Master's thesis

NTNU
Norwegian University of Science and Technology

Pauline Røstum Bellingmo

DP Control System for Blueye Pioneer

Master's thesis in Marine Technology
Supervisor: Roger Skjetne (main), Andreas Viggen (co), Martin Ludvigsen (co)
June 2020

NTNU
Norwegian University of
Science and Technology

blueye

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

MASTER OF TECHNOLOGY THESIS DEFINITION (30 SP)

Name of the candidate:	Bellingmo, Pauline Røstum
Field of study:	Marine control engineering
Thesis title (Norwegian):	DP kontrollsystem for Blueye Pioneer
Thesis title (English):	DP Control System for Blueye Pioneer

Background

For the off-the-shelf Remotely Operated Vehicle (ROV) Blueye Pioneer (www.blueyerobotics.com), there is an interest to develop a robust and easy to handle dynamic positioning (DP) capability for the NTNU AUR-Lab owned version of it. While high-end professional ROV systems are typically operated by skilled teams and technicians, whom are able to handle faults and unforeseen events, the general user (e.g. a student in AUR-Lab experimental activities) does not necessarily possess such skills. Hence, there is a high focus on fault-tolerance, reliability, and availability of a DP function in NTNU's Blueye Pioneer (referred hereafter as *the drone*). Fault-tolerant algorithms must automatically detect and handle relevant failure modes – to make out a higher level of autonomy for the drone.

In this thesis, the goal is to investigate the relevant sensor suite of the Blueye Pioneer and develop a corresponding fault-tolerant DP control system, which includes a fault-tolerant observer, controller, and necessary guidance functionality. The observer should be fault-tolerant with respect to the most common and important failure modes in the sensor suite.

Work description

1) Perform a background and literature review to provide information and relevant references on:
    - Blueye Pioneer underwater drone.
    - Dynamic modeling of ROVs and identification of model parameters.
    - Relevant sensors and observer designs for ROVs in DP (stationary/low-speed) operations, especially acoustic positioning sensor, gyrocompass, angular rate sensors, and depth sensor.
    - Dynamic positioning, incl. main DP control system components and DP observer designs.
    - Relevant failure modes of DP sensors, and references to fault-tolerant DP observer designs.
    - Derivative free optimization for DP.

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

2) Present the main characteristics and functionalities of the Blueye Pioneer drone.
    - Make an "Actuator table" listing each thruster, including information on its location in body frame, main thrust direction, maximum power and force.

3) Define the available sensor suite to be used for DP. This includes the acoustic positioning system (APS), heading (compass/gyro), pressure sensor, IMU, and possibly DVL.
    - Make a "Sensors table" listing each relevant sensor, and describe its signal type, interface, flags, noise StD, selected failure modes, and any other relevant parameter.

4) Describe the DP system hardware architecture by short descriptions and drawing(s) showing the subsystems and relevant components, considering powering and control system integration.

5) Make a "Software topology drawing" illustrating the software decomposed into subsystems and modules, with special focus on where you will introduce the DP algorithms.

6) Make a problem statement of the DP problem for the Blueye Pioneer, considering definition of workspace, configuration space (modes to control), and relevant reference frames, DP Observer design problem, DP Control design problem, and necessary guidance system functionality.

7) Given your model(s), describe and carry out system identification – some by empirical formulas and some by free running test and derivative free optimization (DFO).
    - Describe how the free running test and DFO was performed. Present and discuss the results.

8) Design and develop the DP observer. Start with nominal and then extend it with fault-tolerance. Describe and perform DP observer tuning/calibration. Test the DP observer, present and discuss the results.
9) Perform DP control design and implementation. Describe and perform DP feedback control tuning/calibration. Test the DP control system, present and discuss the results.

Specifications

Every weekend throughout the project period, the candidate shall send a status email to the supervisor and co-advisors, providing two brief bulleted lists: 1) work done recent week, and 2) work planned to be done next week.

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, problem, design, results, and critical assessments. 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 elements: Title page, abstract, acknowledgement, project definition, list of symbols and acronyms, table of contents, introduction (project motivation, objectives, scope and delimitations), background/literature review, problem formulation, method, results, conclusions with recommendations for further work, references, and optional appendices. 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, which is taken very seriously by the university and cause 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:	15 January, 2020	**Due date:**	As specified by the administration.

Supervisor:	Roger Skjetne
Co-advisor(s):	Andreas Viggen Henriksen (Blueye), Martin Ludvigsen.

Trondheim, April 14, 2020

Roger Skjetne
Supervisor

Summary

A fault-tolerant dynamic positioning (DP) control system for the remotely operated vehicle Blueye Pioneer (hereby referred to as ”the drone”) is developed in this thesis. As a part of developing the control system, relevant information and characteristics regarding the drone and the external acoustic positioning system used, are described. The chosen observer is a model-based extended Kalman filter developed using a low-speed model of the drone. The model parameters are estimated using the geometric properties of the drone, Eidsvik’s method for the hydrodynamic parameters, and Derivative-free Optimization (DFO) on experimental data for the damping parameters. The observer is developed to handle asynchronous measurements and to detect and handle signal freeze of the measurements. For other relevant failure modes, e.g. outlier and high noise, mechanisms for detection and handling are proposed. A guidance system consisting of a pure pursuit velocity reference generator is developed. Lastly, PID control laws for surge, sway, heave and yaw are designed.

The estimation of model parameters is challenging and does not give a high accuracy model for the control system. To compensate for this, the observer is designed with high process noise covariance in order to rely more on the measurements than the model. The state estimates with corresponding measurements follows the measurements closely, while the velocity and bias estimates are highly fluctuating. The control system is able to go to the desired position in x, y, and z with an accuracy of less than 1m. However, the drone does not follow the shortest path to the desired position, but rather makes a large detour before arriving. The heading controller does not give satisfactory results as it has large oscillations around the desired heading angle.

Sammendrag

I denne masteroppgaven er et feil-tolerant dynamisk posisjonering (DP) kontrollsystem utviklet for undervannsdronen Blueye Pioner. For å kunne utvikle kontrollsystemet er relevant informasjon og egenskaper relatert til dronen og dens indre sensorer i tillegg til det eksterne akustiske posisjonseringssystemet presentert. Et extended Kalman filter blir valgt som estimator og utviklet basert på en lavhastighetsmodell av dronen. Modellparametrene er estimert ved å bruke de geometriske egenskapene til dronen, Eidsviks metode for hydrodynamiske parametre, og derivative-fri optimalisering på data samlet i Marin Cybernetic's laben for å finne dempning. Relevante feilmodier for sensorene er identifisert og noen er implementert i estimatoren. For de resterende feilmodiene som ikke er implementert, er metoder for å detektere og håndtere disse foreslått. Et referansesystem basert på metoden pure pursuit er implementert for å generere referansehastigheter til kontrollsystemet. Til slutt utvikles kontrollere for jag, svai, hiv og gir.

Identifikasjonen av dempingen i modellen er utfordrende og gir ikke en model med høy nøyaktighet. For å kompansere for en unøyaktig modell, blir kovariansen av prosessstøyen i estimatoren satt til en høy verdi, slik at estimatoren stoler mer på målingene enn modellen. Dette fører til at estimatene med tilhørende målinger følger målingene tett. Hastighet- og bias-estimatetene er fluktuerende. Kontrollsystemet leder dronen til ønsket posisjon i x, y, og z med en nøyaktighet på under 1m. Heading-vinkelen oscillerer rundt ønsket vinkel uten å konvergere.

Preface

This master thesis is a continuation of the project thesis written in the fall semester (Bellingmo, 2019). The thesis is written as part of the study program Marine Technology at the Norwegian University of Science and Technology (NTNU). A part of the experimental testing in this thesis was done in the NTNU's Marine Cybernetics Laboratory. This thesis was written during the spring of 2020 when the university was shut down due to the virus Covid-19. Through this, I have learned the benefits of digital guidance, but also sought the opportunity to get help from fellow students. Additionally, working undisturbed at home has its benefits.

My supervisor at NTNU, Roger Skjetne, has helped me with general questions regarding the scope of the work and possible approaches for the assignments. Additionally, he has provided me with great guidance when using Derivative-free optimization to identify model parameters. My co-supervisor at Blueye, Andreas Viggen, has helped with topics specific for the Blueye drone. Also other employees at Blueye has helped with software issues related to the Blueye drone. The employees at Water Linked has provided me with all the relevant information needed to use their acoustic positioning system. Most of the software related issues encountered have been solved by the help of a fellow student and google.

This thesis has been challenging at times, but rewarding once a solution has been found. I have found it especially rewarding to be able to do real-life testing with the Blueye drone and an acoustic positioning system outside. Through the project, I have gained new knowledge regarding hardware and software related issues, which I believe will be much appreciated in the future.

It is assumed that the reader has knowledge of hydrodynamics and control theory.

Acknowledgements

Through the work on my thesis I have received help from several people, for which some will be mentioned here. Firstly, I would like to give thanks to my supervisor, Roger Skjetne, for guidance and support. I would also like give thanks to my co-supervisor at Blueye Robotics, Andreas Viggen, for help with all sorts of practical issues. Moreover, I am grateful that Erlend Eriksen at Blueye Robotics provided me with essential guidance to solve software related issues at a time of need.

Water Linked has lent me their acoustic position system and guided me through any related issues, for which I am very grateful.

I would like to thank Torgeir Wahl at NTNU for support when using the position system Qualisys in the Marine Cybernetics Laboratory.

Lastly, I would like to thank my fellow student Martin Kvisvik Larsen, for support with several software related issues.

Contents

Summary i

Sammendrag ii

Preface iii

Acknowledgements iv

Table of Contents viii

List of Tables ix

List of Figures xi

1 Introduction 1
- 1.1 Background . . . 1
- 1.2 Objective . . . 1
- 1.3 Work Description . . . 1
- 1.4 Scope and Delimitation . . . 2
- 1.5 Challenges . . . 3

2 Background and Literature Review 4
- 2.1 ROV . . . 4
- 2.2 Blueye Pioneer Underwater Drone . . . 4
- 2.3 Notation . . . 5
- 2.4 Geometric Relations . . . 6
  - 2.4.1 Reference Frames . . . 6
  - 2.4.2 Body-Fixed Reference Points . . . 7
  - 2.4.3 Rotations and Translations . . . 7
  - 2.4.4 Inertia Matrix . . . 8
- 2.5 Stochastic Processes . . . 8
- 2.6 ROV Modelling . . . 8
  - 2.6.1 Kinematic Equation . . . 9
  - 2.6.2 Kinetics . . . 9

2.6.3 Identification of Model Parameters . . . 9
2.7 Dynamic Positioning . . . 9
2.8 Sensors for DP . . . 10
2.8.1 Inertial Measurement Unit . . . 10
2.8.2 Depth Sensor . . . 10
2.8.3 Acoustic Positioning System . . . 11
2.9 Motion Control System . . . 11
2.9.1 Observer Design . . . 12
2.9.2 Guidance System . . . 14
2.9.3 Control Law . . . 14
2.10 Signal Fault Tolerance . . . 14
2.10.1 Signal Fault Modeling . . . 15
2.11 Fault-tolerant DP Observer . . . 16
2.12 Derivative-Free Optimization . . . 16
2.13 Networking . . . 17

3 Experimental Platform **18**
3.1 Specification of Underwater Drone . . . 18
3.1.1 Geometric Properties . . . 18
3.1.2 Thrusters . . . 18
3.1.3 Internal Sensors . . . 19
3.1.4 Surface Unit . . . 19
3.2 Specifications of APS . . . 19
3.3 Hardware Topology . . . 20
3.3.1 Power Flow . . . 21
3.4 Software Topology . . . 22
3.4.1 Internal Observers . . . 23
3.5 Control Modes . . . 24
3.6 Marine Cybernetics Laboratory . . . 24

4 Sensor Suite for DP **25**
4.1 Drone-Related Sensors . . . 25
4.1.1 Depth Sensor . . . 25
4.1.2 IMU . . . 25
4.2 APS . . . 26
4.3 Location of Sensors . . . 26
4.4 Sensor Update and Uncertainties . . . 26

5 Problem Statement **28**
5.1 Configuration Space and Workspace . . . 28
5.2 Reference Frames . . . 28
5.3 Equations of Motion . . . 29
5.4 State-Space Model . . . 30
5.5 Speed Regimes . . . 30
5.6 Assumptions . . . 31

6 Estimation of Model Parameters **32**

6.1 Rigid-Body 32
6.2 Hydrodynamics 32
6.3 Hydrostatics 34
6.3.1 Thrust Dynamics 34
6.4 Identify Damping by Free-Running Test and DFO 35
6.4.1 Gather Data From Free-Running Test 35
6.4.2 Data Processing 35
6.4.3 Damping Identification Using DFO 37
6.4.4 Implementation 40
6.4.5 Damping Results 41
6.4.6 Discussion of Damping 45
6.4.7 Conclusion 46

7 DP Observer **47**
7.1 Observability 47
7.2 EKF 47
7.2.1 Tuning 47
7.3 Signal Processing Module 48
7.3.1 Fault Detection and Handling 49
7.4 Results From Open-Loop Tests 50
7.4.1 Observer With High Process Noise Covariance 50
7.4.2 Observer With Low Process Noise Covariance 51
7.4.3 Discussion 51

8 Guidance System **53**
8.1 Design 53
8.2 Implementation 54

9 Control System **55**
9.1 Design 55
9.2 Implementation 56

10 Results and Analysis **57**
10.1 Results From Closed-loop 57
10.1.1 Discussion 61

11 Conclusion and Recommendations **63**
11.1 Conclusion 63
11.2 Recommendations 63

Bibliography **65**

A Partial derivative of nonlinear term in state space model **69**

B codes **71**
B.1 MATLAB 71
B.1.1 Added Mass Estimation 71
B.1.2 Damping Identification 71

B.2 Control System 71
B.2.1 Observer 71
B.2.2 Guidance system 71
B.2.3 Controller 72

C Software **73**
C.1 Signal flow APS 73

D Nelder-Mead Simplex Algorithm **74**

E CUSUM **75**
*

List of Tables

2.1 SNAME notation. 5
2.2 SNAME notation for hydrodynamic coefficients. 5
2.3 Failure modes summary. 16

3.1 Geometric properties. 18
3.2 Actuator table 19
3.3 OS, framework, programming language and API used on hardware components on the drone, the APS, and the topside computer (PC) used to run the developed control system. 22

4.1 Sensor location relative to CG. 26
4.2 Sensor update rates, noise standard deviation and bias overview. 27

6.1 Thrust notation. 34
6.2 Lever arms for yaw moment. 34
6.3 Luneberger gains in method 2. 39
6.4 Weighting parameters. 39
6.5 Lower and upper bounds for optimization 40
6.6 Optimization performance 44

List of Figures

1.1 Overview of the report and can be seen as a readers guide to the master thesis. 2

2.1 The underwater drone Blueye Pioneer. Courtesy: Blueye Robotics. 4
2.2 The reference frames ECI (black), ECEF (blue), and NED (green). Courtesy: (Breivik, 2010, p.46). 6
2.3 Body frame. Courtesy: Blueye Robotics. 7
2.4 LBL, SBL and USBL acoustic positioning. Courtesy: Mallios et al. (2009). 11
2.5 Connections in a motion control system. The navigation system includes a signal processing module and an observer, the control system includes a controller and a thrust allocation, while the guidance system contains a reference generator. Courtesy: Figure inspired by (Fossen, 2011, p. 233). 12
2.6 Communication protocol-stack. Courtesy: Turtschi (2002). 17

3.1 Waterlinked's Underwater GPS operating principle. At point 1 is where the underwater vehicle with the Locator is placed, point 2 illustrate the submerged receivers, and point 3 show where the topside hosing (Master-D1) could be placed. Courtesy: Water Linked AS. . 19
3.2 Hardware topology of the drone (marked in blue) and the APS (marked in red). Courtesy to Water Linked for photo of Topside housing. 21
3.3 Power flow for the drone. 22
3.4 Power flow for the APS. 22
3.5 Software topology of the drone and APS. 23

5.1 Local NED (n), body (b) and user (b) frame together. α is the angle between the NED frame and the user frame. Courtesy of Blueye drone: Blueye Robotics. 29

6.1 Qualisys Reference System in the tank in the MC-lab. The blue lines represent the NED frame, while the lines marked in black represent the axes in the Qualisys reference frame. ϕ_Q is the angle from x-axis in the NED frame to the x-axis of the Qualisys frame. 36
6.2 Flow of the DFO for one initialization of $\mathbf{x}$. 41
6.3 Pose from Method 1 using Model 1 in sway-yaw test. X- and y-position is given in the NED frame. The x-axis shows the time in seconds. The simulated responses are illustrated with a solid line, while the measured states are represented by a dotted line. 41

6.4 Velocity and control load for Method 1 using Model 1 in sway-yaw test. The velocity and control load are given in the body frame. The x-axes shows the time in seconds. The control load ($\boldsymbol{\tau}$) is the input to the SVM, generating the velocity and position estimates. The simulated responses are illustrated with a solid line, while the measured states are represented by a dotted line. 42
6.5 Pose for Method 1 using Model 1 in sway test. X- and y-position is given in the NED frame The x-axis shows the time in seconds. The simulated responses are illustrated with a solid line, while the measured states are represented by a dotted line. 42
6.6 Velocity and control load vector for Method 1 using Model 1 in sway test. The velocity and control load are given in the body frame. The x-axes shows the time in seconds. The simulated responses are illustrated with a solid line, while the measured states are represented by a dotted line. 43
6.7 Pose and velocity for Method 2 using Model 1 in sway test. X- and y-position is given in the NED frame, while the velocity is given in the body frame. The x-axes shows the time in seconds. The simulated responses are illustrated with a solid line, while the measured states are represented by a dotted line. 43
6.8 Pose and velocity for Method 2 using Model 2 in sway test. X- and y-position is given in the NED frame, while the velocity is given in the body frame. The x-axes shows the time in seconds. The simulated responses are illustrated with a solid line, while the measured states are represented by a dotted line. 44
6.9 KPI value for each iteration of the optimization for Method 2 using Model 1. 44

7.1 Estimated and measured x position, y position, heading angle and heading rate with estimated linear velocities in open-loop test using EKF with high process noise covariance. . . . 50
7.2 Bias loads in open-loop test using EKF with high process noise covariance. 51
7.3 Estimated and measured x position, y position, heading angle and heading rate with estimated linear velocities using EKF with low process noise covariance. 51

8.1 Input and output to the guidance system. The external observer refers to the EKF developed in Chapter 7. 54

9.1 Implemented control system with all components. The external observer refers to the EKF developed in Chapter 7. 56

10.1 Horizontal trajectory of the drone. 58
10.2 Reference and estimated pose in the closed-loop test. 58
10.3 Reference and estimated velocities in the closed-loop test. 59
10.4 Control load from the closed-loop test. 59
10.5 Thrust setpoints for each thruster from the closed-loop test. 60
10.6 Bias loads from the closed-loop test. 60
10.7 Heading estimates from the rostopics $observer/attitude$ and $observer/state\_estimates$ from the drone in the closed-loop test. 61

C.1 Signal flow APS. Courtesy Water Linked. 73

Abbreviations

API	Application Programming Interface
APS	Acoustic Positioning System
ARL	Average Run Length
BODY	Body-fixed (frame)
CB	Center of Buoyancy
CF	Center of Flotation
CG	Center of Gravity
CO	Coordinate Origin
CUSUM	Cumulative Sum
DFO	Derivative-Free Optimization
DOF	Degree of Freedom
DP	Dynamic Positioning
DVL	Doppler Velocity Log
ECEF	Earth-Centered Earth-Fixed
ECI	Earth-Centered-Inertial
EKF	Extended Kalman Filter
ESC	Electronic Speed Controller
FD	Fault Diagnosis

FIR	Finite Impulse Response
FTC	Fault Tolerant Control
GNC	Guidance, Navigation and Control
GNSS	Global Navigation Satellite System
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
IP	Internet Protocol
KF	Kalman Filter
KPI	Key Performance Indicator
LBL	Long Baseline
MC-lab	Marine Cybernetics Laboratory
MCS	Motion Control System
MEMS	Micro-Electric-Mechanical System
NED	North-East-Down
ODM	Observer Design Model
OS	Operating System
PID	Proportional, Integral, and Derivative (controller)
PMU	Power Management Unit
PWM	Pulse-Width Modulation
ROS	Robotic Operating System
ROV	Remotely Operated Vehicle
SAE	Sum of Absolute Error
SBL	Short Baseline
SDK	Software Development Kit
SU	Surface Unit
SVM	Simulation Verification Model
TCP	Transmission Control Protocol
UART	Universal Asynchronous Receiver/Transmitter

UDP	User Datagram Protocol
USB	Universal Serial Bus
USBL	Ultra-short Baseline
UUV	Unmanned Underwater Vehicles

Nomenclature

$\boldsymbol{\eta}$	Position vector
$\boldsymbol{\nu}$	Velocity vector
$\boldsymbol{\omega}$	Angular velocity
$\boldsymbol{\tau}$	Force vector
$\boldsymbol{\Theta}$	Euler angles
$\boldsymbol{g}(\boldsymbol{\eta})$	Restoring force vector
$\boldsymbol{J}(\eta)$	Transformation matrix
$\boldsymbol{T}(\boldsymbol{\Theta})$	Transformation matrix
$\mathbf{b}$	Bias
$\mathbf{C}$	Centripetal and Coriolis matrix
$\mathbf{D}$	Damping matrix
$\mathbf{I}_b$	Inertia matrix about the an arbitrary origin o_b
$\mathbf{I}_g$	Inertia matrix about Center of gravity
$\mathbf{M}$	System inertia matrix (mass matrix)
$\mathbf{r}_g^b$	Distance from CO to CG
$\mathbf{R}_a^b$	Rotation matrix
$\mathbf{S}$	Cross-product operator
$\mathbf{v}$	Measurement noise
$\mathbf{w}$	Process noise
μ	Expected value
ϕ	Roll angle

ψ	Yaw angle
σ	Variance
θ	Pitch angle
a	Acceleration
I_i	Moment of inertia about axis i
I_{ii}	Products of inertia
K	Roll moment
K_p	Linear damping roll
$K_{\dot{p}}$	Added mass roll
K_{ppp}	Cubic damping roll
M	Pitch moment
m	mass
M_q	Linear damping pitch
$M_{\dot{q}}$	Added mass pitch
M_{qqq}	Cubic damping pitch
N	Yaw moment
N_r	Linear damping yaw
$N_{\dot{r}}$	Added mass yaw
N_{rrr}	Cubic damping yaw
p	Angular velocity roll
q	Angular velocity pitch
r	Angular velocity heave
u	Linear velocity surge
v	Linear velocity sway
w	Linear velocity heave
X	Surge force
x	Position surge
x_g	Center of gravity in x-direction

X_u	Linear damping surge
x_y	Center of gravity in y-direction
x_z	Center of gravity in z-direction
$X_{\dot{u}}$	Added mass surge
X_{uuu}	Cubic damping surge
Y	Sway force
y	Position sway
Y_v	Linear damping sway
$Y_{\dot{v}}$	Added mass sway
Y_{vvv}	Cubic damping sway
Z	Heave force
z	Position heave
Z_w	Linear damping heave
$Z_{\dot{w}}$	Added mass heave
Z_{www}	Cubic damping heave

Chapter 1

Introduction

1.1 Background

For the Remotely Operated Vehicle (ROV) Blueye Pioneer there is an interest to develop a robust and easy to handle dynamic positioning (DP) control system, since a typical user for this low cost ROV does not have the necessary skills to handle unforeseen events and fault. Thus, there is a need for a fault-tolerant control system able to automatically detect and handle typical failure modes on the ROV (hereafter referred to as the drone). The drone is equipped with functionalities that automatically keeps the drone at the desired depth and heading, but has no functionality for controlling surge and sway motion. The drone is mainly used for inspection for which such a DP capability proves useful. The DP system will also provide the location of the drone, which is highly important in search and rescue operations, but also in inspections e.g. when faults are detects. A fault-tolerant DP control system will ensure safer operation and make it easier to control the drone. The DP control system will be developed using the NTNU AUR-Lab owned version of the ROV. The findings from this thesis will be relevant for other ROVs as well, especially small sized ROVs.

1.2 Objective

The main goal of this project is to develop a fault-tolerant DP control system, which includes a fault-tolerant observer, a guidance system, and a controller. This includes studying the available sensor suite of the drone and the acoustic positioning system (APS) that will determine the relevant failure modes. Additionally, this includes making a model of the drone that can be used in the control system.

1.3 Work Description

As a part of making a fault-tolerant control system, several aspects must be established. Firstly, it is important to get familiar with the Blueye Pioneer and the APS to be used and its main functions. This includes making an overview of the relevant hardware and software components used in the control system. In order to make a fault-tolerant observer, the characteristics of the available sensor suite will be investigated. To build an model-based observer, the model parameters must be identified. The observer will be designed and tested both in open-loop and closed-loop. Methods for detecting and handling faults in the observer with respect to the relevant failure modes will be described and implemented. A guidance system for DP functionality will be designed. With the observer and guidance system in place, the controller can be designed and implemented. The control system will be tested outside under real operating conditions. Additionally, a literature background will be made on the subjects of:

- The Blueye Pioneer drone.
- Dynamic modeling of ROVs and identification of model parameters.
- Relevant sensors for ROVs in DP, especially the APS, inertial measurement unit, and pressure sensor.
- Dynamic positioning, including main DP control system components and observer designs. Observer designs for ROVs.
- Relevant failure modes of the DP sensors.
- Fault-tolerant DP observer designs.
- Derivative free optimization for DP.

Figure 1.1 shows where these tasks have been answered in the report. Chapter 2 contains the necessary scientific background for the work and mentions relevant previous work with regards to topics covered. Chapter 3 presents the main characteristics of the drone and the external APS used, including related hardware and software information. Chapter 4 presents the sensor suit to be used in the DP control system, which includes the pressure sensor and IMU in the drone, and the APS as a whole. Chapter 5 presents the problem statement for the DP control system including limitations and assumptions made. Chapter 6 describes how the model parameters have been identified, i.e., the mass, Coriolis-centripetal, and damping parameters. Chapter 7 describes the design and development of the chosen observer, i.e. an extended Kalman filter (EKF). Chapter 8 presents the guidance system developed, where target tracking strategy pure pursuit is used to generate reference velocities. Lastly, Chapter 9 describes how the design and implementation of the PID control law and the control system as a whole.

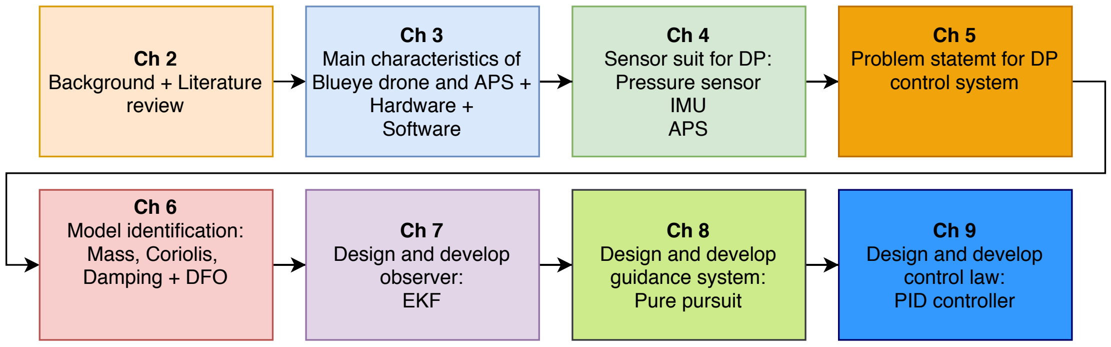
Ch 2
Background + Literature review
Ch 3
Main characteristics of Blueye drone and APS + Hardware + Software
Ch 4
Sensor suit for DP: Pressure sensor IMU APS
Ch 5
Problem statemt for DP control system
Ch 6
Model identification: Mass, Coriolis, Damping + DFO
Ch 7
Design and develop observer: EKF
Ch 8
Design and develop guidance system: Pure pursuit
Ch 9
Design and develop control law: PID controller

Figure 1.1: Overview of the report and can be seen as a readers guide to the master thesis.

1.4 Scope and Delimitation

The drone is mainly used for underwater inspections and therefore usually operated at low speed. Based on this, the dynamics of the drone is modelled using a low-speed hydrodynamic model. An APS is the only external device used in the DP control system in addition to the drone itself. The drone is typically controlled by giving joystick inputs related to the desired force in a given degree of freedom to the drone through an application on a phone or tablet. In this thesis, the input will be given as a desired pose, i.e. x-, y-, z-position and heading angle, that is passed to the drone using a PC. The drone is equipped with an internal guidance system that is based on joystick input. To have the input given by the desired pose instead of joystick input, an external guidance system is developed. The external guidance system will be used instead of the existing internal guidance system. The DP controller developed in this project will be an integrated part of the existing controller for the drone. The existing controller already has DP functionalities for auto-heading and auto-depth on the drone, so the extended control system will be developed for the degrees of freedom in the horizontal plane, i.e. surge, sway, and yaw. The existing thrust allocation on the drone will be used as is.

As the scope of work has proved to be larger than anticipated, the DP observer is implemented to be fault-tolerant with regards to some and not all relevant failure modes. Detection and handling of the other relevant failure modes will be described, but not implemented. If all relevant failure modes were to be handled and detected in the observer, there would not be time to develop the control system, which was considered an essential part of the work assignment.

1.5 Challenges

One of the major challenges in this thesis is to get accurate positioning estimates and velocity estimates with measurements from the APS (2Hz) and an uncertain model representing the drone. Another challenges is getting accurate heading angle estimates in areas where a digital compass can deviate due to magnetic disturbances. The heading is of high importance for a DP control system, since it dominates the motion in the horizontal plane and the transformation between the NED and body frame, and should therefore be reliable and accurate. This is discussed in detail in Chapter 10.

Chapter 2

Background and Literature Review

Relevant parts of the previously conducted project thesis by this author (Bellingmo, 2019) is included in this master thesis, as this is a continuation of the project thesis.

2.1 ROV

Underwater vehicles can be divided in two categories, manned and unmanned (Christ and Wernli, 2014). Further, unmanned underwater vehicles (UUVs) can be divided in to autonomous underwater vehicles (AUV) and remotely operated vehicle (ROV). An ROV differs from an AUV in that it is has an umbilical for communication and/or power supply between the surface and the vehicle (Christ and Wernli, 2014). Depending on the usage, ROVs are usually equipped with camera, light, and manipulators (Ludvigsen, 2019). ROVs can be classified according to their application. For instance an ROV used to gather camera images and other sensor data is categorized as an observation class ROV (Christ and Wernli, 2014).

2.2 Blueye Pioneer Underwater Drone

In this thesis, the ROV Blueye Pioneer will be used, illustrated in Figure 2.1.

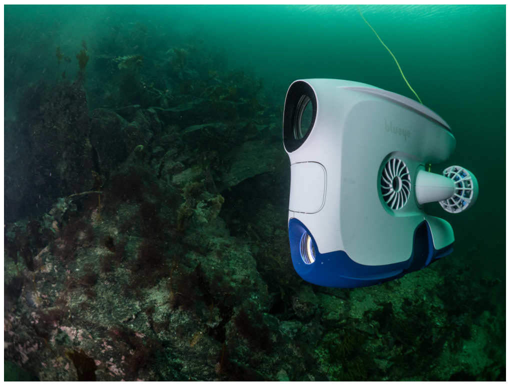

Figure 2.1: The underwater drone Blueye Pioneer. Courtesy: Blueye Robotics.

The drone is an observation class ROV, as the main purpose is to gather camera images. The drone is considered a low cost and small sized drone, used mainly for underwater inspections in sea and fresh water. The drone is used in a various of markets, e.g. shipping, waterworks, and search and rescue. The drone is developed to be easy to use and handle, such that no previous knowledge of the drone is required to get a good user experience. All technical specifications regarding the drone is found at blueye.no[1]. The ROV used in this project thesis is a Blueye Pioneer, developed by Blueye Robotics AS, owned by NTNU AUR-LAB. Several theses are based on the Blueye drone, e.g. Mokleiv (2017), Kvalberg (2019), and Scheide (2016).

2.3 Notation

In this thesis, the notation from SNAME (1950) will be used. These are summarized in Table 2.1 and Table 2.2. Scalars, vectors, and matrices are represented by normal letters, bold lowercase letters, and bold uppercase letters, respectively.

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

Table 2.1: SNAME notation.

DOF	**Added mass**	**Linear damping**	**Quadratic damping**
Surge	$X_{\dot{u}}$	X_u	$X_{\|u\|u}$
Sway	$Y_{\dot{v}}$	Y_v	$Y_{\|v\|v}$
Heave	$Z_{\dot{w}}$	Z_w	$Z_{\|w\|w}$
Roll	$K_{\dot{p}}$	K_p	$K_{\|p\|p}$
Pitch	$M_{\dot{q}}$	M_q	$M_{\|q\|q}$
Yaw	$N_{\dot{r}}$	N_r	$N_{\|r\|r}$

Table 2.2: SNAME notation for hydrodynamic coefficients.

The generalized vectors for position, velocity and force are given by (2.1) for 6 degrees of freedom (DOF) and will be used in this thesis (Fossen, 2011).

$$\boldsymbol{\eta} = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{bmatrix}, \quad \boldsymbol{\nu} = \begin{bmatrix} u \\ v \\ w \\ p \\ q \\ r \end{bmatrix}, \quad \boldsymbol{\tau} = \begin{bmatrix} X \\ Y \\ Z \\ K \\ M \\ N \end{bmatrix} \tag{2.1}$$

[1]`https://support.blueye.no/hc/en-us/articles/360006036673-Technical-specifications?preview%5Btheme_id%5D=210292689&preview_as_role=anonymous#`

2.4 Geometric Relations

The theory in this section is mainly taken from Fossen (2011).

2.4.1 Reference Frames

When describing the motion of a marine craft it is convenient to define some reference frames. Relevant reference frames for this thesis are defined in the following.

ECI: The Earth-Centered Inertial (ECI) frame $\{i\}$ is an inertial frame fixed to the Earth's surface, meaning that it is nonaccelerating. The origin is located at the Earth's center. Newton's law of motion is valid in this frame as it is non-accelerating.

ECEF: The Earth-Centered Earth-Fixed (ECEF) reference frame $\{e\} = (x_e, y_e, z_e)$ rotates with the Earth and has its origin o_e fixed to the center of the Earth. This frame is often used for global navigation.

NED: The North, East, Down (NED) reference frame $\{n\} = (x_n, y_n, z_n)$ where the origin is located at the surface of the Earth. The x-axis points to the true North, y-axis towards East, and the z-axis points downwards normal to the Earth's surface. The coordinates are determined by the two angles longitude and latitude. This frame is used for local navigation where it's commonly assumed to be an inertial frame of the vehicle (Skjetne, 2005). The relation between the ECI, ECEF, and NED frames are illustrated in Figure 2.2. One can see that ECI (black) is fixed, ECEF (blue) rotates with the earth, and NED is situated at the surface of the Earth.

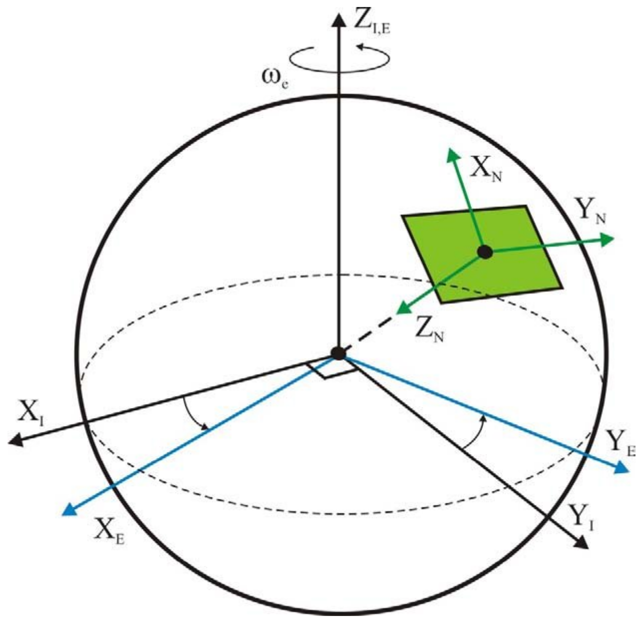

Figure 2.2: The reference frames ECI (black), ECEF (blue), and NED (green). Courtesy: (Breivik, 2010, p.46).

BODY: The BODY reference frame $\{b\} = (x_b, y_b, z_b)$ is a moving frame fixed to the vessel. The axes are aligned with the principal axes of inertia, as seen in Figure 2.3. The origin o_b of the body frame is referred to as the Coordinate Origin (CO).

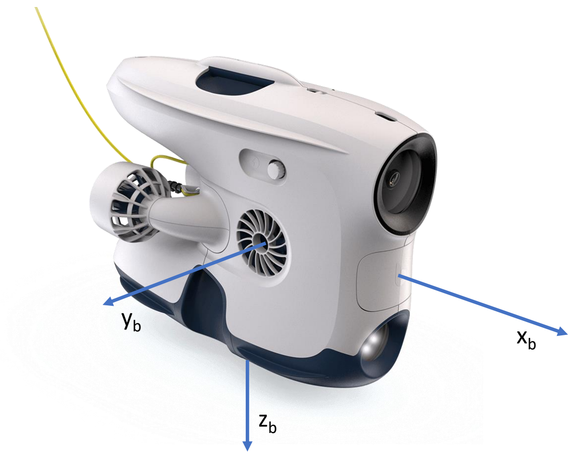

Figure 2.3: Body frame. Courtesy: Blueye Robotics.

USER A user frame is a reference frame that is specified by the user. For this thesis, a relevant user frame can be a Earth-fixed reference frame with the origin at the location of the user, e.g. at shore where the x-axis is straight out from shore and y-axis alongside the shore. A user frame is useful when the user wants to know the position of the drone relative to it self.

2.4.2 Body-Fixed Reference Points

The most important reference point for a marine vessel is the coordinate origin (CO) of the body-fixed frame. The CO is the reference point used in the control system. Other reference points are the center of gravity (CG), the center of buoyancy (CB), and the center of flotation (CF). The distance from CO til CG is given by $\mathbf{r}_g^b = [x_g, y_g, z_g]^\top$. In general, these points can vary with time, except CO, which is why CO is often chosen as the origin of the control system. For instance, for a container ship the load may vary with time, which can change the CG.

2.4.3 Rotations and Translations

To go from one reference frame a to another frame b can be done using a rotation matrix $\mathbf{R}_a^b$ as seen in (2.2).

$$\mathbf{x}^b = \mathbf{R}_a^b \mathbf{x}^a \tag{2.2}$$

For a rotation between the NED and BODY frames, this may be done using the Euler angles roll (ϕ), pitch (θ), and (ψ). For the velocity vector $\boldsymbol{\nu}$ this is

$$\boldsymbol{\nu}^n = \boldsymbol{R}_b^n(\boldsymbol{\Theta})\boldsymbol{\nu}^b \tag{2.3}$$

$\boldsymbol{R}(\boldsymbol{\Theta})$ is the Euler angle rotation matrix and is given as

$$\boldsymbol{R}_b^n(\boldsymbol{\Theta}) = \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi s\theta c\phi \\ s\psi c\theta & c\psi c\phi + s\psi s\theta s\phi & -c\psi s\phi + s\psi s\theta c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{2.4}$$

$c(\cdot)$ and $s(\cdot)$ are the cosine and sine functions. The cross-product operator **S** is defined as

$$\boldsymbol{S}(\boldsymbol{\lambda}) = -\boldsymbol{S}^\top(\boldsymbol{\lambda}) = \begin{bmatrix} 0 & -\lambda & \lambda \\ \lambda & 0 & \lambda_1 \\ -\lambda_2 & \lambda_1 & 0 \end{bmatrix}, \tag{2.5}$$

and equals the cross-product between to vectors λ and a

$$\boldsymbol{\lambda} \times \mathbf{a} = \boldsymbol{S}(\boldsymbol{\lambda})\mathbf{a}. \tag{2.6}$$

The transformation matrix $\boldsymbol{T}(\boldsymbol{\Theta})$ relates the body-fixed angular velocity $\boldsymbol{\omega}^b = [p, q, r]^\top$ with the Euler rate vector $\dot{\boldsymbol{\Theta}} = [\dot{\phi}, \dot{\theta}, \dot{\psi}]^\top$, i.e.,

$$\dot{\boldsymbol{\Theta}} = \boldsymbol{T}(\boldsymbol{\Theta})\boldsymbol{\omega}^b. \tag{2.7}$$

2.4.4 Inertia Matrix

The inertia matrix $\mathbf{I}_g$ about CG is defined as

$$\mathbf{I}_g = \begin{bmatrix} I_x & -I_{xy} & -I_{xz} \\ -I_{yx} & I_y & -I_{yz} \\ -I_{xz} & -I_{yz} & I_z \end{bmatrix}, \tag{2.8}$$

where I_x, I_y, I_z are the moments of inertia about the body-axes and $I_{xy} = I_{yx}$, $I_{xz} = I_{zx}$, and $I_{yz} = I_{yz}$ are the products of inertia. The transformation between I_g to I_b, where I_b is the inertia matrix about the an arbitrary origin o_b, is given by

$$\mathbf{I}_b = \mathbf{I}_g - m\mathbf{S}^2(\mathbf{r}_g^b). \tag{2.9}$$

2.5 Stochastic Processes

Gaussian White Noise White noise is a stationary random process with a constant spectral density (Brown and Hwang, 2012). Assuming that the noise is made of superposition of independent random variables, the Central limit theorem gives that the noise tends towards a normal distribution (Brown and Hwang, 2012). Gaussian distribution is another name for normal distribution. A Gaussian white noise variable has zero mean (Brown and Hwang, 2012). A multidimensional Gaussian white noise variable can be expressed by

$$\mathbf{v} \sim \mathcal{N}(0, \mathbf{P}), \tag{2.10}$$

where **P** is the covariance matrix.

Wiener Process The Wiener process is dened as integrated Gaussian white noise where the initial value is zero (Brown and Hwang, 2012). A wiener process is given by

$$\dot{\mathbf{b}} = \mathbf{w}, \quad \mathbf{w} \sim \mathcal{N}(0, \mathbf{P}), \tag{2.11}$$

where **w** is a zero-mean Gaussian white noise with covariance matrix **P**.

2.6 ROV Modelling

The theory in this section is mainly taken from Fossen (2011).

2.6.1 Kinematic Equation

The kinematic equation for six degrees of freedom (DOF) is given by

$$\dot{\boldsymbol{\eta}}^n = \boldsymbol{J}(\boldsymbol{\eta}^n)\boldsymbol{\nu}^b, \tag{2.12}$$

where the position is in the NED frame and the velocity is expressed in the body frame. The transformation matrix from the inertial frame to the body frame J is

$$\boldsymbol{J}(\eta) = \begin{bmatrix} \boldsymbol{R}(\boldsymbol{\Theta}) & \boldsymbol{0_{3x3}} \\ \boldsymbol{0_{3x3}} & \boldsymbol{T}(\boldsymbol{\Theta}) \end{bmatrix}, \tag{2.13}$$

where R is the Euler angle rotation matrix and T is a transformation matrix.

2.6.2 Kinetics

The equation of motion is given by

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{g}(\eta) = \boldsymbol{\tau} + \boldsymbol{\tau_{env}}. \tag{2.14}$$

The equation of motion is constituted by a rigid-body, hydrostatic and hydrodynamic part such that (2.14) becomes

$$\boldsymbol{M_{RB}}\dot{\boldsymbol{\nu}} + \boldsymbol{C_{RB}}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{M_A}\dot{\boldsymbol{\nu}}_{\boldsymbol{r}} + \boldsymbol{C_A}(\boldsymbol{\nu_r})\boldsymbol{\nu_r} + \boldsymbol{D}(\boldsymbol{\nu_v})\boldsymbol{\nu_v} + \boldsymbol{g}(\eta) = \boldsymbol{\tau} + \boldsymbol{\tau_{env}}, \tag{2.15}$$

where

- $\mathbf{M} = \mathbf{M_{RB}} + \mathbf{M_A}$: system inertia matrix,
- $\boldsymbol{C}(\boldsymbol{\nu}) = \boldsymbol{C_{RB}}(\boldsymbol{\nu}) + \boldsymbol{C_A}(\boldsymbol{\nu_r})$: centripetal and Coriolis matrix,
- $\boldsymbol{D}(\boldsymbol{\nu})$: damping matrix,
- $\boldsymbol{g}(\eta)$: restoring force vector,
- $\boldsymbol{\tau}$: control forces and moments, and
- $\boldsymbol{\tau_{env}}$: environmental forces due to wind, waves and current.

The relative velocity vector is the difference between the velocity vector and the ocean current, i.e., $\boldsymbol{\nu_r} = \boldsymbol{\nu} - \boldsymbol{\nu}_c$. The subscript RB stands for rigid body and A stands for added mass. Added mass is an inertia force that occurs because an accelerating body must move a volume of the surrounding fluid (Fossen, 2011).

2.6.3 Identification of Model Parameters

Identification of the hydrodynamic parameters for a general work class ROV is proposed in Eidsvik (2015), where an empirical method for added mass estimation was made. This is further discussed for the ROV uDrone in Sandøy (2016). Model parameter estimation for the Blueye Pioneer will be discussed in further detail in Chapter 6.

2.7 Dynamic Positioning

According to DNV (2010), dynamic positioning (DP) is defined as a vessel that keeps its position and heading by using the thrusters and propellers on the vessel.

2.8 Sensors for DP

Relevant sensors for an ROV in DP low-speed operations include accelerometer, gyroscopes, gyrocompass, depth sensor and acoustic positioning. An overview of these sensors is given below.

2.8.1 Inertial Measurement Unit

An Inertial Measurement Unit (IMU) often consists of a 3-axis gyroscope, 3-axis accelerometer, and 3-axis magnetometer which is used to measure the acceleration, angular velocity and the magnetic field (Fossen, 2011). Today, due to an development in Micro-Electric-Mechanical System (MEMS) technology, there exist IMUs with high accuracy and low price at a small size (Dukan, 2014). This is suitable for small sized ROVs, like the Blueye drone. There also exist more accurate IMUs using gyroscopes with north-seeking capabilities, e.g. fiber optical gyroscopes and ring laser gyroscopes, which do not need magnetometers (Dukan, 2014). These gyroscope-based IMUs are usually larger and more expensive, and is thus not suitable for the small Blueye drone. Models of the IMU measurements are defined in Mahony et al. (2008). The gyroscope measures the angular velocity in the body frame $\boldsymbol{\omega}_{imu} = [p, q, r]^\top$ and can be modeled as

$$\boldsymbol{\omega}^b_{imu} = \boldsymbol{\omega}^b + \mathbf{b}^b_{gyro} + \mathbf{w}^b_{gyro}, \tag{2.16}$$

where $\boldsymbol{\omega}$ is the true value, **b** is the bias, and **w** is the measurement noise. The accelerometer measures the linear acceleration along the three axis $\mathbf{a} = [a_x, a_y, a_z]^\top$. Expressed in the body frame, the measurements are modeled adding a bias **b** and a noise **w**.

$$\mathbf{a}^b_{imu} = \dot{\boldsymbol{\nu}}^b - \boldsymbol{R}^b_n(\boldsymbol{\Theta})\mathbf{g}^n + \mathbf{b}^b_{acc} + \mathbf{w}^b_{acc}. \tag{2.17}$$

The accelerometer measures the acceleration minus the gravitational acceleration **g**, which are expressed in the inertial frame, and must be transformed to the body frame. The magnetometer measures the magnetic field **m** and is modeled with a bias and measurement noise, similarly to the others. Expressed in the body frame, we get

$$\mathbf{m}^b_{imu} = \boldsymbol{R}^b_n(\boldsymbol{\Theta})\mathbf{m}^n + \mathbf{b}^b_{mag} + \mathbf{w}^b_{mag}, \tag{2.18}$$

where $\mathbf{m}^n$ is the true magnetic field expressed in NED. The bias represents the local magnetic disturbance on the IMU. The magnetic force is used to compute the heading. This is done inside the IMU, and the heading measurement is an output from the IMU. Another way to find the heading, is by using a gyrocompass. A gyrocompass is a non-magnetic compass that finds the heading using the rotation of the Earth and a fast spinning wheel (Bai and Bai, 2014).

2.8.2 Depth Sensor

The depth sensor is in reality a pressure sensor. The depth can be computed using (2.19).

$$z = \int_{p_{atm}}^{p_{uuv}} \frac{1}{\rho(S,T,p)g(L,p)}, \tag{2.19}$$

where the depth is a function of salinity *S*, temperature *T*, pressure *p*, latitude *L*, density ρ, and gravity g, p_{atm} is the atmospheric pressure and p_{uuv} is the pressure at the UUV (Ludvigsen, 2019)..

2.8.3 Acoustic Positioning System

For underwater navigation, acoustic positioning system (APS) is commonly used since the Global Navigation Satellite Systems (GNSS's) does not work under water. An APS computes the range from a transponder attached to the underwater vehicle to the transducer placed at a known location (Christ and Wernli, 2014). A transponder receives acoustic signals and automatically transmits a replay signal (Christ and Wernli, 2014). A transducer transforms a pressure wave into electrical current, and vice versa, meaning that it acts as both a receiver and transmitter (Christ and Wernli, 2014). The distance is calculated by measuring the time of flight for the acoustic signal to travel from the transponders to the target vessel and return, using the speed of sound in water. Additionally, the phase of the wave is measured to get the direction, and thus the position of the target can be computed (Ludvigsen and Sørensen, 2016). The speed of sound depends on temperature, pressure, and salinity (Milne, 1983).

There are three main types of APS's; Long-baseline (LBL), short-baseline (SBL), and ultra-short-baseline (USBL) systems, as illustrated in Figure 2.4. LBL provides wide area positioning by having three of more transponders placed at known locations on the seabed that communicate with a transducer attach to an underwater vehicle (Bai and Bai, 2014). USBL and SBL on the other hand, have the transducer placed in the surface, typically on a ship or platform, while a transponder is attached to the underwater vehicle. An SBL have several transducers with relatively small spacing (baseline) mounted on a ship or structure near the surface, while a USBL consists of one transceiver that is typically mounted on a pole beneath a ship (Bai and Bai, 2014). USBL systems offer a fixed accuracy, while the accuracy of the SBL improves with transducer spacing (Bai and Bai, 2014). Additionally, SBL systems works well in shallow water and reflective environments compared to USBL[2].

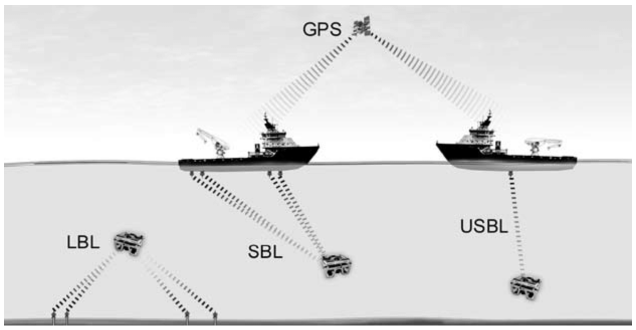

Figure 2.4: LBL, SBL and USBL acoustic positioning. Courtesy: Mallios et al. (2009).

2.9 Motion Control System

A motion control system (MCS), also referred to as a control system, usually consists of the three blocks guidance, navigation and control systems (Fossen, 2011). These blocks can be loosely or tightly coupled depending on the application. The guidance system computes the reference position, velocity and acceleration of the vehicle (Fossen, 2011). The navigation system determines the position and possibly the velocity and acceleration of the vehicle using sensor measurements, and typically consists of a signal processing module and an observer (Fossen, 2011). The control system determines the necessary control load produced by the vehicle in order to satisfy the control objective (Fossen, 2011). This block typically consists of the controller

[2]`https://waterlinked.github.io/docs/explorer-kit/introduction/`

which computes the needed control load and a thrust allocation for distributing the load to the individual actuators. A control objective for a DP control system can for instance be to maintain the reference position. A closed-loop guidance system is illustrated in Figure 2.5. Each block of the motion control system is explained in further detailed in the following subsection.

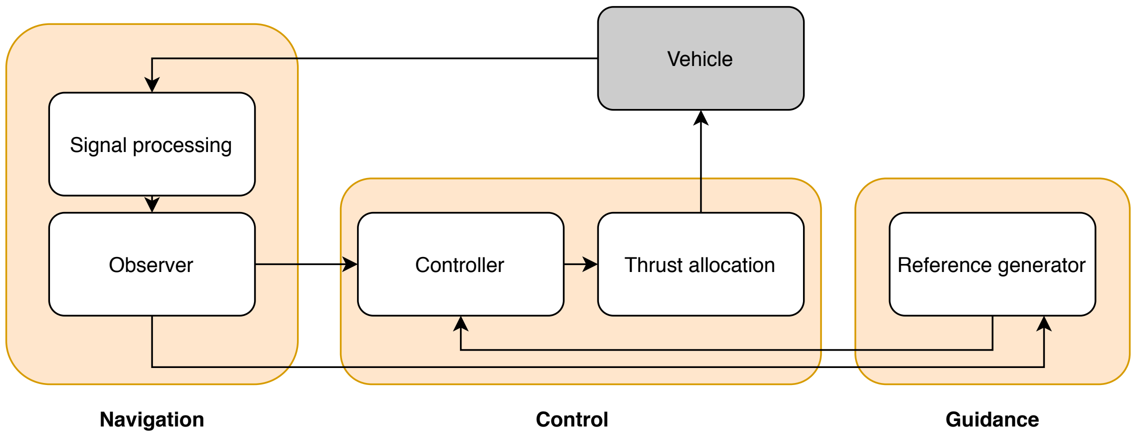

Figure 2.5: Connections in a motion control system. The navigation system includes a signal processing module and an observer, the control system includes a controller and a thrust allocation, while the guidance system contains a reference generator. Courtesy: Figure inspired by (Fossen, 2011, p. 233).

A DP system developed and tested for the ROV Minerva is described in Dukan et al. (2011), which showed good performance of an extended Kalman filter for the nonlinear dynamics of the drone. Breivik (2010) describes MCS for marine vehicles, including guidance systems for target tracking. MCS for ROVs is described and implemented in Dukan (2014) and Fernandes et al. (2015), where the former has some interesting notes on how to make a guidance system with joystick in the closed-loop control, while the latter have results that encourage the use of a high-gain state observer instead of the standard extended Kalman filter.

2.9.1 Observer Design

The tasks of an observer, also named state estimator, is to filter out the noise from the measurement, estimate states from the measurements, predict states in case of sensor faults, and estimate disturbances (Candeloro et al., 2012). Observer design can be divided in two types: sensor-based and model-based.

Sensor-Based Observer In a sensor-based observer the strapdown inertial navigation system (INS) uses only the inertial sensor measurements, which typically includes an IMU (Dukan, 2014). This considers only the kinematics, so no estimation of model parameters are necessary. The inertial sensors are usually sensitive to alignment and calibration errors. Integrating IMU readings to get position, velocity and attitude will drift over time. To limit the errors, INS is usually fused with a positioning system, e.g. GNSS in air or APS under water. This fusion can be done using an error-state Kalman filter (ESKF), as described in Sola (2017). A sensor-based hybrid observer using asynchronous measurements for DP is described in Brodtkorb et al. (2015). Other work on INS and positioning systems are proposed in Vik and Fossen (2001) and Mahony et al. (2008). The work of Dukan (2014) contains both model-based Kalman filters and sensor-based observers.

Model-Based Observer A model-based observer uses a model of the vessel to estimate the states and are good for filtering noise and estimation without causing phase lag (Dukan, 2014). An overview of some model-based observers for DP of ROVs is given in Candeloro et al. (2012), where the linear Kalman filter (LKF), adaptive Kalman filter, extended Kalman filter (EKF), and the passive nonlinear observer are tested. Their results showed that the EKF performed best. A comparison between a linear time-varying Kalman filter, an EKF, and an uncented Kalman filter is discussed in Værnø et al. (2019), where all observers had similar performance. EKF in DP is also discussed in Tannuri and Morishita (2006) and Hassani et al. (2013). A nonlinear passive observer for ships is described in Fossen and Strand (1999) and using time-varying gains in Værnø et al. (2017).

The observer must also be able to handle sensor fusion with sensors which have asynchronous measurement updates. Methods for motion estimation using sensor fusion of GNSS and IMU based on a multirate Kalman filter is discussed in Ren et al. (2019). Integration filter for APS, DVL, IMU and Pressure Gauge for underwater vehicles is proposed in Dukan and Sørensen (2013). Zhao et al. (2012) considers the issue of handling asynchronous measurements from DVL and APS sensors. In this project thesis, EKF is implemented and is therefore explained in more detail.

Extended Kalman Filter

A Kalman filter is a recursive filter that estimates the state of a dynamic system using noisy measurements (Fossen, 2011). Using a Kalman filter, it is possible to estimate unmeasured states and filter out white noise (Fossen, 2011). The objective of the Kalman filter is to minimize the estimation error (Ghadrdan et al., 2012). A Kalman filter assumes that the system is observable and linear, and that the process and measurement noise are independent and Gaussian distributed (Ghadrdan et al., 2012). In dead reckoning, when measurements are lost in a period of time, the Kalman filter is useful as it can predict the states. For nonlinear system dynamics, the extended Kalman filter (EKF) can be used. A system with a nonlinear process model can be written on the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{B}\mathbf{u} + \mathbf{E}\mathbf{w}, \quad \mathbf{w} \sim \mathcal{N}(0, \mathbf{Q}) \tag{2.20a}$$

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{v}, \quad \mathbf{v} \sim \mathcal{N}(0, \mathbf{R}), \tag{2.20b}$$

where $\mathbf{x}$ is the state, $\mathbf{y}$ is the measurement, $\mathbf{u}$ is the control input $\mathbf{v}$ and $\mathbf{w}$ are measurement and process noise, $\mathbf{f}(\mathbf{x})$ is a nonlinear function, and $\mathbf{B}$, $\mathbf{E}$, and $\mathbf{H}$ are model parameters. The corresponding nonlinear discrete-time process model can be written as

$$\mathbf{x}(k+1) = \mathcal{F}(\hat{\mathbf{x}}(k), \mathbf{u}(k)) + \mathbf{\Gamma}\mathbf{w}(k), \quad \mathbf{w}(k) \sim \mathcal{N}(0, \mathbf{Q}_d), \tag{2.21a}$$

$$\mathbf{y}(k) = \mathbf{H}\mathbf{x}(k) + \mathbf{v}(k), \quad \mathbf{v}(k) \sim \mathcal{N}(0, \mathbf{R}_d), \tag{2.21b}$$

The discrete EKF algorithm is found in Fossen (2011) and is given in (2.22).

$$\text{Design matrices} \quad \mathbf{Q}_d(k) = \mathbf{Q}_d^\top(k) > 0 \text{ , } \mathbf{R}_d(k) = \mathbf{R}_d^\top(k) > 0 \tag{2.22a}$$

$$\text{Initial conditions} \quad \bar{\mathbf{x}}(0) = \mathbf{x}_0 \text{ , } \bar{\mathbf{P}}(0) = \mathbf{P}_0 \tag{2.22b}$$

$$\text{Kalman gain} \quad \mathbf{K}(k) = \bar{\mathbf{P}}(k)\mathbf{H}^\top(k)[\mathbf{H}(k)\bar{\mathbf{P}}(k)\mathbf{H}^\top(k) + \mathbf{R}_d(k)]^{-1} \tag{2.22c}$$

$$\begin{aligned} \text{Predictor} \quad & \bar{\mathbf{x}}(k+1) = \mathcal{F}(\hat{\mathbf{x}}(k), \mathbf{u}(k)) \\ & \bar{\mathbf{P}}(k+1) = \mathbf{\Phi}(k)\hat{\mathbf{P}}(k)\mathbf{\Phi}^\top(k) + \mathbf{\Gamma}(k)\mathbf{Q}_d(k)\mathbf{\Gamma}^\top(k) \end{aligned} \tag{2.22d}$$

$$\begin{aligned} \text{Corrector} \quad & \hat{\mathbf{x}}(k) = \bar{\mathbf{x}}(k) + \mathbf{K}(\mathbf{y}(k) - \mathbf{H}(k)\bar{\mathbf{x}}(k)) \\ & \hat{\mathbf{P}}(k) = [\mathbf{I} - \mathbf{K}(k)\mathbf{H}(k)]\bar{\mathbf{P}}(k)[\mathbf{I} - \mathbf{K}(k)\mathbf{H}(k)]^\top + \\ & \qquad \mathbf{K}(k)\mathbf{R}_d(k)\mathbf{K}^\top(k) \text{ , } \hat{\mathbf{P}}(k) = \hat{\mathbf{P}}^\top(k) > 0 \end{aligned} \tag{2.22e}$$

where $\mathbf{Q}_d$ is the discrete-time process noise covariance matrix, $\mathbf{R}_d$ is the discrete-time measurement noise covariance matrix, $\mathbf{P}$ is the error covariance matrix, and $\mathbf{K}$ is the Kalman gain. In an EKF, the system matrices are linearized about the estimated states $\hat{\mathbf{x}}$ (Lefebvre et al., 2004). The discrete time matrices $\mathcal{F}(\hat{\mathbf{x}}(k), \mathbf{u}(k))$, $\mathbf{\Phi}(k)$, and $\mathbf{\Gamma}(k)$ are found by forward Euler integration (Fossen, 2011) are given by

$$\mathcal{F}(\hat{\mathbf{x}}(k), \mathbf{u}(k)) = \hat{\mathbf{x}}(k) + h[\mathbf{f}(\hat{\mathbf{x}}) + \mathbf{B}\mathbf{u}(k)] \tag{2.23a}$$

$$\mathbf{\Phi}(k) = \mathbf{I} + h\frac{\partial \mathbf{f}(\mathbf{x}(k))}{\partial \mathbf{x}(k)}\bigg|_{\mathbf{x}=\hat{\mathbf{x}}} \tag{2.23b}$$

$$\mathbf{\Gamma}(k) = h\mathbf{E}, \tag{2.23c}$$

where *h* is the sampling time. Stability of the EKF is not guaranteed since its inherent approximation can make the filter diverge with unbounded estimation errors (Bar-Shalom et al., 2002).

2.9.2 Guidance System

A guidance system can be designed in different manners, depending on the control objective. Setpoint regulation is a type of motion control system where the desired pose is constant (Fossen, 2011). Another examples is trajectory tracking, or simply target tracking, where the goal is to track a desired output $\mathbf{y}_d(t)$ (Fossen, 2011). Pure pursuit guidance is a type of target tracking where the controlled vehicle aligns its velocity vector along the line of sight vector between the controlled vehicle and the target (Breivik, 2010). This gives the desired velocity

$$\mathbf{v}_d = -U_{max}\frac{\tilde{\mathbf{p}}}{\sqrt{\tilde{\mathbf{p}}^\top\tilde{\mathbf{p}} + \Delta_p^2}}, \tag{2.24}$$

where $\tilde{\mathbf{p}} = \mathbf{p} - \mathbf{p}_t$ is the position error, U_{max} is the maximum approach speed, and Δ_p is a tuning parameter. Here $\mathbf{p}$ denotes the position of the controlled vehicle, while $\mathbf{p}_t$ is the position of the target.

2.9.3 Control Law

A simple and widely used control law is the proportional, integral and derivative (PID) controller, given by

$$\boldsymbol{\tau} = -\mathbf{K}_p\mathbf{e} - \mathbf{K}_d\dot{\mathbf{e}} - \mathbf{K}_i\int_0^{t'} \mathbf{e}(t)dt, \tag{2.25}$$

where $\mathbf{e} = \hat{\boldsymbol{x}} - \boldsymbol{x}_d$ is the estimation error for the state vector $\mathbf{x}$, and $\mathbf{K}_p$, $\mathbf{K}_d$, and $\mathbf{K}_i$ are the controller gains. The subscript *d* on the state vectors represent the desired states. The controller gains can be found by using the control design model and its bandwidth as explained in Fossen (2011, chapter 12.2), or tuning by trial and error. An example of more advanced control law is an adaptive controller, e.g. based on a contraction theory as described in Lakshmanan et al. (2020).

2.10 Signal Fault Tolerance

Fundamental topics on signal fault tolerance is covered in Blanke (2016), while change detection and practical examples are discussed in Gustafsson and Gustafsson (2000). Blanke (2016) defines a *fault* as "a deviation of the system structure or the system parameters from the nominal situation". This can be "fixed" using fault-tolerant control. A fault-tolerant control system is aimed to "prevent a component fault from causing a failure at the system level" (Blanke, 2016). The same book describes a *failure* as "the inability of a system or component to accomplish its function". This on the other hand, can not be easily fixed and the failed component has to be shut off.

2.10.1 Signal Fault Modeling

Signal failure modes are the physical effect of a signal fault. Common failure modes are; signal dropout, bias, drift, frozen signal, outliers, and high noise. The theory in this section is based on Mokleiv (2017), otherwise specified. A sensor measurement can be modeled as the real signal and a measurement noise (Gustafsson and Gustafsson, 2000), i.e.,

$$y_{nominal} = y_{real} + v, \quad v \sim \mathcal{N}(0, \sigma^2),$$

where *y* is a scalar signal and σ^2 is the variance. The noise is modeled as a Gaussian white noise, which is often a good assumption, since the noise can be considered a sum of small contributions (see Section 2.5). This model is defined as the nominal fault-free measurement and can be expressed as

$$y_{nominal} \sim \mathcal{N}(y_{real}, \sigma^2).$$

Frozen Signal A frozen signal is characterized by several consecutive measurements which are equal. This leads to zero variance and can be modeled as

$$y \sim \mathcal{N}(y_{k-1}, 0) \tag{2.26}$$

Signal Dropout The behavior of a dropout depends on the sensors and the software used. For the Blueye Pioneer all sensor measurements are run through the Robotic Operating system (ROS) software which only publishes new measurements, i.e., when the measurement has changed. This means that a signal dropout would behave like a frozen signal and can be handled in the same manner, i.e, not differentiating between a frozen signal or dropout.

Bias A bias is a constant offset that changes the nominal measurement. The bias does not affect the variance in the model, but adds on to the mean

$$y \sim \mathcal{N}(y_{nominal} + y_{bias}, \sigma_0^2) \tag{2.27}$$

Signal Drift A signal drift is equal to a signal bias, only that the offset varies with time. This can be modeled as

$$y \sim \mathcal{N}(y_{nominal} + y_{drift}(t), \sigma_0^2) \tag{2.28}$$

Outliers An outlier is categorised as a measurement that is outside a certain band about the estimated mean (Sørensen, 2013). This can be seen as a measurement with a large z-score, i.e., distance from the mean measured in standard deviation, where the z-score is typically between 3-9 (Sørensen, 2013). If the signal is outside the range defined in (2.29), it is assumed to be an outlier. *a* is the z-score.

$$y \in [y_{nominal} - a\sigma, y_{nominal} + a\sigma] \tag{2.29}$$

Since the drone is going to operate at low speed (due to DP), an outlier can be seen as a large difference between two measurement, as this can not be due to a large velocity (Zhao et al., 2012).

High Noise A failure mode is categorized as high noise when there is an increase in the noise level over a period of time. This mode manifests itself in a similar way as an outlier, only that the variance is increased for several measurements, not just a few. High noise can be modeled as

$$y \sim \mathcal{N}(y_{real}, \sigma^2_{high}) \tag{2.30}$$

Summary

The modeling of all the failure modes are summarized in Table 2.3.

Failure mode	**Model**	**Specification**
Frozen signal	y $\sim \mathcal{N}(y_{k-1}, 0)$	
Signal dropout	y $\sim \mathcal{N}(y_{k-1}, 0)$	
Bias	y $\sim \mathcal{N}(y_{nominal} + y_{bias}, \sigma_0^2)$	
Signal drift	y $\sim \mathcal{N}(y_{nominal} + y_{drift}(t), \sigma_0^2)$	
Outlier	y $\notin [y_{nominal} - a\sigma, y_{nominal} + a\sigma]$	a $\in [3, 9]$
High noise	y $\sim \mathcal{N}(y_{real}, \sigma^2_{high})$	

Table 2.3: Failure modes summary.

2.11 Fault-tolerant DP Observer

A fault-tolerant observer must be able to detect and handle all the relevant failure modes. Mokleiv (2017) made a fault-tolerant observer using EKF for a previous model on the Blueye drone, which is of high relevance for this thesis. The work of Abrahamsen (2019) covers fault-tolerant DP for a vessel utilizing machine learning approaches. A fault-tolerant navigation system for underwater robots using particle filter is discussed in Zhao et al. (2012) and Zhao et al. (2014). More on fault-detection and handling is described in Section 7.3.1.

2.12 Derivative-Free Optimization

Derivative-free optimization (DFO) is the "mathematical study of optimization algorithms that do not use derivatives" (Audet, 2017). The inclusion of "mathematical studies" in the definition limits DFO algorithms to have methods that can be mathematically analyzed to prove convergence or other stopping criteria (Audet, 2017). The DFO solely utilizes the objective function, $f(\mathbf{x})$, to find the optimal solution. For a minimization problem, the objective can be formulated as

$$\min_{\mathbf{x}} f(\mathbf{x}). \tag{2.31}$$

DFO is generally used when the derivative is impossible or hard to obtain. When the gradient is available, DFO will be outperformed by most modern gradient-based optimization algorithms (Audet, 2017). A popular DFO algorithm is the Nelder-Mead simplex algorithm (Nelder-Mead for short), developed by John Nelder and Roger Mead in 1965, which is a search method for unconstrained minimization (Nelder and

Mead, 1965). Particle swarm optimization is another example of a DFO algorithm. The use of DFO to find unknown parameters for DP is discussed in several papers. Relevant master theses are Løvås (2019) and Alfsen (2019), who use DFO to find control parameters for DP, where the latter also use DFO to find observer gains. In Værnø et al. (2019), DFO is used to find damping parameters for the system model and observer gains.

2.13 Networking

A network can be defined as two or more processes communicating, where the processes can be on the same or different devices (Turtschi, 2002). The Internet Protocol (IP) was developed to have a standardized way to communicate over different networks (Turtschi, 2002). Sockets are used to access IP-based networks from an application (Turtschi, 2002). Network sockets are "endpoints of communication used for connecting to other computers, sending, and receiving data from them." (Turtschi, 2002). Network communication can be divided in four protocol layers; Application layer, Transport layer, Network layer, and physical layer (Turtschi, 2002), illustrated in Figure 2.6.

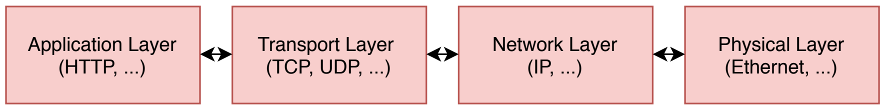

Figure 2.6: Communication protocol-stack. Courtesy: Turtschi (2002).

Two common protocols used in the Transport layer are the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). TCP is a connection- and stream-oriented protocol, for which a communication link must be connected at all times (Turtschi, 2002). TCP is reliable, as it ensures that no packets are lost. Most protocols that need reliability use TCP as their base, e.g. the Hypertext Transfer Protocol (HTTP). UDP is a connection-less protocol, meaning that it sends packets without checking if the receiver is ready, which may lead to loss of packets (Turtschi, 2002). Using UDP, there is no guarantee that the packets sent will be received in order. The benefit of UDP is that it is faster than TCP, since it doesn't have to resend packets or give them in order (Turtschi, 2002).

Chapter 3

Experimental Platform

This chapter gives an overview of specifications of the drone and APS used, including hardware and software related information.

3.1 Specification of Underwater Drone

The underwater drone used in this project is the Blueye Pioneer from Blueye Robotics. The specifications are found at Blueye.com (Blueye). The drone is shown in Figure 2.3.

3.1.1 Geometric Properties

Table 3.1 shows the most relevant geometric properties for the underwater drone. $A_{proj,top}$, $A_{proj,side}$, and $A_{proj,front}$ are the projected areas top, side, and front respectively. I_z is the moment of inertia in yaw. The mass m is weight in air with salt water ballast.

Parameter	**Value**	**Unit**
L	485	mm
W	257	mm
H	354	mm
m	8.6	kg
$A_{proj,top}$	79561.5	mm^2
$A_{proj,side}$	140366	mm^2
$A_{proj,front}$	56387	mm^2
I_z	92502.96	$kgmm^2$

Table 3.1: Geometric properties.

3.1.2 Thrusters

The drone has four thrusters; two longitudinal, one lateral, and one vertical thruster. The power of each thruster is 350W in free water. When placed on the drone, the efficiency is lower, depending on the location. Table 3.2 gives the position of each thruster relative to CG in the body frame, the main thrust direction, and an approximate efficiency coefficient for each thruster.

Thruster	**X** [m]	**Y** [m]	**Z** [m]	**Main thrust directions**	**Efficiency**
Longitudinal starboard	0.1491	0.08082	-0.02628	Surge and yaw	0.9
Longitudinal port	0.1491	-0.08082	-0.02628	Surge and yaw	0.9
Lateral	0.06141	-0.01502	-0.02628	Sway	0.8
Vertical	-0.02359	-0.00082	-0.06848	Heave	0.6

Table 3.2: Actuator table

3.1.3 Internal Sensors

The drone is equipped with an IMU, pressure sensor, temperature sensor, and a camera. Since the IMU and the pressure sensors are the most relevant sensors for the control system to be developed, they will be described in more detail in Chapter 4.

3.1.4 Surface Unit

The drone is connected to the Surface Unit by an Ethernet cable. The Surface Unit has a wifi router for wireless connection to smartphones or tablets. Additionally, it has a USB port for attachment of Ethernet adapter for wired connection in difficult wireless environment.

3.2 Specifications of APS

For this project, the positioning system from Water Linked named "Underwater GPS" is used as the APS. The documentation for the Underwater GPS is found at waterlinked.com (Water Linked). The Underwater GPS is based on SBL acoustic positioning. The system consists of four hydroacoustic receivers submerged near the surface together with a digital acoustic position computing board named Master-D1 and a hydroacoustic locator named Locator U1 strapped to the drone. The Locator U1 (also referred to as simply the Locator) consists of a transducer, a depth sensor, and an internal GPS based time sync module. The transducer in the Locator is only used as a transmitter, i.e. transmitting acoustic signals. The Master-D1 is usually placed in a topside housing. The operating principle is shown in Figure 3.1.

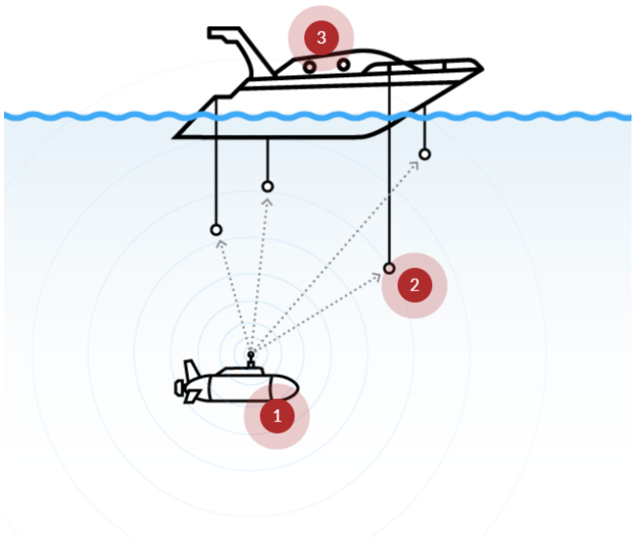

Figure 3.1: Waterlinked's Underwater GPS operating principle. At point 1 is where the underwater vehicle with the Locator is placed, point 2 illustrate the submerged receivers, and point 3 show where the topside hosing (Master-D1) could be placed. Courtesy: Water Linked AS.

The position of the Locator is calculated using the time-of-flight between the receiver and the transmitter. This position is computed in the acoustic reference system where the origin is at the location of

the Master-D1 and x- and y-axes are defined by the
orientation of the topside housing[1]. The Master-D1 is equipped with an IMU and GPS. The IMU is used to find the orientation of the topside housing, and together with the GPS and acoustic positioning data, the global position of the Locator is computed. The global position is given in the NED frame with units latitude and longitude. The position data sent to the user (by e.g. the API) is filtered in a Kalman filter installed in the Master-D1. Communication with the Underwater GPS can easily be done through Water Linked's API[2]. Through the API the user will be notified if there is a loss of signal from the Locator or other issues.

3.3 Hardware Topology

Drone Hardware A microcontroller KL82 is mounted on the power management unit (PMU) to power the thrusters and the other sensors. The KL82 handles the communication with the thrusters and battery and communicates the control signals with a 4-in-1 electronic speed controller (ESC) by the digital protocol DShot. The ESC controls the control signal to the four thrusters. A single board computer named Imx6 runs the Blueye-made operating system (OS) named Blunux. Blunux is a configuration of a Yocto project[3], which is a project to develop Linux distributions. The Imx6 and KL82 are connected by Universal Asynchronous Receiver/Transmitter (UART). The pressure sensor and IMU are connected with I2C to the Imx6. The Imx6 is used to communicate from the drone through an Ethernet cable to the Surface Unit placed above sea level. Topside computers and tablets are connected to the network provided by the Surface Unit and are able to subscribe and publish messages to the drone using the Robotic Operating System (ROS).

APS Hardware The Master-D1 communicates with a PC by an Ethernet cable. The Master-D1 is connected with the receivers using a Water Linked proprietary signal. The receiver communicate with the Locator attached to the drone. The Topside housing (Master-D1) is equipped with a GPS and an IMU.

A hardware topology of the Blueye drone and the APS is illustrated in Figure 3.2. The PC in the figure illustrate the topside computer that will be used to communicate with the drone and the APS. The PC will run the control system to be developed. The drone is equipped with two IMUs, but for simplicity, only one IMU is shown in the figure.

[1] `https://waterlinked.github.io/docs/explorer-kit/gui/receivers/`
[2] `https://waterlinked.github.io/docs/explorer-kit/gui/api/`
[3] `https://www.yoctoproject.org/`

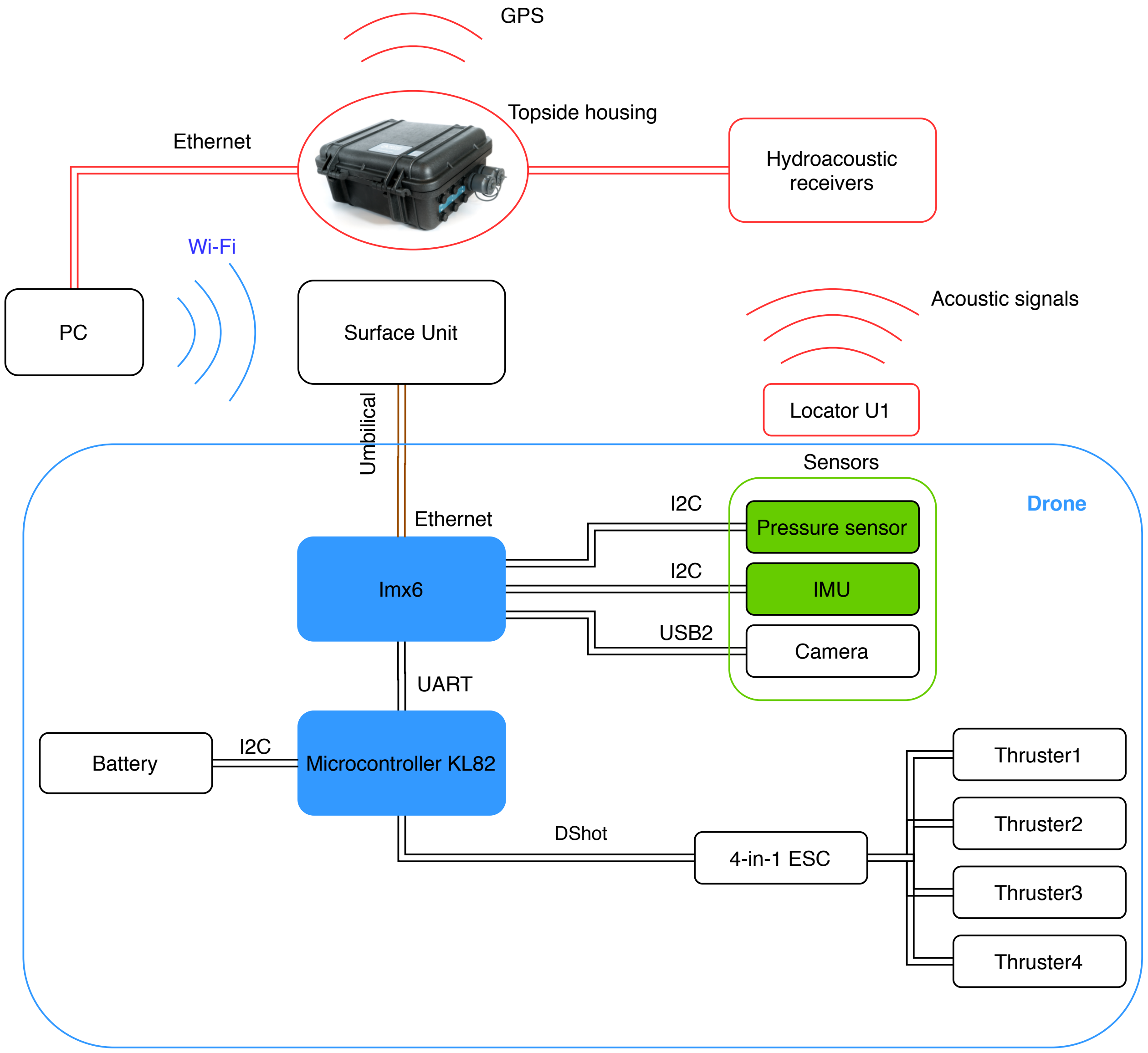

Figure 3.2: Hardware topology of the drone (marked in blue) and the APS (marked in red). Courtesy to Water Linked for photo of Topside housing.

3.3.1 Power Flow

The drone is powered from an in-house battery with nominal capacity of 6500 mAh and nominal voltage of 14.8V. Under normal using conditions, the battery last for approximately two hours. The microcontroller KL82 in the drone is placed on a Power Management Unit (PMU) board which distributes the power to each component in the drone. The computer Imx6, pressure sensor, and IMU are powered with 3.3V, the camera is powered with 5V, while the four thrusters are powered with 14.8V. The Master-D1 of the APS is powered by an external battery at 14.8V. The Master-D1 has a PMU which distributes 12V to each receiver. The transmitter Locator U1 is equipped with an internal battery which can power the transmitter for approximately 10 hours. The power flow for the drone and the APS are shown in Figure 3.3 and Figure 3.4 respectively.

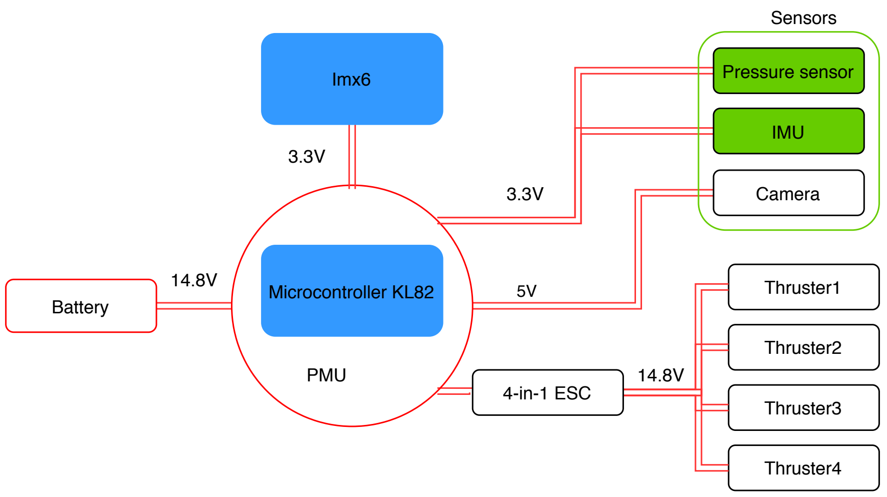

Figure 3.3: Power flow for the drone.

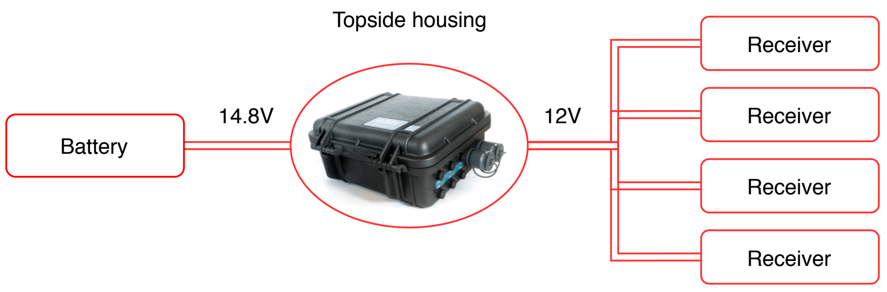

Figure 3.4: Power flow for the APS.

3.4 Software Topology

The OS, framework, programming language, and API for the topside computer, the single-board computer Imx6 inside the drone, and the Master-D1 onboard the APS are presented in Table 3.3. The topside computer refers to the computer used to run the control system components developed in this thesis.

	Topside computer	**Imx6 on drone**	**Master-D1 on APS**
OS	Ubuntu 16.04	Blunux	Linux
Framework	ROS Kinetic	ROS Melodic	-
Programming language	Python 2	C++/Python 2	C/C++/Python
API	-	Blueye SDK	Swagger HTTP

Table 3.3: OS, framework, programming language and API used on hardware components on the drone, the APS, and the topside computer (PC) used to run the developed control system.

The software on the drone is written in ROS which contains tools and libraries to simplify the task of manage the behavior of a robot.[4] The communication interface in ROS provides the possibility publishing and

[4]`https://www.ros.org/about-ros/`

subscribing messages among others. The Blueye drone has a Software Development Kit (SDK) that allows for control of some functions on the drone, without accessing all the code on the drone. More information regarding the SDK is found at blueye-robotics.github.io[5]. Water Linked's APS is equipped with a software API of the type Swagger based on HTTP[6]. The API allows you to easily read acoustic and global position data. The software topology is illustrated in Figure 3.5 where it is divided in the tree subsystems; Topside computer, Drone, and APS. The relevant modules of each subsystem for DP are shown. The observer in the APS receives raw measurements from the IMU and GPS located on the Master-D1 board, in addition to the measurements from the receivers. The internal observer in the APS filters the raw signals and detect and handle typical failure modes in the signals. The x and y position from the observer are sent to the control system to be developed in the computer used in this thesis, i.e. the Topside computer. The drone is equipped with an internal observer, a controller, and a thrust allocation. The drone also contains other software related functions, but only the ones with the highest relevance for the DP control system are included. The drone will provide the heading, heading rate and control force to the control system to be developed. Details regarding the signal flow in the APS is given in Appendix C.1.

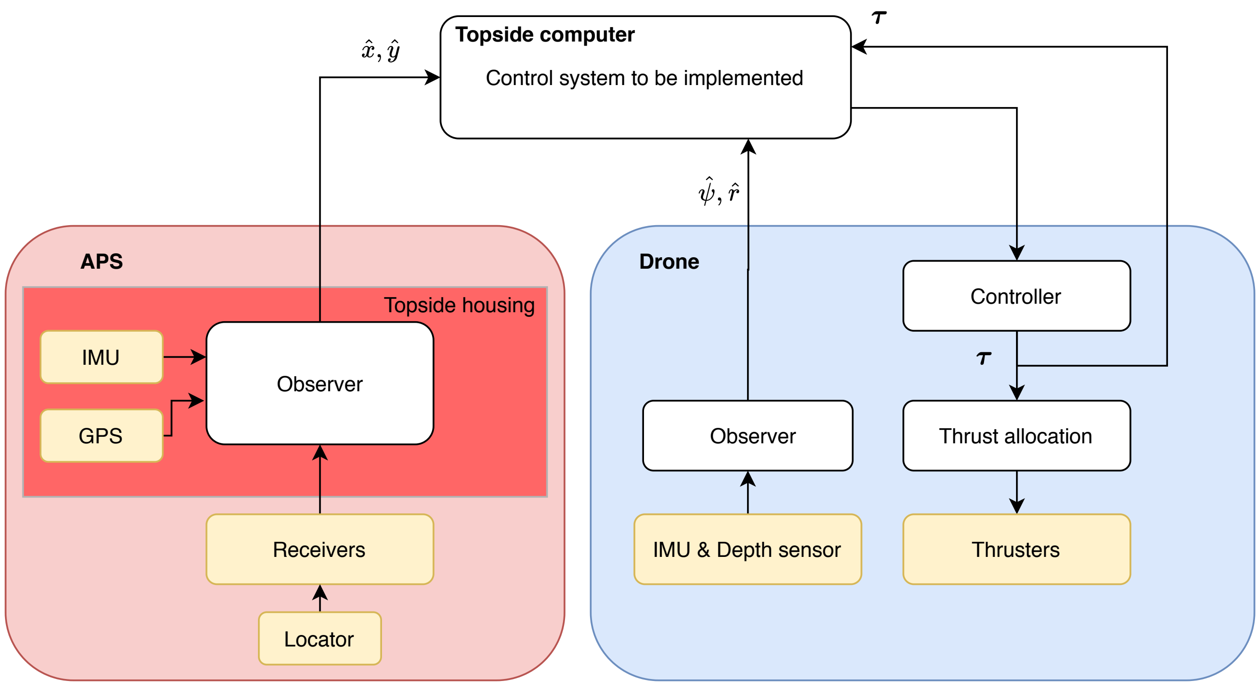

Figure 3.5: Software topology of the drone and APS.

3.4.1 Internal Observers

Both the APS and the drone are equipped with observers, hereby referred to as the "internal observer" of the APS and the drone. For the drone, the internal observer is an integrated part of the internal control system on the drone. The drone's internal observer computes multiple heading estimates using different combinations of sensor data, which are published using ROS on Rostopics. Two relevant topics that represent the heading will be described here. The topic $observer/attitude$ gives the heading angle computed using a fusion of the

[5]https://blueye-robotics.github.io/blueye.sdk/
[6]https://waterlinked.github.io/docs/explorer-kit/gui/api/

digital compass, i.e. the magnetometers in the IMU, and the integrated heading rate from the gyroscope in the IMU. The second topic $observer/state\_estimator$ bases the heading estimated on the integrated heading rate from the gyroscope in the IMU. When the heading angle is computed by integrating the heading rate given by the gyroscope, this will lead to a drift in the heading estimate. The heading estimate from the topic $observer/state\_estimator$ is used in the internal control system on the drone.

3.5 Control Modes

The drone is equipped with auto depth and auto heading functions. Auto depth enables the drone to keep a specified depth, while it can move in x and y direction. For auto heading, the drone keeps a heading specified by the user. It is also possible to control the drone manually, with e.g. joysticks. The drone has functionalities named ”slow” and ”boost”, which enable the control input to be either smaller or larger than the nominal force input.

3.6 Marine Cybernetics Laboratory

The Marine Cybernetics Laboratory at Tyholt, hereby referred to as the MC-lab, is used to perform some tests with the drone. Information regarding the lab is found at ntnu.edu[7]. This lab is equipped with a camera-based positioning system named Qualisys. The system has a position data noise level of ± 1 camera units. More information regarding Qualisys can be found at qualisys.com[8].

[7]https://www.ntnu.edu/imt/lab/cybernetics
[8]https://www.qualisys.com/

Chapter 4

Sensor Suite for DP

The available sensor suite relevant for DP is presented in this chapter, including sensor characteristics in terms of update rate, noise, bias accuracy, signal type and other relevant failure modes.

4.1 Drone-Related Sensors

The drone uses both TCP and UDP for network communication. UDP is used for the main communication of data out from the drone, e.g. depth and heading measurements. For the main communication input to the drone, TCP is applied. Examples of information that is sent over TCP is motion input (in surge, sway etc.) or a service to set the auto function for heading on.

4.1.1 Depth Sensor

The depth sensor has an update rate of 42 Hz. The resolution of the sensor is 0.2 mbar according the sensor specifications[1], which equals approximately 0.2 cm in depth. The resolution describes the magnitude of change in the measurement needed for it to be detected. The noise variance in the depth measurements is approximately $1 \times 10^{-6}\,\mathrm{m}^2$, based on measurements on the same drone in Kvalberg (2019). The depth sensor measurements are accurate for most depths, but has some uncertainty when it is close to or above the surface. For most operational conditions, this will not be an issue. The signal from the depth sensor is an I2C type, which is a serial communication bus.

4.1.2 IMU

The drone is equipped with two identical IMUs for redundant measurements. Since they are identical, only one will be described here. The MEMS-based IMU consists of a 3-axis gyroscope, 3-axis accelerometer, and 3-axis magnetometer. The IMU has an update rate of 100 Hz. The IMU is placed behind the camera, which is not in the CO, which gives a lever arm from the IMU measurement to the CO. The IMU measurements experience drift, which is one of its largest error source. The signals from the IMU are also I2C, like the depth sensor. The heading angle is one of the measurements the IMU can provide. A typical failure mode for the heading angle computed using the magnetometer, is magnetic disturbances. This magnetic disturbance can be caused by external objects with magnetic properties, e.g. ferrous metals. This is a relevant issue when

[1]`Blueye.no`

using the drone for e.g. inspections, as the drone can be close to structures containing the ferrous metal iron, which can cause a magnetic disturbance for the magnetometer. The drone is equipped with motors for the thrusters. Motors produce magnetic fields, which can influence the heading angle from the magnetometer as well. In the internal observer of the drone, the digital compass measurements are calibrated to reduce the effect of the running motors on the heading angle.

4.2 APS

The update rate of an acoustic system is normally lower than surface based positioning systems (e.g GNSS). For underwater acoustics, the acoustic signal travels at speed of sound in water, while in GNSS radio signals are used, whom travel at the speed of light which is much higher than speed of sound in water.Using the transmitter in Locator U1 in Water Linked's APS, the system has an update rate of 2 Hz. The accuracy of an APS is determined by the range. For this system, the maximum range is 100 m from any of the receivers to the transponder. The update rate of APS depends on slant range between the transponder and transducer, where the rate decreases and gets more irregular at lager ranges (Zhao et al., 2012). The Locator is equipped with a internal GPS to time sync the internal clock when the Locator is still in air. However, after some time under water, the clock will start to drift and lead to an error in the position estimate. Diving with the Underwater GPS for 7 hours, the signal will drift approximately 1 m. Water Linked's API will give an error message if the position is lost or other issues are discovered during operation. The signal from the receivers in the APS is a Water Linked proprietary signal. TCP is used as the network interface.

4.3 Location of Sensors

The location of the sensors are needed to transform these measurements to the desired reference frame in the control system. The IMU on the drone is placed behind the camera, while the APS transmitter is attached to the top of the drone. Table 4.1 gives their distance relative to CG in the body frame, which coincides with the CO (see Section 5.2).

System	**Sensor**	**X** [m]	**Y** [m]	**Z** [m]
Drone	Pressure sensor	0.0547	-0.0008	-0.0951
	IMU	0.1239	-0.0442	-0.1173
APS	APS transmitter	0.0814	0.0000	-0.2036

Table 4.1: Sensor location relative to CG.

4.4 Sensor Update and Uncertainties

The average update rates, standard deviation of the noise, and bias for the different sensors are given in Table 4.2. The data for the pressure sensor, accelerometer, gyro, magnetometer in the drone are gathered from experiments on the drone done in Kvalberg (2019). For bias, only data on the accelerometer and gyro in the drone's IMU is known. The APS sensor represent the characteristics of the system as a whole. Data on the noise for the APS measurements are not available.

System	**Sensor**	**Noise std**	**Update rate**	**Bias**
Drone	Pressure sensor	$1 \times 10^{-3}\,\mathrm{m}$	42 Hz	
	Accelerometer (IMU)	$0.0308\,\mathrm{m/s^2}$	100 Hz	$6.8 \times 10^{-6}\,\mathrm{m/s^2}$
	Gyroscope (IMU)	$3.08 \times 10^{-5}\ rad/s$	100 Hz	0.017 rad/s
	Magnetometer (IMU)	$1 \times 10^{-3}\ rad$	100 Hz	
APS	APS	-	2 Hz	

Table 4.2: Sensor update rates, noise standard deviation and bias overview.

Chapter 5

Problem Statement

To make the motion control system, a model of the drone must first be identified. A large focus will be given the identification of the damping parameters, as these are hard to find. With the model in place, the observer will be designed. Once the measurements have been filtered and faulty signals handled in the observer, the control law can be implemented. Lastly, to get a complete DP system, a guidance system must be designed in order to set the desired pose for the control system. In this project, the existing thrust allocation algorithm on the drone will be applied.

5.1 Configuration Space and Workspace

A configuration space is ”the space of possible positions and orientations a vehicle can attain” (Fossen, 2011). The configuration space for the drone is 6 DOF, which are surge, sway, heave, roll, pitch, and yaw. The workspace of the drone is a reduced space of the real configurations space (Fossen, 2011), chosen to be 3 DOF in the horizontal plane, i.e. surge, sway, and yaw. The workspace is reduced to the chosen DOFs since the drone is already stable with regards to roll and pitch, and has existing functionality to control heave. The drone is equipped with a heading controller, but the yaw motion is included because it is essential for controlling the motion in the horizontal plane. The observer design model (ODM) and control design model (CDM) are defined in the workspace of the drone.

5.2 Reference Frames

The CO, i.e., the origin of the body frame and the origin of the control system, is chosen to coincide with the center of gravity (CG), such that $x_g = y_g = z_g = 0$. For simplicity, the CG is assumed constant. In this thesis, the drone will be controlled from shore at the location of the user, giving the need to define a user reference frame. The user frame is chosen to be equivalent to the acoustic reference frame (see Section 3.2), where the origin is at the location of the topside housing (Master-D1) of the APS, the x-axis is parallel to the shore, the y-axis is pointing straight out from the shore, and the z-axis points upwards. Elements in the user frame are denoted with the superscript u. The NED frame is assumed inertial, which means that the rotation of the Earth is neglected and Newton’s law apply. This is a fair assumption when the drone is operated in a limited geographical area. The NED frame used in this control system is defined as a local NED frame with the same origin as the user frame, i.e., at the topside housing of the APS. The coordinates in the local NED frame are given in meters from its origin. The relation between the local NED, body and user frame is shown in Figure 5.1. As can be seen from the figure, the user frame is flipped relative to the NED frame and the body frame, that is, the z-axis in NED points downwards, while the z-axis in the user frame points upwards. The angle between the NED frame and the body frame is denoted α.

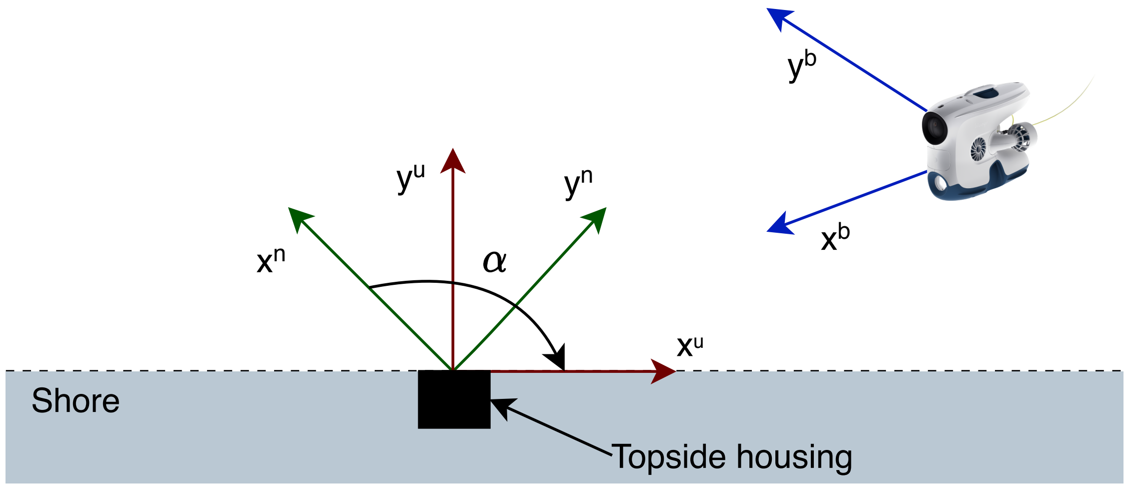

Figure 5.1: Local NED (n), body (b) and user (b) frame together. α is the angle between the NED frame and the user frame. Courtesy of Blueye drone: Blueye Robotics.

5.3 Equations of Motion

The vessel model in 3 DOFs is represented by the states and control inputs

$$\boldsymbol{\eta}^n = \begin{bmatrix} x \\ y \\ \psi \end{bmatrix}, \quad \boldsymbol{\nu}^b = \begin{bmatrix} u \\ v \\ r \end{bmatrix}, \quad \boldsymbol{\tau}^b = \begin{bmatrix} X \\ Y \\ N \end{bmatrix}, \tag{5.1}$$

where $\boldsymbol{\tau}$ is the control load vector. The superscript $\{n\}$ and $\{b\}$ mean that the vector is in the NED frame and body frame, respectively. The heading angle is chosen to be contained between [-pi,pi). The equations of motions for the drone in the workspace is given by

$$\dot{\boldsymbol{\eta}}^n = \boldsymbol{J}(\boldsymbol{\psi}^n)\boldsymbol{\nu}^b \tag{5.2a}$$
$$\boldsymbol{M}\dot{\boldsymbol{\nu}}^b = -\boldsymbol{C}(\boldsymbol{\nu}^b)\boldsymbol{\nu}^b - \boldsymbol{D}(\boldsymbol{\nu}^b)\boldsymbol{\nu}^b + \boldsymbol{\tau}^b + \mathbf{b}^b \tag{5.2b}$$
$$\dot{\mathbf{b}}^b = \mathbf{w}^b, \mathbf{w}^b \tag{5.2c}$$

where the transformation matrix $\boldsymbol{J}(\boldsymbol{\psi})$ is given by

$$\boldsymbol{J}(\boldsymbol{\psi}) = \begin{bmatrix} c(\psi) & -s(\psi) & 0 \\ s(\psi) & c(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{5.3}$$

where $c(\cdot)$ and $s(\cdot)$ are abbreviations for the cosine and sine functions. The bias $\mathbf{b} = [b_x, b_y, b_\psi]$ accounts for unmodeled effects and is modeled as a Wiener process where $\mathbf{w}$ is a zero-mean Gaussian white noise. The force from the umbilical acting on the drone is not included in the model, and will thus be a part of the bias. The restoring force is not included as it acts in the vertical plane, not the horizontal plane which is modeled here.

5.4 State-Space Model

The system model in (5.2) is written in state-space form by defining a state **x**, input **u** and measurement vector **y**.

$$\dot{\mathbf{x}} = \mathbf{A}(\mathbf{x})\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{E}\mathbf{w}, \quad \mathbf{w} \sim \mathcal{N}(0, \mathbf{Q}) \tag{5.4a}$$

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{v}, \quad \mathbf{v} \sim \mathcal{N}(0, \mathbf{R}) \tag{5.4b}$$

where **w** is the process noise and **v** is the measurement white noise. They are both modeled as zero-mean Gaussian noise. The chosen state and measurements vectors are given by (5.5). The control input is the control load produced by the thrusters.

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{\eta}^n \\ \boldsymbol{\nu}^b \\ \mathbf{b}^b \end{bmatrix}, \mathbf{y} = \begin{bmatrix} \mathbf{y}^n_{aps} \\ \psi \\ r \end{bmatrix}, \text{and } \mathbf{u} = \boldsymbol{\tau}^b \tag{5.5}$$

where $\mathbf{y}_{aps} = [x, y]^\top$ is the position measurement from the APS. The heading ψ and the heading rate *r* are both from the drone's internal observer, but from different rostopics published by the drone. The heading is gathered from the rostopic named $observer/attitude$, which is based on the heading computed using the magnetometers and the gyroscope in the IMU, while the heading rate is from the rostopic $observer/state\_estimator$, which is computed using the gyroscope in the IMU. This choice of vectors gives the following state-space matrices

$$\mathbf{A}(\mathbf{x}) = \begin{bmatrix} \mathbf{0} & \boldsymbol{J}(\boldsymbol{\psi}) & \mathbf{0} \\ \mathbf{0} & -\mathbf{M}^{-1}(\boldsymbol{C}(\boldsymbol{\nu^n}) + \boldsymbol{D}(\boldsymbol{\nu^n})) & \mathbf{M}^{-1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ & & \end{bmatrix} \tag{5.6a}$$

$$\mathbf{B} = \begin{bmatrix} \mathbf{0} \\ \mathbf{M}^{-1} \\ \mathbf{0} \end{bmatrix} \tag{5.6b}$$

$$\mathbf{E} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{I} \end{bmatrix} \tag{5.6c}$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & \mathbf{0}_{1x3} \\ 0 & 1 & 0 & 0 & 0 & 0 & \mathbf{0}_{1x3} \\ 0 & 0 & 1 & 0 & 0 & 0 & \mathbf{0}_{1x3} \\ 0 & 0 & 0 & 0 & 0 & 1 & \mathbf{0}_{1x3} \end{bmatrix}. \tag{5.6d}$$

5.5 Speed Regimes

As stated in Section 1.4, the DP system will only be developed for low speed operations. This requires a definition of what is considered low speed regarding the drone. The maximum forward (surge) speed of the drone is 3 knots, i.e. $1.54\,\mathrm{m\,s^{-1}}$. The maximum speed in the other DOFs are unknown, but assumed to be lower than surge. For this project, low speed is defined as $0{-}0.9\,\mathrm{m\,s^{-1}}$ for surge, sway and heave, and $0{-}15\,^{\circ}\,\mathrm{s^{-1}}$ for yaw.

5.6 Assumptions

- The NED frame is assumed inertial, which is a fair assumption when the drone is to be operated in a small geographical area.
- The CO in the BODY frame is chosen to coincide with the center of gravity (CG). This means that $x_g = y_g = z_g = 0$.
- The functionalities for slow and boost on the control input to the thrusters, discussed in Section 3.5, will not be applied in this project. This is done to simplify the development of the DP system.
- Since the lateral thruster is placed 2 cm above the CO (see Table 3.2), it will give small roll moment to the drone. The vertical thruster has a lever arm in both x and y direction in the body frame, giving pitch and roll moment. The lever arm along the y-axis for the vertical thruster is close to zero, so the roll moment caused by this is negligible. With the four existing thruster on the drone, the control system is not able to compensate for these roll and pitch moments.

Chapter 6

Estimation of Model Parameters

The following chapter shows how the model parameters for a vessel model defined in the 3 DOF workspace were estimated.

6.1 Rigid-Body

The rigid-body inertia matrix $\boldsymbol{M_{RB}}$ and the Coriolis-centripetal matrix for rigid body $\boldsymbol{C_{RB}}(\boldsymbol{\nu})$ in 6 DOF are given by (Fossen, 2011),

$$\boldsymbol{M_{RB}} = \begin{bmatrix} m\mathbf{I} & -m\mathbf{S}(\mathbf{r}_g^b) \\ m\mathbf{S}(\mathbf{r}_g^b) & \mathbf{I}_b \end{bmatrix} \tag{6.1a}$$

$$\boldsymbol{C}_{RB}(\boldsymbol{\nu}) = \begin{bmatrix} \mathbf{0} & -m\boldsymbol{S}(\boldsymbol{\nu}_1) - m\boldsymbol{S}(\boldsymbol{\nu}_2)\mathbf{S}(\mathbf{r}_g^b) \\ -m\boldsymbol{S}(\boldsymbol{\nu}_1) + m\mathbf{S}(\mathbf{r}_g^b)\boldsymbol{S}(\boldsymbol{\nu}_2) & \mathbf{S}(\mathbf{I}_b\boldsymbol{\nu}_2) \end{bmatrix} \tag{6.1b}$$

,

where m is the mass, $\mathbf{I}_b$ is the inertia matrix about CO, $\mathbf{S}$ is the cross-product operator, $\mathbf{r}_g^b$ is the distance vector from CO to CG, and $\boldsymbol{\nu}_1 = [u, v, w]^\top$ and $\boldsymbol{\nu}_2 = [p, q, r]^\top$. Since the drone is symmetric about the xz-plane, $I_{xy} = I_{yz} = 0$. Moreover, since the body-frame origin CO is placed in the CG, $\mathbf{r}_g$ equals zero. Applying these assumptions, the model parameter matrices in 3 DOF simplify to

$$\mathbf{M}_{RB} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I_z \end{bmatrix} \quad , \quad \boldsymbol{C}_{RB}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & -mv \\ 0 & 0 & mu \\ mv & -mu & 0 \end{bmatrix}. \tag{6.2}$$

The mass, moment of inertia, and linear velocities v and u are estimated by the observer, providing the rigid-body matrices.

6.2 Hydrodynamics

The added mass matrix in 3 DOF is defined as

$$\mathbf{M_A} = -\begin{bmatrix} X_{\dot{u}} & X_{\dot{v}} & X_{\dot{r}} \\ Y_{\dot{u}} & Y_{\dot{v}} & Y_{\dot{r}} \\ N_{\dot{u}} & N_{\dot{v}} & N_{\dot{r}} \end{bmatrix}. \tag{6.3}$$

The added mass was found using the empirical method developed by Eidsvik (2015), which takes use of experimental data from a DNV standard. The method was validated on the uDrone in Sandøy (2016). Eidsvik's method assumes that the ROV is a rectangular prism where 2 of 3 sides are equal. The difference between the two approximately equal sides should not be larger than 10%. For the Blueye drone, the width is 25% lower than the height, meaning that the assumption did not hold. Additionally, the thrusters on the drone are placed on the sides and change the flow around the vehicle compared to a rectangle shaped prism. As discussed in Mokleiv (2017), Eidsvik tested the ROV Neptunus, which has some geometric similarities to the Blueye drone and also has the thrusters sticking out. The empirical method was satisfactory for the ROV Neptunus except for heave, roll, and yaw directions.

The added mass was computed using the length, height, width, and projected areas for each side of the drone given in Table 3.1. The added mass was computed for both fresh water, whose density is assumed to be 1000 kg/m^3, and sea water, whose density is assumed to be 1025 kg/m^3. This gave the added mass matrix in (6.4). The location of script to compute the added mass is given in Appendix B.1.1.

$$\mathbf{M}_A = \text{diag}\{5.09, 19.38, 0.16\} \quad \text{for fresh water} \tag{6.4a}$$

$$\mathbf{M}_A = \text{diag}\{5.22, 19.87, 0.16\} \quad \text{for sea water} \tag{6.4b}$$

Modeling the added mass matrix as a diagonal matrix (Fossen, 2011, p.121), the Coriolis-centripetal matrix simplified to

$$\mathbf{C}_A(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & Y_{\dot{v}}v \\ 0 & 0 & -X_{\dot{u}}u \\ -Y_{\dot{v}}v & X_{\dot{u}}u & 0 \end{bmatrix}, \tag{6.5}$$

where $Y_{\dot{v}}$ and $X_{\dot{u}}$ are elements in the added mass matrix. Since the Coriolis-centripetal matrix $\mathbf{C}(\nu)$ is linear in the velocity vector $\boldsymbol{\nu}$, the matrix is easily modeled taking $\boldsymbol{\nu}$ as input. The damping can be divided into a linear and nonlinear part

$$\mathbf{D}(\boldsymbol{\nu}) = \mathbf{D}_L + \mathbf{D}_{NL}(\boldsymbol{\nu}). \tag{6.6}$$

The linear damping is due to potential damping and skin friction, while the nonlinear damping is due to quadratic and higher-order damping terms (Fossen, 2011). The linear damping was modeled as

$$\mathbf{D}_L = \begin{bmatrix} X_u & 0 & 0 \\ 0 & Y_v & Y_r \\ 0 & N_v & N_r \end{bmatrix}. \tag{6.7}$$

The nonlinear damping was simplified to include only the quadratic terms, i.e., $\mathbf{D}_{NL}(\boldsymbol{\nu}) = \mathbf{D}_Q(\boldsymbol{\nu})$, which is given by

$$\mathbf{D}_Q(\boldsymbol{\nu}) = \begin{bmatrix} X_{|u|u}|u| & 0 & 0 \\ 0 & Y_{|v|v}|v| + Y_{|r|v}|r| & Y_{|v|r}|v| \\ 0 & N_{|r|v}|r| & N_{|r|r}|r| + N_{|v|r}|v| \end{bmatrix}. \tag{6.8}$$

Identification of the damping coefficients is explained in Section 6.4.

6.3 Hydrostatics

The restoring force includes the gravitational $\mathbf{f}_g^n$ and buoyancy force $\mathbf{f}_b^n$. The gravitational force is given by the submerged weight of the drone $W = mg$, while the buoyancy is given by $B = \rho g \Delta$ (Fossen, 2011). Δ is the volume displacement of the drone and equals the volume of the drone when the drone is fully submerged. These forces act in the vertical plane, which is not included in the workspace of the drone.

6.3.1 Thrust Dynamics

The control forces and moments on the drone are determined by the four thrusters described in Section 3.1.2. The forces and moments $\boldsymbol{\tau}$ in surge, sway, and yaw from a force vector $\mathbf{f} = [X, Y, Z]^\top$ is given by

$$\boldsymbol{\tau} = \begin{bmatrix} \mathbf{f} \\ (\mathbf{r} \times \mathbf{f})_{yaw} \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Yl_x - Xl_y \end{bmatrix} = \begin{bmatrix} X \\ Y \\ N \end{bmatrix}, \tag{6.9}$$

where $\mathbf{r} = [l_x, l_y, l_z]^\top$ is the location of the thruster in the body frame relatice to the CO. The thrust forces $\mathbf{f}_t$ and control input $\mathbf{u}$ for each thruster is defined in Table 6.1. The control input is the PWM signals to each thruster. Additionally, the lever arms from CO to each thruster giving the yaw moment is defined.

Location	**Thrust force**	**Control input**	**Lever arm**
Longitudinal starboard	f_{t1}	u_1	-$l_{y,longitud}$
Longitudinal port	f_{t2}	u_2	$l_{y,longitud}$
Lateral	f_{t3}	u_3	$l_{x,lat}$
Vertical	f_{t4}	u_4	

Table 6.1: Thrust notation.

The vertical thruster has no lever arm for the yaw moment, since the force direction is parallel to the z-axis. The lever arms to the yaw moment can be found in Table 3.2, but restated here for clarity,

Lever arm	**Value** [m]
$l_{y,longitud}$	0.08082
$l_{x,lat}$	0.06141

Table 6.2: Lever arms for yaw moment.

The thrust forces $\mathbf{f}_t$ are related to the control inputs $\mathbf{u}$ by a force coefficient matrix $\mathbf{K}$, i.e.,

$$\mathbf{f}_t = \mathbf{K}\mathbf{u}, \tag{6.10}$$

The control load vector produced by all the thrusters $\boldsymbol{\tau}$ is related to the individual thruster forces by

$$\boldsymbol{\tau} = \mathbf{T}\mathbf{f}_t, \tag{6.11}$$

where $\mathbf{T}$ is the thrust configuration matrix that depends on the location of the thrusters relative to the center of gravity, given by

$$\mathbf{T} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -l_{y,longitud} & l_{y,longitud} & l_{x,lat} & 0 \end{bmatrix}. \tag{6.12}$$

6.4 Identify Damping by Free-Running Test and DFO

To identify the damping coefficients, a free-running tests of the drone in the MC-lab with all sensors connected, together with camera-based positioning system Qualisys, was performed. The free running tests consisted of six different parts, i.e., (1) surge test, (2) sway test, (3) yaw test, (4) combined surge and sway test, (5) combined surge and yaw test, and (6) combined sway and yaw test. Each test was conducted by setting a constant speed of the DOF(s) to be tested first in positive direction, zero speed, and then in negative direction. This was done using the Blueye SDK, from where you can set the input in the four DOFs surge, sway, heave and yaw. All the tests were run in the low speed regime, defined in Section 5.5.In order to get a free floating drone, the drone was controlled to have a negative heave motion during the test, such that the drone would stay above the bottom of the tank. This means that for all the DOFs tested, there was always a coupling to heave. The auto functions for heave and heading were both on during the tests. Due to shallow water in the MC-lab, it was not possible to test the drone for heave alone.

6.4.1 Gather Data From Free-Running Test

The data was gathered by running the positioning system Qualisys in MC-lab and logging data from the drone simultaneously. Both Qualisys and drone data were sampled at 50 Hz. The following data was saved:

1. From Qualisys:
   - the position in x, y and z
2. From drone:
   - control force $\boldsymbol{\tau}$
   - depth from the drone's state estimator
   - heading angle from the drone's state estimator

Ideally, data from Water Linked's APS should also have been recorded, but since the system does require constant GPS lock, this could not be performed inside the MC-lab.

6.4.2 Data Processing

Before the data could be used for optimization, data from different sources had to be synchronized, missing data had to be handled and noisy signals filtered. Additionally, the velocity had to be computed from the position such that it could be used in the optimization.

Change Reference frame

The position data from Qualisys was measured in a local reference frame in the MC-lab, illustrated in Figure 6.1. The Qualisys position coordinates needed to be transformed to the NED frame, such that it would be in the same frame as the position to be used in the simulation. To do this, the angle between the magnetic north and the x-axis in the Qualisys reference frame had to be determined. This was done by measuring the heading estimated by the drone when it was aligned with the x-axis of the Qualisys reference frame. This gave the angle $\phi_Q = 156^\circ$ from North to the Qualisys frame. This angle measurement will have elements of uncertain due to possible errors in the alignment to the x-axis of the Qualisys reference frame and uncertainty in heading angle measurement of the drone.

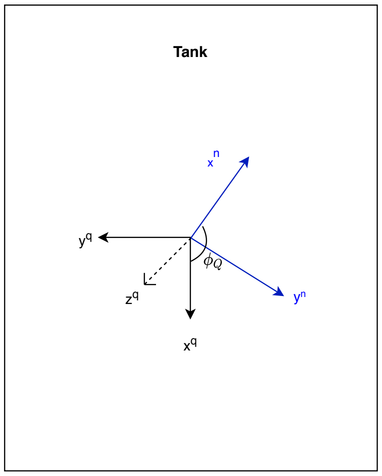

Figure 6.1: Qualisys Reference System in the tank in the MC-lab. The blue lines represent the NED frame, while the lines marked in black represent the axes in the Qualisys reference frame. ϕ_Q is the angle from x-axis in the NED frame to the x-axis of the Qualisys frame.

Synchronization of Data

The moment of initiation for gathering data from Qualisys and the drone was not the same during the tests, as they were gathered using two different computers. The Qualisys data were gather using an offline program, for which an accurate timestamp was not added to the data. To synchronize these data, the depth measurements from Qualisys and the drone were compared, to find the delay between the to. This was done by visual inspection of the data, and will give an uncertainty in the analysis using these data.

Interpolation and Cleansing

The position data from Qualisys contained a lot of missing data points. This occurred frequently when the Qualisys' cameras that captured the drone switched as the drone was moving. According to the employee Torgeir Wahl working in the MC-lab, Qualisys captured the motion of the drone poorly due to its unusual shape. Firstly, the start and end of the tests were found by looking at the when control force input for the DOF to be tested was unlike zero. This was easily detected for all tests including surge and sway, but for the yaw test this could not be detect, since the auto-depth function was on during the whole test. Secondly, interpolation was used to estimate the missing data points in the position data from Qualisys. For this, the MATLAB function *interp1* with the spline method was used. When the measurements only contained a few missing data points, the interpolation worked well. However, since the true position was unknown, it was not possible to tell if the interpolation was correct. In the optimization, the DFO does not know if the data are true measurements or interpolated, and trust them equally, which leads to an uncertainty in the model parameters. The uncertainty increased with the amount of missing data. The heading measurements from the drone was not filtered, as the filtering in the drone's state estimator was considered sufficient.

Filter Position

To filter the position data, a lowpass filter was applied. With a passband frequency of 0.01 Hz and sampling rate of 50 HZ , the noise was removed from the measurements. However, using a lowpass filter, the filtered measurement vector got transients at the beginning and end of the vector. This is handled by using the raw measurements for the first and last part of the vector instead of the filtered measurement. The heading angle was contained between $[-pi, pi)$.

Find Velocity

The velocity vector was computed using a Finite Impulse Response (FIR) differentiating filter on the filtered pose. The filter was implemented using the FIR differentiating filter found at github.com[1], with a filter order of 50, passband frequncy of 2 Hz, and a stopband frequency of 5 Hz. This filter is based on the FIR differentiation filter developed in Udjus (2017). Since the heading angle was contained between -pi and pi, the computed heading rate got transients when the heading changes quickly from pi to -pi or visa versa. Lastly, the velocity vector was transformed to the body frame using a rotation matrix.

Uncertainties

The uncertainties in the data that may influence the damping identification are summarized here:

- Measurement uncertainty in Qualisys and sensors on the drone.
- Angle between North and Qualisys reference frame ϕ_Q.
- Synchronization of Qualisys and drone data.
- Accuracy of interpolated position data.
- Accuracy of velocity estimates.

6.4.3 Damping Identification Using DFO

To identify the damping, DFO was applied to the experimental data. The data was first divided in low and high speed according to Section 5.5. Only low speed data was be applied, since the goal is to make a low speed DP system. Secondly, the low speed data was divided in test and training data, where the training data was used to tune the optimization parameters. The test data was used to find the accuracy of the optimization using an "unknown" data sets. DFO is useful when the gradient of the objective function is unknown or uncertain, which was the case for this optimization. The steps needed to estimate the damping using DFO are:

1. **Parameterization** Parameterize the drone, i.e., design the model for the simulation and verification model (SVM).
2. **SVM** Run simulations on the same cases as in the experiment using the control input to the thrust allocation as input.
3. **KPI** Compute the Key Performance Indicator (KPI) to see how the optimization performed.
4. **DFO** Find damping parameters using the chosen DFO algorithm based on the KPI.

Steps 2-4 are repeated until the optimization find damping parameters with a satisfactory KPI. If it cannot conclude with a satisfactory KPI, a new parameterization (step 1) can be performed.

Parameterization

To identify the damping by optimization, the drone's motion was parameterized. Two different 3 DOF models were used in the simulation, both using (5.2) as a basis. The two models only differ in how the kinetic equation (5.2c) is modeled. In the simple model (Model 1) the Coriolis-centripetal matrix $\boldsymbol{C}(\boldsymbol{\nu})$ and the nonlinear damping term $\boldsymbol{D_L}(\boldsymbol{\nu})$ are neglected. Additionally the bias loads $\mathbf{b}$ are neglected, meaning that

[1] `https://github.com/NTNU-MCS`

the unmodeled dynamics will be included in the damping term. The second model, Model 2, is modeled equally as (5.2c), except that the nonlinear damping only includes linear damping and quadratic damping. The kinetic equations are given by

$$\text{Model 1} \quad \boldsymbol{M}\dot{\boldsymbol{\nu}} = -\boldsymbol{D_L}\boldsymbol{\nu} + \boldsymbol{\tau}, \tag{6.13a}$$

$$\text{Model 2} \quad \boldsymbol{M}\dot{\boldsymbol{\nu}} = -\boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} - (\boldsymbol{D_Q}(\boldsymbol{\nu}) + \boldsymbol{D_L})\boldsymbol{\nu} + \boldsymbol{\tau} + \mathrm{b}, \tag{6.13b}$$

with corresponding position vector $\boldsymbol{\eta}^n = [x, y, \psi]$, velocity vector $\boldsymbol{\nu}^b = [u, v, r]$, and control load vector $\boldsymbol{\tau}^b = [X, Y, N]$. The reference frame superscripts are omitted for simplicity, but the states are given in the frame indicated in the former sentence. Since there was no current or waves in the MC-lab tank, the velocity vector became simply $\boldsymbol{\nu}$. The model parameters found in Section 6.1 and Section 6.2 were applied, except that the linear damping was chosen to be symmetrical, meaning that $Y_r = N_v$. The mass and Centripetal-centripetal matrices were assumed known and correctly identified.

Simulation

In order to estimate the position and velocity, a simulation verification model (SVM) was developed. The SVM may include the drone dynamics, propulsion system, external forces (wind, waves, and current), and sensor modules (Fossen, 2011). The SVM simulates the response of the system using the model specified. The SVM takes the control force load $\boldsymbol{\tau}$ as input, and outputs the acceleration. The acceleration was found by solving the kinetic equation (6.13) using the currently chosen model. To methods for estimating the velocity and pose were developed; Method 1 and Method 2.

Method 1: Open-Loop Simulation Method 1 is an open-loop simulation where the velocity was found by integrating the acceleration and the pose was found by integrating the velocity and by solving the kinematic equation (2.12). This was done in discrete time by using the Forward Euler method, given by

$$\hat{\boldsymbol{\eta}}_{k+1} = \hat{\boldsymbol{\eta}}_k + h\boldsymbol{J}(\hat{\boldsymbol{\psi}}_k)\hat{\boldsymbol{\nu}}_k \tag{6.14a}$$

$$\hat{\boldsymbol{\nu}}_{k+1} = \hat{\boldsymbol{\nu}}_k + h\dot{\boldsymbol{\nu}}_k, \tag{6.14b}$$

where *h* is the sampling time.

Method 2: Closed-Loop Simulation with Observer Method 2 is a closed-loop simulation with a Luenberger observer to get better estimates of the velocity and pose. This method also uses Forward Euler to discretize the system. The discrete time state equations for Method 2 is given by

$$\hat{\boldsymbol{\eta}}_{k+1} = \hat{\boldsymbol{\eta}}_k + h(\boldsymbol{J}(\hat{\boldsymbol{\psi}}_k)\hat{\boldsymbol{\nu}}_k + \mathbf{L}_{2\eta}\widetilde{\boldsymbol{\eta}} + \mathbf{L}_{1\nu}\boldsymbol{J}(\hat{\boldsymbol{\psi}}_k)\widetilde{\boldsymbol{\nu}}) \tag{6.15a}$$

$$\hat{\boldsymbol{\nu}}_{k+1} = \hat{\boldsymbol{\nu}}_k + h(\dot{\boldsymbol{\nu}}_k + \mathbf{M}^-1(\mathbf{L}_{1\eta}\boldsymbol{J}^\top(\hat{\boldsymbol{\psi}}_k)\widetilde{\boldsymbol{\eta}} + \mathbf{L}_{1\nu}\widetilde{\boldsymbol{\nu}})) \tag{6.15b}$$

$$\hat{\mathbf{b}}_{k+1} = \hat{\mathbf{b}}_k + h(\mathbf{L}_{1\eta}\boldsymbol{J}^\top(\hat{\boldsymbol{\psi}}_k)\widetilde{\boldsymbol{\eta}} + \mathbf{L}_{1\nu}\widetilde{\boldsymbol{\nu}}), \tag{6.15c}$$

where $\mathbf{b}$ is the bias vector and the estimation error is defined as $\widetilde{\boldsymbol{\eta}} = \boldsymbol{\eta} - \hat{\boldsymbol{\eta}}$ and $\widetilde{\boldsymbol{\nu}} = \boldsymbol{\nu} - \hat{\boldsymbol{\nu}}$. $\boldsymbol{\eta}$ and $\boldsymbol{\nu}$ represent the measured pose and velocity vectors, while $\hat{\boldsymbol{\eta}}$ and $\hat{\boldsymbol{\nu}}$ represent the estimated states in the simulation. By introducing this feedback gain through the observer, the estimate of the damping will get more and more accurate. The gains used in the simulation are stated in Table 6.3.

L-gain	Value
$\mathbf{L}_{1\eta}$	0.1 diag{1,1,180/pi}
$\mathbf{L}_{2\eta}$	100 diag{1,1,180/pi}
$\mathbf{L}_{3\eta}$	50 diag{5,5,pi/180}
$\mathbf{L}_{1\nu}$	0.1 diag{1,1,0.1*pi/180}
$\mathbf{L}_{2\nu}$	diag{1,1,0.1*pi/180}
$\mathbf{L}_{3\nu}$	500 diag{1,1,0.1*pi/180}

Table 6.3: Luneberger gains in method 2.

For both methods, Model 1 and Model 2 were both tested in the simulation. In order to have the same coordinate origin for the simulation as the measurement data, the initial pose of the simulation was set equal to the initial pose from the Qualisys measurement in NED frame and the heading measurement from the drone.

KPI

The sum of absolute error (SAE) was chosen to be the KPI function to be used in this DFO. The KPI was calculated using a data set from each of the six tests, e.g surge-test, surge-sway-test etc. The SAE for each of these tests are summed together to get the total KPI. This is done to get damping parameters that will fit for all the tested DOFs. The function for the KPI is given by

$$KPI = \sum_{i=1}^{m}[w_x, w_y, w_\psi][w_\eta \sum_{k=1}^{n} ||\hat{\boldsymbol{\eta}} - \boldsymbol{\eta}|| + w_\nu \sum_{k=1}^{n} ||\hat{\boldsymbol{\nu}} - \boldsymbol{\nu}||], \tag{6.16}$$

where *m* is the total number of tests, i.e., surge-test, sway-test etc, and *n* are the total number of samples in each test set. $[w_x, w_y, w_\psi]$ is the weighting in surge, sway, and yaw, while w_η and w_ν are the weighting for the pose and velocity error respectively. In the KPI, the heading and heading rate are given degrees. This means that if the weights in all DOFs were equal to 1, the KPI would punish 1° error in heading equally as 1m error in x or y position. The chosen weighting parameters for the two simulation method are given in Table 6.4.

Simulation method	$[w_x, w_y, w_\psi]$	w_η	w_ν
Method 1	[1, 1, 10]	0.25	1
Method 2	[0.2, 0.2, 1]	0.1	1

Table 6.4: Weighting parameters.

DFO

Matlab's bounded fminsearch function[2], *fminsearchbnd*, was the chosen DFO algorithm. Fminsearch uses the Nelder-Mead algorithm, and was chosen since it has few tuning parameters which makes it easy to use. The Nelder-Mead algorithm is explained in some detail in Appendix D. The constrained Nelder-Mead has no analytical guarantee for convergence, but was applied to limit the search area and exclude nonphysical solutions. With both upper and lower bounds on the optimization parameters, the fminsearchbnd method

[2]`https://www.mathworks.com/matlabcentral/fileexchange/8277-fminsearchbnd-fminsearchcon`

uses a sinus transformation to keep the optimization parameters constrained. The goal of the DFO is to minimize the damping parameters $\mathbf{x}$ using the objective function $f(\mathbf{x})$, under some constraints, i.e.,

$$\min_{\mathbf{x}} f(\mathbf{x}) \tag{6.17a}$$

$$\mathbf{x}_{LB} < \mathbf{x} < \mathbf{x}_{UB} \quad \text{constraint} \tag{6.17b}$$

The chosen KPI is the objective function of the DFO. The constraints are here given as upper and lower bound of $\mathbf{x}$, i.e., $\mathbf{x}_{LB}$ and $\mathbf{x}_{UB}$. The damping parameter **x** to be optimized for both models are given by

Model 1 $\mathbf{x} = [X_u, Y_v, Y_r, N_r]$ (6.18a)
Model 2 $\mathbf{x} = [X_u, Y_v, Y_r, N_r, X_{uu}, Y_{vv}, Y_{rv}, Y_{vr}, N_{rv}, N_{vr}, N_{rr}].$ (6.18b)

The chosen constraints for the different methods and models are given in Table 6.5.

	Model 1	**Model 2**
Method 1	LB = [1,50,-3,1]	LB = [1,50,-3,0.1,5,25,-5,-5,-5,-5,0.1]
	UB = [20,100,3,10]	UB = [20,100,3,5,20,100,5,5,5,5,10]
Method 2	LB = [1,30,-3,1]	LB = [1,30,-3,0.1,5,25,-5,-5,-5,-5,0.1]
	UB = [25,100,3,10]	UB = [25,100,3,5,20,100,5,5,5,5,10]

Table 6.5: Lower and upper bounds for optimization

Since the Nelder-Mead algorithm only can guarantee a local optimum, the DFO was fed with three different initialization of $\mathbf{x}$. This was done by generating random numbers between the lower and upper limits of $\mathbf{x}$.

6.4.4 Implementation

Figure 6.2 is a simple illustration of the flow of the optimization for one initialization of x. In the real case, the optimization is ran with several initializations of x. The optimization takes the measured data and the FIR differentiated velocity from all the six tests as input. An initial damping parameter $\mathbf{x}_0$ is chosen randomly between the lower and upper bounds. The bounded fminsearch, i.e. the chosen DFO algorithm, is ran by first running a simulation. The simulation depends on which method and model is chosen. The simulation outputs the estimated position, velocity, and bias. The KPI is computed using the measured and estimated states. The KPI value is returned to the DFO, where the performance of the damping parameter is evaluated based on the KPI value. If the optimization limit is reached, e.g. having a satisfactory low KPI value, the optimization is terminated. If not, a new damping parameter **x** is computed based on the performance of the previous damping parameter. The optimization continuous until a optimization limit is reached, e.g. a satisfactory low KPI value or maximum number of iterations is obtained. The figure is a simplification as it does not show that the different models have different lower and upper bounds. The coded implementation for identifying the damping in Matlab is described in Appendix B.1.2.

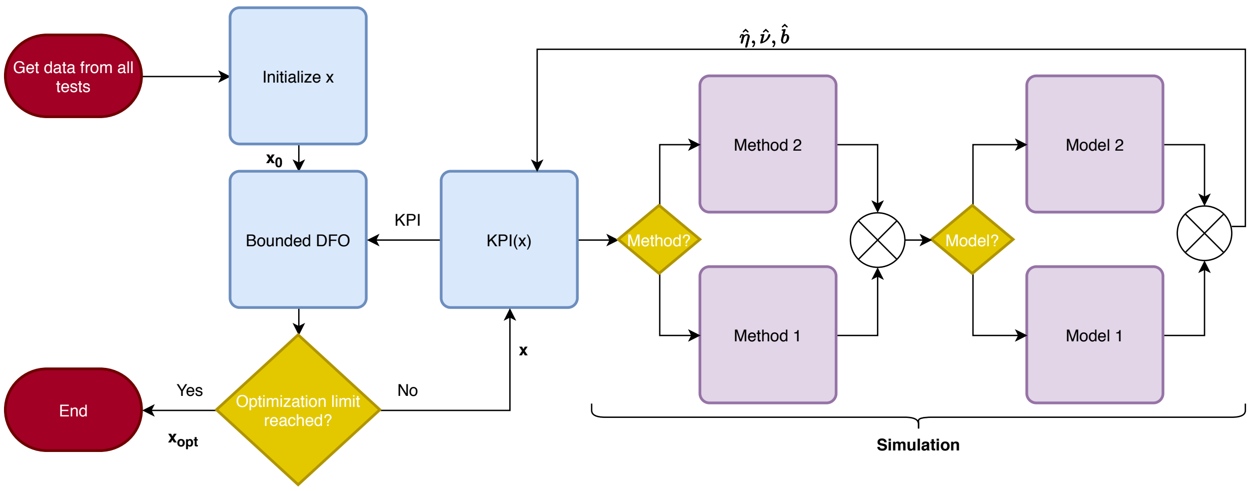

Figure 6.2: Flow of the DFO for one initialization of **x**.

6.4.5 Damping Results

The optimization results for Method 1 and Method 2 are given below. Only a small selection of all the tests performed are be presented.

Results Method 1

All the presented results using Method 1 in the simulation were run using Model 1. Results from a sway-yaw test using Method 1 is shown in Figure 6.3 and Figure 6.4.

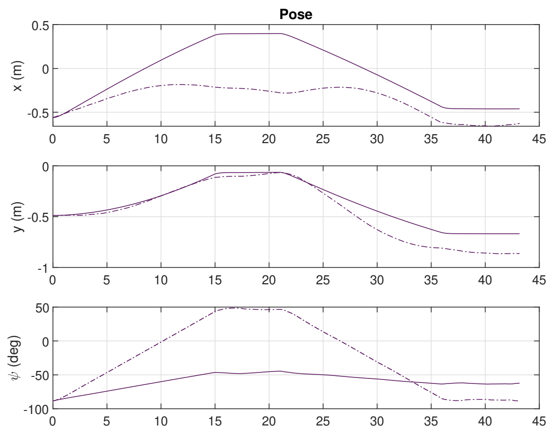

Figure 6.3: Pose from Method 1 using Model 1 in sway-yaw test. X- and y-position is given in the NED frame. The x-axis shows the time in seconds. The simulated responses are illustrated with a solid line, while the measured states are represented by a dotted line.

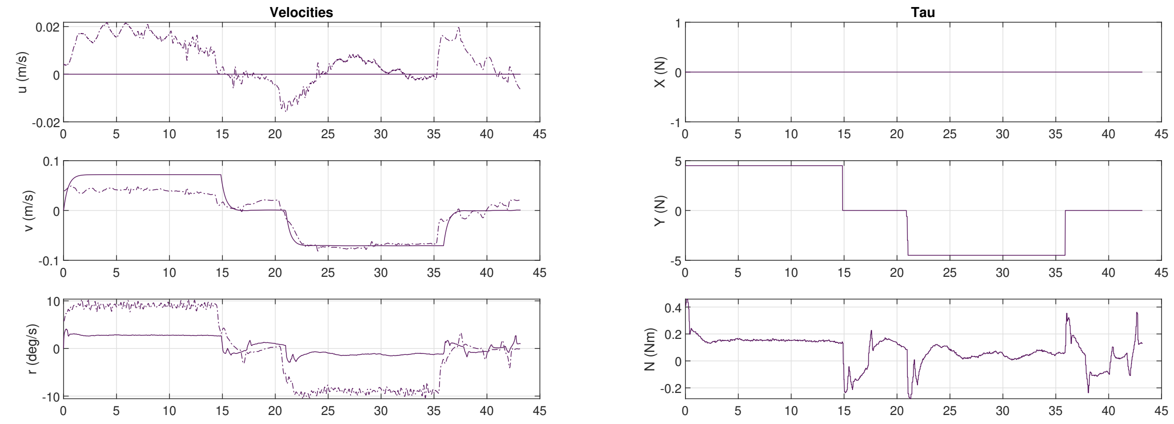

Figure 6.4: Velocity and control load for Method 1 using Model 1 in sway-yaw test. The velocity and control load are given in the body frame. The x-axes shows the time in seconds. The control load ($\boldsymbol{\tau}$) is the input to the SVM, generating the velocity and position estimates. The simulated responses are illustrated with a solid line, while the measured states are represented by a dotted line.

Results from a sway test using Method 1 is shown in Figure 6.5 and Figure 6.6.

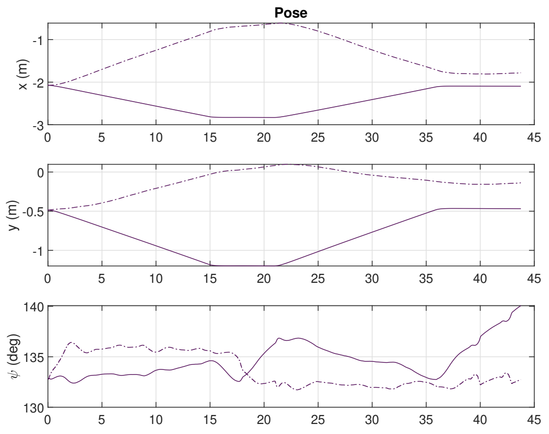

Figure 6.5: Pose for Method 1 using Model 1 in sway test. X- and y-position is given in the NED frame The x-axis shows the time in seconds. The simulated responses are illustrated with a solid line, while the measured states are represented by a dotted line.

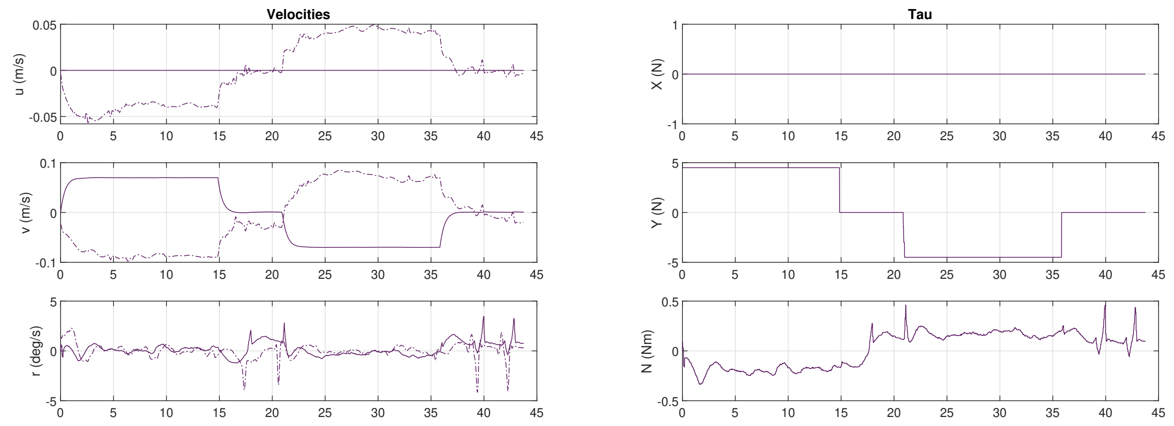

Figure 6.6: Velocity and control load vector for Method 1 using Model 1 in sway test. The velocity and control load are given in the body frame. The x-axes shows the time in seconds. The simulated responses are illustrated with a solid line, while the measured states are represented by a dotted line.

Results Method 2

The results using Method 2 are all from the same pure sway test presented in Method 1. This means that the control load vector $\boldsymbol{\tau}$ is the same for both methods, and is thus only shown in Figure 6.6. The pose and velocity from simulation Method 2 using Model 1 are shown in Figure 6.7.

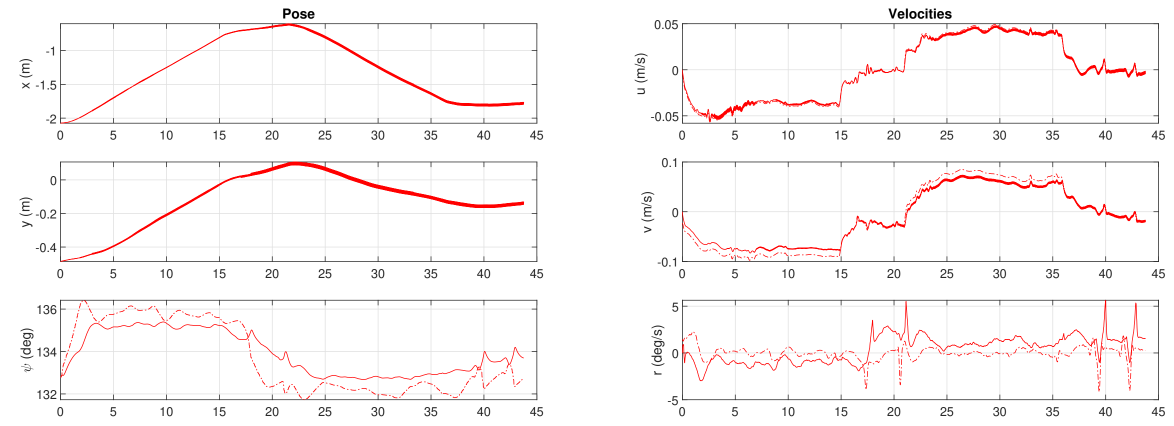

Figure 6.7: Pose and velocity for Method 2 using Model 1 in sway test. X- and y-position is given in the NED frame, while the velocity is given in the body frame. The x-axes shows the time in seconds. The simulated responses are illustrated with a solid line, while the measured states are represented by a dotted line.

The pose and velocity from simulation Method 2 using Model 2 are shown in Figure 6.8.

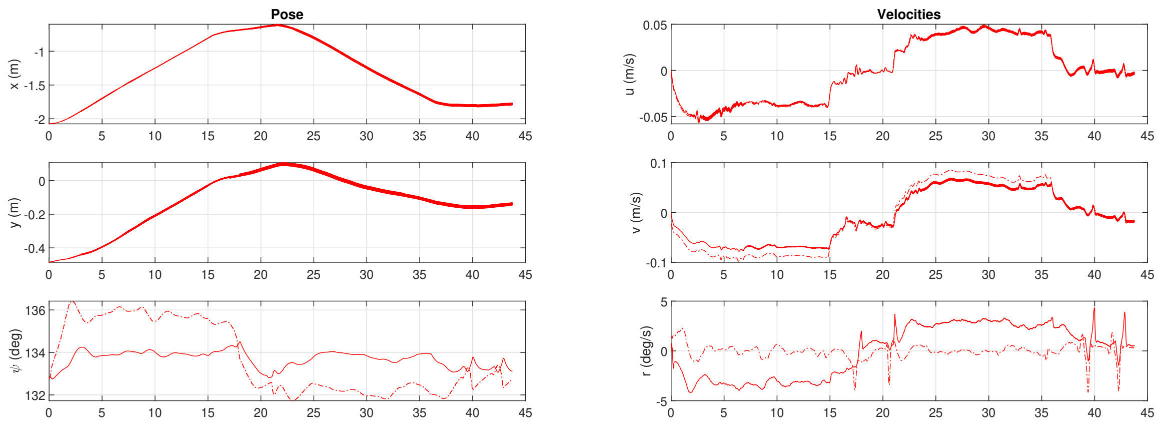

Figure 6.8: Pose and velocity for Method 2 using Model 2 in sway test. X- and y-position is given in the NED frame, while the velocity is given in the body frame. The x-axes shows the time in seconds. The simulated responses are illustrated with a solid line, while the measured states are represented by a dotted line.

The KPI and best damping parameter, $\mathbf{x}_{opt}$, obtained during the optimization are given in Table 6.6. The KPI stated is the total KPI, i.e., the total SAE for all the test, not just the sway test.

	Method 1	Method 2 + Model 1	Method 2 + Model 2
KPI	2.3×10^6	1.45×10^5	1.45×10^5
$\mathbf{x}_{opt}$	[20.0,6.0,-3.0,7.6]	[25.0,30.0,1.4,3.3]	[11.4,31.2,2.2,2.5,8.8, 86.8,-4.9,2.4,-5.0,-5.0,0.1]

Table 6.6: Optimization performance

The value of the KPI during the optimization for Method 2 using Model 1 is shown in Figure 6.9. Function value is the same as the KPI value.

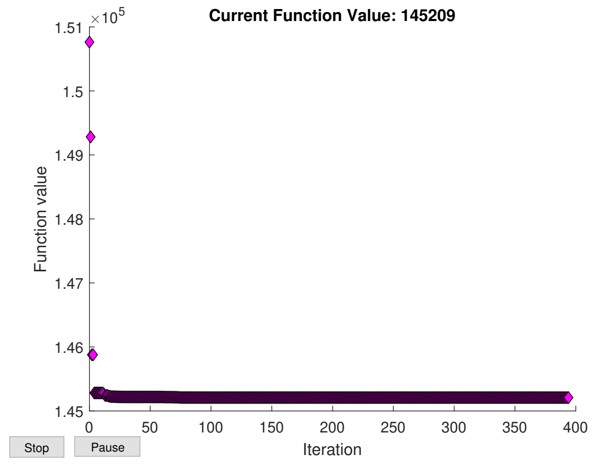

Figure 6.9: KPI value for each iteration of the optimization for Method 2 using Model 1.

6.4.6 Discussion of Damping

Method 1: Sway-Yaw Test

The responses from the sway-yaw test shown in Figure 6.3 and Figure 6.4 shows a good response in sway as the estimated velocity in sway follows closely the measured one. In surge, there is a small measured velocity around zero, while the estimated surge velocity is simply zero since the surge control force is zero. This is because the surge is uncoupled from sway and yaw in the model, and does not take into account that a force in sway or yaw can induce a motion in surge. For a more accurate model, this coupling should have been accounted for, but since the induced surge velocity is quite small in this case, it does not have a large impact on the performance. In yaw, the model is not able to estimate the heading rate accurately compared to the measured one. At the worst point, the discrepancy is approximately $8\,^\circ\,\mathrm{s}^{-1}$. This may be due to the fact that the auto-heading function was active during the whole test, impacting the control load in yaw such that the yaw velocity was hard to estimate correctly. The discrepancy in the heading rate leads to a large error in the heading estimate, which again leads to an error in the x- and y-position since these are transformed to the NED frame using the heading angle. The auto-heading function was turned on so that the drone would not drift in yaw during the tests, which was especially useful during the surge and sway related tests. A possible way to improve the performance of the model, could have been to turn off the auto-heading function during the tests when a control input was given in yaw, i.e. surge-yaw, sway-yaw, and yaw. This way, it may have been easier to estimate a correct heading rate based on the control moment in yaw.

Method 1: Sway Test - Discrepancy in Experimental Data

In the response from the pure sway test shown in Figure 6.5 and Figure 6.6, one can see that the simulated position and velocity in surge have opposite signs than the measured ones. Since both the velocity and the control load are given in the body frame, a positive control force in surge should give a positive velocity in surge and visa versa. Additionally, considering that the heading angle is approximately 135° for the first 15 seconds on the test, the positive control force in surge should give a negative motion in surge and sway. However, for this sway test, this was not the case. This is the opposite response of what was seen in the sway-yaw test, shown in Figure 6.3 and Figure 6.4, where the measured position and velocity have signs corresponding to the sign of the control load. The reason for this discrepancy remains unknown, but most likely stems from an error in gathering or processing the experimental data. This unexplained behavior leads to a large uncertainty in the results from the optimization for both Method 1 and 2, since the measured velocity is used as input in both methods.

Method 2

The response from Method 2 using Model 1 and Model 2, shown in Figure 6.7 and Figure 6.8 respectively, shows a good performance for both models. The simulations with the observer are able to estimate the pose and velocity with a small error from the measured ones for both models. The simple Luenberger observer functions well, even though the measurements have a large uncertainty. This indicates that an observer is able to estimate the states well, regardless of the accuracy of the model. Comparing the results from Method 2 using the different models, one can see that the performance is quite similar. The largest difference lays in the estimation of the heading angle and heading rate. Figure 6.7 shows that Model 1 has a slightly smaller estimation error than Model 2, seen in Figure 6.8, meaning the Model 1 performs better than Model 2. Assuming that the damping parameters are correctly identified, the elaborate model (Model 2) should better represent the real process and thus have better estimates than the simple model (Model 1). However, this is not the case for this sway test. This may be a coincidence for this exact optimization, or may be a consequence of that the optimization more easily finds a local optimum for Model 1 than Model 2, since the search space is significantly smaller using Model 1. For the simple model (Model 1), the search space in the optimization has 4 dimensions, while in the elaborate model (Model 2), the search space has 11 dimensions.

Optimization Performance

The development of the optimization performance indicated by the KPI for Method 2 using Model 1 shown in Figure 6.9, shows a stagnation in the decrease of the KPI only after a few iterations. This may be explained by looking at the optimal damping parameters ($\mathbf{x}_{opt}$) given in Table 6.6 and the upper and lower bounds given in Table 6.5. One can see that many of the optimal damping parameters are on the lower and upper bounds. This may indicate that a global optimum is outside the lower and upper bounds, or that the optimization is not able to find an optimum using the chosen models. Table 6.6 shows that the total KPI from all the tests (e.g. surge-sway, surge-yaw etc) is smaller for Method 2 than Method 1, indicating that the Method 2 performs better than Method 1, which corresponds with the observations from the sway test. Even though the results from Method 2 in the sway test using Model 1 where slightly better than using Model 2, the overall performance for all the tests, indicated by the KPI, is the same for Model 1 and 2.

6.4.7 Conclusion

The results from the sway test showed that Method 2 performed better than Method 1. Additionally, the simple model (Model 1) performed slightly better than the elaborate model (Model 2) in the sway test. The overall performance from all the test, indicated by the total KPI, showed similar performance for Model 1 and 2 using Method 2. With a similar performance for both models, the damping results from Method 2 using Model 1 will be used, since this has the simplest model. For Model 1, neglecting nonlinear damping and Coriolis-centripetal matrix, is a fair assumption for low speed, as their are both small for low speed. The chosen damping is given by

$$\mathbf{D}_L = \begin{bmatrix} 25 & 0 & 0 \\ 0 & 30 & 1.4 \\ 0 & 1.4 & 3.3 \end{bmatrix}. \tag{6.19}$$

It must be noted that this damping parameter is highly uncertain, due to the uncertainty in the measured data and the developed model. To improve the results, different models could have been implemented and tested. However, there exists no model that can replicate the real process exactly. The results are affected by the poor quality of the position data gathered from Qualisys. The missing data points in the Qualisys data have led to a large uncertainty in the interpolated position and filtered velocity vectors. To improve the results, more accurate data should be used. In the simulation, the velocity is found by integrating the acceleration, which can lead to a drift in the velocity. The position was found by integrating the velocity, which can lead to an even larger drift in position. This will increase the uncertainty of the estimated states.

Chapter 7

DP Observer

The DP observer is based on an observer design model (ODM) which is defined in the workspace of the drone (see Chapter 5). The ODM focuses on modelling measurement noise, failure situations including dead-reckoning capabilities, filtering and motion prediction (Fossen, 2011). The ODM was chosen as a linearized and discretized version of the state-space model given in Section 5.4, i.e. on the form

$$\mathbf{x}(k+1) = \mathcal{F}(\hat{\mathbf{x}}(k), \mathbf{u}(k)) + \mathbf{\Gamma}\mathbf{w}(k), \quad \mathbf{w}(k) \sim \mathcal{N}(0, \mathbf{Q}_d), \tag{7.1a}$$

$$\mathbf{y}(k) = \mathbf{H}\mathbf{x}(k) + \mathbf{v}(k), \qquad \mathbf{v}(k) \sim \mathcal{N}(0, \mathbf{R}_d). \tag{7.1b}$$

7.1 Observability

To see if the system is observable, some assumptions were made such that the state matrix $\mathbf{A}$ became linear. The Centripetal and Coriolis $\boldsymbol{C}(\boldsymbol{\nu})$ matrix was assumed negligible, the damping matrix $\boldsymbol{D}(\boldsymbol{\nu}))$ was assumed linear, and the transformation matrix $\boldsymbol{J}(\psi)$ was assumed equal to the identity matrix $\mathbf{I}$. With a linear $\mathbf{A}$ matrix, the observability matrix was computed by using the MATLAB function *obsv*$(\mathbf{A}, \mathbf{H})$. The observability matrix had rank 9, which means that a system with this $\mathbf{A}$ matrix had full rank and is thus observable (Chen, 2013).

7.2 EKF

For this thesis, an EKF was chosen as the observer and was implemented using (2.22). The EKF was discretized and linearized according to (2.23), where the method for deriving the partial derivative $\frac{\partial \mathbf{f}(x)}{\partial \mathbf{x}(k)}$ is described in Appendix A. The observer rate was chosen to be equal to the sampling rate of the fastest sensor, that is, 100Hz.

7.2.1 Tuning

The EKF was tuned by altering the discrete-time process noise covariance matrices $\mathbf{R}_d$ and $\mathbf{Q}_d$. The discrete-time measurement noise covariance matrix $\mathbf{R}_d$ was chosen taking the noise variance for each sensor given in Table 4.2 as a basis, i.e.,

$$\mathbf{R}_d = \text{diag}\{\sigma^2_{x_{aps}}, \sigma^2_{y_{aps}}, \sigma^2_{\psi}, \sigma^2_{r}\} = \text{diag}\{1 \times 10^{-5}, 1 \times 10^{-5}, 1 \times 10^{-6}, 1 \times 10^{-6}\} \tag{7.2}$$

where h is the sampling time, $\sigma^2_{i_{aps}}$, σ^2_{ψ}, σ^2_r are the noise variance for APS, heading and heading rate, respectively. Since the noise level of the APS measurements were not available, an approximate noise level was found by tuning the observer until a satisfactory result for filtered the position. The discrete-time process noise covariance matrix $\mathbf{Q}_d$ was found by trial and error and is given by

$$\mathbf{Q}_d = \text{diag}\{1 \times 10^3 , 1 \times 10^3 , 1 \times 10^3 \} \tag{7.3}$$

The process noise covariance matrix was chosen to be quite high in order to avoid fluctuating estimates. More on this in Section 7.4.3.

Before the raw measurements from the sensors could enter the observer and the rest of the control system, the measurements had to go through a signal processing module. In this thesis the signal processing module was designed as an integrated part of the observer.

7.3 Signal Processing Module

The signal processing module is used to detect and remove faulty signals, such that these will not propagate through the control system (Sørensen, 2013). In order to have a robust observer, it should be able to handle different update rates from the sensors, possible failure modes, and dead reckoning.

Transform Data The position measurements from the APS were received in the user frame. Before they were used in the observer, they had be transformed to the local NED frame. The position was first rotated 180° about the x-axis, such that the z-axis in the user frame would be pointing in the same direction as the z-axis of the NED frame. The APS computes the angle from the North to the y-axis in the acoustic frame using an internal IMU sensor. The angle from the North to the x-axis in the acoustic frame (which equals the user frame), i.e. α (see Figure 5.1), is found by adding 90° to the angle from North to the y-axis provided by the APS. Using this angle, the position data was rotated about the z-axis such that the data was aligned with the NED frame. In the control system, the measurements should be transformed to the CO, i.e., the reference point of the control system. The x and y position from the APS were not measured in CO, but at the transmitter placed on top of the drone, where the exact distance from CO in the body frame is found in Table 4.1. The offset from the CO to the transmitter was transformed to the NED frame and then subtracted from the position measurements in the NED frame. The heading and heading rate measurements are provided by the IMU sensor, which is not located in the CO either. Since the axes of the IMU are aligned with the principal axes of the body frame, the heading and heading rate measurements do not need to be corrected for misalignment.

Synchronization of Measurements The sensor that provides the measurements have different updates rates. The signal processing module synchronizes the different measurements such that the observer receives them all at the same rate, i.e. the observer rate. The sampling rates for all the sensors are given in Table 4.2. The heading and heading rate, gathered from the state estimator on the drone, have a sampling rate of 100 Hz, while the x and y position from the APS have a sampling rate of 2 Hz. With an observer rate of 100 Hz, new x and y position data will only be received for every 50th iteration in the observer. The APS will simply provide the previous measurement until a new measurement is available. In the observer, this is detected as a signal freeze. Handling signal freeze is described in Section 7.3.1.

7.3.1 Fault Detection and Handling

The signal processing module should check for signal range and variance, frozen signals, and outliers. Before the heading and heading rate enter the observer developed in this thesis, they have been processed by the observer in the integrated control system on the drone, i.e., its internal observer. The position data are also processed by an observer integrated in the APS, before they enter the observer developed in this thesis. Both internal observers filter the data and remove faulty signals. In this thesis, signal freeze was the only failure mode detected and handled. A frozen signal was detected by zero variance for *n* consecutive signal samples, where the number *n* was chosen to be 5. A signal freeze was handled by skipping the corrector step by setting the Kalman gain $\mathbf{K}$ to zero, as described in (Fossen, 2011).

Proposed Method

For this thesis, fault detection and fault-handling was only implemented for signal freeze, so a proposed approach for other relevant failure modes is described. Fault detection is vital to detect and remove faulty signals so that these signals will not propagate through the control system. As the different failure modes have been modeled in Section 2.10.1, they can be detected by considering which distribution fits the signal best. All these failure modes can be detected by detecting a change in mean or variance from the nominal values y_{real} and σ_0^2. A limitation of change detection is that "the design is a compromise between detecting true changes and avoiding false alarms" (Gustafsson and Gustafsson, 2000), hence, it is a crucial issue to separate between the two.

High Derivative After a frozen signal, a high derivative in the measurement may occur, as the new signal may differ significantly from the previously known signal. This may be falsely detected as an outlier or high noise. As proposed in Abrahamsen (2019), this can be handled by implementing a few seconds of cool-down after a signal freeze before fault detection is activated.

Bias and Drift Signal bias or drift can be detected by observing a change in the mean. This requires an estimate of y_{real} which may be done using redundant sensor information or a model (Mokleiv, 2017). The CUSUM algorithm can be used to detect a change in mean or variance. The CUSUM algorithm is explained in Appendix E.

Outliers As described in Section 2.10.1 can outliers be detected by checking if the z-score is larger than a certain number. Outlier rejection can also be done using a χ^2 test and an EKF as described in Lekkas et al. (2015). When an outlier is detected, the measurement is usually replaced by a calculated number, e.g. the mean value (Sørensen, 2013).

High Noise High noise detection can be detected by observing a change in variance, since high noise gives high variance (Sørensen, 2013). This can be detected using CUSUM.

Dead Reckoning with EKF When dead reckoning occurs, the observer should ignore the new measurements, and instead base the predictions on the model alone. In the EKF, this can be handled in the same manner as a signal freeze, i.e., by skipping the corrector step by setting the Kalman gain $\mathbf{K}$ to zero (Fossen, 2011).

7.4 Results From Open-Loop Tests

The drone was tested outside at a location where the river meets the ocean, i.e. in brackish water. At the test location, the water was calm, that is, the effects of currents and waves were low. At the test location, the APS always had GPS signals available. The observer was tested in open-loop with two different values of the process noise covariance $\mathbf{Q}_d$.

7.4.1 Observer With High Process Noise Covariance

This subsection shows the results from the observer tested in open-loop with a high process noise covariance, i.e.,

$$\mathbf{Q}_d = \text{diag}\{1 \times 10^3\,,1 \times 10^3\,,1 \times 10^3\,\}. \tag{7.4}$$

The estimated states corresponding to the measured states and the linear velocities are show in Figure 7.1. Figure 7.2 shows the estimated bias loads.

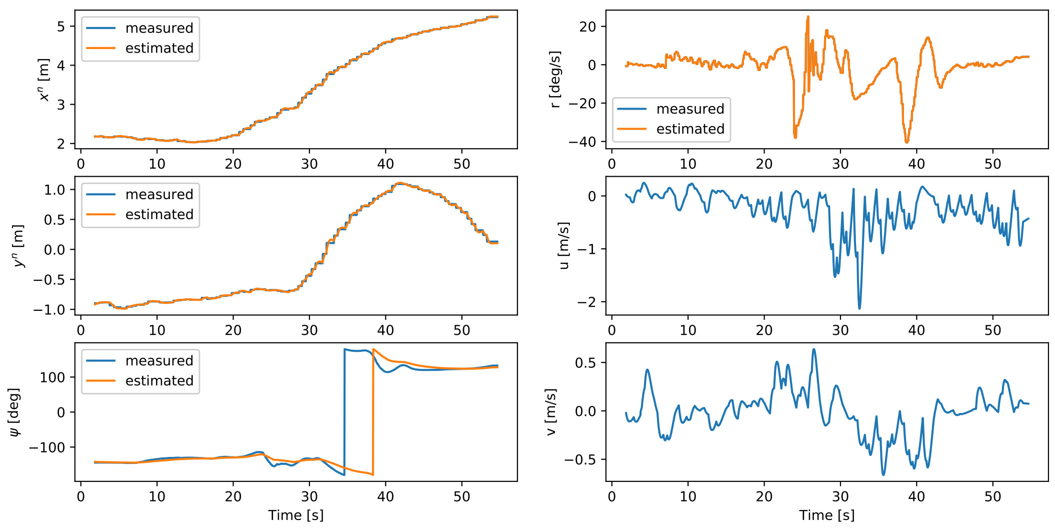

Figure 7.1: Estimated and measured x position, y position, heading angle and heading rate with estimated linear velocities in open-loop test using EKF with high process noise covariance.

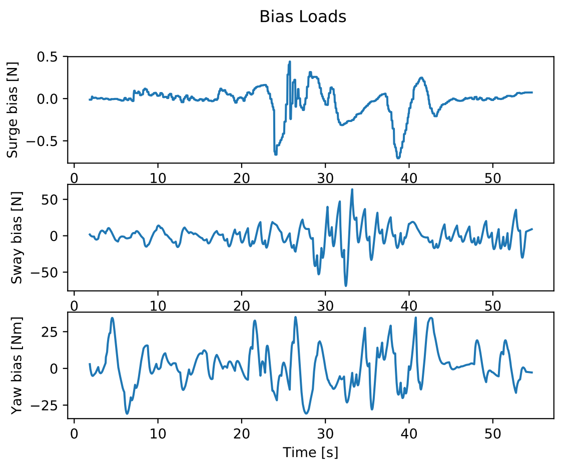

Figure 7.2: Bias loads in open-loop test using EKF with high process noise covariance.

7.4.2 Observer With Low Process Noise Covariance

The observer was also tested with a low process noise covariance, i.e.,

$$\mathbf{Q}_d = \text{diag}\{1\,,1\,,1\,\}. \tag{7.5}$$

The corresponding response is shown in Figure 7.3.

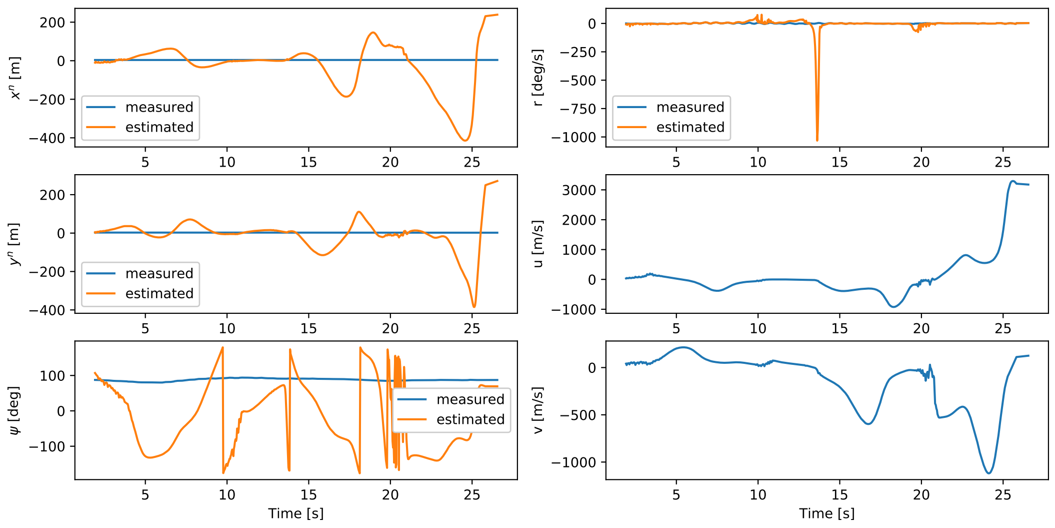

Figure 7.3: Estimated and measured x position, y position, heading angle and heading rate with estimated linear velocities using EKF with low process noise covariance.

7.4.3 Discussion

In Figure 7.1 one can see that the estimated states follow the measured states closely. This is due to the relatively large elements in the process noise covariance matrix, $\mathbf{Q}_d$, compared to the measurement noise covariance matrix, $\mathbf{R}_d$, telling the observer rely more on the measurements than the model. The EKF was able to estimate x and y positions, even though the sampling rate of the APS is much lower than the sampling rate of the observer. After 35 seconds, it may seem as if the estimated heading deviates largely from the measured one when the heading does a sudden change. This sudden change is due to the fact that the heading is contained between -180° to 180°. When the estimated heading is approximately -180° and the measured heading is approximately 180°, they are in fact quite close. Figure 7.1 shows that the estimated velocities are quite fluctuating. With only position and heading measurements available and a model with poor accuracy, the linear velocities are hard to estimate. Velocity or acceleration measurements could have improved these estimates. Figure 7.2 show that the bias loads are highly fluctuating, which means that there are unmodeled effects or poorly modeled effects in the model.

Figure 7.3 shows that the estimated states become highly fluctuating with a low value of $\mathbf{Q}_d$. The discrete-time process noise covariance depends on the sampling rate of the observer. The observer is ran at 100Hz, while the slowest sampling rate is 2Hz for the APS. This means that new position data from the APS are received only each 50th iteration in the observer. Between the new position measurements, the corrector step is omitted and the observer bases its estimates on the model alone. The results from the model identification in Section 6.4 showed that the accuracy of the model is poor. Since the ODM is derived from a continuous, nonlinear model, the ODM will additionally have both linearization errors and discretization errors that influence the estimates. With an inaccurate model, the state estimates that highly depend on measurements with a low sampling rate (e.g. position and velocity depending on APS), will be hard to estimate correctly between each new measurement. Each time a new measurement is available, i.e. when the Kalman gain is not set to zero, the observer is able to correct this estimation error, see Equation (2.22e). How the states estimates are corrected depends on the noise covariances $\mathbf{Q}_d$ and $\mathbf{R}_d$. When process noise covariance $\mathbf{Q}_d$ is relatively low, the observer will trust more in the model. At a high observer rate, this will lead to highly fluctuating estimates due to an inaccurate model.

To keep the state estimates of the observer from diverging at an observer rate of 100Hz, the $\mathbf{Q}_d$-matrix was given a high value such that it would trust more in the measurements than the model. The process noise covariance was chosen to be

$$\mathbf{Q}_d = \text{diag}\{1 \times 10^3\,, 1 \times 10^3\,, 1 \times 10^3\,\}. \tag{7.6}$$

In this thesis, the noise covariances were tuned directly in the discrete-time model, making them dependent on the observer rate. To make the noise covariances independent of the observer rate, they could have been tuned in the continuous-time model. The noise covariances would later be discretized using the observer rate. This way, the observer rate could have been changed without having to tune the noise covariance matrices once more. Having measurements from additional sensors, e.g. linear accelerations from the IMU, could have improved the position and linear velocity estimates when the APS measurements are not available. Integrating the noisy acceleration measurements from the IMU, would lead to a drift in the velocity and position estimates, but could still improve the observer estimates compared to only using the model alone. More propositions for improving the estimates are discussed in Section 11.2.

Chapter 8

Guidance System

The guidance system should compute the reference position, velocity, and possibly acceleration of the vehicle (Fossen, 2011). The reference states should be feasible for the vehicle and give a desired motion towards the desired states.

8.1 Design

As stated in Section 1.4, an external guidance system was developed such that the input can be given by a desired pose, i.e. x-, y-, z-position and heading angle, instead of joystick input based on the desired force in a DOF, which is the solution of the existing internal guidance system on the drone. In the external guidance system the input is passed to the drone using a PC with ROS installed to communicate with the drone. The external guidance system was used instead of the existing internal guidance system. Since the guidance system is replacing the existing internal guidance system on the drone, the guidance system has to generate reference position and velocity in heave such that the existing depth controller in the drone will work as intended. The guidance system developed for this thesis will hereafter be referred to as ”the” guidance system.

In the guidance system, the user can to specify the desired pose input in a augmented position vector, that is, $\boldsymbol{\eta}_d^u = [x_d^u, y_d^u, z_d^u, \psi_d]$ in the user frame. The input pose in the user frame was transformed to the local NED frame, as explained in Section 7.3. The desired heading was contained between $[-pi, pi)$. To generate the reference (desired) velocity $\boldsymbol{\nu}_d^n = [u_d^n, v_d^n, w_d^n, r_d]$, the target tracking strategy pure pursuit was employed. The reference linear velocities were computed using (2.24), restated here for clarity, i.e.,

$$\begin{bmatrix} u_d \\ v_d \end{bmatrix} = -U_{max}\frac{\tilde{\mathbf{p}}}{\sqrt{\tilde{\mathbf{p}}^\top\tilde{\mathbf{p}} + \Delta_p^2}}, \quad w_d = -w_{max}\frac{\tilde{z}}{\sqrt{\tilde{z}^2 + \Delta_z^2}}, \tag{8.1}$$

where $\mathbf{p} = [x, y]^\top$, $\tilde{\mathbf{p}} = \hat{\mathbf{p}} - \mathbf{p}_d$, $\tilde{z} = \hat{z} - z_d$, $U_{max} = \sqrt{u_{max}^2 + v_{max}^2}$ and w_{max} are the maximum approach speed in the horizontal plane and vertical axis respectively, and Δ_p and Δ_z are tuning parameters. u_{max} and v_{max} refers to the maximum surge and sway speed, respectively. Inspired by Breivik (2010, p 143), the heading rate was computed using

$$r_d = -r_{a,max}tanh(\frac{\Delta_\psi\tilde{\psi}}{r_{a,max}}), \tag{8.2}$$

where the heading error is defined as $\tilde{\psi} = \hat{\psi} - \psi_d$, $r_{a,max}$ is the maximum heading rate, and Δ_ψ is a tuning parameter that determines the shape of this approach. The chosen tuning parameters are given in Equation (10.1). Lastly, the desired velocity was transformed to the body frame using

$$\boldsymbol{\nu}_d^b = \mathbf{J}(\psi)^\top \boldsymbol{\nu}_d^n \quad \text{where} \quad \boldsymbol{J}(\boldsymbol{\psi}) = \begin{bmatrix} c(\psi) & -s(\psi) & 0 & 0 \\ s(\psi) & c(\psi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{8.3}$$

8.2 Implementation

The input and output flow to the guidance system is illustrated in Figure 8.1. The guidance system receives the desired pose $\boldsymbol{\eta}_d^u$ in the user frame and transforms it to the NED frame using the angle α from the APS. α is the angle between the user frame (i.e. the acoustic frame) and the NED frame, as specified in Section 5.2. The estimated x- and y-position, and heading are received from the external observer, while the estimated depth is received from the internal observer in the drone. The guidance system computes the reference velocities by using the error between the desired position and estimated position. The desired pose and velocity are sent to the controller inside the drone.

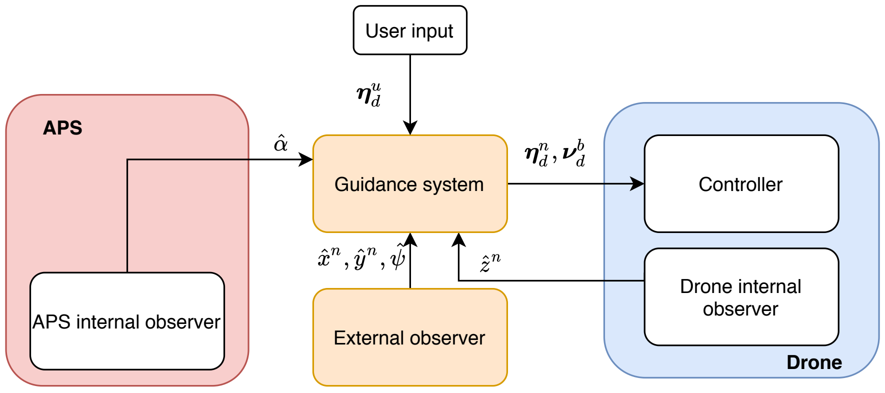

Figure 8.1: Input and output to the guidance system. The external observer refers to the EKF developed in Chapter 7.

The guidance system is ran at a rate of 100Hz, the same as the observer.Details regarding the coded implementation is given in Appendix B.2.2.

Chapter 9

Control System

The control system determines the necessary control load produced by the vehicle in order to satisfy the control objective (Fossen, 2011), i.e. go to the desired position using the reference velocity. This is done by developing a control law and a thrust allocation.

9.1 Design

As stated in Section 1.4, the scope of this projects includes making a control system that can control the drone in surge, sway, yaw, as there already exists functionalities for controlling heave motion. The control law for surge, sway and yaw will be implemented as an integrated part of the existing controller on the drone. The existing thrust allocation in the drone will be used as is. The control system uses a control design model (CDM) based the continues-time model defined in Section 5.4. Like the ODM, the CDM is defined in the workspace of the drone, i.e. surge, sway, and yaw. The control objective is to make $||\mathbf{h}_d - \mathbf{h}|| \rightarrow 0$ as $t \rightarrow \infty$, where $\mathbf{h} = [x, y, z, \psi, u, v, w, r]^\top$. A feedback PD controller is chosen as the control law, given by

$$\boldsymbol{\tau}^b = -\mathbf{K}_p \boldsymbol{J}^\top(\boldsymbol{\psi})(\hat{\boldsymbol{\eta}}^n - \boldsymbol{\eta}_d^n) - \mathbf{K}_d(\hat{\boldsymbol{\nu}}^b - \boldsymbol{\nu}_d^b), \tag{9.1}$$

where $\mathbf{K}_p$ and $\mathbf{K}_d$ are the controller gains. The subscript *d* on the state vectors represent the desired states. The existing heave controller used is a PID controller which accounts for the restoring force $g(\eta)$ in heave, given by

$$\tau^b_{heave} = -K_{p,depth}(\hat{z}^n - z^n_d) - K_{d,depth}(\hat{w}^b - w^b_d) - K_{i,depth}\int_0^{t'} z(t)dt + g(\eta). \tag{9.2}$$

The integral term is added to remove any steady-state error in the depth. When the drone is submerged, the buoyancy is slightly larger than the gravitational force, giving a positive buoyancy for the drone. The chosen controller gains are found by trial and error, given in Equation (10.2). In the existing depth controller, the reference depth is usually close to the measured depth. In the DP control system developed here, the distance between the measured and the reference depth can be larger. For the new control purpose, also the depth controller had to be tuned to get a satisfactory behaviour.

9.2 Implementation

The complete control system, including the observer and guidance system, is illustrated in Figure 9.1. This an extension of the guidance system illustrated in Figure 8.1, where the inputs and outputs to the observer and control system are included as well. The external observer, i.e. the EKF developed in Chapter 7, gets the "measured" position $\hat{x}^u$ and $\hat{y}^u$ in the user frame frame and the angle between the user frame and the NED frame $\hat{\alpha}$ as input from the APS. The external observer also receives the "measured" heading $\hat{\psi}$, heading rate $\hat{r}$, and control load $\boldsymbol{\tau}^b$ from the drone. The inputs are not true measurements, as they are estimates from the internal observers in the APS and the drone. The internal controller in the drone receives the desired pose $\boldsymbol{\eta}_d^u$ and velocity $\boldsymbol{\nu}_d^u$ from the guidance system. The estimates states in the workspace (surge, sway, yaw), i.e. $\hat{\boldsymbol{\eta}}^n$, $\hat{\boldsymbol{\nu}}^n$, and $\hat{\boldsymbol{b}}^n$, are received from the external observer, while the estimated depth $\hat{z}^n$ and depth rate $\hat{w}^n$ are received from the internal observer on the drone. Using these inputs, the controller computes the control load $\boldsymbol{\tau}^b$ that is passed to the thrust allocation. The thrust allocation the individual force of each thruster using the control load.

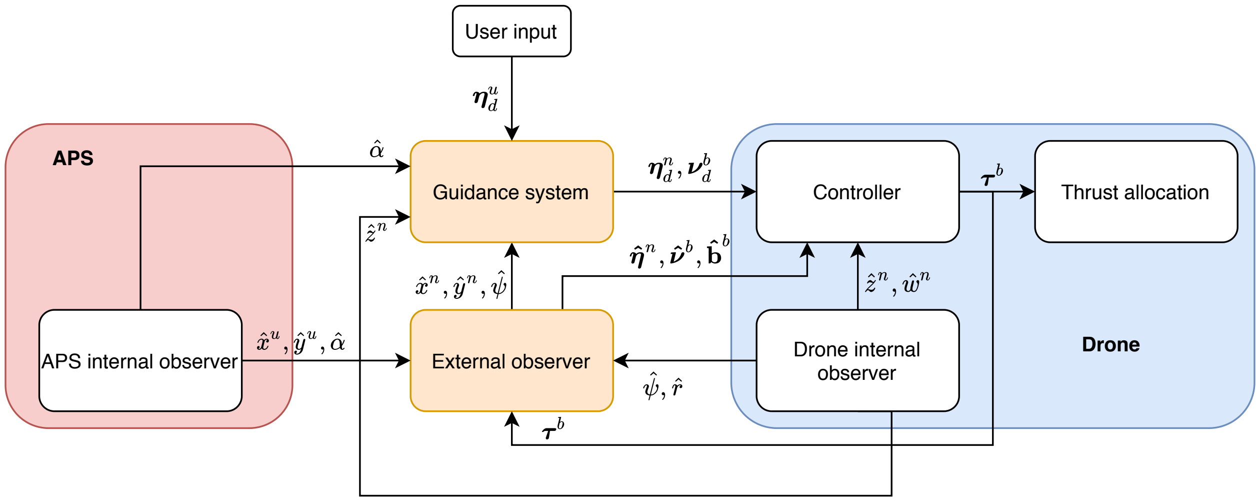

Figure 9.1: Implemented control system with all components. The external observer refers to the EKF developed in Chapter 7.

Details on the implementation and running the control system is explained in Appendix B.2.3.

Chapter 10

Results and Analysis

10.1 Results From Closed-loop

This section shows the results from tests with the whole control system, i.e. in closed-loop. The drone was tested at the same test location as for the open-loop test, i.e outside at a place where the river meets the ocean with calm water. The results in Figures 10.1 to 10.7 shows the results from a test with the controller activated for surge, sway, heave and yaw. The reference generator tuning parameters for the guidance system are given by

$$\Delta_p = 10, \quad \Delta_z = 10, \quad \Delta_\psi = 0.5, \tag{10.1a}$$
$$U_{max} = 1.27\,\mathrm{m\,s^{-1}}, \quad w_{max} = 0.9\,\mathrm{m\,s^{-1}}, \quad r_{a,max} = 15\,^\circ\,\mathrm{s^{-1}}. \tag{10.1b}$$

The maximum velocities in the reference generator are chosen to be lower than the believed maximum speed in order to ensure that the drone can follow the reference and lead to a calm motion of the drone. The control law gains used in this test are given by

$$\mathbf{K}_p = \mathrm{diag}\{2, 2, 0.3\} \quad \mathbf{K}_p = \mathrm{diag}\{0.1, 0.1, 0.3\} \tag{10.2a}$$
$$K_{p,depth} = 10, \quad K_{i,depth} = 0.01, \quad K_{d,depth} = 0.5 \quad . \tag{10.2b}$$

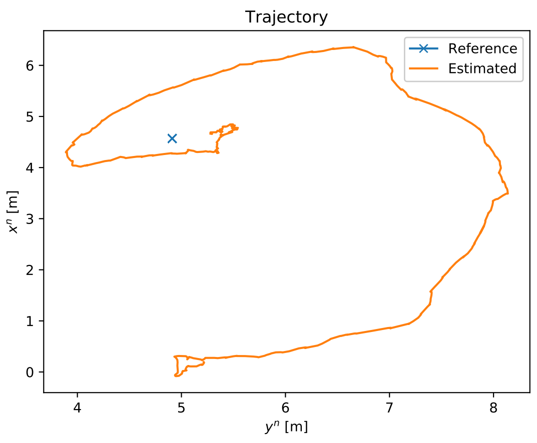

Figure 10.1: Horizontal trajectory of the drone.

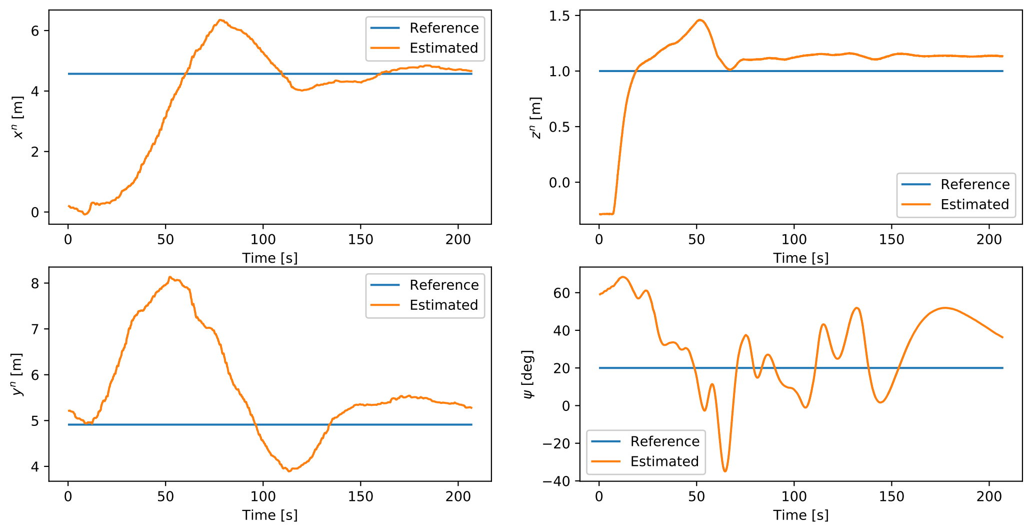

Figure 10.2: Reference and estimated pose in the closed-loop test.

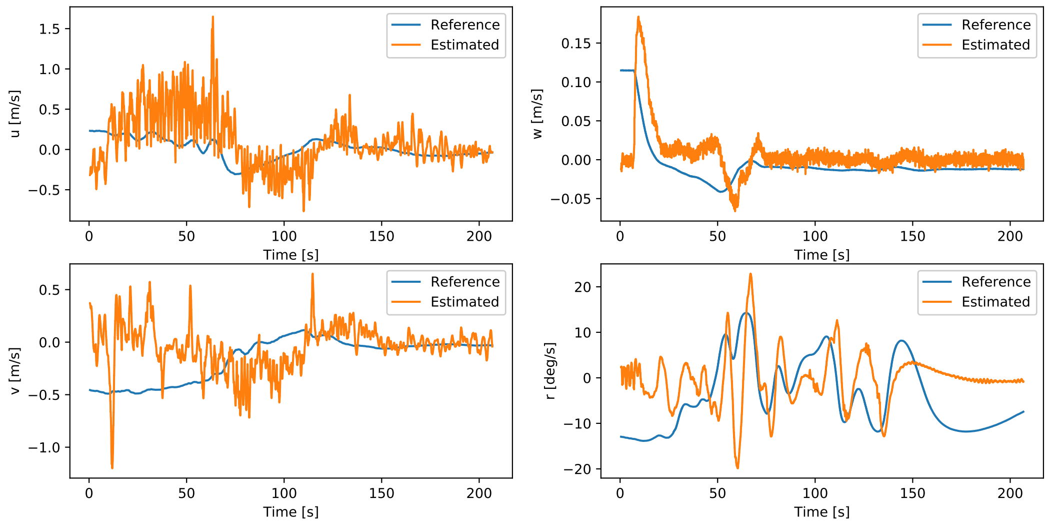

Figure 10.3: Reference and estimated velocities in the closed-loop test.

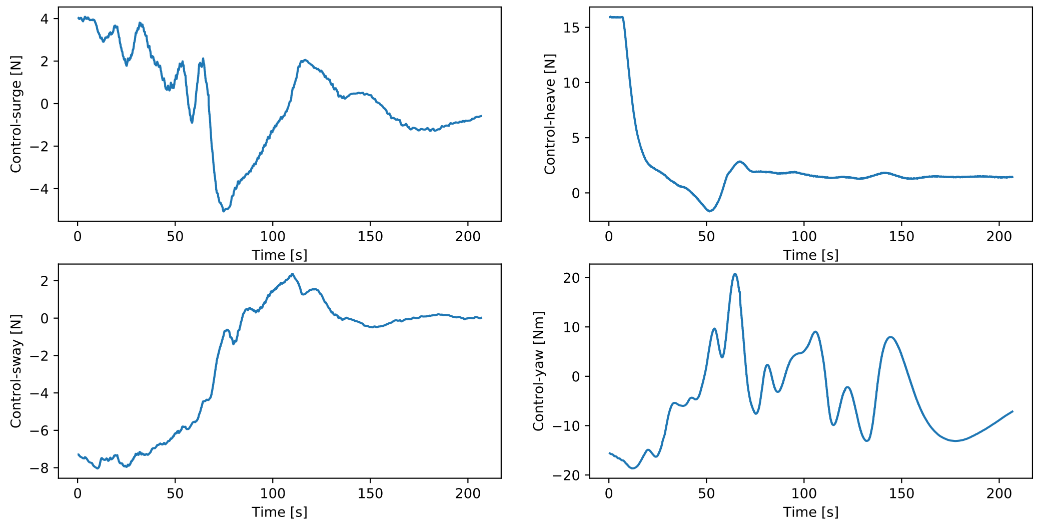

Figure 10.4: Control load from the closed-loop test.

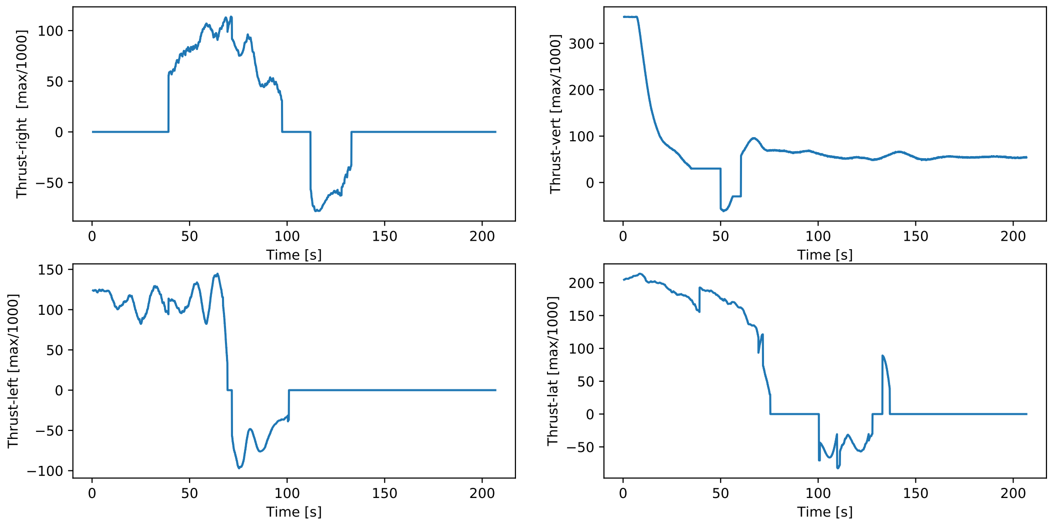

Figure 10.5: Thrust setpoints for each thruster from the closed-loop test.

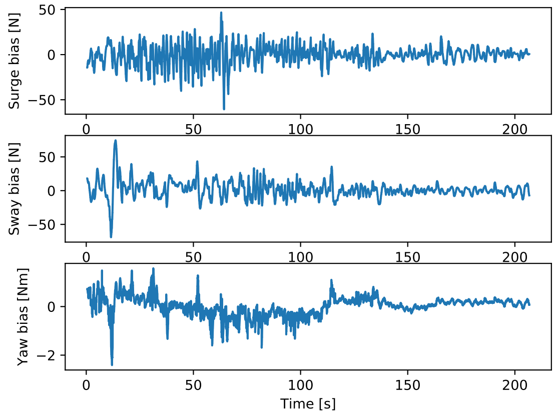

Figure 10.6: Bias loads from the closed-loop test.

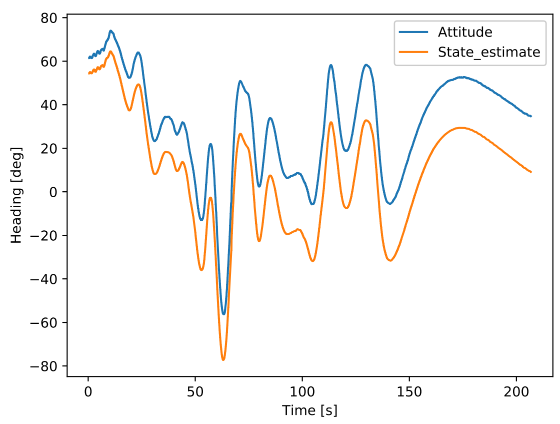

Figure 10.7: Heading estimates from the rostopics $observer/attitude$ and $observer/state\_estimates$ from the drone in the closed-loop test.

10.1.1 Discussion

Figure 10.2 shows that the estimated position converges towards the desired position with an accuracy of less than 1m. The heading estimate does not converge, but has large oscillations around the desired angle. Through excessive tuning of the heading controller, the convergence of the estimated heading towards the desired one can be improved. However, due to limited time for testing and tuning, this was not achieve in this project. For the surge, sway, and heave controller one can see that it takes a long time to reach the desired position. Additionally, Figure 10.1 shows that the drone does not take the shortest path towards the desired position, but makes a large detour. This indicates that all the controllers should be tuned further for improved accuracy and avoiding large detours. Figure 10.5 shows that the thrusters are run at 10-30 % of their maximum force, indicating the thrust force available is not a limiting factor for the control system given the current tuning parameters.

In Figure 10.3 one can see that the chosen reference generator parameters given in Equation (10.1) gave smooth reference velocities. This was done to avoid large transients in the reference velocity when the distance between the current pose and actual pose is large, e.g. at the beginning, and to avoid oscillations around the desired pose when drone is close to the desired pose. This ensured a slow motion towards the desired position and made it easier to control the drone. Figure 10.3 shows that the velocity estimates from the observer are quite fluctuating, similar to was seen in the open-loop test in Section 7.4. Since the velocity estimates are used in the feedback controllers, they influence the performance of the control system. Improvement of the velocity estimate is therefore desirable and might be crucial in order to improve the performance of the control system. This is further discussed in Section 11.2.

From figure Figure 10.6, one can see that the bias in surge, sway and yaw are highly fluctuating. Looking at the control load in Figure 10.4, one can see that the bias loads in surge and sway are considerably larger than the control loads in surge and sway. This means that the observer design model uses the bias actively to explain discrepancies between its predictions and the measurements. This is further evidence that the observer design model is inaccurate and does not represent the physics of the real drone.

Different Heading Estimates

Figure 10.7 shows that the estimated heading angle from the drone's internal observer for the estimates published at the rostopics $observer/attitude$ and $observer/state\_estimates$ differs up to 20° during the test. Since there are only two measurements, one can not determine which one is the most accurate measurement, as sensor voting would require a third reference. As described in Section 3.4.1, the heading from $observer/attitude$ is computed using the digital compass and the integrated gyro readings, while the heading from $observer/state\_estimates$ is computed using the integrated gyro alone. The latter method of estimating heading is prone to uncertainty due to the need to integrate heading rates, while the former has additional uncertainties due to possible magnetic disturbances. In this thesis, the heading estimate from $observer/attitude$ was used in the control system, while in the original control system for the drone, the heading estimate from $observer/state\_estimates$ is used. When there is no or low magnetic disturbance, the heading from $observer/attitude$ is more accurate than the other. As described in Section 4.1.2, the drone is often operated in areas close to ferrous metals, e.g. ship hulls or pipes, where there is a lot of magnetic disturbance. For this reason, Blueye only takes use of the gyro based heading, i.e. from $observer/state\_estimates$, in their control system. The heading angle is of high importance for the control system, since it determines the motion in the horizontal plane and the transformation between the NED and body frame. For instance the control forces in surge and sway are computed using the position errors in NED transformed to the body frame, see Equation (9.1). To have the most accurate heading estimates, the heading from the topic $observer/attitude$ was used in the control system developed in this thesis. For this to be true, the operational area for the DP system was assumed to have few magnetic disturbances. A possible improvement to this issue of which heading estimate to use is proposed in Section 11.2.

Chapter 11

Conclusion and Recommendations

11.1 Conclusion

A DP control system has been developed, including making a model of the drone, an observer, a guidance system, and control laws. The model identified in Chapter 6 gave poor results in terms of representing the real process when using DFO on experimental data from the MC-lab. This had some impact on the observer, as the process noise covariance had to be set relatively high such that the observer relied more on the measurements than the model, in order to avoid fluctuating estimates. This led to states with high correspondence to the measurements. For the states with no direct measurements, e.g. the linear velocity estimates, the estimates were volatile and fluctuating. The observer was developed to handle asynchronous measurements and to detect and handle signal freeze. The surge, sway and depth controllers were able to reach the desired positions within an accuracy of less than 1m. The drone did not follow the shortest path to the desired position, but made a large detour before arriving. The heading controller developed did not give a satisfactory behaviour, as it had large oscillations around the desired heading. Additionally, due to relying on the magnetometer and the gyroscope, the heading estimate of the observer is prone to magnetic disturbance which varies with the area of operation. The performance of the control system depends heavily on the accuracy of the heading estimates and the performance of the heading controller, and the improvement of these is therefore of significant importance for the overall control system.

11.2 Recommendations

Sensor Data

For improved accuracy in the DP control system, more sensors could have been used. For example a light weight DVL used to estimate the velocity could have improved the accuracy in both velocity and position when the drone is close to an object or the sea bottom. In this thesis, not all the sensor data available has been used, e.g. the acceleration from the IMU. By integrating the acceleration, one can get estimates of the velocity and position, which is especially useful when the APS measurements are not available. Since the sampling rate of the APS is much lower than the IMU and the chosen observer rate, this could improve the position and velocity estimates in the observer for the iterations when the APS measurements are not available. Another method to improve the state estimates in the observer, is to include a smoothing filter for step changes due to the slow APS updates, as described in (Ren et al., 2019). Both the Locator in the APS and the drone are equipped with a pressure sensor. For simplicity, only the depth estimate from the pressure sensor in the drone was used in this project. In order to obtain a redundant depth estimate, both these measurements should be used in the control system.

Heading Estimates

As discussed in Section 10.1.1, two possible heading estimates were relevant for use in the control system; (1) based on the digital compass and the integrated gyroscope reading ($observer/attitude$), and (2) based on the integrated gyroscope reading alone ($observer/state\_estimate$). In this thesis, heading estimate number (1) was used, while in the original control system of the drone heading estimate number (2) is applied. The second one is useful when the drone experiences magnetic disturbances, as the heading estimate from digital compass will deviate. Since the heading angle is of high importance in the control system, the most accurate heading available at each time step should be used. An easy solution would be to use a more accurate sensor, e.g. a gyrocompass. However, an accurate gyrocompass typically comes at a size and cost which is not compatible with the drone. A more feasible solution could be to implemented a switch between heading alternative (1) and (2), where alternative (1) is used when the drone is far from objects that can cause magnetic disturbances, while alternative (2) is used when close to magnetic disturbances, e.g. ship hulls.

GUI and Guidance System

The guidance system in this thesis only allowed for desired setpoints in the defined user frame. This could easily be extended to the local and global NED frame, for extended functionality. Another functionality that could be useful, is that the user can define its own user frame, e.g. by specifying the orientation of the user relative to North. As of now, the user has to input the desired setpoints in a script on the computer. In the future, this should be integrated in the mobile application that is usually used to control the drone. Additionally, a combination of joystick input and automatic control should be implemented for the surge and sway controller, which is available for the yaw and heading controller. For more advanced use, the guidance system can be developed to do path following instead of just pure pursuit, which is the present solution.

General

To improve the performance of the DP control system, more tuning of the control laws should be performed, in particular for the heading controller. Additionally, the model should be improved to better represent the real system, as discussed in Section 6.4.7. To avoid the issues of making an accurate model, a sensor-based observer, described in Section 2.9.1, could have been implemented instead of the model-based observer. To see if the observer is consistent, i.e. ”convergence of the estimate to the true value” (Bar-Shalom et al., 2002), a filter consistency test should be performed. For improved control in the low speed regime, the slow function discussed in Section 3.5 could be applied. The control system developed in this thesis has only been tested in calm environments. For a more robust control system, it should be tested under harsher circumstances, e.g. with waves and current. In this thesis, motion in roll and pitch have been neglected. The drone is underactuated with regards to these DOFs. An interesting topic would be to study how one could stabilize a vehicle in its underactuated DOFs.

Bibliography

Abrahamsen, B., 2019. Feil-tolerant dynamisk posisjonering for den autonome testplattformen ReVolt. NTNU. URL: `http://hdl.handle.net/11250/2622939`.

Alfsen, J.N., 2019. DP control system calibration using derivative-free optimization.

Audet, C., 2017. Derivative-free and blackbox optimization Tex.address: Cham tex.edition: 1st ed. 2017. tex.isbn: 3-319-68913-4 tex.publisher: Springer International Publishing : Imprint: Springer.

Bai, Q., Bai, Y., 2014. 21 - Subsea Survey and Positioning, in: Bai, Q., Bai, Y. (Eds.), Subsea Pipeline Design, Analysis, and Installation. Gulf Professional Publishing, Boston, pp. 487 – 509. URL: `http://www.sciencedirect.com/science/article/pii/B9780123868886000213`, doi:`10.1016/B978-0-12-386888-6.00021-3`.

Bar-Shalom, Y., Kirubarajan, T., Li, X.R., 2002. Estimation with Applications to Tracking and Navigation. John Wiley & Sons, Inc., New York, NY, USA.

Bellingmo, P.R., 2019. Fault-tolerant DP Observer Design for Blueye Pioneer.

Blanke, M., K.M.L.J.S.M., 2016. Diagnosis and Fault-Tolerant Control. 3rd ed., Springer.

Blueye, . Pioneer technical specifications. URL: `http://support.blueye.no/hc/en-us/articles/360006036673-Pioneer-technical-specifications`.

Breivik, M., 2010. Topics in guided motion control of marine vehicles URL: `http://hdl.handle.net/11250/259494`. publisher: Tapir Uttrykk tex.isbn: 978-82-471-2084-2 (printed ver.).

Brodtkorb, A.H., Teel, A.R., Sørensen, A.J., 2015. Sensor-based hybrid observer for dynamic positioning, in: 2015 54th IEEE Conference on Decision and Control (CDC), pp. 948–953. doi:`10.1109/CDC.2015.7401995`.

Brown, R.G., Hwang, P.Y.C., 2012. Introduction to random signals and applied Kalman filtering : with MATLAB exercises. 4th ed. ed., Wiley, Hoboken, N.J.

Candeloro, M., Sørensen, A.J., Longhi, S., Dukan, F., 2012. Observers for Dynamic Positioning of ROVs with Experimental Results. IFAC Proceedings Volumes 45, 85 – 90. URL: `http://www.sciencedirect.com/science/article/pii/S1474667016312095`, doi:`https://doi.org/10.3182/20120919-3-IT-2046.00015`.

Chen, C.T., 2013. Linear system theory and design. The Oxford series in electrical and computer engineering. international 4th ed. ed., Oxford University Press, New York.

Christ, R., Wernli, R., 2014. The ROV Manual, Second Edition: A User Guide for Remotely Operated Vehicles. Ocean News & Technology 20, 76. URL: http://search.proquest.com/docview/1490693020/.

DNV, 2010. Rules for Classication of SHIPS Newbuildings, Special Equipment and Systems.

Dukan, F., 2014. ROV motion control systems. volume 2014:295 of *Doktoravhandlinger ved NTNU (trykt utg.)*. Norwegian University of Science and Technology, Faculty of Engineering Science and Technology, Department of Marine Technology, Trondheim.

Dukan, F., Ludvigsen, M., Sørensen, A.J., 2011. Dynamic positioning system for a small size ROV with experimental results, in: OCEANS 2011 IEEE - Spain, pp. 1–10. doi:10.1109/Oceans-Spain.2011.6003399.

Dukan, F., Sørensen, A.J., 2013. Integration Filter for APS, DVL, IMU and Pressure Gauge for Underwater Vehicles. IFAC Proceedings Volumes 46, 280 – 285. URL: http://www.sciencedirect.com/science/article/pii/S1474667016461719, doi:https://doi.org/10.3182/20130918-4-JP-3022.00039.

Eidsvik, O.A., 2015. Identification of hydrodynamic parameters for remotely operated vehicles URL: http://hdl.handle.net/11250/2350869. tex.publisher: NTNU.

Fernandes, D.d.A., Sørensen, A.J., Pettersen, K.Y., Donha, D.C., 2015. Output feedback motion control system for observation class ROVs based on a high-gain state observer: Theoretical and experimental results. Control Engineering Practice 39, 90–102. URL: http://www.sciencedirect.com/science/article/pii/S0967066114002780, doi:10.1016/j.conengprac.2014.12.005.

Fossen, T., Strand, J., 1999. Passive nonlinear observer design for ships using Lyapunov methods: Full-scale experiments with a supply vessel. Automatica 35, 3–16.

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

Ghadrdan, M., Grimholt, C., Skogestad, S., Halvorsen, I.J., 2012. Estimation of Primary Variables from Combination of Secondary Measurements: Comparison of Alternative Methods for Monitoring and Control, in: Karimi, I.A., Srinivasan, R. (Eds.), Computer Aided Chemical Engineering. Elsevier. volume 31 of *11 International Symposium on Process Systems Engineering*, pp. 925–929. URL: http://www.sciencedirect.com/science/article/pii/B978044459506550016X, doi:10.1016/B978-0-444-59506-5.50016-X.

Gustafsson, F., Gustafsson, F., 2000. Adaptive filtering and change detection. volume 1. Citeseer.

Hassani, V., Sørensen, A.J., Pascoal, A.M., 2013. A novel methodology for robust dynamic positioning of marine vessels: Theory and experiments, in: 2013 American Control Conference, IEEE. pp. 560–565.

Kvalberg, J.L., 2019. Monocular visual odometry for a mini ROV URL: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2622900. tex.publisher: NTNU.

Lagarias, J.C., Reeds, J.A., Wright, M.H., Wright, P.E., 1998. Convergence properties of the Nelder–Mead simplex method in low dimensions. SIAM Journal on Optimization 9, 112–147. Tex.publisher: Society for Industrial and Applied Mathematics.

Lakshmanan, A., Gahlawat, A., Hovakimyan, N., 2020. Safe feedback motion planning: A contraction theory and (L)-Adaptive control based approach. arXiv.org URL: `http://search.proquest.com/docview/2385861337/`. publisher: Cornell University Library, arXiv.org tex.address: Ithaca.

Lefebvre, T., Bruyninckx, H., De Schutter, J., 2004. Kalman filters for nonlinear systems: a comparison of performance. International Journal of Control URL: `https://lirias.kuleuven.be/handle/123456789/158494`.

Lekkas, A.M., Candeloro, M., Schjølberg, I., 2015. Outlier Rejection in Underwater Acoustic Position Measurements Based on Prediction Errors. IFAC-PapersOnLine 48, 82 – 87. URL: `http://www.sciencedirect.com/science/article/pii/S2405896315002530`, doi:`https://doi.org/10.1016/j.ifacol.2015.06.014`.

Ludvigsen, M., 2019. TMR4120 Underwater Engineering.

Ludvigsen, M., Sørensen, A.J., 2016. Towards integrated autonomous underwater operations for ocean mapping and monitoring. Annual Reviews in Control 42, 145 – 157. URL: `http://www.sciencedirect.com/science/article/pii/S1367578816300256`, doi:`https://doi.org/10.1016/j.arcontrol.2016.09.013`.

Løvås, H.S., 2019. DP autotuning by use of derivative-free optimization. URL: `http://hdl.handle.net/11250/2624699`.

Mahony, R., Hamel, T., Pflimlin, J.M., 2008. Nonlinear Complementary Filters on the Special Orthogonal Group. IEEE Transactions on Automatic Control 53, 1203–1218.

Mallios, A., Romagós, D., Ridao, P., 2009. Localization Advances in the Unstructured Underwater Environment., pp. 111 – 116.

Milne, P., 1983. Underwater acoustic positioning systems.

Mokleiv, B., 2017. Fault-tolerant observer design URL: `http://hdl.handle.net/11250/2563993`. tex.publisher: NTNU.

Nelder, J., Mead, R., 1965. A simplex method for function minimization. Computer J. 7, 308–313.

Ren, Z., Skjetne, R., Jiang, Z., Gao, Z., Verma, A.S., 2019. Integrated GNSS/IMU hub motion estimator for offshore wind turbine blade installation. Mechanical Systems and Signal Processing 123, 222–243. URL: `http://www.sciencedirect.com/science/article/pii/S088832701930007X`, doi:`10.1016/j.ymssp.2019.01.008`.

Sandøy, S.S., 2016. System Identification and State Estimation for ROV uDrone. NTNU. URL: `http://hdl.handle.net/11250/2409503`.

Scheide, A.W., 2016. Design and Analyze of a Pressure Vessel for an Underwater Remotely Operated Vehicle Produced by Injection Molding. NTNU. URL: `http://hdl.handle.net/11250/2410765`.

Skjetne, R., 2005. The maneuvering problem. PhD Thesis. Norwegian University of Science and Technology, Faculty of Information Technology, Mathematics, and Electrical Engineering, Department of Engineering Cybernetics. Trondheim.

SNAME, 1950. The Society of Naval Architects and Marine Engineers.Nomenclature for Treating the Motion of a Submerged Body Through a Fluid. Technical and Research Bulletin No. 1–5.

Sola, J., 2017. Quaternion kinematics for the error-state KF .

Sørensen, A., 2013. Marine Control Systems. Propulsion and Motion Control of Ships and Ocean Structures.

Tannuri, E.A., Morishita, H.M., 2006. Experimental and numerical evaluation of a typical dynamic positioning system. Applied Ocean Research 28, 133 – 146. URL: `http://www.sciencedirect.com/science/article/pii/S0141118706000587`, doi:`https://doi.org/10.1016/j.apor.2006.05.005`.

Turtschi, A., 2002. C# .NET Web Developer's Guide. Elsevier. URL: `https://linkinghub.elsevier.com/retrieve/pii/B9781928994503X50008`, doi:`10.1016/B978-1-928994-50-3.X5000-8`.

Udjus, G., 2017. Force Field Identification and Positioning Control of an Autonomous Vessel using Inertial Measurement Units URL: `https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2456115`. accepted: 2017-09-21T14:01:21Z Publisher: NTNU.

Vik, B., Fossen, T., 2001. Nonlinear observer design for integration of GPS and inertial navigation systems, in: Proc. of the 40th IEEE Conf. on Decision and Control. Orlando, FL.

Værnø, S.A., Brodtkorb, A.H., Skjetne, R., Calabrò, V., 2017. Time-Varying Model-Based Observer for Marine Surface Vessels in Dynamic Positioning. IEEE Access 5, 14787–14796. URL: `https://ieeexplore.ieee.org/document/7993007`, doi:`10.1109/ACCESS.2017.2731998`. publisher: IEEE.

Værnø, S.A., Skjetne, R., Kjerstad, K., Calabrò, V., 2019. Comparison of control design models and observers for dynamic positioning of surface vessels. Control Engineering Practice 85, 235–245. URL: `http://www.sciencedirect.com/science/article/pii/S0967066119300085`, doi:`10.1016/j.conengprac.2019.01.015`.

Water Linked, . API - Documentation. URL: `https://waterlinked.github.io/docs/explorer-kit/gui/api/`.

Zhao, B., Blanke, M., Skjetne, R., 2012. Fault Tolerant ROV Navigation System Based on Particle Filter using Hydro-acoustic Position and Doppler Velocity Measurements. IFAC Proceedings Volumes 45, 280 – 286. URL: `http://www.sciencedirect.com/science/article/pii/S1474667016312423`, doi:`https://doi.org/10.3182/20120919-3-IT-2046.00048`.

Zhao, B., Skjetne, R., Blanke, M., Dukan, F., 2014. Particle Filter for Fault Diagnosis and Ropbust Navigation of Underwater Robot, IEEE. URL: `http://hdl.handle.net/11250/2393363`.

Appendix A

Partial derivative of nonlinear term in state space model

This chapter explains how the partial derivative of the nonlinear term, i.e. $\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \boldsymbol{x}}$, in the states space equation (5.4b) is derived in an analytic manner. The nonlinear term of the state-space model is given by

$$\mathbf{f}(\mathbf{x}) = \mathbf{A}(\mathbf{x})\mathbf{x} = \begin{bmatrix} \mathbf{f}_1(\mathbf{x}) \\ \mathbf{f}_2(\mathbf{x}) \\ \mathbf{f}_3(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}(\boldsymbol{\psi})\boldsymbol{\nu} \\ -\mathbf{M}^{-1}(\boldsymbol{C}(\boldsymbol{\nu}) + \boldsymbol{D}(\boldsymbol{\nu}))\boldsymbol{\nu} + \mathbf{M}^{-1}\mathbf{b} \\ \mathbf{0} \end{bmatrix} \tag{A.1}$$

where the $\mathbf{A}$ and **x** matrices are given in eq. (5.6a) and eq. (5.5) respectively. In the EKF, you need to find the partial derivative of $\mathbf{f}(\mathbf{x})$ with regards to the state vector **x**, i.e. $\frac{\partial \mathbf{f}(x)}{\partial \mathbf{x}(k)}$. To find the partial derivative, each elements of the function **f** is computed. Firstly, the elements are rewritten as

$$\mathbf{f}_1(\mathbf{x}) = \boldsymbol{J}(\boldsymbol{\eta})\boldsymbol{\nu} = \begin{bmatrix} c(\psi)u - s(\psi)v \\ s(\psi)u + c(\psi)v \\ r \end{bmatrix} \tag{A.2a}$$

$$\mathbf{f}_2(\mathbf{x}) = -\mathbf{M}^{-1}\boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} - \mathbf{M}^{-1}\boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{M}^{-1}\mathbf{b} := \mathbf{f}_{2_C} + \mathbf{f}_{2_D} + \mathbf{f}_{2_b} \tag{A.2b}$$

$$\mathbf{f}_3(\mathbf{x}) = \mathbf{0} \tag{A.2c}$$

$c(\cdot)$ and $s(\cdot)$ are abbreviations for the cosine and sine functions. Then each element in the vector function **f(x)** is partially derivated w.r.t each element in the state vector **x**, i.e.,

$$\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial \mathbf{f}_1}{\partial \boldsymbol{\eta}} & \frac{\partial \mathbf{f}_1}{\partial \boldsymbol{\nu}} & \frac{\partial \mathbf{f}_1}{\partial \boldsymbol{b}} \\ \frac{\partial \mathbf{f}_2}{\partial \boldsymbol{\eta}} & \frac{\partial \mathbf{f}_2}{\partial \boldsymbol{\nu}} & \frac{\partial \mathbf{f}_2}{\partial \boldsymbol{b}} \\ \frac{\partial \mathbf{f}_3}{\partial \boldsymbol{\eta}} & \frac{\partial \mathbf{f}_3}{\partial \boldsymbol{\nu}} & \frac{\partial \mathbf{f}_3}{\partial \boldsymbol{b}} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathbf{f}_1}{\partial \boldsymbol{\eta}} & \frac{\partial \mathbf{f}_1}{\partial \boldsymbol{\nu}} & \mathbf{0} \\ \mathbf{0} & \frac{\partial \mathbf{f}_{2_C}}{\partial \boldsymbol{\nu}} + \frac{\partial \mathbf{f}_{2_D}}{\partial \boldsymbol{\nu}} & \frac{\partial \mathbf{f}_{2_b}}{\partial \boldsymbol{b}} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}. \tag{A.3}$$

For breviaty, the dependency of **x** of the partial derivatives is not stated in (A.3). These elements of partial derivatives that are non-zeros are given below.

$$\frac{\partial \mathbf{f}_1(\mathbf{x})}{\partial \boldsymbol{\eta}} = \begin{bmatrix} 0 & 0 & -s(\psi)u - c(\psi)v \\ 0 & 0 & c(\psi)u - s(\psi)v \\ 0 & 0 & 0 \end{bmatrix} \tag{A.4}$$

$$\frac{\partial \mathbf{f}_1}{\partial \boldsymbol{\nu}} = \boldsymbol{J}(\boldsymbol{\eta}) \tag{A.5}$$

$$\frac{\partial \mathbf{f}_{2_C}(\mathbf{x})}{\partial \boldsymbol{\nu}} = -\mathbf{M}^{-1} \begin{bmatrix} 0 & C_{13}r & C_{13}v \\ C_{23}r & 0 & C_{23}u \\ C_{31}v + C_{32}v & C_{31}u + C_{32}u & 0 \end{bmatrix} \tag{A.6}$$

where C_{ij} is element in row i and column j in the $\boldsymbol{C(\nu)}$ matrix.

$$\frac{\partial \mathbf{f}_{2_D}(\mathbf{x})}{\partial \boldsymbol{\nu}} = -\mathbf{M}^{-1}(\mathbf{D}_L + 2\mathbf{D}_Q(\boldsymbol{\nu})) \tag{A.7}$$

where

$$\mathbf{D}_Q(\boldsymbol{\nu}) = \begin{bmatrix} X_{|u|u}|u| & 0 & 0 \\ 0 & Y_{|v|v}|v| & 0 \\ 0 & 0 & N_{|r|r}|r| \end{bmatrix} \tag{A.8}$$

$$\frac{\partial \mathbf{f}_{2_b}}{\partial \boldsymbol{b}} = \mathbf{M}^{-1} \tag{A.9}$$

Appendix B

codes

B.1 MATLAB

B.1.1 Added Mass Estimation

The script for Eidsvik (2015) empirical method of finding the added mass is found in in the folder $code$ with the name *AddedMass_Eidsvik*. The geometric parameters used can be found in Table 3.1.

B.1.2 Damping Identification

The code used to perform the damping identification using DFO is found in the folder $code/DFO$. To run the simulations using Method 1, run the script named $optim\_all\_tests\_sim$. To run the simulations using Method 2, run the script named $optim\_all\_tests\_obs$. In both these scripts the variable D_optim_al is used to decide which model is used, where giving this variable the value 1 means that Model 1 is run, while the value 2 means the Model 2 is run.

B.2 Control System

B.2.1 Observer

The codes with the implemented observer are found in the folder $code/controlsystem/observer{+}guidance$. The scripts are written in Python. Before being able to run the observer, the computer running the observer must be connected to the drone through wifi and to the APS using an Ethernet cable. To run the observer using the ODM developed in this thesis (see Chapter 7), you need to run the script $node\_observer$ with the case variables set to the following: $case\_model = 1$, $case\_parder = 1$, $case\_obs = 1$, $case\_pos = 1$, and $signal\_freeze = True$.

B.2.2 Guidance system

The codes with the implemented guided system are found in the folder $code/controlsystem/observer + guidance$. The scripts are written in Python. To run the guidance system, the computer used must be connected the same way as when running the observer. To run the guidance system, you need to run the script $reference\_generator$ with the case parameters set to: $depth\_case = 1$ and $case\_pos = 1$.

B.2.3 Controller

The coded implementation of the control laws is not appended, since this is a part of the private Blueye codes. However, all the relevant parts that have been implemented have been stated in Chapter 9. Since the control laws were implemented as a part of the internal control system on the drone, the modified controller had to be cross compiled to the drone using a docker image. The cross compilation required internet access. Once the controller was cross-compiled to the drone, the script could not be modified without cross-compiling again. When testing outdoors, the controller in surge, sway, heave and yaw were activated by calling rosservices. For tuning the controller gains, ROS' dynamic reconfigure package[1], was used to update the controller gains at runtime without having to restart any rosnodes. This was done by running the script dyn_param_client which can be found in the folder $code/controlsystem/live\_tuning$.

[1]`http://wiki.ros.org/dynamic_reconfigure`

Appendix C

Software

C.1 Signal flow APS

A detailed signal flow in Water Linked's APS is shown in Figure C.1.

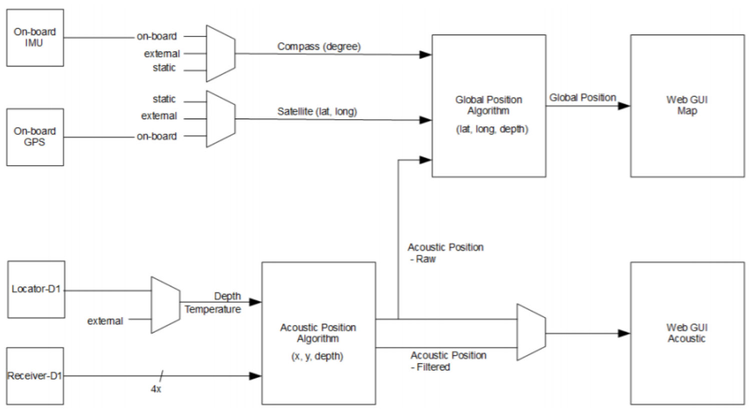

Figure C.1: Signal flow APS. Courtesy Water Linked.

Appendix D

Nelder-Mead Simplex Algorithm

The Nelder-Mead algorithm is based on simplices, which are ”bounded convex polytopes with nonempty interior and exactly n + 1 vertices” (Audet, 2017). The methods begins by constructing a simplex and evaluating the objective function at each vertex of the simplex. Simply explained, it attempts to replace the worst vertex of the simplex with a better one. The algorithm evolves by doing ”reflections”, ”contractions”, and ”expansion” (Audet, 2017). A detailed explanation of the algorith is found in (Lagarias et al., 1998).

Appendix E

CUSUM

This section is written based on ”the CUSUM Algorithm” section in (Blanke, 2016). The cumulative sum (CUSUM) algorithm is used to detect a known changes. This algorithm takes use of the log-likelihood ratio of an observation z, defined as

$$s(z) = ln\frac{p_{\theta_1}(z)}{p_{\theta_0}(z)} \tag{E.1}$$

where p_{θ_i} is a probability density function that depends on the parameter θ. θ_0 is the parameter before a change has occurred, and θ_1 is the parameter is the changed parameter.
The CUSUM is defined as the sum of all the log-likelihood ratios

$$S(k) = \sum_{i=1}^{k} s(z(i)) = \sum_{i=1}^{k} ln\frac{p_{\theta_1}(z(i))}{p_{\theta_0}(z(i))} \tag{E.2}$$

where $S(k)$ is the CUSUM for the present time instant k.
The CUSUM is expected to have a negative drift before a change, and a positive drift after the change. Defining two hypotheses, H_1 and H_2, where H_1 is the nominal case and H_2 is the case where a change has occurred. To decide between the two hypotheses, one can use a decision function $g(k)$ defined as

$$g(k) = S(k) - m(k) \tag{E.3}$$

where $m(k) = min_{i \leq j \leq k} S(j)$. The decision function can identically be defined as

$$g(k) = \max_{1 \leq j \leq k} \sum_{i=1}^{k} ln\frac{p_{\theta_1}(z(i))}{p_{\theta_0}(z(i))} = \max_{1 \leq j \leq k} \sum_{i=1}^{k} s(z(i)) \tag{E.4}$$

The test of determining if a change has happened is then defined as

$$\text{if} \quad g(k) \leq h \quad \text{accept } H_0 \tag{E.5a}$$

$$\text{if} \quad g(k) < h \quad \text{accept } H_1 \tag{E.5b}$$

The threshold h is a design parameter that must be specified by the user.

Applications

The CUSUM algorithm can be used to detect a change in mean or variance since the probability functions are known. For instance, for a change in mean for a normal distribution from μ_0 to μ_1 we get

$$s(z) = ln\frac{p_{\mu_1}(z)}{p_{\mu_0}(z)} = \frac{2(\mu_1 - \mu_0)z + (\mu_1^2 + \mu_0^2)}{2\sigma^2} \tag{E.6}$$

where μ and σ are respectively the mean and the variance of the normal distribution. An equivalent log-likelihood function can be derived for the variance. The changed mean μ_1 is another design parameter that must be determined. The design parameter h and μ_1 can be determined to satisfy specified mean time for detection and/or mean time between false alarms. This is done using the average run length (ARL) function L

$$L(\mu) = E_\mu(k_a) \tag{E.7}$$

where k_a is the alarm time, i.e. the time when $g(k)$ exceeds the threshold h. The ARL function can be approximated as

$$\hat{L} = (exp[-2(\frac{\mu_s h}{\sigma_s^2} + 1.166\frac{\mu_s}{\sigma_s})] - 1 + 2(\frac{\mu_s h}{\sigma_s^2} + 1.166\frac{\mu_s}{\sigma_s}))(\frac{\sigma_s^2}{2\mu_s^2}) \tag{E.8}$$

where μ_s and σ_s^2 are the mean and variance of the increments of the cumulative sum. They are given by

$$\mu_s = \frac{\mu_1 - \mu_0}{\sigma^2}(\mu - \frac{\mu_1 - \mu_0}{2}) \tag{E.9}$$

and

$$\sigma_s^2 = \frac{(\mu_1 - \mu_0)^2}{\sigma^2} \tag{E.10}$$

When the mean μ equals μ_1, the ARL function $L(\mu_1)$ gives the mean time for detection $\bar{\tau}$. The mean time for detection can be estimated as

$$\hat{\bar{\tau}} = \hat{L}(\frac{\beta^2}{2\sigma^2}) \tag{E.11}$$

β is a new variable defined as $\beta = \mu_1 - \mu_0$. When $\mu = \mu_0$, the ARL function $L(\mu_0)$ gives the mean time between false alarms $\bar{T}$, which is estimated as

$$\hat{\bar{T}} = \hat{L}(-\frac{\beta^2}{2\sigma^2}) \tag{E.12}$$

NTNU
Norwegian University of Science and Technology

blueye