3 & 0 & -\epsilon_1 \\ -\epsilon_2 & \epsilon_1 & 0 \end{bmatrix}$$

While quaternion representation has clear benefits over Euler angles, including quaternions in the state vector of the Extended Kalman Filter is problematic. The primary reason for this is the tendency of the unit quaternion constraint to be violated, due to numerical inaccuracies in the filter propagation and state estimation steps. Quaternions that do not fulfill the unit criterion can not be used in vector rotation operations (Marins et al., 2002).
To circumvent this issue, a special form of state estimation update for the quaternions was implemented (Psiaki et al., 1990). This operation is shown in Equation 2.28 and ensures that the updated quaternion estimate is always normalized.

$$\hat{\boldsymbol{q}}_k = \bar{\boldsymbol{q}}_k \otimes \begin{bmatrix} \sqrt{1 - ||\mathbf{K}_q \boldsymbol{r_q}||^2} \\ \boldsymbol{r_q} \end{bmatrix} \tag{2.28}$$

Where $\bar{\boldsymbol{q}}_k$ is the quaternion predicted by the filter. The $\mathbf{K}_q \boldsymbol{r_q}$ term is the product of the Kalman gain and residual term of the quaternion components. The row corresponding to the real part of the quaternion (η) has been removed from the Kalman gain and residual vector, so $\boldsymbol{K}_q, \boldsymbol{r_q} \in \mathbb{R}^{3x3}$.
To avoid numerical inaccuracies, the residual calculation step for the quaternions is also different from Equation 2.26 and may instead calculated as in Equation 2.29:

$$\Delta \boldsymbol{q}_k = \boldsymbol{q}_{m,k} \otimes \bar{\boldsymbol{q}}_k^* = \boldsymbol{q}_{m,k} \otimes \bar{\boldsymbol{q}}_k^{-1} = [\eta_k \;\; \boldsymbol{r}_k]^T \tag{2.29}$$

where $\boldsymbol{q}_{m,k}$ is the measured quaternion at the time of the update. While q_k^* is the conjugate of the predicted quaternion, which is the same as the inverse $\bar{\boldsymbol{q}}_k^{-1}$ if the predicted quaternion fulfills the unit property, as it should.

The end result of the multiplicative operations proposed by Psiaki et al. (1990) is implicit quaternion normalization throughout the filter, despite the pitfalls of a quaternion implementation in an ordinary Extended Kalman Filter.

Chapter 3

The ReVolt Demonstration Platform

This chapter will review both the physical and digital framework that the ReVolt demonstration platform is composed of, along with a review of simulation engine used in the evaluation of the system.

3.1 The Physical ReVolt

This section will review the physical ReVolt platform, with all the relevant subsystems and its control system architecture. The idea of ReVolt is split up in a physical and a digital part, where the latter is used for simulation and digital twin experimentation and should have very few discrepancies compared to its physical counterpart.

The design can be traced back to a DNV GL conceptual study in 2014, which proposed a ship class for environmentally friendly short-sea shipping (DNV GL, 2015). ReVolt features several unorthodox design features, such as completely electric propulsion and absence of a superstructure due to its planned autonomous operation. The physical model used in this thesis is a scale model vessel of the conceptual ship design produced by Stadt Towing Tank. Subsequent references to ReVolt will refer to this physical model ship, shown in Figure 3.1, and its digital counterpart, instead of the conceptual full-scale design. The technical details of the model can be found in Table 3.1.

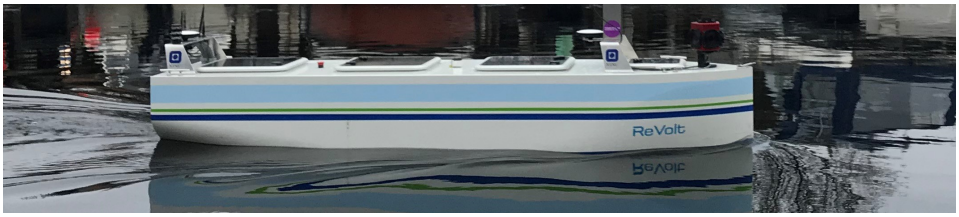


Figure 3.1: ReVolt underway in the Trondheimsfjord. Courtesy of Tom Arne Pedersen.

Length (LOA)	Beam	Draft	Depth Moulded	Light Ship Weight
3.02 m	0.72 m	0.23m	0.58 m	257 kg
Battery Voltage	**Top Speed**	**Scale**	**Battery Capacity**	**Total Engine Power**
12V	2 kts	1:20	1.8 kWh	360W

Table 3.1: Technical Specifications of ReVolt

3.1.1 System Topology

An overview of some of the components of ReVolt is given in Figure 3.2. The vessel has three actuators in total, consisting of two thrusters in the aft part and one retractable bow thruster. The latter is normally retracted, except for operations requiring high maneuverability, such as dynamic positioning and docking. The thrusters of the vessel will be a major focus of the thesis during the topic of actuator fault tolerance.

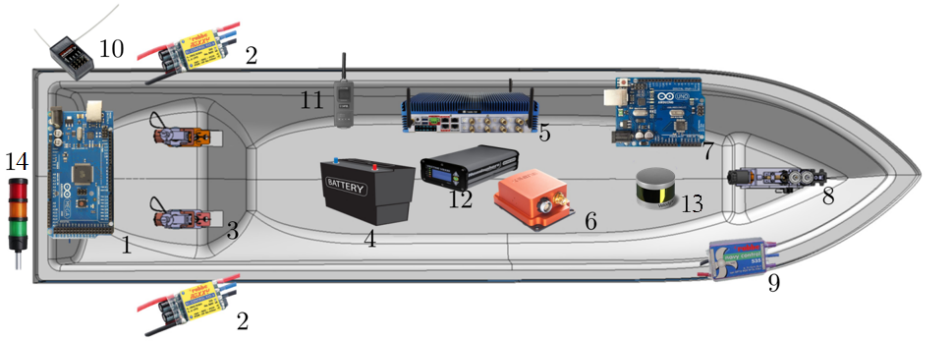


Figure 3.2: Illustration of vital components aboard ReVolt. Adapted from Stadt Towing Tank documentation.

No.	Component	No.	Component
1	Arduino Mega	**8**	Retractable Bow Thruster
2	Electronic Speed Controllers (Stern)	**9**	Electronic Speed Controller (Bow)
3	Main Propulsion Thrusters	**10**	Remote Control Receiver
4	Battery Bank	**11**	Satel Radio Modem
5	Onboard Computer	**12**	VS330 GNSS Receiver
6	Xsens IMU	**13**	VeloDyne Lidar
7	Arduino Uno	**14**	Light Beacon

Table 3.2: List of vital components, as numbered in Figure 3.2.

3.2 Computer Control Framework

To implement the desired control systems and enable efficient communication between individual components onboard, a solid hardware and software framework is needed. The hardware framework that allows this consists of an industrial embedded computer (*Tank 720*) and two Arduino micro-controllers, where the latter are connected to the computer via Universal Serial Bus (USB). The onboard computer serves as the main node in the control topology of the vessel, and handles the communication with higher level components such as the navigation system. The microcontroller consists of an *Arduino Uno R3* in the bow and *Arduino Mega 2560* in the aft compartment, which serves as I/O hubs for sensors and low-level control of pulse width modulation signals for motors.

3.2.1 Software Framework

The sum of all parts of ReVolt is a surprisingly complex system, which requires strict communication protocols to function properly. To connect the different parts and subsystems together, the robotics middleware Robot Operation System (ROS) has been used. The operating system allows abstraction from the otherwise low-level programming required for individual devices, efficient messaging protocols between control systems and usage of multiple computers/microcontrollers.

ROS implementations consists of *nodes*, which are processes that performs tasks at a specified rate or asynchronously. Nodes may subscribe to *topics*, which are the communication buses used for communication between modules. If a node is subscribed to a topic, it will receive all *messages* published to the topic by other

nodes. An example would be an observer node subscribing to a topic for sensor measurements, while subsequently publishing to a topic for state estimates.

Using ROS as middleware also allows for wrapping of different programming languages between the nodes, which is convenient. Although the source code of ReVolt is mainly written in Python, there are also C++ (Arduino) and Java (thrust allocation) components. The included datalogging feature (ROSBAG) allows for automatic time-stamping of measurements and will also be used extensively during the results.

3.3 Simulator

CyberSea is an inhouse simulator developed by DNV GL, shown in the Figure 3.3. The software have been used extensively in hardware-in-the-loop tests of both ships and rigs, and may recreate multitudes of different scenarios, related to environmental disturbances, equipment failures and marine traffic.

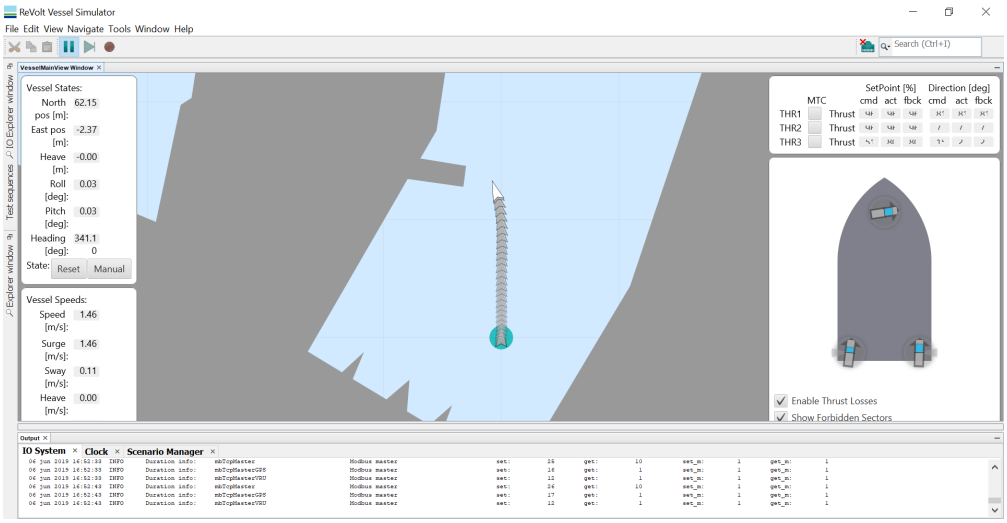


Figure 3.3: Screenshot of the simulator user interface.

The simulator can be connected to digital control systems by utilizing the Modbus protocol (Modbus Organization, 2012). By running the shipborne control system on an emulated Linux machine with a dedicated IP address, the Modbus module of CyberSea is able to establish connection between the two components. This is a form of two-way communication, where the control system acts upon information from the simulator, such as GNSS and IMU readings, and outputs commands

for the actuators. CyberSea amongst other things simulates navigational sensors, thruster dynamics and all hydrodynamical calculations – based on the control input from the control system. The topology of the system is illustrated in Figure 3.4.

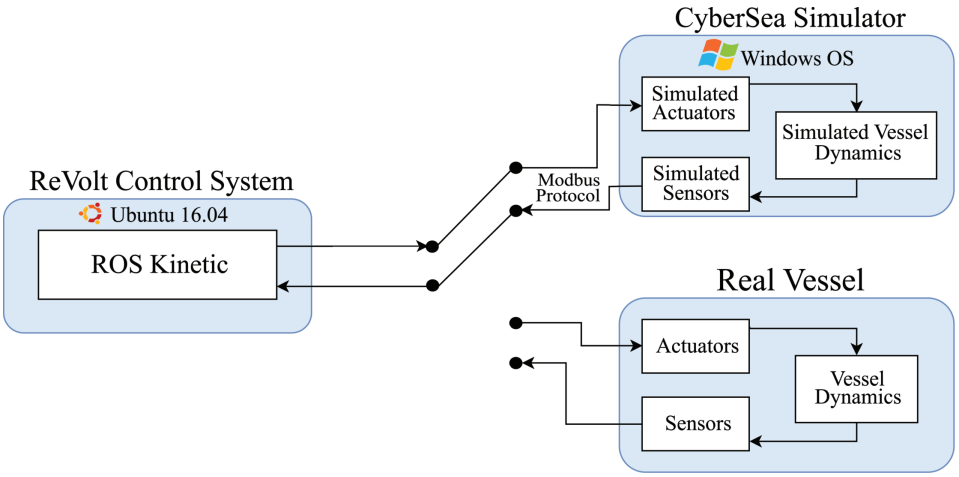


Figure 3.4: Communication flow during simulations versus real life tests.

Scenarios involving actuator and sensor faults can be simulated by a scripted language that is read by the simulator in realtime, allowing for automated testing of vessels. By controlling the environmental disturbances, the user can simulate a whole array of different sea state scenarios and log the vessel responses. There are ten different pre-defined sea state scenarios based on the Douglas sea state scale (Augustyn, 2019), as seen in Table 3.3.

Sea State Codes	Significant Wave Height [m]	Mean Wind Speed $\frac{m}{s}$	Description
0	0	0	Glassy
1	0.1	2	Rippled
2	0.4	3	Smooth
3	1	5	Slight
4	2	8	Moderate
5	3.5	11	Rough
6	5	15	Very Rough
7	7.5	18	High
8	12	25	Very High
9	17	35	Phenomenal

Table 3.3: Environmental scenarios in CyberSea, based on the Douglas scale of sea states

The wind disturbance is composed of a wind component with static direction and magnitude and a smaller gust component with these parameters randomized. Note that the water current velocity remains unchanged at 0 $\frac{m}{s}$ in all the predefined Douglas sea states, and has to be manually activated if required.

3.3.1 ReVolt as a Digital Twin

An accurate digital model of ReVolt has been made to enable rapid development of control systems, and to allow testing scenarios that would be difficult to recreate in real life. The hydrodynamic model of the twin is based on real derived from towing tank tests.

The control system itself is the same as the one running on the physical vessel, save for code for low-level control of motors and auxiliary sensors. For the perspective of the virtual Linux machine running the onboard code, there is no difference between simulation and reality. This is a strength, as there is no need to create an ad hoc control system specific for simulation, and enables real-life testing of software developed in simulator with only minor code modifications.

3.4 Past Contributions to ReVolt

Multiple project and master theses using ReVolt as a demonstration platform have already been delivered. Havnegjerde (2018) developed a path following algorithm for the model ship, along with a user interface for remote operation. Danielsen-Haces (2018) looked into creating a graphical visualization of ReVolt and providing condition monitoring for onshore operators. Kamsvåg (2018) developed a sensor fusion scheme by combining the LIDAR and cameras onboard ReVolt, to enhance the sensory capabilities of the platform.

Alfheim and Muggerud (2017) made numerous sensor suite improvements and implemented a dynamic positioning system for the physical vessel. However, the DP system developed would not function in CyberSea and had to be replaced for this thesis. An exception is the reference model that the authors developed, which has been reused in the new dynamic positioning system of this thesis.

Chapter 4

Implementation

The most vital components of the control system used for dynamic positioning is shown by block diagram in Figure 4.1.

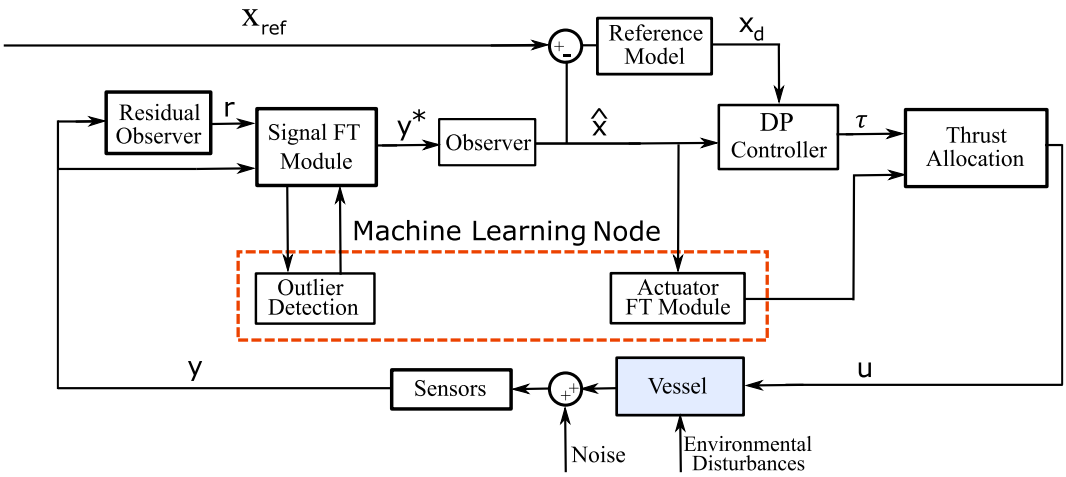


Figure 4.1: Block diagram of the implemented DP control system

The DP controller itself is relatively simple PID controller, acting on the desired states given by the reference model and the currently estimated states from the observer. It incorporates anti-windup for the integral term and output saturation, and the tuned controller parameters can be found in Appendix E. The controller outputs a 3 DOF vector, containing desired X and Y forces and the yaw moment N, for the thrust allocation block.

For practical reasons the state observer and residual observer have been divided into two. The reason for this is that the state observer has no capability to differentiate

between the sensor units, as it is fed aggregated sensor measurements y^* from the signal fault tolerant module. The residual observer on the other hand is fed raw sensor measurements y^{unit} that has not been treated or voted over by the signal fault tolerant module, and is also aware of what sensor unit each measurement stems from. This duality in observer design is consistent to the signal fault-tolerance architecture proposed in Mokleiv (2017).

The actuator fault tolerance system is composed of a recurrent neural network classifying actuator healthy based on filter estimates, and a switching mechanism to act on information produced by the network. Note that the thrust allocation block is composed of several submodules not shown in the block diagram. Details regarding the thrust allocation system and its interaction with the actuator fault tolerance module will be given later, in Section 4.3.2.

4.1 Extended Kalman Filter

The two observers used in the control system are both discrete Extended Kalman Filters, with different objectives. The aim of the residual observer is to act as a residual generator for the signal fault tolerance module, while the state observer provides estimates of the state for the DP controller.

The state vector of the filters is shown in Equation 4.1, and consists of 16 variables in total.

$$\mathbf{x} = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \\ \dot{\mathbf{p}} \\ \mathbf{b^a} \\ \mathbf{b^g} \end{bmatrix} \tag{4.1}$$

The dynamics of the state variables is shown in Equation 4.6.

$$\dot{\mathbf{p}} = \ddot{\mathbf{p}} \tag{4.2}$$

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q} \otimes (\boldsymbol{\omega_m} - \mathbf{b^g}) \tag{4.3}$$

$$\ddot{\mathbf{p}} = \mathbf{R}(\mathbf{q})(\mathbf{a}_m - \mathbf{b^a}) + \mathbf{g} \tag{4.4}$$

$$\dot{\mathbf{b}}^{\text{a}} = 0 \tag{4.5}$$

$$\dot{\mathbf{b}}^{\text{g}} = 0 \tag{4.6}$$

where $\boldsymbol{\omega}_m$ is the measured angular rate, $\mathbf{a}_m$ the measured acceleration, and $\mathbf{R}$ is the rotation matrix. Estimates of position and velocity in the NED frame, attitude

in quaternions, and biases for both accelerometer and rate gyros are output by the filters. The positional state estimates are used in control loop of the Dynamic Positioning system, to provide an error based on the currently desired position. Before being fed to the same DP control loop, the velocity estimates are converted to the BODY frame and the attitude to Euler angles.

The $\mathbf{g}$ vector is a static gravity vector, $g = [0, 0, 9.81]^T$. Ideally the gravity vector should be a function of the longitude and latitude, and calculated through methods as in Hsu (1996). Since the DP operation is assumed to be contained in a relatively small geographical area however, the gravity has been assumed constant.

The quaternion dynamics in Equation 4.3 can be rewritten in iterative discrete form as in Equation 4.7:

$$\mathbf{q_{k+1}} = \mathbf{q_k} - \frac{1}{2}\Delta T \cdot \boldsymbol{\Omega}(\boldsymbol{\omega_m} - \mathbf{b^g}) \cdot \mathbf{q}_k \tag{4.7}$$

where $\boldsymbol{\Omega}$ is the 4x4 skew-symmetric matrix shown in Equation 4.8.

$$\boldsymbol{\Omega}(\boldsymbol{\omega_m} - \mathbf{b^g}) = \boldsymbol{\Omega}(\boldsymbol{\omega_t}) = \begin{bmatrix} 0 & \boldsymbol{\omega_t}^{\mathrm{T}} \\ -\boldsymbol{\omega_t} & \mathbf{S}\left(\boldsymbol{\omega_t}\right) \end{bmatrix} = \begin{bmatrix} 0 & \omega_{t,x} & \omega_{t,y} & \omega_{t,z} \\ -\omega_{t,x} & 0 & \omega_{t,z} & \omega_{t,y} \\ -\omega_{t,y} & \omega_{t,z} & 0 & -\omega_{t,x} \\ -\omega_{t,z} & -\omega_{t,y} & \omega_{t,x} & 0 \end{bmatrix} \tag{4.8}$$

The state propagation Jacobian is given by Equation 4.9:

$$\mathbf{F}_k = \begin{bmatrix} \mathbf{I}_{3\mathrm{x}3} & \mathbf{0}_{3\mathrm{x}4} & \Delta T \cdot \mathbf{I}_{3\mathrm{x}3} & \mathbf{0}_{3\mathrm{x}3} & \mathbf{0}_{3\mathrm{x}3} \\ \mathbf{0}_{4\mathrm{x}3} & \mathbf{I}_{4\mathrm{x}4} - \frac{\Delta T}{2}\boldsymbol{\Omega}(\boldsymbol{\omega}_m - \mathbf{b}^{\mathrm{g}}) & \mathbf{0}_{4\mathrm{x}3} & \mathbf{0}_{4\mathrm{x}3} & \frac{\partial}{\partial \mathbf{b}^{\mathrm{g}}}\left(\frac{\Delta T}{2}\boldsymbol{\Omega}(\mathbf{b}^{\mathrm{g}})\mathbf{q}\right) \\ \mathbf{0}_{3\mathrm{x}3} & \frac{\partial}{\partial \mathbf{q}}\left(\Delta T(\mathbf{a}_{\boldsymbol{m}} - \mathbf{b}^{\mathrm{a}})\right) & \mathbf{I}_{3\mathrm{x}3} & \frac{\partial}{\partial \mathbf{b}^{\mathrm{a}}}\left(\Delta T(\mathbf{a}_{\boldsymbol{m}} - \mathbf{b}^{\mathrm{a}})\right) & \mathbf{0}_{3\mathrm{x}3} \\ \mathbf{0}_{3\mathrm{x}3} & \mathbf{0}_{3\mathrm{x}4} & \mathbf{0}_{3\mathrm{x}3} & \mathbf{0}_{3\mathrm{x}3} & \mathbf{0}_{3\mathrm{x}3} \\ \mathbf{0}_{3\mathrm{x}3} & \mathbf{0}_{3\mathrm{x}4} & \mathbf{0}_{3\mathrm{x}3} & \mathbf{0}_{3\mathrm{x}3} & \mathbf{0}_{3\mathrm{x}3} \end{bmatrix} \tag{4.9}$$

where ΔT is time elapsed since last filter iteration, and $\mathbf{I}$ is the identity matrix. The state propagation Jacobian in its entirety can be studied in the code implementation of the Extended Kalman Filter, attached with the thesis.

An initial measurement noise covariance matrix was chosen based on technical specifications from datasheets given by the sensor manufacturers. However, the measurement noise covariance matrix $\mathbf{R}_{\text{kalman}}$ required additional tuning due to the transformations applied to parts of the raw sensor readings, such as in the transformation of GNSS data from LLA to the NED frame. The tuning of the filters is shown in Appendix F.

4.2 Signal Fault Tolerance

The purpose of the module is to evaluate the integrity of sensor readings, ensure that spurious signals are not propagated throughout the control system, and finally identify and accommodate signal failure modes that may arise. The conceptual design of the signal fault tolerance module, and all subcomponents, is shown in Figure 4.2.

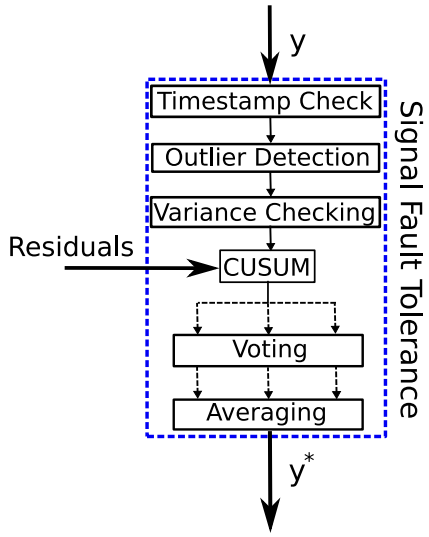


Figure 4.2: Block Diagram showing the components of the signal fault tolerance module

The dashed lines indicate individual sensor measurements, while the solid lines represents vectors containing measurements from all sensors relevant to the signal type. Finally, the processed output of the block is a single value, which will be averaged should the physical redundancies be equal to or exceed two units.

4.2.1 Timestamp Check

Timestamping aims to detect signal drop outs, and does so by evaluating discrepancies between the UNIX timestamp provided by ROS for each measurement, and the current UNIX time also reported by ROS. Should the time difference exceed 4 seconds, the unit in question will be flagged for signal drop out.

4.2.2 Outlier Detection

The signal outlier detection relies on a model using One-Class Support Vector Machine, utilziing the Scikit-Learn library for Python as backend for the algorithm (Cournapeau, 2013). A measurement will be discarded and not propagated through the rest of the control system, should a sample be classified as an outlier by the system. The datasets from which the models have been trained on have been manually reviewed and inspected for outliers, in order to ensure adequate data quality.

The model is continually learning and improving the accuracy of the region of acceptance for measurements. Measurements deemed as inliers are collected in a pool and saved, until the pool contains a total of 400 datapoints classified as inliers. When this threshold is reached, the model is sequentially retrained with the new set of inliers included. The reason why only inliers are allowed to be used as new training data is that outliers would skew the region of acceptance of the model, which would ultimately harm the accuracy by obfuscate the boundaries between inliers and outliers.
While automatic and unsupervised retraining has obvious benefits in form of the manhours spared collecting, cleaning and reviewing data, it also opens up the possibility of false negatives affecting the model. While this has not been observed during the thesis, it is certainly a possibility and new models made from unsupervised training should be reviewed carefully and verified.

The kernel used in the implementation is radial basis function (RBF). This is one of the kernels which enables linear separation by transformation of the feature space to a higher dimensional space, as described in Section 2.5.1.

4.2.3 Floating Point Arithmetic

All the signal processing functions deal with measurements expressed in double precision, as per the 64 bit IEEE 754 standard (IEEE, 2008). While this has obvious benefits in terms of accuracy, it may introduce problems due to the numerical inaccuracies stemming from errors in rounding and discretization of continuous variables. Specifically, to test if two values are equal is not straightforward and requires knowledge about the data type precision. Equal checking operations are needed for instance in the variance testing.
The procedure implemented for robust floating point comparison is shown in the pseudo-code for the DoubleEqual function in Algorithm 1.

Algorithm 1 DoubleEqual

```
1: Boolean doubleEqual(arg1, arg2, ϵ)
2: difference ← |(b-a)|
3: abs_a, abs_b ← |(a)|, |(b)|
4: DBL_MIN ← Smallest Value representable using Double
5: DBL_MAX ← Largest Value representable using Double
6:
7: if (a == b) then
8:     return True
9: else if (a == 0 or b == 0 or difference < DBL_MIN) then
10:    return (difference < ϵ · DBL_MIN)
11: else
12:    RelativeError = difference / min((abs_a+abs_b),DBL_MAX)
13:    return (RelativeError < ϵ)
```

Where ϵ is a tolerance value selected by the user. The first if condition aims to capture the best case scenarios where the values are exactly the same. However, the simplistic first condition will not catch comparisons where one argument is zero or when both are extremely close to each other – the second condition handles these cases. Lastly, cases not captured by the previous tests are handled using the relative error, which should be below the user defined tolerance value if the arguments are to be deemed equal.

4.2.4 Variance Testing

Evaluation of the variance of a signal can be an indicator for whether a signal has frozen or is subject to a high variance failure mode. The variance for a window of the 20 last measurements is calculated at each timestep. For the GNSS units that are assumed to operate at 10 Hz, this results in a 2 second delay from the onset of frozen signal to it being detected by the variance test. As the IMU units operate at a far higher frequency, close to 200 Hz, the delay is much smaller with an average around 0.1 seconds.

Differentiating between the high variance failure mode and legitimate high dynamics can be difficult. The latter is a result of periods of large changes in the underlying state, such as the velocity during rapid acceleration or roll angle when a large beam wave hits the vessel.

An additional measure was implemented to prevent false alarms following unfreezing

of a previously frozen signal. The high derivative effect is illustrated in Figure 4.3.

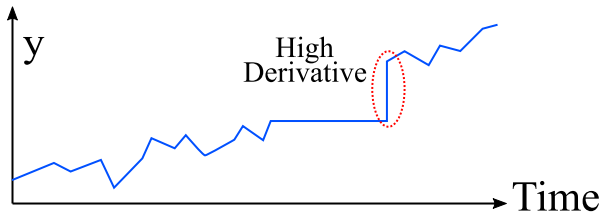


Figure 4.3: High Derivative phenomenon following signal freeze

The sudden unfreezing results in a nearly stepwise change in measurement as the underlying state of the process has developed while the sensor output was frozen. This in turn will result in a high variance in a sufficiently small variance window, triggering high variance or outlier alarms. This was circumvented by implementing a 2 seconds cool-down after unfreezing for the FDD of the specific signal.

4.2.5 CUSUM

The implementation of the cumulative sum (CUSUM) algorithm detects sudden biases and signal drift, as the mean of the underlying distribution is changed. The input to the CUSUM comes from a dedicated residual generator in form of an Extended Kalman Filter. At each iteration, the probabilities of the residual value belonging to three distinct Gaussian distributions are calculated. The null hypothesis assumes there is no underlying change of the statistical distribution of the residual, so the mean is still equal to zero. The two other distributions are situated at each flank of the curve of the null hypothesis distribution. These probabilities are denoted P_{H0} for the null hypothesis, while P_{H1} and P_{H2} are the probabilities of a residual belonging to respectively the distribution on the left and right flank. The cumulative sums are then calculated based on the previous CUSUM iterations, and the new probabilities, as shown in Equation 4.10-4.11:

$$S_{k+1}^{\text{high}} = max\left(0, \left(S_k^{\text{high}} + log_{10}\left(\frac{P_{H1}}{P_{H0}}\right)\right)\right) \tag{4.10}$$

$$S_{k+1}^{\text{low}} = min\left(0, -\left(S_k^{\text{low}} + log_{10}\left(\frac{P_{H2}}{P_{H0}}\right)\right)\right) \tag{4.11}$$

where the cumulative sums themselves are denoted as S^{high} and S^{low}, often called upper cumulative sum and lower cumulative sum. Both of the S values are initialized

as zero. The cumulative sums are monitored continuously, and should they cross their user-defined thresholds it is deemed the underlying properties of the statistical distribution has changed – indicative of a sudden bias or drift in the signal. Both the user-defined thresholds and the statistical properties of the alternative hypotheses have to be tuned by the user. Tuning of the latter can be challenging as it requires an educated guess about the statistical properties of the failure modes.

4.2.6 Voting and Averaging

If the physical redundancy of operational IMU or GNSS units are equal to two or higher, the measurements will be average before being published and fed to the control system. To be included in the averaging process, the unit must not have faults of any kind, as specified by the failure modes described earlier. Units that have been deemed unfit for averaging by a voting mechanism will not included either.

The voting mechanism requires a physical redundancy of at least three units, in order to isolate the deviating sensor in addition to just confirming a deviation the between sensor readings. The disabling of sensors due to voting can have unfortunate side effects in form of sudden step jumps in filtered value. To circumvent this phenomena, a lowpass filter has been implemented as based on recommendations from Sørensen (2013). Like all forms of filtering, this does introduce a phase shift in the output signal. Due to the tuning of the time constants of the lowpass filter, this phase shift should be small. Note that the filter output is only used during deactivation of sensors due to voting, and will not cause phase shift issues during normal operation.

4.3 Actuator Fault Tolerance

The module tasked with providing actuator fault tolerance is composed of two major parts, namely the recurrent neural network doing the online classification and the controller bank acting on the latest information from said network.

To limit the scope of the thesis, it is not possible to consider all the different failure modes of thruster system of a DP vessel. The focus will thus be on the situation of drive-off, as described in Hansen (2011) as one of two common failure modes of DP systems. Drive-off is often considered the most dangerous failure mode, as the faulty thruster(s) forces the vessel away from the desired position (Haugen and

Smogeli, 2017). The drive-off situation has modelled as a freeze in the actuator input u of the faulty thruster.

4.3.1 Recurrent Neural Network

There are a total of 35 input variables in the neural network, comprised of the categories as shown in Figure 4.4. Special precaution had to be made regarding what features to be used in the input layer, as there is no shortage of data collected aboard the vessel, but including the wrong ones could jeopardize the system. It was important not to select highly specific variables like the GNSS position, as this would most likely cause overfitting and be counterproductive, as the ability to diagnose a fault should never be a function of the vessel position. By using feature engineering, the more universal positional error was calculated and selected as input variable instead of the raw position.

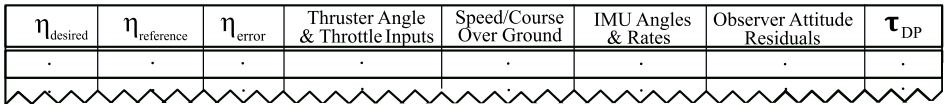


Figure 4.4: Dataset structure used for training the neural network.

All data that enters the model, both during training and operation, is subject to feature scaling. Data from each variable is transformed using the Min-Max scheme, where the sample with the lowest value is set equal to zero and the highest equal to one, and the rest in-between the extremes.

The model gives its verdict of the actuator health in form of 4 binary outputs. The columns consist of a healthy state indicating no faults, and one column for each distinct failure configuration – as shown in Figure 4.5. A ROS message containing the relevant failure mode is sent to the thrust allocation node in case the output of the model deviates from the healthy state.

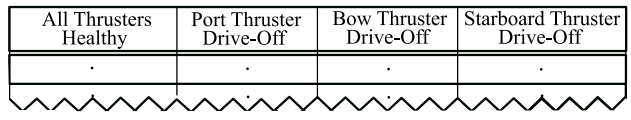


Figure 4.5: Output columns of the neural network.

As the current neural network architecture follows an one-hot encoding scheme,

only a single output column will be set equal to one at each time step. This form of classification is called a multi-class problem, where the classes are mutually exclusive. Note that it does not hinder classification of multiple faults, only multiple faults occurring in the exact same time step.

The RMSProp method is for the optimization during training, which uses a gradient descent approach to tune the weights of the neurons. A common problem during training with gradient-based learning methods such as RMSProp the issue of vanishing gradients. This happens during backpropagation, when low valued gradients may accumulate backwards in the network as they are multiplied numerous times because of the differentiation chain rule. The end result may be an extremely low gradient that makes changing the weight of front neurons nigh impossible, hindering the learning process. The activation functions in the hidden layers consist of ”leaky” rectified linear units (ReLU). The ordinary rectified linear unit function is especially prone to the vanishing gradient problem, in what is dubbed as the ”dying ReLU” phenomenon. The most dramatic effect of preventing learning is circumvented by introducing a small slope at negative inputs, as seen in Figure 4.6b.

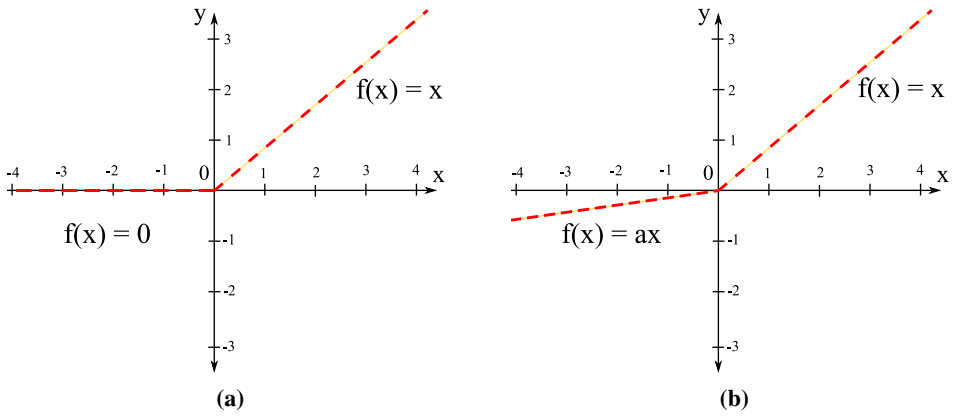


Figure 4.6: Standard rectified linear unit function **(a)** and leaky variant **(b)**, where a is some positive constant.

The Long Short-Term Memory ANN implementation is composed of five dense layers, where the first and the last are respectively the input and the output layer. A visualization of the complete architecture used in the implementation can be seen in Appendix D. While the amount of layers is quite high, only five of them are dense layers and two are LSTM layers – while the rest are activation functions and regularization layers.

Regularization

Regularization has been used in order to reduce the risk of overfitting during the model training process. Dropout functionality has been introduced at every hidden layer, which at each epoch randomly deactivates a user-defined fraction of neurons and their corresponding weights. This hinders the model from overly depending on specific neurons, which could potentially make the function memorize the training data rather than learning a generalizable classification function. After tuning, the dropout fraction was set equal to 0.4, meaning that 40% of the neurons of each hidden layer was deactivated during each epoch. This precaution, in addition to delegating 30% of the datasets to the test set, should help prevent overfitting.

Performance

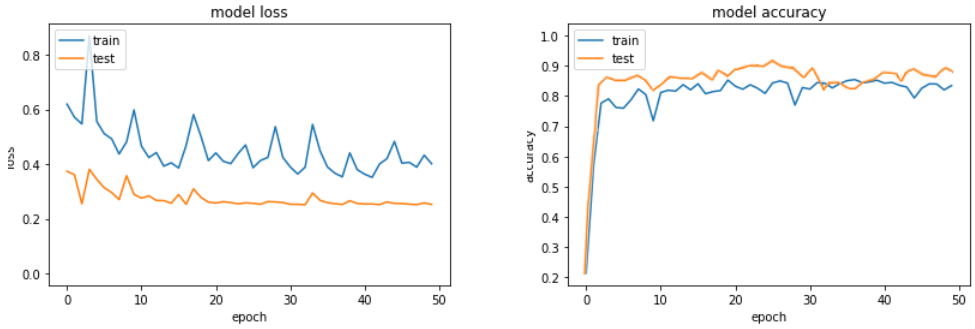
The performance of the final model was evaluated by several performance metrics commonly used in machine learning, namely accuracy, F_1-score, recall and precision. Note that the three latter metrics are based strictly on binary classification, where a fault is either present or absent. The F_1-score, recall and precision metrics do not tell whether the right diagnosis has been reached, and considers only a faulty/fault-free dichotomy. The accuracy metric is based on the concept of categorical accuracy, which does not follow the same dichotomy. The categorical accuracy is calculated by dividing the number of correctly classified predictions by the total number of datapoints, as the predicted class of each datapoint is compared to its true class (TensorFlow, 2019).
Table 4.1 shows the performance metrics of the model after training.

F_1-score	Precision	Recall	Accuracy
0.956	0.979	0.937	0.836

Table 4.1: Classification performance of the LSTM implementation during training.

The relatively high precision value is the result of a conscious tuning choice, as

false positives are punished severely. The recall is also relatively high, which in this context means that an overwhelming majority of the datapoints containing faults are correctly classified as faulty. The loss and accuracy evolution during training is shown in Figure 4.7, featuring both the training and test sets.



(a) **(b)**

Figure 4.7: Evolution of loss **(a)** and accuracy values **(b)** during training phase.

The loss function for the training set seems to stabilize itself inside a band of 0.45 ± 0.10 after 20 epochs, while the same value for test set stabilizes around 0.275. The accuracy function for both the training and test set has a sharp rise and diminishing improvements after the initial 10 epochs, and the training set accuracy reaches a final value of 0.836 and the test set stops at 0.879.

4.3.2 Thrust Allocation

In the case of ReVolt, there are three azimuth thrusters with fixed pitch propellers. Each azimuth thruster needs a throttle setpoint and an azimuth angle as input, and in the usual scenario with all thrusters operational this results in a control input $u \in \mathbb{R}^6$. By considering the usual horizontal plane model usually used in ships, the vessel can be assumed to have 3 degrees of freedom – surge, sway and yaw (Fossen, 2011). Thus according to the horizontal model $\tau \in \mathbb{R}^3$, and as τ has a lower dimension than u it can be classified as a overactuated system, meaning the allocation problem is an optimization problem due to the multiple solutions. Note that some thruster failure modes can result in reduction of the rank of u, potentially making the system only fully actuated or ultimately under-actuated.

The thrust allocation algorithm that has been utilized in the thesis has been developed by DNV GL, and is written in Java. It uses the Moore-Penrose pseudo-inverse

operation to obtain the control input vector u. While the pseudo-inverse does indeed find the optimal solution in the context of least-squares, it does not account for actuator saturation (Millan, 2008). Saturation constraints must thus be solved in an ad hoc manner, to ensure that the boundaries of each thruster are not overstepped.

To make the DP system fault-tolerant, it needs to account for different actuator failure modes and combinations that may occur during operation. Thus a multitude of different thruster allocation configurations has been implemented, and stored in an offline controller bank, as seen on the right in Figure 4.8. The decision logic block selects the optimal thruster allocation scheme based on the health of the actuator system. For instance, should one of the azimuth thrusters suffer a rotational and throttle lock, the switching mechanism will compensate by changing to a different thrust allocation that distributes the control effort only among the two healthy thrusters. The input to the decision logic comes from actuator health messages sent by the neural network.

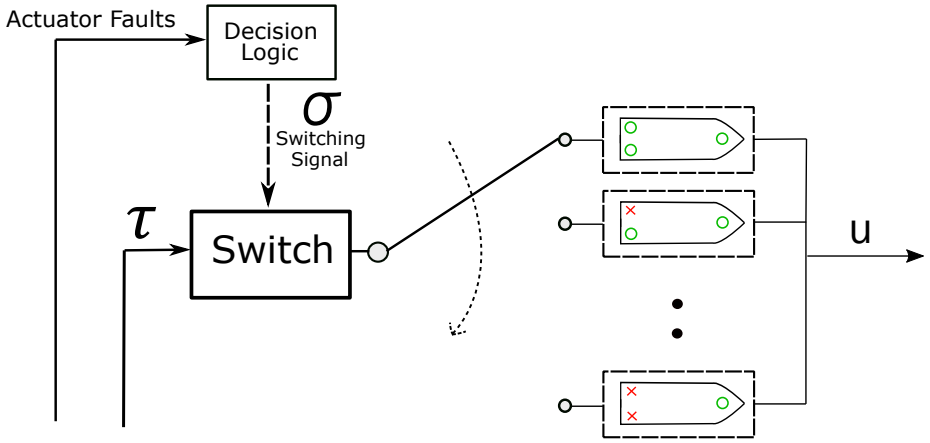


Figure 4.8: Illustration of the thrust allocation switching mechanism. Functioning thrusters symbolized by a green circle and deactivated ones marked with a red cross.

The switch itself is conducted almost instantly, so only the currently selected thrust allocation is ran at each time step, meaning there is no extra computational overhead attributed to the increased number of thrust allocation options. The fault handling philosophy falls under what is termed controller reconfiguration in Blanke et al. (2006).

Chapter 5

Results

To evaluate the performance of the final fault-tolerant control system, a set of scenarios has been devised which aims to exhibit the many features of the system. The cases are to be presented in realistic terms with environmental disturbances and measurement noise, as would be present in an real-life application.

- Case I – Baseline case without environmental disturbances and faults to assess performance of the DP system itself. The performance of the EKF is also reviewed.
- Case II – Baseline case without faults, but with environmental disturbances, to assess performance of the DP system in harsher weather.
- Case III – Case featuring multiple GNSS signal faults.
- Case IV – Case featuring multiple actuators faults.
- Case V – Combined case featuring both signal and actuator faults, with environmental disturbances.

Each case is conducted using the 4-corner test for dynamic positioning, as described in Vaerno et al. (2017) and shown in Figure 5.1. The maneuver serves as a benchmark for dynamic positioning performance, as it tests out a range of both pure and coupled motions.

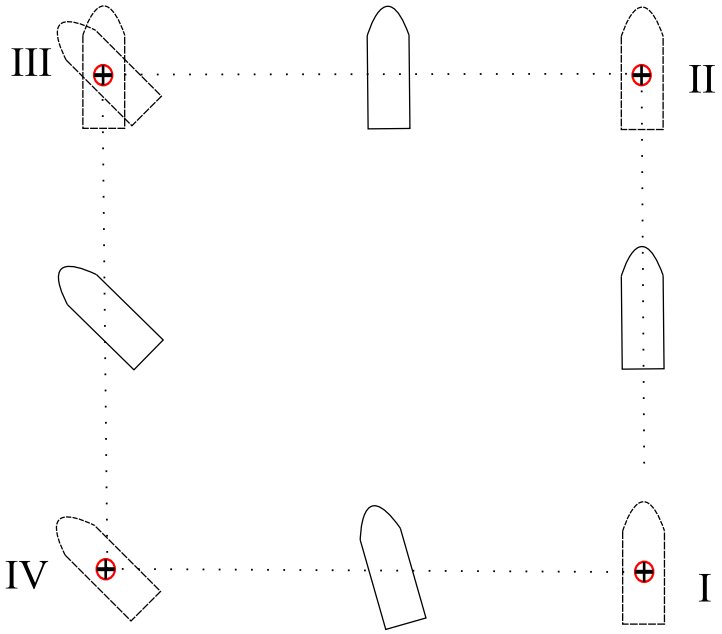


Figure 5.1: Illustration of the 4-corner test for measuring DP performance.

1. Surge motion from point **I** to **II**.
2. Sway motion from **II** to **III**, while heading is kept at $0°$. Upon arrival, heading is changed from $0°$ to $-45°$.
3. Surge and sway motion from **III** to **IV**. Heading is to be kept constant at $-45°$ during the transit.
4. Coupled motion as vessel moves from **IV** to the starting point **I**. The reference heading changes gradually from $-45°$ to $0°$.

The distance between the corners has been set equal to 30 m.

5.1 Measurement Noise

Every case includes measurement noise for onboard GNSS and IMU units, as characterised in Table 5.1.

Measurement Noise		
Unit	Output	Std. Dev. σ
GNSS	Latitude $[deg]$	$1{\cdot}10^{-6}$
	Longitude $[deg]$	$3{\cdot}10^{-6}$
	Altitude $[m]$	0.3
	Speed-Over-Ground $\left[\frac{m}{s}\right]$	0.025
	Course-Over-Ground $[rad]$	0.0025
IMU	Roll ϕ $[rad]$	0.415
	Pitch θ $[rad]$	0.415
	Yaw ψ $[rad]$	0.415
	Roll Rate $\dot{\phi}$ $\left[\frac{rad}{s}\right]$	0.01
	Pitch Rate $\dot{\theta}$ $\left[\frac{rad}{s}\right]$	0.01
	Yaw Rate $\dot{\psi}$ $\left[\frac{rad}{s}\right]$	0.01
	Acceleration$_X$ $\dot{u}$ $\left[\frac{m}{s^2}\right]$	0.0075
	Acceleration$_Y$ $\dot{v}$ $\left[\frac{m}{s^2}\right]$	0.0075
	Acceleration$_Z$ $\dot{w}$ $\left[\frac{m}{s^2}\right]$	0.0075

Table 5.1: Sensor outputs subject to white noise

The white noise parameters have been chosen based on specifications given in supplier datasheets, presented in Appendix B and C. By modelling the measurement noise based on the actual components of ReVolt, the simulator conditions should be close to that of a sea-trial.

The standard deviation of the noise affecting the latitude and longitude may seem small, but at the area where the simulations take place (The Dora-basin in Trondheim) horizontal accuracy corresponds to ± 0.2 m, which is conservatively chosen. In areas with Differential GPS coverage such as the Trondheimsfjord, the horizontal positional error is orders of magnitudes smaller – and with Real-Time Kinematic augmentation it may be accurate down to 0.01 m. The altitude measurements have a correspondingly higher noise component, aiming to capture the tendency of GNSS systems' Vertical Dilation Of Precision (VDOP) value to be higher than the horizontal counterpart.

5.2 Baseline Case I

The first case should serve as a baseline for comparison of subsequent runs, with no faults occurring and all environmental disturbances absent.

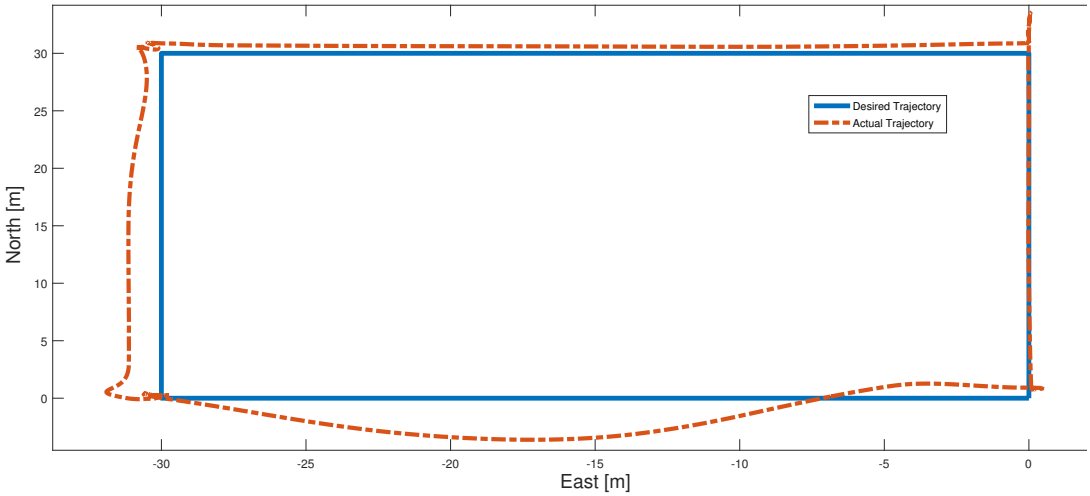


Figure 5.2: Reference trajectory and actual path followed by the vessel during Case I.

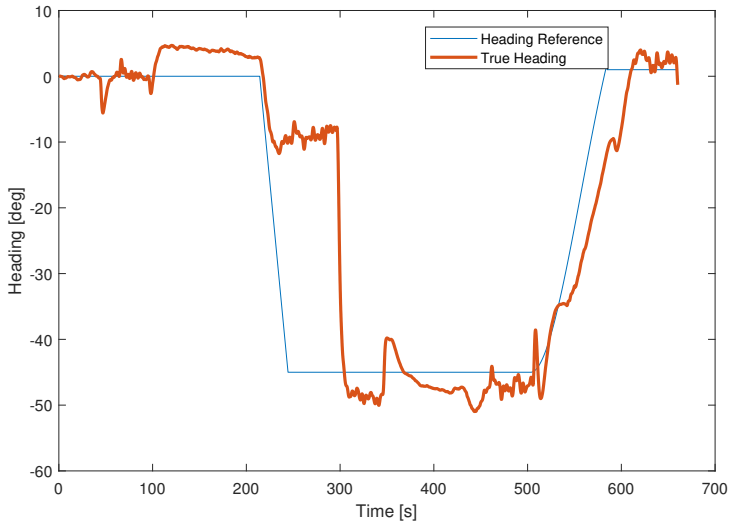


Figure 5.3: Heading setpoint and actual heading of the vessel during Case I.

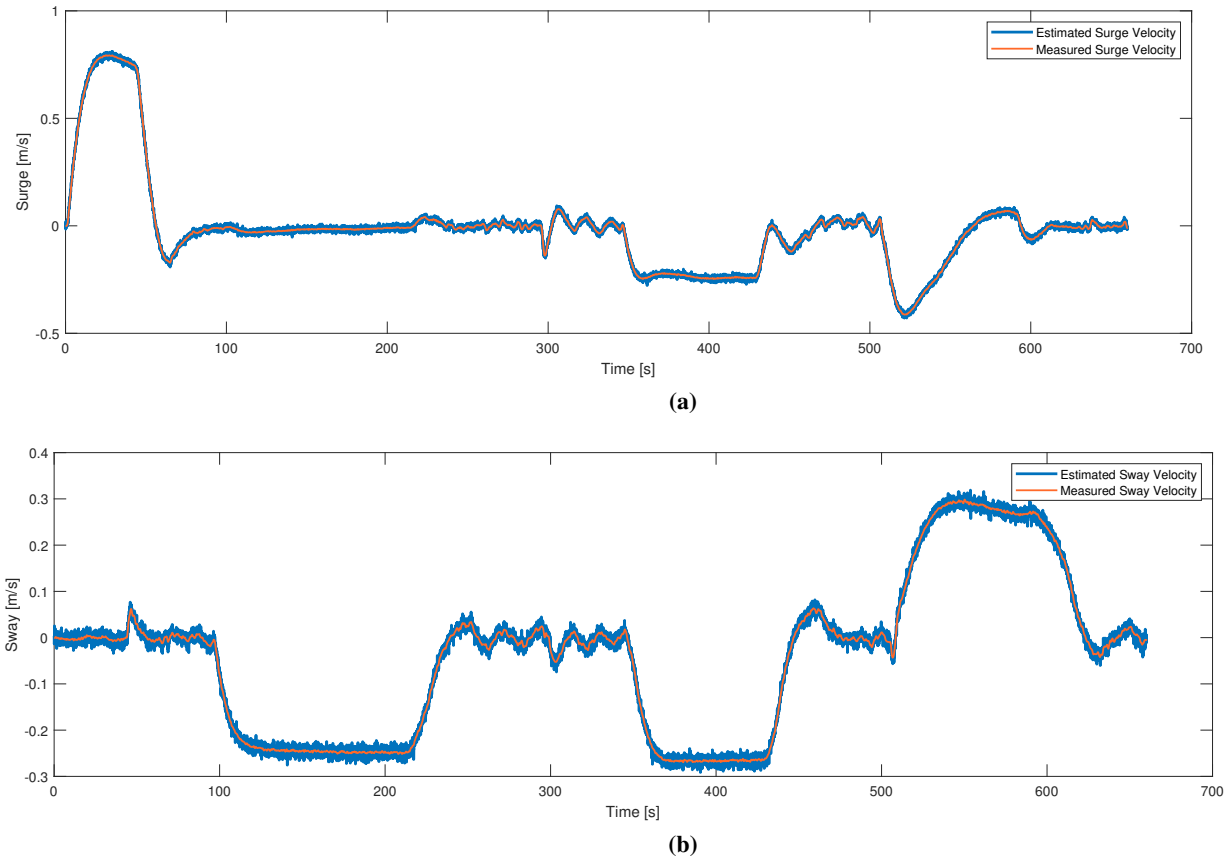


Figure 5.4: Estimated and and measured values of surge **(a)** and sway **(b)** velocity during Case I.

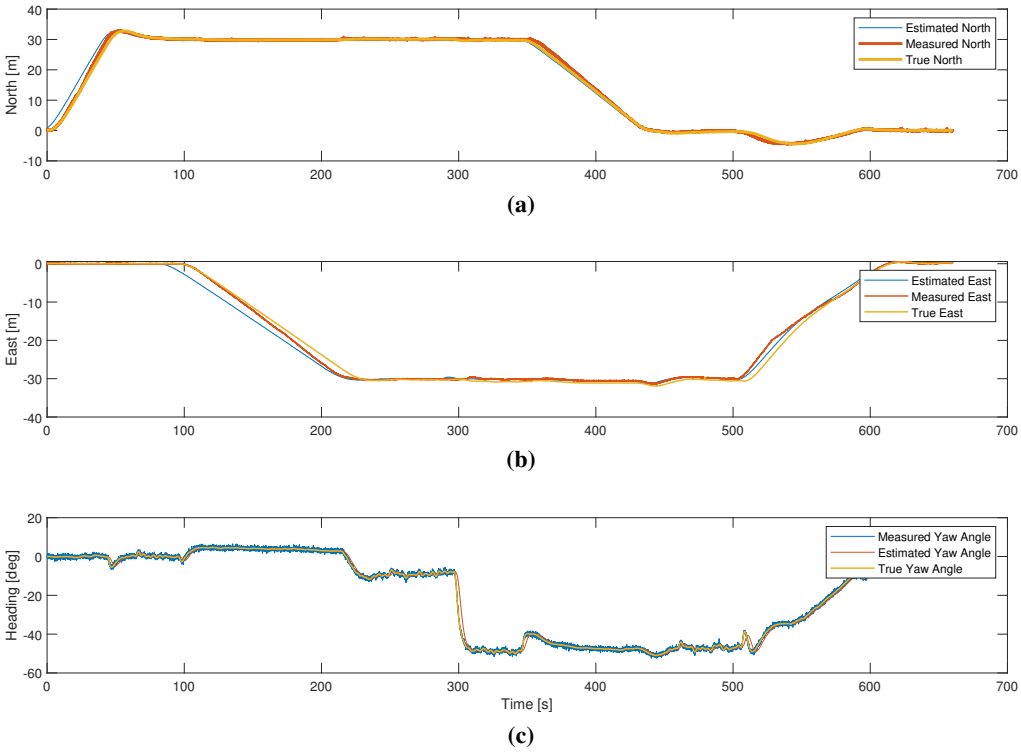


Figure 5.5: Estimated, actual and measured values of north position **(a)**, east position **(b)** and yaw angle **(c)** during Case I.

5.2.1 Discussion – Baseline Case I

The Extended Kalman Filter does an adequate job at estimating the true pose of the vessel despite the noisy sensor readings. Note that there is some lag between the true values and the filtered/unfiltered values, where the former are retrieved instantly from the simulator and the latter are published within the ROS control system with inherent computational delays.
While the filter does not directly estimate surge and sway velocities, as the velocities in the filter states are in the NED frame, they may be found by transformation from NED to BODY. The results are shown in Figure 5.4, and shows a decent performance by the filter in removing the white noise from the measurements.

As recognized from Figure 5.3, the heading of the vessel struggles to accurately follow the reference. Based on experience from troubleshooting it seems not to be a matter of tuning of neither controller or observer, but rather a consequence

of the course instability of the hull – an effect consistent with past sea trials. The hull of the physical ReVolt has recently undergone modification to improve hydrodynamical properties by fitting a skeg, but this is unfortunately not reflected in simulation as the digital model has yet to be updated.
While this issue should be rectified, the heading tracking performance of the vessel is adequate for the purposes of this thesis.

5.3 Baseline Case II - Environmental Disturbances

This case has similar objectives to the previous scenario, but includes relatively harsh environmental disturbances, in form of a Slight sea state as described in Table 3.3. The case shall demonstrate the ability of the model ship to withstand environmental conditions close to the edge of the seaworthiness of the vessel.
The direction of the wind in the fully developed state (reached after $\approx 60s$) is exactly 180°, blowing southwards with an average speed of $5.75\frac{m}{s}$. The wind generated waves move in the same direction, with a significant wave height of 1 m.

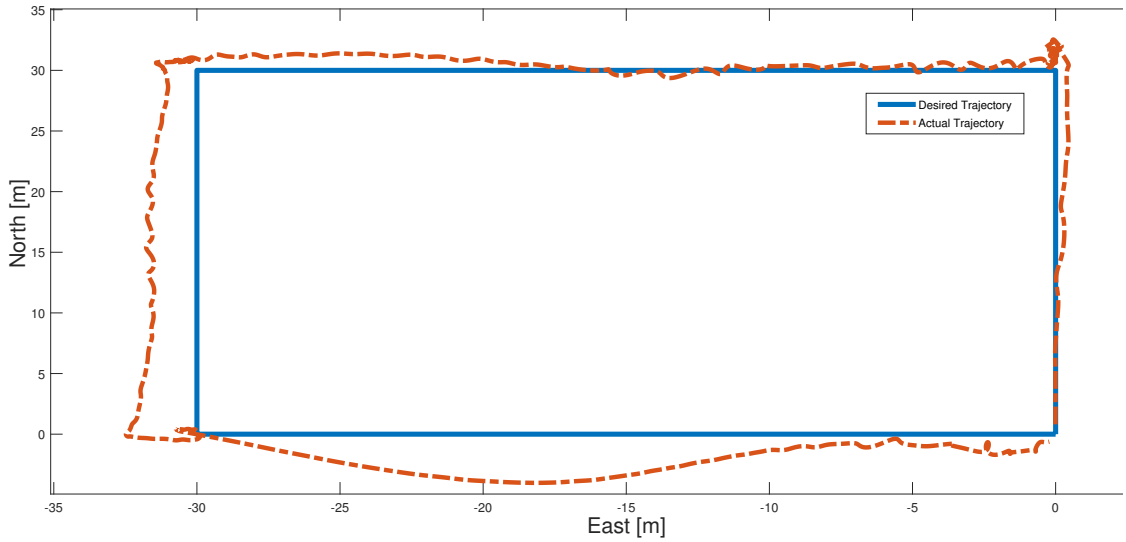


Figure 5.6: Reference trajectory and actual path followed by the vessel during Baseline Case II.

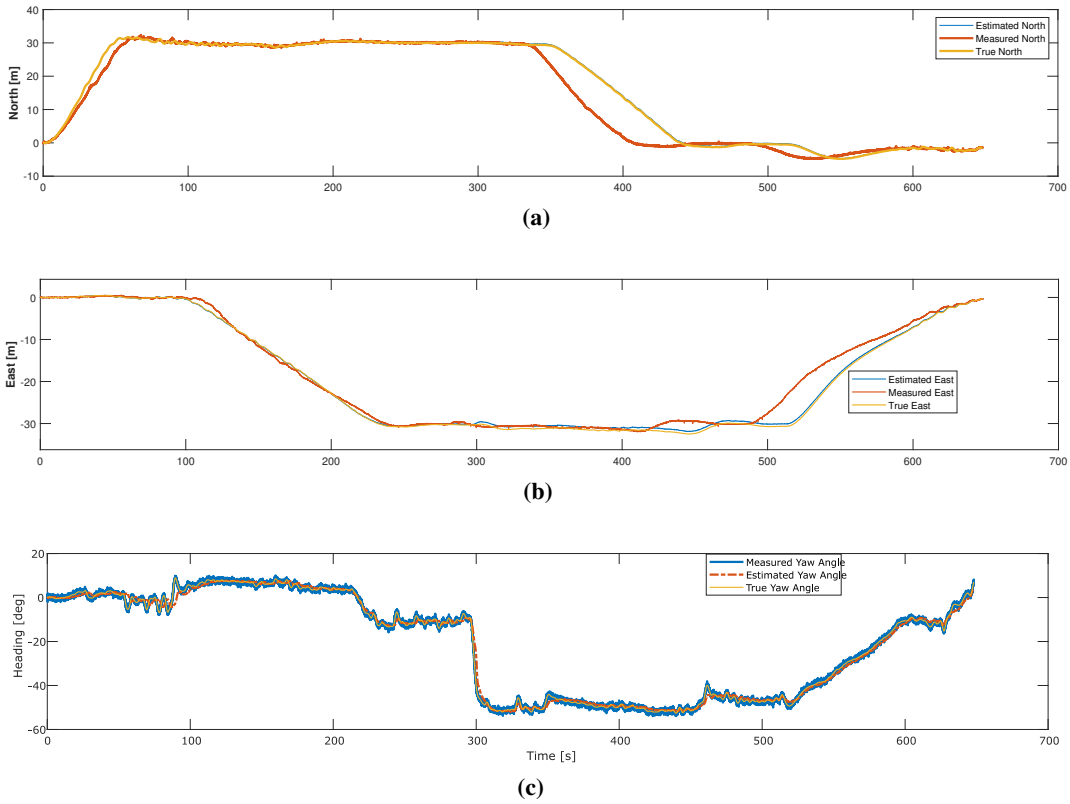


Figure 5.7: Estimated, actual and measured values of north position **(a)**, east position **(b)** and yaw angle **(c)** during scenario II.

5.3.1 Discussion – Baseline Case II

The vessel has considerable deviations from the desired trajectory, particularly while in transit between setpoints, as seen in Figure 5.6. The portions with the least amount of deviation are while the waves hit the vessel directly on the bow between setpoint I and III. After this, the waves start hitting the ship diagonally on the starboard side and the deviations increase considerably.
The yaw plot (Figure 5.7**c**) provides a decent state estimate, but seems to no wholly capture the periods of high dynamics and oscillations.

The hull form, relatively small size of vessel and lack of stabilization systems makes the ship susceptible to environmental disturbances regardless of control architecture. Short of changing the onboard hardware or hull, a weathervaning

system like WOPC (Fossen and Strand, 2001) could be realized to reduce the environmental disturbances while holding position.

5.4 Signal Fault Tolerance – Case III

This case will demonstrate the capabilities of the signal fault tolerance, by subjecting the GNSS latitude signals of unit 1 with random signal failures. To provide more challenging conditions for signal fault diagnosis and test for false positives, these faults occur both when moving forward and turning around. The different faults and their duration is shown in Table 5.2.

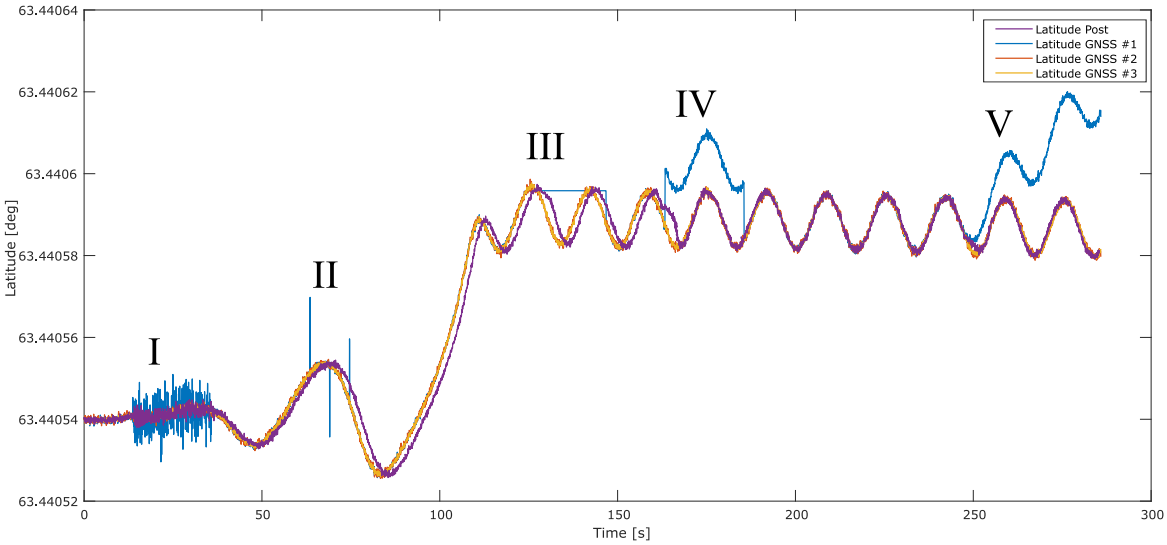


Figure 5.8: Evolution of the latitude reported by GNSS units, and the processed signal (purple). The various signal failure modes are indicated by Roman numbers.

#	Failure Mode	Start [sec]	End [sec]
I	High Variance	13.5	33.5
II	Outliers	63.5	75.5
III	Freeze	124.3	146.7
IV	Sudden Bias	163.5	185.5
V	Drifting	247.5	–

Table 5.2: List of signal faults occurring in Case III, along with their start and end times.

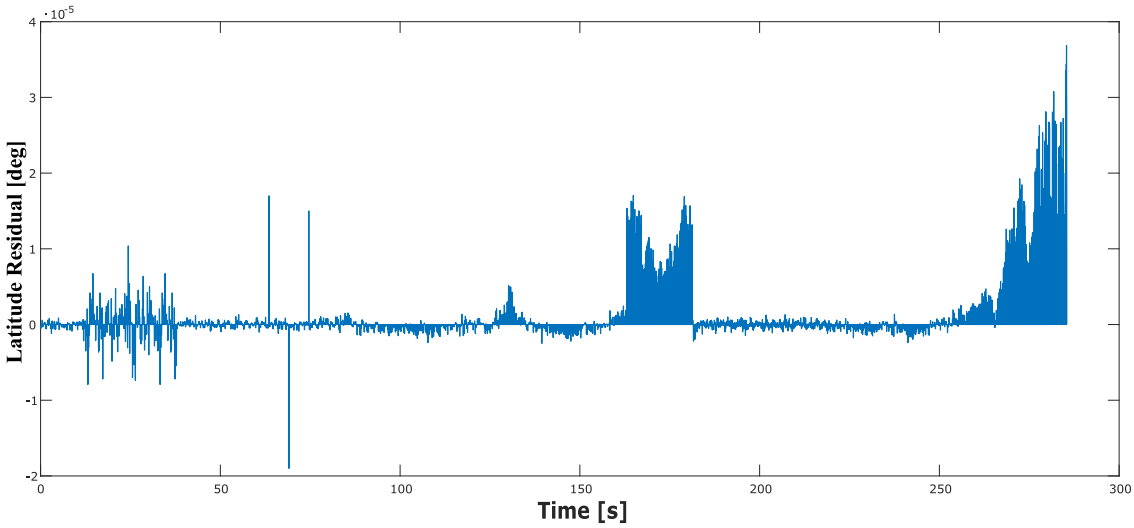


Figure 5.9: Evolution of the Latitude Residuals for GNSS unit 1.

5.4.1 Discussion – Signal Faults Case III

The signal fault tolerance module seems to have satisfactory performance, and correctly deactivates the relevant sensor upon diagnosing a failure mode, without false alarms. The high variance initially results in increased noise of the processed signal (shown in purple), but decreases after $\approx$ 3 seconds as the malfunctioning GNSS unit is deactivated. The outliers between 63.5 and 75.5 seconds do not have an impact on the processed signal, as they are flagged by the outlier detection prior to averaging. The high dynamics between 85 to 115 seconds and the subsequent oscillatory motion does not result in false positives, which indicates the system parameters are well tuned for this case. The freeze and bias are diagnosed and handled after respectively 2.3 and 2.8 seconds.

The artifacts of the signal residual (Figure 5.9) reflect the different failure modes encountered well. The sudden bias event at 163.5 seconds shows a rapid and significant change in residual magnitude, that does not vanquish. This is also true for the drifting signal at 223.5 seconds, which in addition shows a gradually growing trend in magnitude. Both these cases are possible for the CUSUM algorithm to detect, as the characteristics fulfill the aforementioned criteria of residual generators listed in Blanke et al. (2006).
The high variance segment is also clearly distinguishable, but the sign of the residual values are not consistent unlike during the sudden bias and drift modes.

5.5 Actuator Fault Tolerance – Case IV

The performance of the vessel using the actuator fault tolerance module is shown in Figure 5.10. The ⊗ symbols indicate an injection of a thruster fault, and the green circle indicates that the fault has been diagnosed and handled by the module. Note since the vessel only has three actuators, a previously deactivated thruster was re-activated each time a new thruster was diagnosed and disabled (shown by the green circles). The reason for this was to include more fault situations in the same case, as the vessel was not able to complete operation in the underactuated case with only a single functioning thruster. The status of each thruster throughout the run is shown in Figure 5.11.

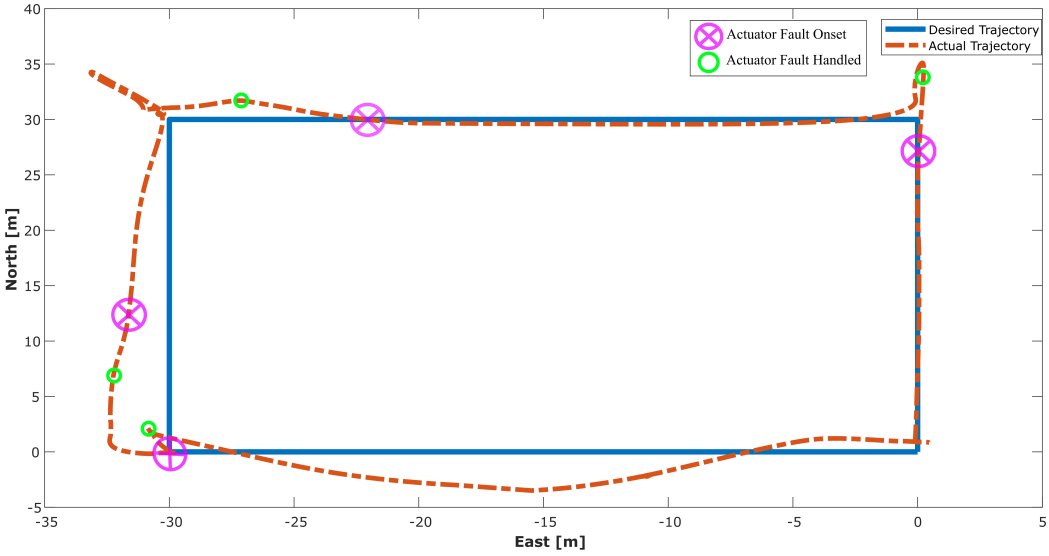


Figure 5.10: Reference trajectory and actual path followed by the vessel.

5.5.1 Discussion – Actuator Faults Case IV

Figure 5.11 reveals a mean time between fault occurrence and handling reveals of 19.8 seconds in this case. Especially the last fault of the starboard thruster has a relatively long handling time, at 33.5 seconds. It is not clear why this situation in particular results in a longer delay, but it could be because the positional error has a relatively small magnitude compared to the prior faults.

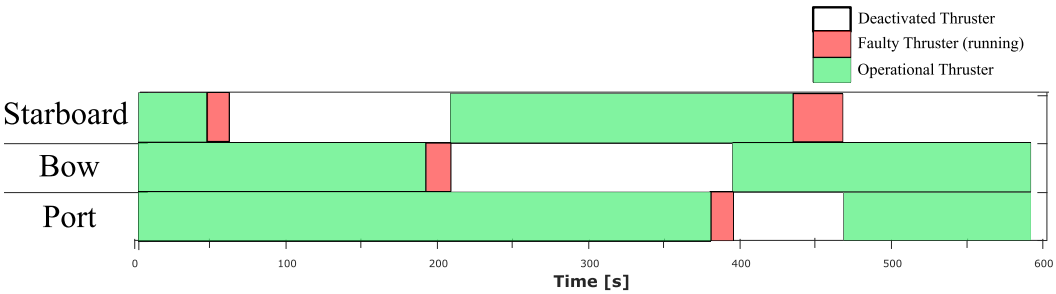


Figure 5.11: Status of each thruster as a function of time during case IV.

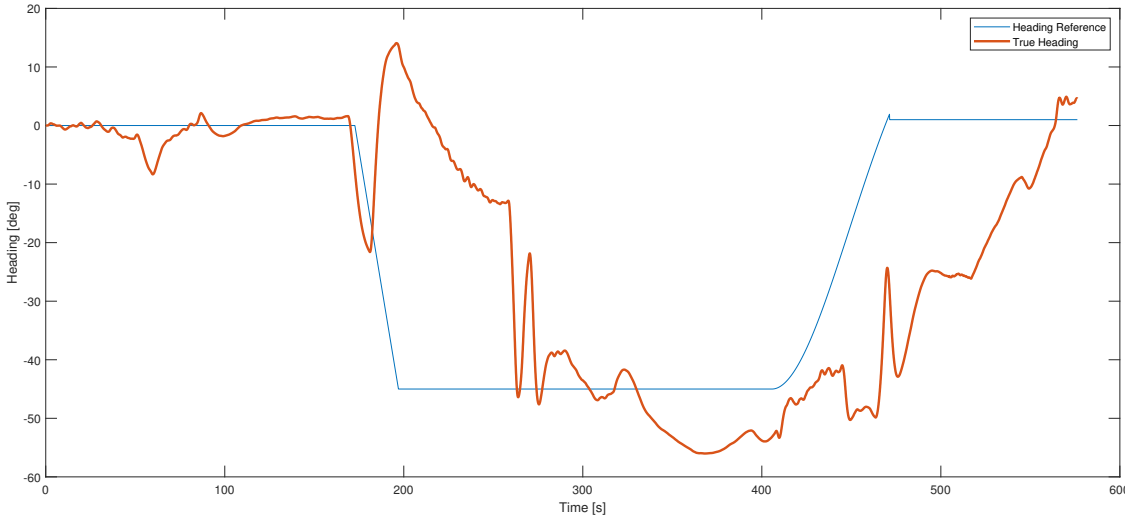


Figure 5.12: Heading setpoint and actual heading of the vessel during Case IV.

Based on the vessel trajectory of Figure 5.10 and the heading evolution of Figure 5.12, it can be asserted that the fault and subsequent deactivation of the bow thruster results in the most significantly degraded performance. This can be attributed to the difficulty of the stern thrusters to provide the demanded yaw moment of the controller, and especially in a critical situation so close to one of the corner setpoints. Despite the overshooting caused by the loss of the bow thruster, the vessel is able to follow the desired trajectory to a certain degree, and arrive at the final corner setpoint.

5.6 Combined Faults – Case V

This case puts the comprehensive system to a test with multiple challenges, stemming from actuators, sensors and environment. The environmental conditions are the same as in Case II, with a slight weather condition. Four separate actuator faults happen during the run, and is illustrated in the same way as Case IV. Signal faults happen to the roll measurements of IMU unit 2, shown in Table 5.3.

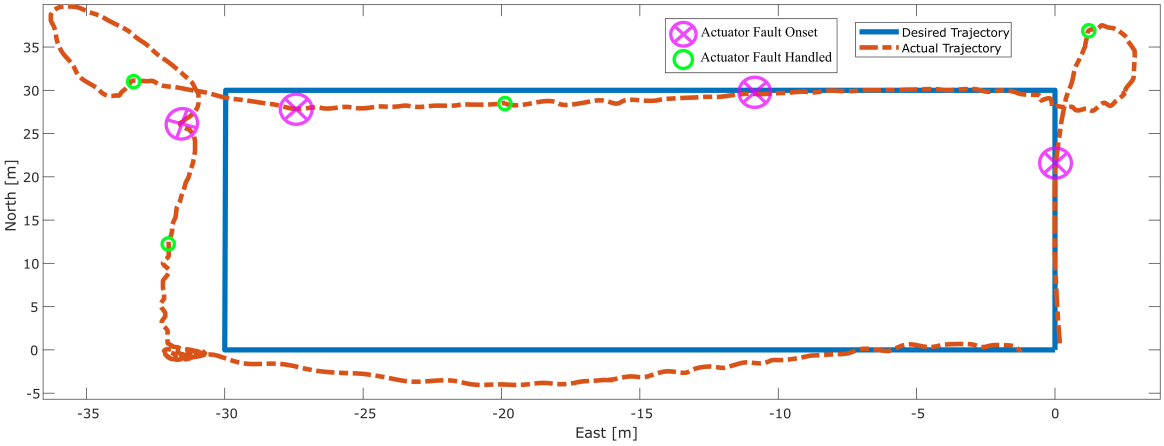


Figure 5.13: Reference trajectory and actual path followed by the vessel during Case V.

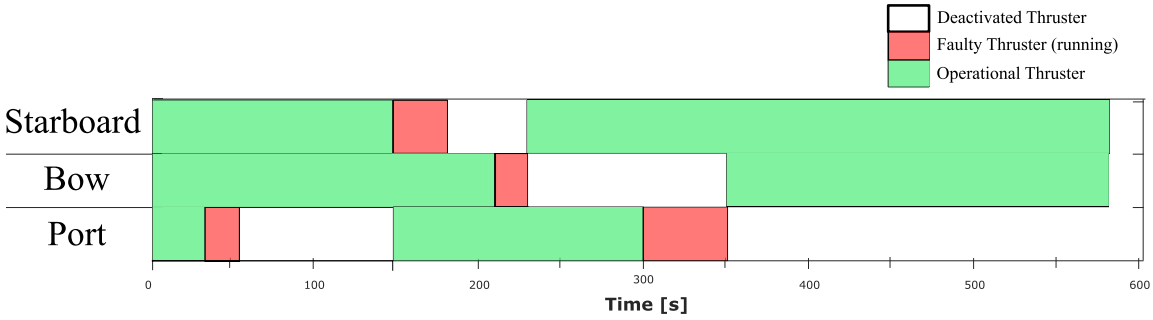


Figure 5.14: Status of each thruster as a function of time during Case V.

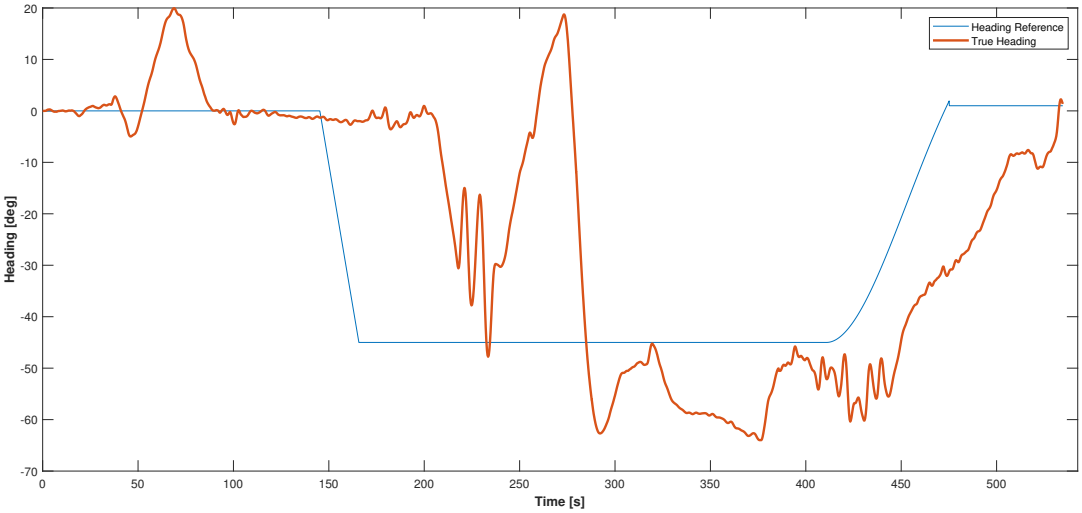


Figure 5.15: Heading setpoint and actual heading of the vessel during Case V.

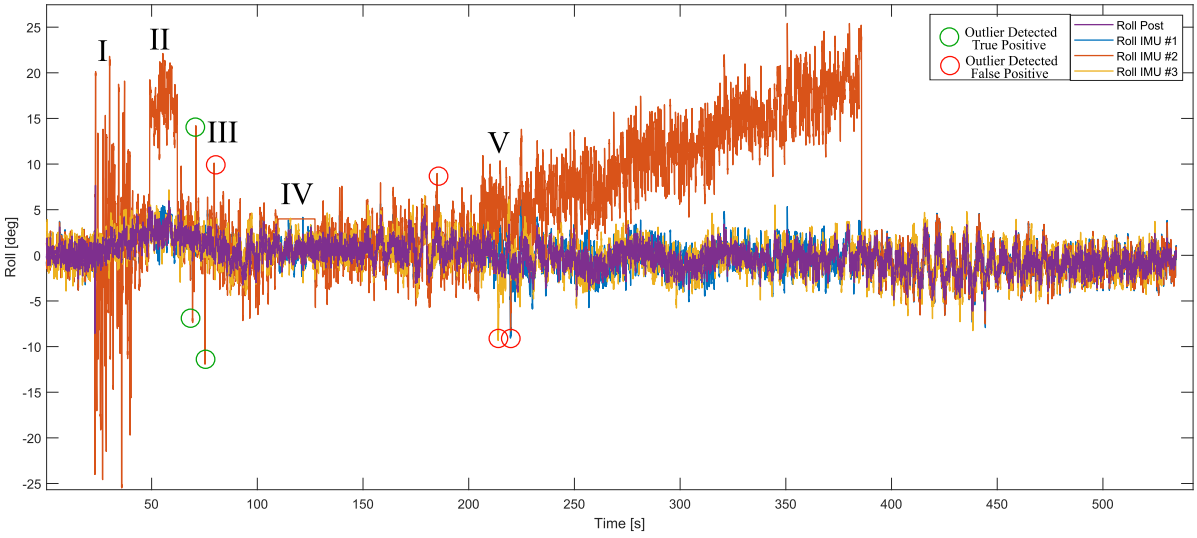


Figure 5.16: Evolution of the roll reported by IMU units, and the processed signal (purple). Faults indicated by Roman numbers, and outliers depicted as circles.

#	Failure Mode	Start [sec]	End [sec]
I	High Variance	22.9	40.4
II	Sudden Bias	49.1	62.2
III	Outliers	71.0	80.2
IV	Freeze	109.8	127.3
V	Drifting	205.1	385.8

Table 5.3: List of signal faults occurring in Case V, along with their start and end times.

5.6.1 Discussion – Combined Faults Case V

As seen in Figure 5.13, the vessel struggles to follow the desired trajectory under the weight of all the challenges. The relatively heavy environmental disturbances are especially damaging during the period between actuator fault occurrence and handling, as the vessel is unable to properly counteract the forces from wind and waves. Akin to Case IV, the bow thruster fault and subsequent disablement results in the most dramatic setpoint deviation of the run, as can be seen by the peak of Figure 5.15. However, the vessel is still able to continue the operation after the faults have been handled – which was an impossibility before the fault tolerant system.

The signal faults does not seem to cause any adverse effects to the overall control system. The different failure modes encountered in Figure 5.16 are all eventually picked up and handled by the signal fault tolerance module. However, the IMU outlier detection seems to wrongly tag some valid measurements as outliers, in the four instances symbolized with a red circle.

5.7 Evaluation of Thruster Fault-Tolerance

In the cases reviewed, the actuator fault tolerance module managed to eventually classify each actuator failure mode correctly, without false alarm instances or diagnosing of the wrong thruster. On average there seems to be a 25.6 second delay between the onset of a thruster fault and it being handled by the thruster fault-tolerance module. Precisely, the delay stems from the detection, identification and isolation operation of the recurrent neural network.
The relatively long delay is problematic in a real-life setting, considering the time limit values from Hansen (2011).

One of the reasons for this delay is the tuning of the RNN, as false positives were punished as severely as false negatives to prevent unwanted accidental performance degradation, as evidenced by relatively high precision value of the model. Loosening the cost of false positive classification during training could potentially make this delay shorter, with the trade-off being risking unnecessary and potentially dangerous drops in actuator capabilities due to false alarms.

It is possible the model lacks input features capable of serving as precursors for performance degradation, as for instance the pose error and attitude residual may be some of the last features to be affected by an actuator fault.
Building upon this, an alternative way of rectifying the delay could installation of a low-level sensor suite capable of detecting faults before they culminate in the severe performance drops detectable by the current system. Such a sensor suite measuring mechanical and electrical properties of thrusters could potentially also declare a thruster operational again after a fault, which is not possible under the current scheme.

While the detection and diagnosis part of the module has room for improvement, the fault handling seems sufficient in its current form. The switching between the thrust configurations is near instant, and it seems able to operate adequately in situations with only two thrusters operational. The natural step forward for the switching mechanism would be enabling partial deactivation of thrusters, instead of the current binary on/off approach.

5.8 Evaluation of Signal Fault-Tolerance

Overall the signal fault tolerance block seems to work well, and captured all the failure modes encountered during the cases. The averaged signal did not seem to

be drastically altered by the faults, indicating that the module acts quick enough to deactivate the sensors before faulty measurements are propagated.

All of the outliers set to occur during the simulations were detected by the One-Class Support Vector Machine implementation. However, a few IMU datapoints are incorrectly tagged as outliers in the severe weather scenario. The reason for this is most likely due to lacking data size and diversity, as most datasets were collected with calmer sea states where roll and pitch motions had far smaller magnitudes and lower frequencies. The algorithm thus labels some motion spikes as outliers, rather than legitimate vessel responses due to the challenging environmental conditions. A way of rectifying this could be by collecting more sensor datasets of harsh weather and with differing incident wave and wind angles.

Using software profiling it was discovered that the outlier detection module has a considerable computational overhead, and is at times the node with highest percentage of CPU and memory usage. The profiler also revealed that the most computational intensive part of the outlier detection was the online training of the machine learning models. By disabling the training feature and only doing classification, the CPU and memory usage dropped considerably – as seen in Figure 5.17. While this workaround placed it among the most efficient modules, a more well-rounded alternative to disabling the feature could be making the data threshold for retraining considerably higher so the retraining was conducted less frequently.

Node	PID	CPU %	Mem %
/ObserverNode	5532	22.50	2.87
/OutlierDetectionNode	5511	22.30	3.39
/ResidualObserverNode	5522	20.60	2.88
/rqt_gui_py_node_6534	6534	10.80	5.24
/modbusServer	5543	5.90	2.41
/DPcontrollerNode	5474	3.90	2.57
/thrusterAllocNode	5495	3.90	2.01
/modbusServerGPS	5551	2.90	1.99
/modbusServerVRU	5558	2.90	1.99
/ControllerNode	5458	2.00	0.36
/sig_node	5503	2.00	0.33
/rosout	5455	1.00	0.33
/GuidanceLawNode	5599	0.00	0.32
/TCPDatatransceiver	5566	0.00	0.33
/HeadingControllerNode	5585	0.00	0.34
/SpeedControllerNode	5573	0.00	0.34
/rqt_reconfigure	5618	0.00	5.55
/refFilterNode	5473	0.00	2.59

(a)

Node	PID	CPU %	Mem %
/ResidualObserverNode	9112	18.60	2.85
/ObserverNode	9119	17.90	2.85
/rqt_gui_py_node_10024	10024	8.80	5.22
/modbusServer	9123	5.00	2.41
/thrusterAllocNode	9084	4.00	2.01
/ControllerNode	9061	2.00	0.35
/sig_node	9095	2.00	0.33
/modbusServerGPS	9131	2.00	1.99
/modbusServerVRU	9139	2.00	1.99
/DPcontrollerNode	9076	2.00	2.57
/rosout	9045	1.00	0.32
/refFilterNode	9071	1.00	2.59
/GuidanceLawNode	9180	0.00	0.34
/TCPDatatransceiver	9151	0.00	0.33
/HeadingControllerNode	9166	0.00	0.34
/SpeedControllerNode	9159	0.00	0.33
/rqt_reconfigure	9192	0.00	5.54
/OutlierDetectionNode	9236	0.00	3.32

(b)

Figure 5.17: Software profiling of ROS control system before **(a)** and after **(b)** deactivation of online training for machine learning algorithms

The primary drawback of the signal fault tolerance module is that it unavoidably

introduces delays to the signals processed, as the node will need time to execute the calculations and publish the results. While run-time is heavily dependant on the hardware of the host computer, it seems to have a mean value around 167 milliseconds on a machine with specifications as in Appendix D. It should be noted the publishing and subscription operations of ROS are by default set to a frequency of 10 Hz, meaning signals will only be sent and received at 0.1 second clock intervals. The initial control system is thus by no means as fast as a Real-Time Operating System, and the relatively small latency added seems tolerable based on the current results.

Chapter 6

Conclusion

The fault tolerant control system developed in this thesis has significantly improved the dynamic positioning capabilities of the ReVolt autonomous platform. Prior to the new system, there were several single points of failure that would prevent dynamic positioning, such as a frozen thruster or IMU measurement.
An additional advantage is that debugging and tracing the origin of a failure will be easier under the new system, as the detection and diagnosis system prints and alert the user about sensory and actuator related misconducts.

A comprehensive signal fault tolerance module has been implemented, containing procedures for detection, diagnosis and isolation of important signal failure modes. In the case of physical redundancies, the module additionally offers voting capacity for eliminating deviating sensors and averaging of measurements. The results indicate that the module functions well overall, albeit with potential for improvement with regards to outlier detection of IMU measurements and code optimization.

It has also been demonstrated that the ReVolt ROS platform has the latency and computational capability to do both online prediction and training. The evaluation of the machine learning models indicate that it is indeed possible to detect certain failure modes in dynamic positioning, by using a data-driven approach utilizing a conventional sensor array. The potential of machine learning in the context of fault tolerance is substantial, as it does not require a plant model and may capture phenomena and patterns that may go unnoticed or unmodeled by a conventional scheme. The last decades improvements in data analysis capabilities, computational power and sensor prices are enabling data-driven approaches to become a serious contender in the near future.

However, there are still significant challenges with the data-driven actuator fault tolerance module implemented. Most importantly, the relatively long mean time to diagnosis of actuator faults has to be addressed before being viable in a real-world

setting. Expanding the scheme with precursor features would seem beneficial, along with enlarging the datasets and exploring different tuning approaches. Another topic that needs consideration is the issue of interpretability among common machine learning algorithms, as it may be insufficient for the requirements of verification and detailed analysis. Lack of interpretability is especially critical on a shipborne system where approval from a class society is needed, and especially so for an unmanned vessel where there are no human operators to rectify undetected faults before they develop to failure. Rigorous ways to verify AI systems is an ongoing research topic, and a consensus on methodology among scientists has not yet been reached.

Recommendations for further work

The scope of the thesis has been broad and has spanned many subfields within both marine and control engineering. Thus there is a lot of potential for improvement and further refining of the individual system components.

Expansion of datasets

The datasets used in both the outlier detection and thruster diagnostics would benefit from being expanded. As collecting data is long tedious work, the time allocated to the task had a lower priority than actual development and documentation of new features. The online learning capabilities of both algorithms should make the process easier, as data collection and training can be done in the background while doing unrelated tasks during simulations or sea trials.

Extended Kalman Filters

There are measurements that could be exploited in the Extended Kalman Filters which are already present on the ReVolt platform, but no yet available in the simulator. These include an auxiliary GNSS receiver attached to the Xsens IMU unit, which could provide sensor fusion with the dedicated satellite navigation system. Extending the sensor fusion capabilities of the EKF would provide more accurate state estimations and dead reckoning performance.

Fusion between CUSUM and EKF

As seen in the results the CUSUM implementation offers a capability to detect sudden biases in signals, but has no method of estimating the magnitude of the bias. However, the Extended Kalman Filter does provide a bias estimate for IMU accelerometers and gyro rate sensors. By fusing these capabilities together, the fault handling could be improved by automatic removal of the bias offset estimated by the EKF, instead of just deactivating units.

Actuator Neural Network

Introducing new input features which could serve as precursors for performance degradation would be a good step forward. The results indicate the currently used features have an intrinsic delay between the onset of a fault to it affects the feature variables. Note that the current scheme only considers the input commands to the actuators, and has no direct feedback of the actual output. A sensor array situated at the thrusters could for instance measure shaft speed, vibration characteristics, or current and voltage input – potentially revealing faults at a much earlier stage.

Bibliography

Alauddin, M., Khan, F., Imtiaz, S., Ahmed, S., 2018. A Bibliometric Review and Analysis of Data-Driven Fault Detection and Diagnosis Methods for Process Systems.

Alfheim, H., Muggerud, K., 2017. Development of a Dynamic Positioning System for the ReVolt Model Ship. Ph.D. thesis, NTNU.

Augustyn, A., 2019. Douglas scale, *Britannica Online Academic Edition*. URL `http://academic.eb.com/levels/collegiate/article/31054`

Blanke, M., 2005. Diagnosis and fault-tolerant control for ship station keeping. In: Proceedings of the 20th IEEE International Symposium on Intelligent Control, ISIC '05 and the 13th Mediterranean Conference on Control and Automation, MED '05.

Blanke, M., Kinnaert, M., Lunze, J., Staroswiecki, M., 2006. Diagnosis and Fault-Tolerant Control.

Blanke, M., Staroswiecki, M., 2007. Structural design of systems with safe behavior under single and multiple faults. In: Fault Detection, Supervision and Safety of Technical Processes 2006.

Bo, L., Lei, W., Xuefeng, W., Shengwen, X., Xin, L., 2018. Estimation of Thruster-thruster/current Interaction in a Dynamic Positioning System through Supervised Learning with Neural Networks. In: Twenty-eighth (2018) International Ocean and Polar Engineering Conference.

Chen, H., Moan, T., 5 2004. Probabilistic modeling and evaluation of collision between shuttle tanker and FPSO in tandem offloading. Reliability Engineering and System Safety 84 (2), 169–186.

Cournapeau, D., 2013. The Scikit-Learn project – Support Vector Machines -

Documentation (*Retrieved 2019-02-06*).
URL http://scikit-learn.org/stable/modules/svm.html

Danielsen-Haces, A., 2018. Digital Twin Development - Condition Monitoring and Simulation Comparison for the ReVolt Autonomous Model Ship. Ph.D. thesis, NTNU.

DNV GL, 2015. ReVolt – Technical Report 2015-0170. Tech. rep., DNV GL,.

Fossen, T. I., 2011. Handbook of marine craft hydrodynamics and motion control. Wiley.

Fossen, T. I., Strand, J. P., 5 2001. Nonlinear passive weather optimal positioning control (WOPC) system for ships and rigs: experimental results. Automatica 37 (5), 701–715.
URL https://www.sciencedirect.com/science/article/pii/S0005109801000061

Freeman, P., Pandita, R., Srivastava, N., Balas, G. J., 2013. Model-based and data-driven fault detection performance for a small UAV. IEEE/ASME Transactions on Mechatronics.

García-Gonzalo, E., Fernández-Muñiz, Z., Nieto, P. J. G., Sánchez, A. B., Fernández, M. M., 2016. Hard-rock stability analysis for span design in entry-type excavations with learning classifiers. Materials.

Hansen, E. O. S., 2011. DP Dependability. In: Dynamic Positioning Conference. Rolls-Royce Marine.

Hassani, V., Pascoal, A. M., Sørensen, A. J., 2018. Detection of mooring line failures using Dynamic Hypothesis Testing. Ocean Engineering.

Haugen, O. I., Smogeli, . R., 2017. Safety of Dynamic Positioning. pp. 1–11.
URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118476406.emoe363

Havnegjerde, A., 2018. Remote Control and Path Following for the ReVolt Model Ship. Remote Control and Path Following for the ReVolt Model Ship. Ph.D. thesis, NTNU.

Hsu, D., 05 1996. Comparison of four gravity models. pp. 631 – 635.

IEEE, Aug 2008. Ieee standard for floating-point arithmetic. IEEE Std 754-2008, 1–70.

Kamsvåg, V., 2018. Fusion between camera and lidar for autonomous surface vehicles. Ph.D. thesis, NTNU.

Ketkar, N., 2017. Deep Learning with Python.

Kohavi, R., Provost, F., 1998. Glossary of terms. Journal of Machine Learning (30), 271274.

Kraft, E., 2003. A quaternion-based unscented Kalman filter for orientation tracking. In: Proceedings of the 6th International Conference on Information Fusion, FUSION 2003. Vol. 1. IEEE Computer Society, pp. 47–54.

Kulkarni, S., Harman, G., 2011. Elementary Introduction to Statistical Learning Theory.

lsermann, R., 2006. Fault-Diagnosis Systems. Springer Verlag.

Marins, J., Xiaoping Yun, Bachmann, E., McGhee, R., Zyda, M., 11 2002. An extended Kalman filter for quaternion-based orientation estimation using MARG sensors. Institute of Electrical and Electronics Engineers (IEEE), pp. 2003–2011.

Millan, J., 2008. Thrust allocation techniques for dynamically positioned vessels. Tech. rep., National Research Council Canada, Institute for Ocean Technology.

Modbus Organization, I., 2012. MODBUS Application Protocol Specification V1.1b3 Modbus. Tech. rep.
URL `http://www.modbus.org/specs.php`

Mokleiv, B., 2017. Fault-tolerant Observer Design. Ph.D. thesis, NTNU.

Pedregosa, F., 2011. Scikit-learn: Machine Learning in Python. Journal Of Machine Learning Research (12), pp.28252830.

Psiaki, M. L., Martel, F., Pal, P. K., 6 1990. Three-axis attitude determination via Kalman filtering of magnetometer data. Journal of Guidance, Control, and Dynamics 13 (3), 506–514.

Schmidhuber, J., Hochreiter, S., 1997. Long Short-Term Memory. Neural Computation 9, pp.17351780.

Scholkopf, B., Williamson, R., Smola, A., Shawe-Taylor, J., Platt, J., Holloway, R., 2000. Support Vector Method for Novelty Detection. Tech. rep.

Shiyu, J., 2017. Kernel Trick Illustration – Licensed under: Creative Commons 4.0. (*Retrieved 2018-10-28*).
URL `https://commons.wikimedia.org/wiki/File:Kernel_trick_idea.svg`

Skjetne, R., Egeland, O., 2006. Hardware-in-the-loop testing of marine control systems ¶. Tech. Rep. 4.

Sørensen, A. J. e. a., 2013. Marine Control Systems Propulsion and Motion Control of Ships and Ocean Structures Lecture Notes. Tech. rep.

Stetco, A., Dinmohammadi, F., Zhao, X., Robu, V., Flynn, D., Barnes, M., Keane, J., Nenadic, G., 2018. Machine learning methods for wind turbine condition monitoring: A review. Renewable Energy 133, 620–635.
URL https://doi.org/10.1016/j.renene.2018.10.047

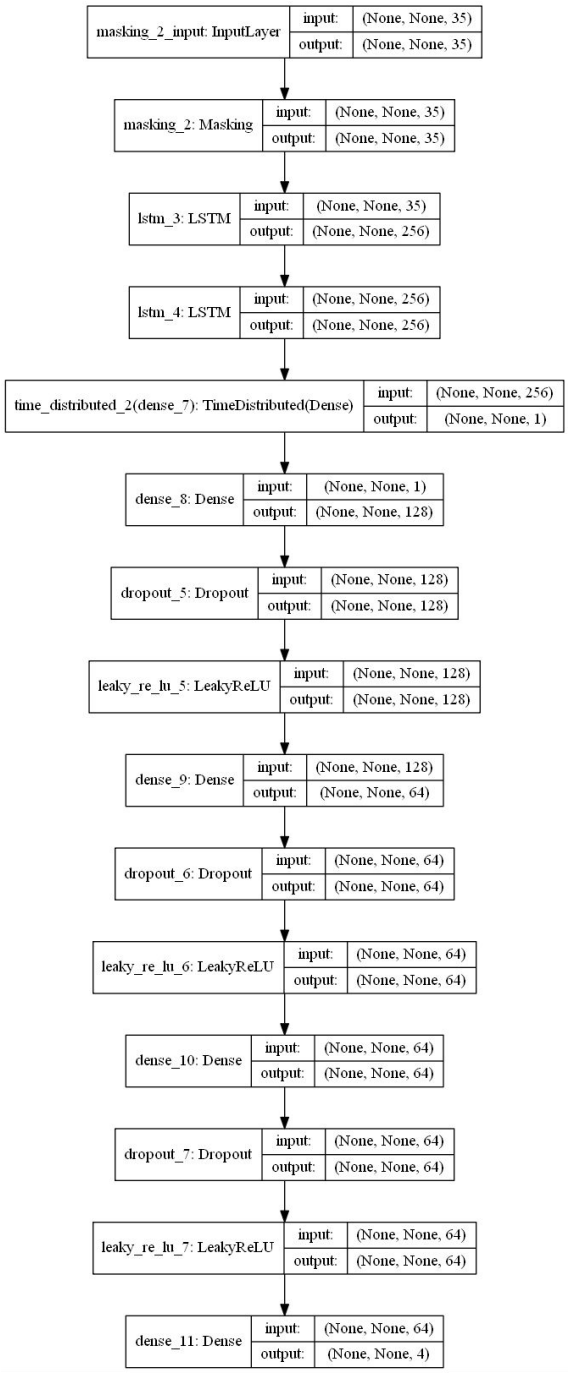
TensorFlow, 2019. TensorFlow Documentation – Categorical Accuracy (*Retrieved 2019-05-10*).
URL https://www.tensorflow.org/api_docs/python/tf/keras/metrics/CategoricalAccuracy

Vaerno, S. A., Brodtkorb, A. H., Skjetne, R., Calabro, V., 7 2017. Time-varying model-based observer for marine surface vessels in dynamic positioning. IEEE Access 5, 14787–14796.

Zhang, Y., Wu, N. E., Jiang, B., 2008. Fault detection and isolation applied to a ship propulsion benchmark. In: IFAC Proceedings Volumes (IFAC-PapersOnline).

Appendix

Appendix A – Layer Architecture LSTM



Appendix B – GNSS Receiver Datasheet

Vector VS330 GNSS Receiver

GNSS Receiver Specifications

Receiver Type:	Vector GNSS L1/L2 RTK Receiver
Signals Received:	GPS, GLONASS, BeiDou, and Atlas
Channels:	540
GPS Sensitivity:	-142 dBm
SBAS Tracking:	3-channel, parallel tracking
Update Rate:	10 Hz standard, 20 Hz optional
Timing (1PPS) Accuracy:	20 ns
Rate of Turn:	100°/s maximum
Compass Safe Distance:	30 cm (with enclosure)[5]
Cold Start:	60 s (no almanac or RTC)
Warm Start:	20 s typical (almanac and RTC)
Hot Start:	5 s typical (almanac, RTC and position)
Heading Fix:	20 s typical (valid position)
Maximum Speed:	1,850 mph (999 kts)
Maximum Altitude:	18,288 m (60,000 ft)
Differential Options:	SBAS, Beacon, External RTCM, Atlas L-band and Athena RTK

Positioning and Heading Accuracy

	Horizontal	Vertical
RMS:		
Single Point [1]:	1.2 m	2.5 m
SBAS (WAAS) [1]:	0.3 m	0.6 m
Code Differential GNSS [1]:	0.3 m	0.6 m
L-Band [2]:	0.08m	0.16 m
RTK [1, 3]:	10 mm + 1 ppm	20 mm + 2 ppm
Heading Accuracy:	0.2° rms @ 0.5 m antenna separation 0.1° rms @ 1.0 m antenna separation 0.05° rms @ 2.0 m antenna separation 0.02° rms @ 5.0 m antenna separation	
Pitch/Roll Accuracy (RMS):	1°	
Heave Accuracy (RMS):	30 cm (DGPS) [5],10 cm (RTK) [1,3]	

Beacon Receiver Specifications

Channels:	2-channel, parallel tracking
Frequency Range:	283.5 to 325 kHz
Operating Modes:	Manual, Automatic, and Database
Compliance:	IEC 61108-4 beacon standard

L-Band Receiver Specifications

Receiver Type:	Single Channel
Channels:	1530 to 1560 MHz
Sensitivity:	-130 dBm
Channel Spacing:	5 kHz
Satellite Selection:	Manual or Automatic
Reacquisition Time:	15 sec (typical)

Communications

Serial Ports:	2 full-duplex RS232, 1 half-duplex RS422 port
USB Ports:	1 USB-A
Baud Rates:	4800 - 115200
Correction I/O Protocol:	RTCM SC-104, L-Dif™ [6], RTCM v2 (DGPS), RTCM v3 (RTK), CMR (RTK), CMR+ (RTK) [3]
Data I/O Protocol:	NMEA 0183, Hemisphere GNSS binary [4]
Timing Output:	1 PPS (CMOS, active high, rising edge sync, 10 kΩ, 10 pF load)

Power

Input Voltage:	8-36 VDC
Power Consumption:	5.3 W nominal (GPS L1/L2 + GLONASS L1/L2) 7 W nominal (GPS L1/L2 + GLONASS L1/L2 + BeiDou B1/B2 + L-band)
Current Consumption:	0.44 A nominal (GPS L1/L2 + GLONASS L1/L2) 0.51 A nominal (GPS L1/L2 + GLONASS L1/L2 + BeiDou B1/B2 + L-band)
Power Isolation:	500 V
Reverse Polarity Protection:	Yes
Antenna Voltage:	5 VDC maximum 60mA
Antenna Short Circuit Protection:	Yes
Antenna Gain Input Range:	10 to 40 dB
Antenna Input Impedance:	50 Ω

Environmental

Operating Temperature:	-30°C to + 70°C (-22°F to + 158°F)
Storage Temperature:	-40°C to + 85°C (-40°F to + 185°F)
Humidity:	95% non-condensing
Mechanical Shock:	EP455 Section 5.14.1 Operational (when mounted in an enclosure with screw mounting holes utilized) EP455 Section 5.15.1 Random
Vibration:	
EMC:	CE (IEC 60945 Emissions and Immunity) FCC Part 15, Subpart B CISPR22
Enclosure:	IP66 (IEC 60529)

Mechanical

Dimensions:	20.2 L x 12.0 W x 7.5 H (cm) 8.0 L x 4.7 W x3.0 H (in)
Weight:	~1.1 kg (~2.5 lbs.)
Status Indications (LED):	Power, Primary and Secondary GPS lock, Differential lock, DGPS position, Heading, RTK lock, L-band DGNSS lock
Power Switch:	Front panel soft switch
Power/Data Connector:	9-pin ODU metal circular
Power Connector:	2-pin ODU metal circular
Data Connector:	DB9 (sealed)
Antenna Connectors:	2 TNC (female)

Aiding Devices

Gyro:	Provides heading smoothing with GNSS. Drift rate is 1° per minute in heading for periods up to 3 minute when loss of GNSS has occurred [4]
Tilt Sensors:	Provide pitch, roll data, assist in fast start-up and heading reacquisition

1 Depends on multipath environment, number of satellites in view, satellite geometry, no SA, and ionospheric activity.

2 Requires a subscription

3 Depends on multipath environment, number of satellites in view, satellite geometry, baseline length (for differential services), and ionospheric activity.

4 Based on a 40 second time constant

5 This is the minimum safe distance measured when the product is placed in the vicinity of the steering magnetic compass. The ISO 694 defines "vicinity" relative to the compass as within 5 m (16.4 ft) separation.

6 Hemisphere GNSS proprietary

Authorized Distributor:

Rev. 08/17

Hemisphere®

Hemisphere GNSS, Inc.
8515 E. Anderson Drive
Scottsdale, AZ, USA 85255

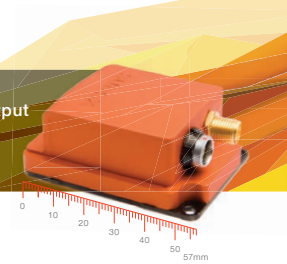
Toll-Free: +1-855-203-1770
Phone: +1-480-348-6380
Fax: +1-480-270-5070
precision@hgnss.com
www.hgnss.com

Appendix C – IMU Datasheet

MTi-G-710

Turnkey GNSS/INS solution for navigation and stabilization applications

- ✔ All-in-one sensor system with high-frequency position and orientation output
- ✔ Excellent heading tracking without requiring a magnetic field
- ✔ Configurable output settings, synchronizes with any 3rd party device



Complete Xsens sensor fusion algorithm

- Compensation against long-lasting transient accelerations
- Ability to cope with GNSS outages
- Non-magnetic heading reference
- Tuned for performance under vibrations
- Selectable filter profiles for range of applications

Easy software integration

- Extensive suite of configurable output formats, calculated onboard the MTi-G-710
- MT Software Suite with intuitive GUI
- Complete SDK for all operating systems
- Support for Robotic Operating System (ROS)
- Xsens Xbus protocol or ASCII (NMEA)
- Access to BASE (by Xsens), an extensive knowledge base and community forum

Best-in-class hardware design

- Highest quality industrial grade components
- Vibration-rejecting gyroscopes and accelerometers
- Built-in multi-GNSS receiver (GPS, GLONASS, BeiDou, Galileo) and barometer
- Wide array of (time) synchronization options

Specification highlights

- True North without requiring a magnetic field
- IP67 encased version or OEM board
- Choice of several interfaces and onboard USB
- All Xsens products are fully interchangeable
- Cost-effective system integrator solution
- Position, velocity and orientation outputs

Product overview

		MTi-G-710 GNSS/INS
Calibrated Sensor Data		yes
Roll/pitch	Static	0.2°
	Dynamic	0.3°
Yaw		0.8°
Position and velocity		
Horizontal position	1σ STD (SBAS)	1.0 m
Vertical position	1σ STD (SBAS, baro)	2.0 m
Velocity accuracy	1σ RMS	0.05 m/s

All above specifications based on typical application scenarios

Appendix C – IMU Datasheet

Sensor specification

	Gyroscopes	Accelerometers
Standard full range	+/- 450 °/s*	+/- 20 g
Initial bias error	0.2 °/s	5 mg
In-run bias stability	10 °/h	15 µg
Bandwidth (-3 dB)	415 Hz	375 Hz
Noise density	0.01 °/s/√Hz	60 µg/√Hz
g-sensitivity (calibrated)	0.003 °/s/g	N/A
Non-orthogonality	0.05 deg	0.05 deg
Non-linearity	0.01%	0.1%

	Magnetometer	Barometer
Standard full range	+/- 8 G	300-1100 hPa
Total RMS noise	0.5 mG	3.6 Pa
Non-linearity	0.2%	N/A
Resolution	0.25 mG	8 cm (sea level, 15 °C)

GNSS receiver

Receiver type	72-channel, 4 Hz GPS/QZSS L1 C/A, GLONASS L10F, BeiDou B1, SBAS L1 C/A: WAAS, EGNOS, MSAS	Horizontal accuracy (CEP)	2.0 m (SBAS) 2.5 m (Autonomous)
Start-up time cold start	26 s	Velocity accuracy (@30 m/s)	
Tracking sensitivity	-167 dBm		0.05 m/s

* Optional +/- 1000 °/s available on request.

System specifications

Input voltage	4.5 to 34V or 3V3	Output frequency	Up to 2 kHz
Typical power consumption	750 mW @ 5V	Interfaces	RS232/RS422/RS485/USB UART
IP-rating	IP67 (encased)	Latency	<2 ms
Temperature (in use)	-40 to 85 °C	Clock drift	1 ppm or external reference
Vibration	MIL-STD-202-201A/204C/214A	Interface protocol	Xbus or ASCII (NMEA)
Sampling frequency	10 kHz/ch (60kS/s)	MTBF	300,000 hours
Sync options	SyncIn, SyncOut, Clock sync 1 PPS	Mounting orientation	No restriction, full 360° in all axes

MTi-G-710 Development Kit: MTi-G-710, antenna, software and cabling

MTi-G-710 encased: 57x42x23.5 mm, 55g, 9-pins push-pull connector

MTi-G-710 OEM: 37x33x12 mm, 11g, 16-pins header

Appendix D – Computer Hardware Specifications

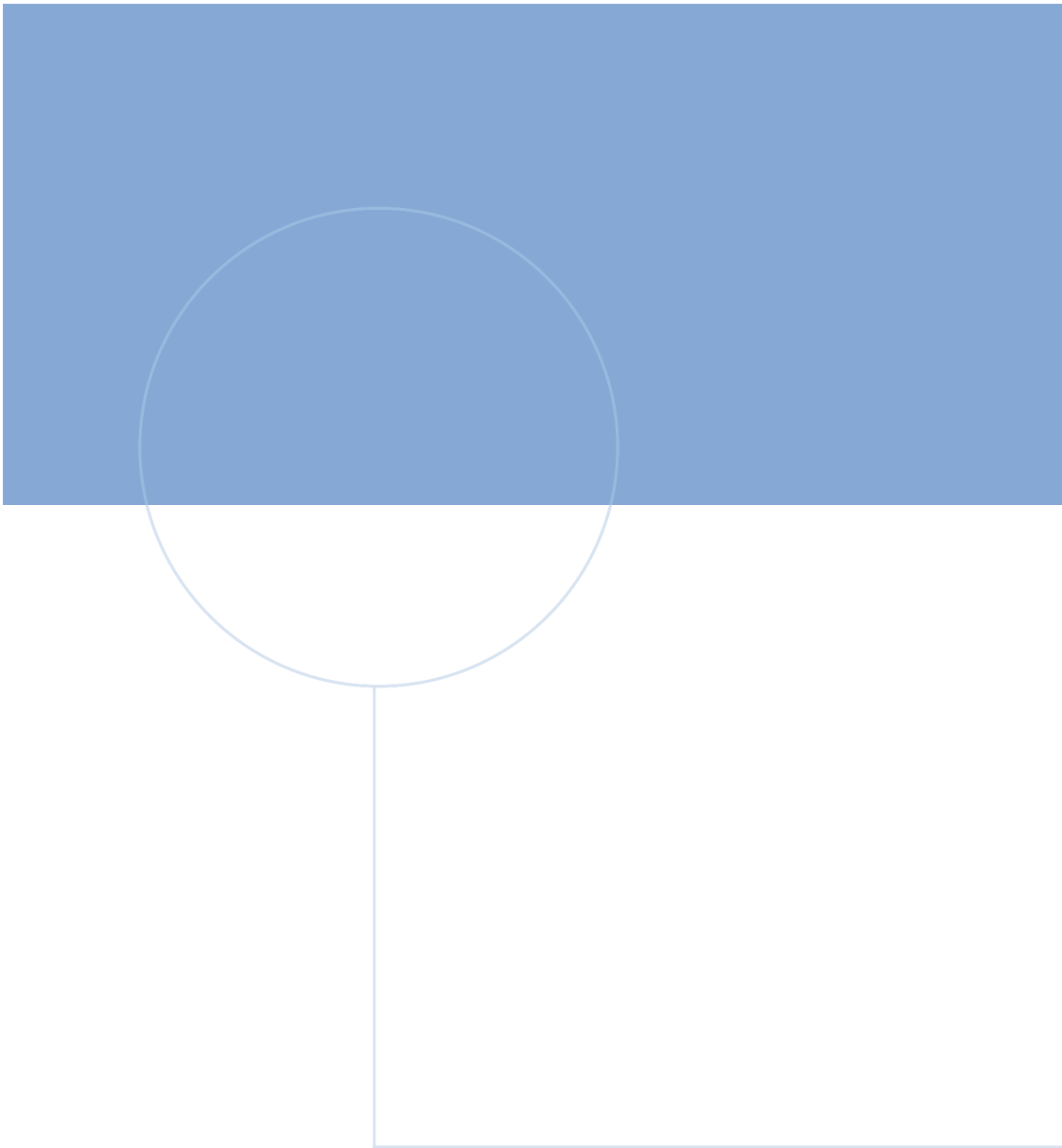
Component	Specification
Central Processing Unit	Intel Core i5-8250U @ 1.80 GHz
Graphical Processing Unit	NVIDIA GeForce 920MX
Random Access Memory	8 GB, DDR4-2400
Storage	256 GB SSD

Appendix E – DP Controller Parameters

Parameter	Value	Parameter	Value	Parameter	Value
K_p Surge	2.5	K_i Surge	0.01	K_d Surge	30
K_p Sway	5.5	K_i Sway	0.02	K_d Sway	30
K_p Yaw	40	K_i Yaw	0.195	K_d Yaw	160

Appendix F – EKF Parameters

$$\mathbf{Q}_{\text{Kalman}} = \begin{bmatrix} 0.455^2\mathbf{I}_{3x3} & 0_{3x4} & 0_{3x3} & 0_{3x3} & 0_{3x3} \\ 0_{4x3} & 0.05^2\mathbf{I}_{4x4} & 0_{4x3} & 0_{4x3} & 0_{4x3} \\ 0_{3x3} & 0_{3x4} & 0.22^2\mathbf{I}_{3x3} & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x4} & 0_{3x3} & \mathbf{I}_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x4} & 0_{3x3} & 0_{3x3} & \mathbf{I}_{3x3} \end{bmatrix} \quad \mathbf{R}_{\text{Kalman}} = \begin{bmatrix} 1.2^2\mathbf{I}_{3x3} & 0_{3x4} & 0_{3x3} & 0_{3x3} & 0_{3x3} \\ 0_{4x3} & 0.05^2\mathbf{I}_{4x4} & 0_{4x3} & 0_{4x3} & 0_{4x3} \\ 0_{3x3} & 0_{3x4} & 0.15^2\mathbf{I}_{3x3} & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x4} & 0_{3x3} & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x4} & 0_{3x3} & 0_{3x3} & 0_{3x3} \end{bmatrix}$$



NTNU
Norwegian University of
Science and Technology