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

MSC THESIS DESCRIPTION SHEET

Name of the candidate:	Lundby, Erlend Torje Berg
Field of study:	Marine control engineering
Thesis title (Norwegian):	Autonom stegvis banegenerering og banefølging for en undervannsdrone
Thesis title (English):	Autonomous stepwise path-generation and path-following for an underwater drone

Background

Dynamic path and mission planning for dynamic environments is necessary when, for instance, unknown obstacles are located beyond the sensor range, unpredictable or moving obstacles are detected, and tasks or mission objectives change during operation. Such a setup is highly relevant in the Vortex project – which aims for competing in the AUV competition Robosub.

In this project, the objective is to develop a path-generation, guidance, and control design that ensures that a fully actuated underwater drone stepwise will be able to replan and update its mission objective(s), stepwise perform path-planning and path-generation, and control the robot accordingly.

Work description

1) Perform a background and literature review, such as papers, articles, web-pages, reports, manuals, etc., to provide information and relevant references on:
   - Robosub competition and Vortex underwater drone.
   - Relevant sensors and instrumentation for the Vortex drone.
   - Autonomous underwater robot system architectures.
   - Relevant guidance and control methods.
   - Maneuvering control problem.

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

2) Formulate the overall guidance and control problem for the Vortex drone. This should include a vehicle specification and a control system topology drawing showing sensors, actuators, embedded control computers, communication, etc. The problem formulation should also include model/system assumptions, control design model with inputs, states, and outputs, description of workspace and configuration space, specification of the working environment, and a description of the control objective(s).

3) Develop a function to represent, store, and update the mission objective(s), based on mission specifications by a user and sensed information. This should at least include the last departure waypoint and target waypoint(s), speed reference, obstacles of relevance, and other relevant information.

4) Develop a path-planning method that based on the mission information determines the next path segment(s) with speed assignment. There should be a logic mechanism in this function that ensures:
   a) that a new target waypoint is activated when reaching a circle of acceptance of the previous target waypoint, and
   b) stores in a table the previous path segments with their corresponding parameters.

5) Develop a stepwise path-generation function that parameterizes each path segment by an index i and a path variable $\theta \in [0,1]$. Derive the algorithm so that each path segment is C^3, and also so that two consecutive path segments will be C^3 in its connection point, that is, continuous in its three first derivatives. Determine also a heading reference along the path, so that this is at least C^2.

6) Develop a maneuvering-based guidance and control law that, based on fully-actuated dynamics, controls the vehicle position and heading to converge to and follow the path with the assigned speed.
7) Present a simulation model and propose some relevant simulation scenarios that illustrate and verify the resulting performance of the stepwise path-planning and path-following control system. Analyze and compare the performance of the two control strategies, discuss the results, and make conclusions.

Specifications

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

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

The report shall be organized in a logical structure to give a clear exposition of background, results, assessments, and conclusions. The text should be brief and to the point, with a clear language. Rigorous mathematical deductions and illustrating figures are preferred over lengthy textual descriptions. The report shall have font size 11 pts., and it is not expected to be longer than 70 A4-pages, 100 B5-pages, from introduction to conclusion, unless otherwise agreed upon. It shall be written in English (preferably US) and contain the following elements: Title page, abstract, acknowledgements, thesis specification, list of symbols and acronyms, table of contents, introduction with objective, background, and scope and delimitations, main body with problem formulations, derivations/developments and results, conclusions with recommendations for further work, references, and optional appendices. All figures, tables, and equations shall be numerated. The original contribution of the candidate and material taken from other sources shall be clearly identified. Work from other sources shall be properly acknowledged using quotations and a Harvard citation style (e.g. *natbib* Latex package). The work is expected to be conducted in an honest and ethical manner, without any sort of plagiarism and misconduct. Such practice is taken very seriously by the university and will have consequences. NTNU can use the results freely in research and teaching by proper referencing, unless otherwise agreed upon.

The thesis shall be submitted with an electronic copy to the main supervisor and department according to NTNU administrative procedures. The final revised version of this thesis description shall be included after the title page. Computer code, pictures, videos, dataseries, etc., shall be included electronically with the report.

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

Supervisor: Roger Skjetne
Co-advisor(s): N/A

Trondheim, 21.06.2019

Roger Skjetne
Supervisor

Summary

This thesis presents the main components of an Autonomous underwater vehicle (AUV) related to control and supervision of the autonomous system. This includes a Supervisory system responsible for reasoning and decision making, a Path Generation module that provides smooth reference paths to the guidance system, and a guidance and control system based on the maneuvering problem, Skjetne [2005]. A simulation model of the Vortex drone is derived, and used as platform for testing and development of the autonomous system.

The Supervisory system is divided into three modules, respectively mission management, risk management and control mode management. The mission management system is in charge of decisions related to accomplish mission objectives. Since the AUV knows its local environment, the scope of the plans are short term, and path planning is limited to one or a few way points. The risk management system is in charge of avoiding potential collisions with dynamical obstacles. If dynamical obstacles are observed within a given risk zone, the AUV should stop and wait until the obstacle leaves the zone. The control mode managment system is in chare of smooth transitions between control objectives. If the AUV goes from maneuvering to dynamical positioning or vice versa, the control mode management system will make sure that the guidance system makes this transition smooth. At the same time, the control mode management makes sure that the guidance system acts according to the current control objective.

The path generation module takes the way-points generated by the mission manager as input to generate smooth reference paths for the guidance system. These paths are designed to ensure bumpless transfers in the transitions between path segments which are connected in the way-points. The guidance and control system design is based on The Maneuvering Problem. This involves separating the control problem into a geometric task and a dynamic task. Given a parametrized path, a dynamic assignment along the path can be designed to ensure high performance of the system despite conflicting performance measures, something that is demonstrated in this thesis.

Sammendrag

Denne oppgaven presenterer de hovedkomponentene til en Autonom Undervannsfarkost (AUV) som er knyttet til kontroll og overordnede beslutninger i det autonome systemet. Dette inkluderer et system som er ansvarlig for resonering og beslutningsprosesser, en banegeneringsmodul som genererer glatte kurver til veiledningssystemet, og et veilednings- og kontrollsystem basert på *The Maneuvering Problem* Skjetne [2005]. Det er utledet en simuleringsmodell av Vortex sin drone som brukes som plattform for testing og utvikling av det autonome systemet.

Det overordnede beslutningstakingssystemet er delt inn i tre moduler, henholdsvis oppdragsstyring, risikostyring og kontrollmodusstyring. Oppdragsstyringssystemet har ansvaret for beslutninger knyttet til oppnåelse av oppdragsmål. Siden AUVen kun kjenner sitt lokale miljø, er planenes omfang kortvarig, og baneplanleggingen er begrenset til ett eller noen få veipunkt av gangen. Risikostyringssystemet er ansvarlig for å unngå potensielle kollisjoner med dynamiske hindringer. Hvis dynamiske hindringer observeres innenfor en gitt risikosone, skal AUVen stoppe og vente til hinderet forlater denne sonen. Styringsmodussystemet har ansvar for at glatte transisjoner som følge av endret kontrollobjectiv. Hvis AUVen går fra manøvrering til dynamisk posisjonering eller omvendt, vil styringsmodusstyringssystemet sørge for at veiledningssystemet gjør denne overgangen jevn. Samtidig sørger styringsmodussystemet for at veiledningssystemet forholder seg til det gjellende kontrollobjetivet.

Banemoduleringsmodulen tar veipunkter som input for å generere glatte referansebaner til styringssystemet. Disse banene er utformet for å sikre støtfrie overføringer i overgangene mellom banesegmenter som er forbundet i veipunktene. Retnings- og kontrollsystemdesign er basert på The Maneuvering Problem . Dette innebærer å separere kontrollproblemet i en geometrisk oppgave og en dynamisk oppgave. Gitt en parametriert sti, kan en dynamisk oppgave langs stien utformes for å sikre høy ytelse av systemet til tross for ytelsesmål som kan skape konflikter. Dette er demonstrert i denne oppgaven.

Acknowledgments

There are several contributors related to the work of this thesis, and I would like to thank them in this Section.

First of all, I would like to thank my supervisor Roger Skjetne. Through many good discussions and private lessons he have helped me gaining much of the basic knowledge related to the work in this thesis. His great professional engagement and good suggestions have inspired me and played a key role in many of the system designs.

Furthermore, I would like to thank my colleagues in Vortex NTNU. The members of the organization are always positive and engaged in helping each other out. Regarding the parts of the thesis related to hardware, I have learned a lot from members that are experts on this field. Furthermore, I would like to specifically thank the members of the control group which I have been working close to. These are namely Kristoffer Rakstad Solberg, Petter Hoem Sletsjøe, Øystein Barth Utbjoe, Espen Eilertsen and Knut Turøy. Thank you for a lot of interesting discussions and in general a motivational collaboration.

Lastly, I would like to thank both Jan Tommy Gravdahl and Carl Einar Bonnevie Rasmussen who have been very helpful by proofreading and validating my work. I am grateful for your kindness.

Table of Contents

Summary **i**

Sammendrag **i**

Acknowledgments **ii**

Table of Contents **v**

List of Tables **vii**

List of Figures **x**

Abbreviations **xi**

1 Introduction **1**
- 1.1 Motivation 1
- 1.2 Objectives 1
- 1.3 Scope and limitations 2
- 1.4 Contributions of this thesis 2

2 Background **3**
- 2.1 Autonomous system architecture 4
  - 2.1.1 Three layered architecture 4
- 2.2 Reference frames 6
- 2.3 Guidance and control systems 6
  - 2.3.1 Target tracking 7
  - 2.3.2 Trajectory tracking 8
  - 2.3.3 Path Following 9
  - 2.3.4 Control methods 11
- 2.4 Maneuvering control problem 13
  - 2.4.1 Problem formulation 13
  - 2.4.2 Path parametrization 14
- 2.5 Sensors 15
  - 2.5.1 Inertial Measurement Unit (IMU) 16
  - 2.5.2 Doppler Velocity Log (DVL) 17
  - 2.5.3 Pressure gauge 17
  - 2.5.4 Camera 17

2.5.5 Acoustic system . . . 18
2.6 Robosub competition . . . 18
2.7 Vortex NTNU . . . 22

3 Problem formulation **23**
3.1 Simulation Model . . . 23
3.2 Control Design Model . . . 23
3.2.1 CDM for maneuvering controller . . . 24
3.2.2 CDM for depth control . . . 24
3.3 Hardware . . . 25
3.4 Resulting autonomous system . . . 27
3.5 Problem statement . . . 28
3.5.1 Supervisory system . . . 28
3.5.2 Path Generation . . . 28
3.5.3 Maneuvering problem . . . 29
3.5.4 Depth control: Tracking problem . . . 29
3.5.5 Performance measures . . . 30

4 Simulation model **31**
4.1 Dynamics . . . 31
4.1.1 Kinematics . . . 31
4.1.2 Kinetics . . . 33
4.2 Assumptions and Simplifications . . . 34
4.3 Added mass estimate . . . 34
4.3.1 Oblate spheroid . . . 35
4.3.2 Rectangular octagon . . . 37
4.3.3 square . . . 38
4.4 Rigid-Body System Inertia Matrix . . . 39
4.5 Linear Damping . . . 40
4.6 Restoring forces . . . 41
4.7 Resulting model . . . 41

5 Supervisory system **43**
5.1 State machines and Harel Statechart . . . 43
5.2 Structure . . . 44
5.2.1 Mission manager . . . 44
5.2.2 Risk Manager . . . 46

6 Path Generation, Guidance and Control Design **49**
6.1 Path Generation . . . 49
6.1.1 $\mathcal{C}^r$ path generated from way-points . . . 49
6.1.2 Step-wize generation of $\mathcal{C}^3$ path . . . 50
6.2 Guidance . . . 53
6.3 Dynamic assignment . . . 53
6.4 Backstepping control desing . . . 54
6.4.1 Control objective . . . 54
6.4.2 Control design . . . 55

7 Results and discussion **59**
7.1 Parameter study . . . 59
7.1.1 Case 1: Path A . . . 60

7.1.2 Case 2: Path B 65
7.1.3 Remarks 69
7.2 Controller robustness against modelling errors 70
7.2.1 Simulating model error 72
7.2.2 Underestimate of pyhsical model, $\alpha = 0.5$ 72
7.2.3 Overestimate of physical model, $\alpha = 0.5$ 74
7.2.4 Remarks 76
7.3 Resulting system 76

8 Concluding remarks **81**
8.1 Conclusion 81
8.2 Further Work 82

List of Tables

2.1 Frames and their notations 15
3.1 Maximum thrust - Forward and reverse. 26

List of Figures

2.1 Illustration of differences between the two paradigms in robotic architecture. 5
2.2 Control architecture for autonomous technology platform. 5
2.3 Three guidance methods used in target tracking. 8
2.4 Principle of LOS guidance. 10
2.5 Piecewize linear path connected by way-points. 14
2.6 The implemented IMU - STIM300 16
2.7 The implemented accoustic sensor - DVL1000. 17
2.8 Front camera and bottom camera. 18
2.9 TRANSDEC facility. 18
2.10 The task environment. 19
2.11 Initial heading by coin flip. 19
2.12 Computer-Aided Design (CAD) drawing of gate with measures. 20
2.13 Path markers with dimensions. 20
2.14 CAD drawing of buoys with dimensions. 21
2.15 CAD drawing of the vertical board. 21

3.1 Communication drawing of the robotic system. 25
3.2 Thrusters mounted on Manta. 26
3.3 Graphical representation of the resulting autonomous system. 27

4.1 Manta in NED and body frame. 31
4.2 Oblate spheroid. 35
4.3 Matlab plot showing interpolation of table values of add 36
4.4 Rectangular octagon. 37
4.5 Visualisation of square approximation of added mass. 38
4.6 Rectangular shape. Added mass of a square plate in heave 39

5.1 Hierachy of the supervisory system. The dotted line between the tree superstates indicates that these states are parallel 44
5.2 Graphical representation of the mission manager. 45
5.3 Risk Management system. 46
5.4 Control Mode Management system. 47

6.1 Online path generation from Matlab. 51

7.1 Paths used in the case study for the dynamic task. 60
7.2 Tracking performance of position references. 61

7.3 Cross-track and along track error. 61
7.4 Performance of heading and speed control. 62
7.5 Cross-track and along track error. 62
7.6 Performance tracking of position references 63
7.7 Cross-track and along track error. 63
7.8 Heading and speed. 64
7.9 Tuning of maximum desired speed 64
7.10 Effect of the tuning parameter λ. 65
7.11 Tracking performance of position references. 66
7.12 Cross-track and along track error. 66
7.13 Performance of heading and speed control. 67
7.14 Tracking performance of position references. 67
7.15 Cross-track and along track error. 68
7.16 Heading and speed. 68
7.17 69
7.18 Undesirable path. $\lambda = 10$. 70
7.19 Test path. 71
7.20 Tracking performance of position references 71
7.21 72
7.22 Performance tracking of position references 73
7.23 73
7.24 Performance tracking of position references 74
7.25 74
7.26 Performance tracking of position references 75
7.27 75
7.28 Speed and heading plots after introducing integral action 76
7.29 Entire Run. The tasks are indicated in the plot 77
7.30 Crafts approach when dynamical object is detected 77
7.31 Heading performance 78
7.32 Crafts approach when dynamical object is detected 79

Acronyms

ARS Angular Rate Sensor. xi, 16
AUV Autonomous underwater vehicle. i, xi, 1, 5, 17, 18, 21–24, 28, 34–39, 41, 45, 46, 54, 81

CAD Computer-Aided Design. ix, xi, 20, 21, 39
CDM Control Design Model. iv, xi, 23, 24
CLF control Lyapunov functions. xi, 13, 55, 56
CPU Central Processing Unit. xi, 25

DOF Degrees of Freedom. xi, 11, 23, 24, 31, 33, 54, 55
DP Dynamic Positioning. xi, 11
DVL Doppler Velocity Log. iii, xi, 17

ESC Electronic Speed Control. xi, 26

FSM Finite State Machine. xi

GAS Global asymptotic stability. xi, 55
GPU Graphics Processing Unit. xi, 25

IMU Inertial Measurement Unit. iii, xi, 16, 17

kg f Kilogram Force. xi, 26

LOS Line-of-Sight. ix, xi, 7, 9, 10
LQR Linear Quadratic Regulator. xi, 12

MATE Marine Advanced Technology Education. xi, 22
MCU Micro Control Unit. xi, 25
MEMS Micro-Electro-Mechanical System. xi, 16

MIMO Multiple input multiple output. xi, 11

N Newton. xi, 26

NED North East Down. ix, xi, 6, 10, 12, 14, 15, 31–33

NM Newton meter. xi

NTNU Norwegian University of Science and Technology. xi, 22

PID Proportional-integral-derivate. xi, 11, 12, 29

PWM Pulse-width Modulation. xi

ROS Robot Operating System. xi, 25, 82

ROV Remotely operated vehicle. xi, 22

RPM Rounds Per Minute. xi, 21

SBC Single Board Computer. xi, 25

SPA Sense Plan Act. xi, 4, 5

TRANSDEC Transducer Evaluation Center. ix, xi, 18, 19

Chapter 1

Introduction

1.1 Motivation

An Autonomous underwater vehicle (AUV) has to navigate in an environment according to mission objectives without human intervention. This means that the vehicle is dependent upon its own decision making to successfully complete a predefined mission. This motivates a deliberative mission manager system that, given a predefined mission, can give commands and generate references for lower layer modules of the system so that the system is able to execute a mission successfully. This leads us to the other modules required to navigate according to a mission objective, namely the guidance and control system modules.

The Vortex drone will compete in a AUV competition, where only local information will be available for the drone at any given time. The drone will have to navigate according to local hints that are placed out in the environment. This motivates a step-wise path generation, where the mission layer continuously search for the next sub-target, and the guidance and control system are responsible for following this plan. In order to ensure smooth transfers in the way-points to ensure safe, predictable operation, a path generation module that generates smooth paths are $\mathcal{C}^3$ continuous paths will be included in the system.

1.2 Objectives

The main objective of this thesis is to develop and implement the modules of an AUV. The system should be able to react to and navigate according to information about the environment and tasks that is perceived during operation. The system will we tested by a simulation study, which requires a simulation model. The simulated environment should reflect the environment in the Robosub competition and the simulation model should reflect the dynamics of the Vortex underwater drone. The purpose of this is to develop a simulation platform that can be used by Vortex NTNU to test control system and algorithms related to mission managing and navigation. This can be divided into the following objectives:

1. Perform a literature review on autonomous system architecture for underwater vehicles and modules in this system architecture. Furthermore, perform a literature review on relevant sensors for the Vortex drone, the Robosub competition and vortex organization
2. Develop a mission manager that acts according to mission objectives, and reacts to information about environment and targets that is perceived during operation. Develop a path planner that generates way-points according to perceived information.

3. Develop a step-wise path generation algorithm that generates $\mathcal{C}^3$ paths.
4. Develop a guidance system that uses the generated path to generate references in position and heading for the controller. Use a path variable $\theta \in [0,\ 1)$ along the path segment and a dynamic assignment to help determining current reference for the controller.
5. Develop a control design that drives the states of the system to the desired states along the path.
6. Derive a simulation model of the Vortex drone based on reasonable assumptions.

1.3 Scope and limitations

- Thruster allocation matrix is excluded. However, thruster dynamics is included by sending the desired thrust through a low pass filter that should represent the thruster dynamics.
- Navigation algorithms are not included in the simulation model. Calculated states are used in the absence of an estimator.

1.4 Contributions of this thesis

In this thesis the following contributions has been made:

- Implementation of an autonomous system in Simulink consisting of the modules that will be mentioned below.
- Development of a supervisory system that is able to control the desired behavior of the system according to mission objectives. The system react to moving obstacles. This system was implemented in Stateflow, which is a graphical editor for modeling decision logic in state machines. Stateflow is compatible with Simulink, and is included in the autonomous system that is simulated in Simulink.
- Implementation of a path generation algorithm in the Simulink model, receiving way-points and generating smooth, $\mathcal{C}^3$ continuous paths.
- Implementation of a guidance module and a control module according to the maneuvering problem, per Section 2.4. Design of the dynamic assignment as a tool in the process of tuning the control system according to performance measures per Section 3.5.5.
- Derivation of a dynamic model for the Vortex drone. Implementation of this model in Simulink, and its appliance in the Simulation model as a platform for testing and development of the control, guidance, path generation and supervisory systems.

Chapter 2

Background

2.1 Autonomous system architecture

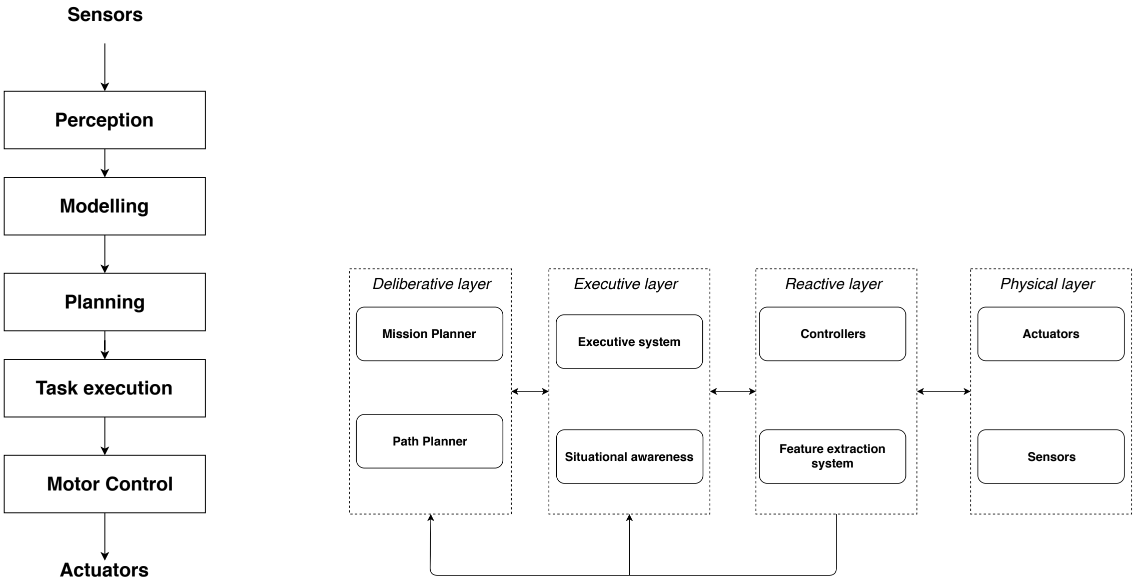
There are several capabilities demanded from autonomous robotic systems that distinguish them from other software systems. Robotic systems need to act asynchronously, in real time and in uncertain and dynamic environments. This requires capabilities for acting in real time, controlling actuators and sensors, deal with concurrency, detect and react to exceptions, dealing with uncertainty, and integrating high level planning with low level control. This results in complex software systems. A well-conceived system architecture will make it easier to deal with this complexity as it provides advantages in specification, execution, and validation of robotic systems.

Modular decomposition is a common feature in autonomous system architecture, where functionality is separated into modular units. The decomposition is often hierarchical, where modular components are built up of other modular components. This layered structure reduce system complexity by abstracting features into modules. There are several ways of decomposing the system (Siciliano and O.Khatib). One method is decomposition along the temporal dimension where each layer is characterized by response frequency lower than the layer below. **Temporal decomposition** require lower response time for the most real-time dependent tasks, where real-time dependency decreases with higher level tasks. Another method is **task abstraction** - tasks at one level are achieved by invoking set of tasks at lower levels. This means that tasks generated in high level deliberative layer are decomposed into commands and references for modules at the executive and reactive layer. The layered architecture structure that will be presented is decomposed both in the temporal dimension and for task abstraction.

2.1.1 Three layered architecture

An important aspect of the three layered architecture is the role of the *internal states* in each layer Gat [1998]. In this context, the internal states refers to the robot's own perception and interpretation of surrounding world. Consider the early robotic procedure Sense Plan Act (SPA). First, information from the world is perceived (sense), then the robot planes actions (plan) before executing planned actions (act). This procedure has the disadvantage that the robot do not respond to changes in the world during execution. Further we can argue that an unexpected outcome from the execution of a plan step can cause subsequent plan steps to be executed in an inappropriate context. Both of these problems can be viewed as the result of the method used to manage internal state information.

This is solved in the three layered architecture by managing the internal state information on three different ways, one way for each layer. The three layered structure organizes its algorithms according to whether they contain a minimum of internal states, contain internal states reflecting the memory about the past or about perceptions until this moment and, or if the algorithm contain internal states reflecting predictions about the future Gat [1998]. The three modules are respectively the *reactive layer*, the *executive layer* and the *deliberative layer*.


Sensors
Perception
Modelling
Planning
Task execution
Motor Control
Actuators
Deliberative layer
Mission Planner
Path Planner
Executive layer
Executive system
Situational awareness
Reactive layer
Controllers
Feature extraction system
Physical layer
Actuators
Sensors

(a) Decomposition of a mobile robot in terms of SPA procedure. Adapted from Moreno [2010]

(b) Decomposition of mobile robots in terms of three-layered architecture. Inspired by Sanchez-Lopez et al. [2016]

Figure 2.1: Illustration of differences between the two paradigms in robotic architecture.

Figure 2.1 illustrates the point made in the last paragraph. In the layered control algorithm, the layers interact in a bidirectional manner which leads to a more flexible system in terms of interacting with changes in the environment.

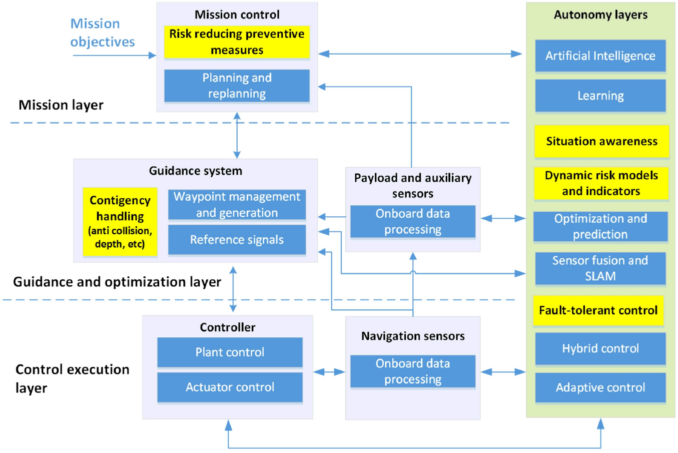

Mission objectives
Mission control
Risk reducing preventive measures
Planning and replanning
Mission layer
Guidance system
Contigency handling (anti collision, depth, etc)
Waypoint management and generation
Reference signals
Payload and auxiliary sensors
Onboard data processing
Guidance and optimization layer
Controller
Plant control
Actuator control
Navigation sensors
Onboard data processing
Control execution layer
Autonomy layers
Artificial Intelligence
Learning
Situation awareness
Dynamic risk models and indicators
Optimization and prediction
Sensor fusion and SLAM
Fault-tolerant control
Hybrid control
Adaptive control

Figure 2.2: Control architecture for autonomous technology platform. Courtesy:Ludvigsen and Sørensen [2016]

An autonomous system architecture for AUV is presented in Figure 2.2. This structure is consistent with the three layered architecture presented above. The deliberative layer is called the *Mission layer*, the executive layer is called the *Guidance and optimization layer* and the executive layer is called the *Control execution layer*.

Deliberative layer: This layer generates the plan as a sequence of actions according to the objective and input from the executive layer as situation awareness. The deliberative layer is responsible for planning and re-planning according to mission objectives.

Executive layer: The executive layer consists of translating the desired plan generated in the high level deliberative layer to references for the motion controller. In the deliberative layer, path planning algorithms generate way-points according to the mission. Path generation algorithms use the way-points to generate smoother paths according to the vessel dynamics. The guidance module generates references for the controller in terms of desired position, speed and acceleration. Also contingency handling in terms of for example anti collision inhabit the executive layer. In addition, the executive layer is responsible for situational awareness. SLAM methods generates an understanding of the robots position and pose in a map and relative to objects.

Reactive layer: The reactive layer consist of motion controllers and and navigation. The navigation algorithms uses measurements from sensors to estimate pose. Motion control receive reference states from the execution layer and the navigation module then estimates states and generates desired forces for the thrust allocation algorithm which again generates desired forces for each actuator.

2.2 Reference frames

The following reference frames are used frequently in marine control systems.

NED (Earth fixed): The North East Down (NED) coordinate frame $\{n\} = (x_n, y_n, z_n)$ is defined as a tangent plane of the earth surface. The x-axis points towards North, y-axis towards East and z-axis points down towards the center of the Earth. In this case, the origin is fixed. This is convenient in order to express the position of the vessel.

Body frame (body fixed): The body fixed reference frame $\{b\} = (x_b, y_b, z_b)$ has its origin o_b fixed to a point on the vessel, referred to as CO - Center of Origin. The axes are chosen to coincide with the principal axis of inertia, and are defined according to Figure 4.1. In this Figure x_b points in the same direction as the front camera, y_b is directed to starboard and z_b follows the right-hand rule downwards from the xy-plane.

Path fixed reference frame: In many cases, it is convenient to express the vessel position in terms of the path reference frame. For simplicity, only paths i a two dimensional plane is considered. For a curved, parametrised path, the origin of the frame is found at desired position on the path $p_d(\theta)$, where θ is the path parameter. The x-axis points along the tangent of the path and the y-axis points 90 degrees left, according to the right hand rule. The coordinates are expressed as

$$\begin{aligned} \boldsymbol{\epsilon}^{p}_{b/p} &= col(s, e) \\ \boldsymbol{\epsilon}(t) &= \boldsymbol{R}_2(\alpha_k)(p^n(t) - p^n_k) = [s(t)\ e(t)] \end{aligned} \tag{2.1}$$

2.3 Guidance and control systems

This section is based on Fossen [2011] unless other literature is specified. Control and guidance systems works close together, thus the design of each of these systems is highly dependent on each other. In motion control, it is typical to define the control problem according to one of the following control objectives:

- **Setpoint regulation**
  Desired position and attitude are chosen to be constant. This is the most basic guidance system in form of constant setpoint feeded to the controller. In this case, the controller is a *regulator*.

- **Trajectory tracking**
  In trajectory tracking, the control objective can in general be to track time varying position, velocity and acceleration reference signals. The guidance system computes these reference signals, typically generated by low-pass filters or optimization methods. The corresponding controller is a tracking controller. Trajectory tracking results in both spatial and temporal constraints.

- **Path Following**
  The objective is to follow a predefined path which is independent of time. There are no temporal constraints linked to propagation along the task, however spatial constraints can be added.

Guidance

As indicated above, guidance includes a variety of methods that in general calculate references for a control system to achieve a certain control objective.

2.3.1 Target tracking

Target tracking is a tracking control problem where the objective is to track an either stationary (setpoint regulation) or moving point. In the case of a moving point, only the instantaneous motion is known. The objective of target tracking can be formulated as:

$$\lim_{x \to \infty} [\boldsymbol{p}^n(t) - \boldsymbol{p}_t^n(t)] = \mathbf{0}$$

Where $\boldsymbol{p}^n(t)$ is the position of the craft and $\boldsymbol{p}_t^n(t)$ is the position of the target Fossen [2011]. Some guidance methods that can be used in target tracking are:

- **Line-of-Sight (LOS) guidance**
  A three point guidance scheme which involves three points - the craft (interceptor), a target and a stationary reference point. This method will be investigated later in the section.

- **Pure pursuit guidance**
  Two point guidance scheme, with the craft itself and the target. Desired velocity vector is pointed in the direction of the target.

- **Constant bearing guidance**
  Two point guidance scheme. The goal for the interceptor is to perceive the target at a constant bearing.

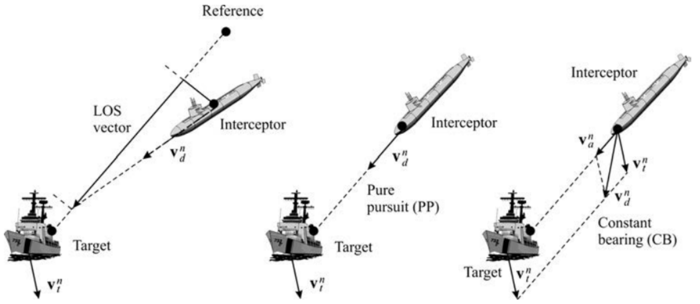


Figure 2.3: Three guidance methods used in target tracking. Courtesy:Fossen [2011]

2.3.2 Trajectory tracking

The control objective of the trajectory tracking problem is to force the system output $\boldsymbol{y}(t)$ to follow a desired output $\boldsymbol{y}_d(t)$. Desired speed and acceleration is obtained from time-differentiation of $\boldsymbol{y}_d(t)$ respectively one and two times.

Reference model

Reference models are used for trajectory generation. This is a simple, *open-loop* guidance method, which means that there is no feedback from the states. Reference models are found in their most simple form as low pass filters:

$$\frac{x_d}{r}(s) = h_{lp}(s)$$

Here, x_d denotes the desired state, and r is a reference signal. The purpose is to suppress high frequencies of the reference signal r, and generate a feasible, desired state x_d. For marine crafts, it is desirable to use a reference filter motivated by the *mass-damper-spring* system:

$$h_{lp} = \frac{\omega_{n_i}^2}{s^2 + 2\zeta_i\omega_{n_i}s + \omega_{n_i}^2} \quad \textit{(i = 1, ..., n)}$$

where $\omega_{n_i}(i = 1, ..., n)$ is the natural frequencies and $\zeta_i(i = 1, ..., n)$ are the relative damping ratios. Position and attitude reference filters are typically chosen to be of third order of filtering the reference signal $\boldsymbol{r}$. This motivates a reference filter of a mass-damper-spring system cascaded with a simple first order low pass filter:

$$\frac{\eta_{d_i}}{r_i}(s) = \frac{\omega_{n_i}^2}{(1 + T_i s)s^2 + 2\zeta_i\omega_{n_i}s + \omega_{n_i}^2} = \frac{\omega_{n_i}^3}{s^3 + (2\zeta_i + 1)\omega_{n_i}s^2 + (2\zeta_i + 1\omega_{n_i}^2 s + \omega_{n_i}^3} \quad \text{i} = \ 1, ..., n$$

where the first order low pass filter time constant $T_i = 1/\omega_i$.

Trajectories generated from simulator The reference model presented above has its limitations. In order to guarantee feasible trajectories, the cutoff frequency of the reference model cannot exceed the closed loop bandwidth of the system. This is difficult to verify due to nonlinearities, time delays and saturating elements in practical systems. An alternative is to use a closed loop model of the craft to generate time varying reference trajectories. Here, the time constants, relative damping ratios and natural frequencies reflect the physical limitations of the craft.

Optimal Trajectory generation The trajectory generation can also be solved as a optimization problem. Static and dynamic constraints are systematically included in this method. For example, the optimization problem can be formulated as minimum power or minimum time $(J = min(power, time))$ subject to constraints such as maximum speed, turning rate, saturation limits in control input and so on.

2.3.3 Path Following

Path following concerns the task of following a path that is independent of time. This means that there are no temporal constraints included in this control problem.

Path Generation Path generation uses way-points from a path plan to generate a continuous path between the way-points. Path generation can be divided into two categories (Lekkas [2014]): paths generated from straight lines and arcs (i) and splines (ii).

Dubins path method uses straight lines and circle arcs to connect way-points. It shows that the shortest path in terms of minimum time between two configurations (x, y, ψ) of a moving craft consist of straight lines and circular arcs. A drawback with this method is the discontinuous jump in yaw rate when switching from straight line $(r_d = 0)$ to circular arc $(r_d = const)$. Another way of combining straight lines and curved arcs is to combine straight lines with *Fermat's spirals*, Lekkas [2014]. In these transition curves, the curvature changes from zero in the transition between straight lines and the beginning of the transition curve to maximum at the middle of the transition curve. This solves the problem with discontinuous yaw rate in the transition.

Paths can also be generated through piece-wise polynomial interpolation. The *cubic Hermite* interpolation ensures that first-order derivatives are continuous in transition between way-points. The *Cubic spline* method requires second order continuity in the end-points, and hence continuous curvature unlike the cubic Hermite interpolation. However, the cubic spline method has more oscillations.

LOS steering law The LOS guidance method is a widely used method for generating heading reference in path following. In LOS guidance, the vessel follows a LOS vector pointing from the vessel to a point (x_{los}, y_{los}) on the path. The LOS vector is calculated according to the direction of the desired path, the cross-track error e(t) from vessel to the desired path and a radius to determine how fast the vessel should converge to the desired path.

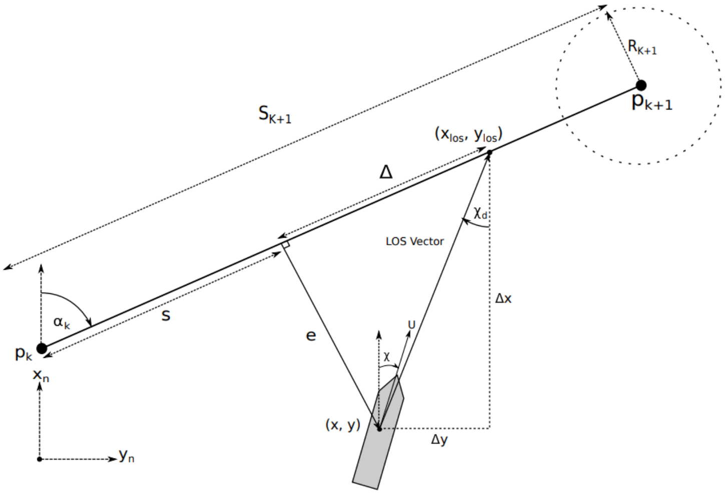


Figure 2.4: Principle of LOS guidance. Courtesy:Havnegjerde [2018].

From Figure 2.4 we have the following. The previous way-point is $p_k = (x_k, y_k)$ and the current target way-point is $p_{k+1} = (x_{k+1}, y_{k+1})$. The relative angle between the NED frame and the current path segment is noted α_k .

$$\alpha_k = atan2(y_{k+1} - y_x, x_{k+1} - x_x) \tag{2.2}$$

The coordinates in the path-fixed frame are given by:

$$\boldsymbol{\epsilon}(t) = \boldsymbol{R}_2(\alpha_k)(p^n(t) - p^n_k) = [s(t)\ e(t)] \tag{2.3}$$

where $p^n(t) = [x,\ y]^T$ and

$$\boldsymbol{R}_2(\alpha_k) = \begin{bmatrix} cos(\alpha_k) & -sin(\alpha_k) \\ sin(\alpha_k) & cos(\alpha_k) \end{bmatrix} \tag{2.4}$$

The control objective is to minimize the cross-track error e(t) defined in Equation 2.3, hence

$$\lim_{t \to \infty} e(t) \to 0 \tag{2.5}$$

Lookahead-based steering

A widly used LOS method is the Lookahead-based steering metod. In this method, the LOS vector should point to a look-ahead distance Δ at the path, see Figure 2.4 . The LOS vector is given by the cross-track error and a lookahead distance $\Delta(t)$ such that.

$$R^2 = e(t)^2 + \Delta(t)^2, \tag{2.6}$$

Where R is the length of the LOS vector, and its size determines the steepness of the desired heading relative to the path. The desired heading χ_d is given by

$$\chi_d = \alpha_k + \chi_r(e), \tag{2.7}$$

Where

$$\chi_r(e) = arctan\left(\frac{-e}{\Delta}\right). \tag{2.8}$$

Control

The control system is composed of the *motion control system* and *control allocation*. The purpose of the motion control system is to calculate necessary control forces and moments $\boldsymbol{\tau} \in \mathbb{R}^n$ according to a control objective. Control allocation on the other hand concerns distributing desired control forces and moments $\boldsymbol{\tau} \in \mathbb{R}^n$to the actuators in terms of control input $\mathbf{u} \in \mathbb{R}^r$.
Consider tracking of a time-varying reference trajectory in 3 DOF: $\boldsymbol{\eta}_d = [N_d(t)\ E_d(t)\ \psi_d(t)]^T$. The tracking objective is to minimize the error $\boldsymbol{e}(t) = \boldsymbol{\eta}(t) - \boldsymbol{\eta}_d(t)$, where $\boldsymbol{\eta} = [N(t)\ E(t)\ \psi(t)]^T$.

- **Trajectory tracking control, three or more controls:** In the case of as many (or more) controls as there are degrees of freedom, the craft is considered fully actuated. This is referred to as a **fully actuated DP system**. The applications are low-speed maneuvering and stationkeeping, where the goal is to drive $\boldsymbol{e}(t) \in \mathbb{R}^2 \times \mathcal{S} \to 0$.
- **Path following control, two controls:** The use of two controls in a 3 DOF *tracking problem* is an *under-actuated* problem, which cannot be solved using linear theory. However, by defining a two dimensional workspace, with along-track and cross-track error ($\boldsymbol{e}(t) = [s(t)\ e(t)]$, it is possible to follow a path by using only two controls (surge speed and yaw moment).

2.3.4 Control methods

In this section, some of the most common control methods in motion control systems will be discussed. This includes methods within optimal control, nonlinear control and state-of-the-art PID control.

PID control

Proportional-integral-derivate (PID) control is a well known control method used in a lot of industrial applications. As the name indicates, there are three terms in the control law, with gains K_p, K_i, K_d, proportional to the error state $K_p(x - k_d)$, the integral of the error $K_i \int_{t_0}^{t}(x - x_d)$ and the derivate of the error $K_d(\dot{x} - \dot{x}_d)$. This accumulates to the control law in 1 DOF:

$$\tau = K_p(x - k_d) + K_i \int_{t_0}^{t}(x - x_d) + K_d(\dot{x} - \dot{x}_d).$$

An analytical way of tuning the gains is to consider a linear mass-damper-spring system and determine the gains from pole placement. The PID controller can easily be extended to MIMO systems. One of the strengths with a PID control law is that each gain is very intuitive, and therefore it is easy to tune the gains. A drawback compared to more advanced control laws is that it is less robust to uncertainties and disturbances and non-linearities than some of the more advanced control laws.

Linear Quadratic Optimal Control

In optimal control, the objective is to find an optimal control law with respect to a given optimization criterion. This criterion is usually given as a cost function depending on the state and control variables. The control law is then designed to minimize the cost function. Linear quadratic optimal control is the collective term of linear input-state-output systems and cost functions that are quadratic term of the input and state variables, Trentelman [2015].
Consider a linear system

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}$$

$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x},$$

where $\boldsymbol{x}$ is the state vector, $\boldsymbol{u}$ is the input vector, $\boldsymbol{y}$ is the output vector, $\boldsymbol{A}$ is the system matrix, $\boldsymbol{B}$ is the input matrix and $\boldsymbol{C}$ is the output matrix. In an Linear Quadratic Regulator (LQR) design, the cost function is expressed as

$$J = \min_{u} \left\{ \frac{1}{2} \int_{0}^{T} (\boldsymbol{y}^T \boldsymbol{Q} \boldsymbol{y} + \boldsymbol{u}^T \boldsymbol{R} \boldsymbol{u} \right\},$$

where $\boldsymbol{Q} = \boldsymbol{Q^T} > 0$ and $\boldsymbol{R} = \boldsymbol{R}^T$ are the weighting matrices. The stady state solution is

$$\boldsymbol{u} = -\boldsymbol{R}^{-1} \boldsymbol{B}^T \boldsymbol{P}_{\infty} \boldsymbol{x}.$$

$\boldsymbol{P}_{\infty} = lim_{t \to \infty} \boldsymbol{P}(t)$ is found by solving the *algebraic Riccati equation*:

$$\boldsymbol{P}_{\infty} \boldsymbol{A} + \boldsymbol{A}^T \boldsymbol{P}_{\infty} - \boldsymbol{P}_{\infty} \boldsymbol{B} \boldsymbol{R}^{-1} \boldsymbol{B}^T \boldsymbol{P}_{\infty} + \boldsymbol{C}^T \boldsymbol{Q} \boldsymbol{C} = \boldsymbol{0}$$

In this case, the control problem is formulated as a regulator problem. In order to solve a trajectory tracking problem, it is necessary to define the error vector

$$\boldsymbol{e} = \boldsymbol{y} - \boldsymbol{y}_d,$$

which yields the cost function

$$J = \min_{u} \left\{ \frac{1}{2} \boldsymbol{e}^T(T) \boldsymbol{Q}_f \boldsymbol{e}(T) + \frac{1}{2} \int_{0}^{T} (\boldsymbol{e}^T \boldsymbol{Q} \boldsymbol{e} + \boldsymbol{u}^T \boldsymbol{R} \boldsymbol{u}) \right\},$$

where $\boldsymbol{Q}_f = \boldsymbol{Q}_f^T \geq 0$ can be added to penaltize the error in the final state. The LQR design has the advantage of achieving optimal control with respect to the cost function. The terms in the cost function can be interpreted as the balance of the relationship between fuel consumption (proportional to $\boldsymbol{R}$) versus penalties the error state (proportional to $\boldsymbol{Q}$). A drawback is that the LQR design presuppose a linearized system, and looses its efficiency when operation outside the proximity of the operating point.

State Feedback Linearization

The idea of feedback linearization is to transform a nonlinear system to a linear system by cancelling the nonlinear terms of the dynamic system in the control law. When the system is linearized, conventional control techniques such as optimization methods or pole placement can be applied to the remaining linear system. Consider the nonlinear dynamics of a marine craft, Fossen [2011]:

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}_{\Theta} \boldsymbol{\eta} \boldsymbol{\nu}$$

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

Let the nonlinear dynamics be $\boldsymbol{n}(\boldsymbol{\nu}, \boldsymbol{\eta}) = \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{g}(\boldsymbol{\eta})$, and let the control law cancelling the nonlinear dynamics be

$$\boldsymbol{\tau} = \boldsymbol{M} \boldsymbol{a}^b + \boldsymbol{n}(\boldsymbol{\nu}, \boldsymbol{\eta}),$$

where $\boldsymbol{a}^b$ is commanded body acceleration. Assuming at thrid order reference model providing trajectories for $(\boldsymbol{\eta}_d, \dot{\boldsymbol{\eta}}_d, \ddot{\boldsymbol{\eta}}_d)$, a trajectory tracking PID controller with acceleration feedforward can be chosen as

$$\boldsymbol{a}^n = \ddot{\boldsymbol{\eta}}_d - \boldsymbol{K}_d \dot{\tilde{\boldsymbol{\eta}}} - \boldsymbol{K}_p \tilde{\boldsymbol{\eta}} - \boldsymbol{K}_i \int_{0}^{t} \eta(\tau) d\tau,$$

where $\boldsymbol{a}^n$ is commanded acceleration in NED frame. The control gains $(\boldsymbol{K}_p, \boldsymbol{K}_i, \boldsymbol{K}_d)$ can then be chosen for example according to pole placement.

State feedback linearization is applicable to ships and underwater vehicle since these systems basically are nonlinear mass-damper-spring systems.

Backstepping

The backstepping control design is constructed in a recursive process, starting with the output dynamics, and using control Lyapunov functions (CLF) to design a control law that guarantees lyapunov stability. At each step a new CLF is introduced, and a virtual controller $\boldsymbol{\alpha}_i$ is designed to stabilize the dynamics up to that step. At the last step, the accumulated CLF is stabilized by the control law $\boldsymbol{\tau}$. There are as many steps as there are *relative degrees* in the system. Nonlinear backstepping design and Feedback linearization are strongly related. However, while feedback linearization cancels all nonlinearities in the system, the backstepping design is more flexible, and will for example not cancel a damping term as this is a stabilizing term.

2.4 Maneuvering control problem

This section is based on Skjetne [2005]. The maneuvering control problem is the coupled tasks of

- *Geometric task*
- *Dynamic task*

The *geometric task* concerns for the output y of the system to reach and follow a desired path $y_d(\theta)$ in space, where θ is a path variable left as an extra degree of freedom for the *dynamic task*. The *dynamic task* concerns following desired dynamics along the path and is usually specified as a speed assignment along the path. The path variable θ is used to assign the dynamics along the path. The separation of the tasks implies that the two main tasks can be approached individually. Furthermore it introduces more flexibility in the development of control law due to that desired motion along the path can be shaped by state feedback. The maneuvering problem aims to bridge the gap between the *target tracking problem* and the *path-following problem*. In a *target tracking problem*, the output y should follow a desired point $y_d(t)$ that varies as a function of time. In this case, the *geometric* and *dynamic* task are merged into one task that leads to objectives often more stringent than required. On the other side, there is the *path-following problem* where the output y should follow a desired path $y_d(\theta)$ without dynamic constraints along the path. This can in many cases be too loosely restricted. *Maneuvering* separate the tasks into two tasks. The most important task is the *geometric task*, while the less important task is the *dynamic task*. Separating the problem makes it possible to design geometric and dynamic references individually such that geometric restrictions can be prioritized.

2.4.1 Problem formulation

The maneuvering control problem can be formulated the following way:

1. **Geometric task:** For any continuous function $\theta(t)$ force the output y to converge to the desired path $y_d(\theta)$,
$$\lim_{t\to\infty} |y(t) - y_d(\theta(t))| = 0 \tag{2.11}$$
2. **Dynamic task:** Follow one or more of the following assignments:
   (a) **Time assignment:** Force the path variable θ to converge to a time signal $\upsilon_t(t)$,
$$\lim_{t\to\infty} |\theta(t) - \upsilon_t(t)| = 0 \tag{2.12}$$

(b) **Speed assignment:** Force the path speed $\dot{\theta}$ to converge to a desired speed $\upsilon_s(\theta, t)$,

$$\lim_{t \to \infty} |\dot{\theta}(t) - \upsilon_s(\theta, t)| = 0 \tag{2.13}$$

(c) **Acceleration assignment:** Force the path acceleration $\ddot{\theta}$ to a desired acceleration $\upsilon_a(\dot{\theta}, \theta, t)$,

$$\lim_{t \to \infty} |\ddot{\theta} - \upsilon_a(\dot{\theta}, \theta, t)| = 0 \tag{2.14}$$

2.4.2 Path parametrization

The main task in *maneuering* is to converge to and follow a parametrized path. This motivates the research of parametrization methods.

Piecewise linear path

Consider a piecewize linear path connected by way-points. Each linear path segment has a local path reference frame $\mathcal{R}_i$, starting in the first way-point in the path segment. The x-axis $x_{\mathcal{R}_i}$ points from the first way-point in the path segment to the next way-point, see Figure 2.5.

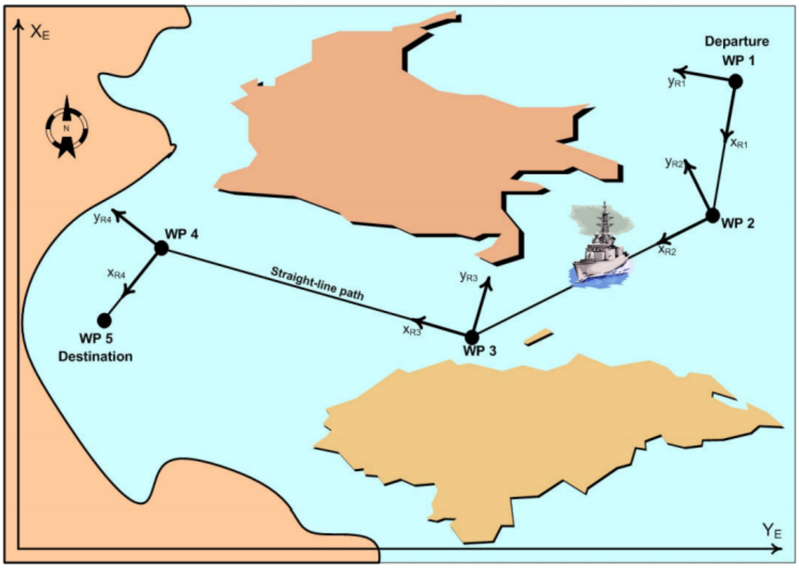


Figure 2.5: Piecewize linear path connected by way-points. Courtesy:Skjetne [2005]

Discrete parametrization In discrete parametrization, the points on the path is expressed in the frame $\mathcal{R}_i$. Considder $p_k = (x_k, y_k)$ as the starting way-point in the current path segment, and $p_{k+1} = (x_{k+1}, y_{k+1})$ is the target way-point in the path segment. (α_k) is the relative angle between the NED frame to the current path segment.

Recall the coordinates in the path-fixed frame from Section 2.3.3:

$$\boldsymbol{\epsilon}(t) = [s(t) \; e(t)]$$

The path is defined for the points p where the cross-track error $e(t)$ is zero, hence

$$\mathcal{P} := \{p \in \mathbb{R}^2 : \; \exists \; i \in \mathcal{I} \; s.t. \; |e(t)| = 0\} \tag{2.15}$$

Here, the path is discretely parametrized by i, where i is the index that denotes the path segment.

Continuous parametrization The continious parametrizarion is defined by

$$p_d(\theta) = \begin{cases} p_1 + \theta(p_2 - p_1); & \theta \in [0,1) \\ p_2 + (\theta - 1)(p_3 - p_2); & \theta \in [1,2) \\ \vdots & \\ p_i + (\theta - i + 1)(p_{i+1} - p_i); & \theta \in [i-1, i) \\ \vdots & \\ p_{n-1} + (\theta - n + 2)(p_n - p_{n-1}); & \theta \in [n-1, n). \end{cases} \tag{2.16}$$

Hence, the path is given by

$$\mathcal{P} := \{p \in \mathbb{R}^2 :\ \exists\, \theta \in [0,n)\ s.t.\ p = p_d(\theta)\} \tag{2.17}$$

The path is parametrized by the path variable θ.

Hybrid parametrization In the case of hybrid parametrization, each path segment is identified by an index i, and each path segment is parametrized by $\theta \in [0,1)$.

$$\mathcal{P} := \{p \in \mathbb{R}^2 :\ \exists\, i \in \mathbb{I}\ and\ \theta \in [0,1)\ s.t.\ p = p_d(i,\theta)\} \tag{2.18}$$

2.5 Sensors

In order to successfully perform the given tasks autonomously, it is important to have good estimates of the vehicle pose relative to navigation markers and tasks. To make this possible, accurate sensors and robust and accurate estimation methods are necessary.
It is necessary to know the pose of the sensors in order to utilize their measurements. That is because measurements are sampled in the respective sensor frame, and needs to be converted to a common frame to make sense in terms of a state estimate. The pose of a sensor defines the origin and axis of the corresponding sensor frame. In Table 2.1, the different frames and their notation are presented.

notation frame	**Frame name**
{b}	Body frame
{m}	IMU frame
{d}	DVL frame
{pg}	Pressure gauge frame
{n}	North East Down (NED) frame

Table 2.1: Frames and their notations

Table 2.1 present frames and their respective notations.

2.5.1 Inertial Measurement Unit (IMU)

Figure 2.6: The implemented IMU - STIM300. Courtesy:sen

The IMU includes three accelerometers that measures specific force in 3-DOF (see equation (2.19a)), three Angular Rate Sensor (ARS), that measures angular rate in 3-DOF (see equation (2.19b)), and three magnetometers that measure magnetic field components in 3-DOF (see equation (2.19c)). A stand alone solution with IMU, where the output from the accelerometers are integrated twice to obtain position and gyro outputs are integrated once to obtain attitude will drift due to sensor biases, misalignment's and temperature variations Fossen [2011].
The measurements from the IMU are given by:

$$\boldsymbol{a}^m_{imu} = \dot{\boldsymbol{v}}^m_{m/n} + \omega^m_{m/n} \times \nu^m_{m/n} - R^m_n(\Theta_{nm})g^n + b^m_{acc} + w^m_{acc} \tag{2.19a}$$

$$\boldsymbol{\omega}^m_{imu} = \omega^m_{m/n} + b^m_{gyro} + w^m_{gyro} \tag{2.19b}$$

$$\boldsymbol{m}^m_{imu} = R^m_b(\Theta_{nm})\boldsymbol{m}^n + b^m_{mag} + w^m_{mag} \tag{2.19c}$$

The IMU that will be implemented is called STIM300. This is a high-performance which Vortex NTNU came to possession of through a coorperation with Sensonor. This unit consist of three Micro-Electro-Mechanical System (MEMS), three accelerometers and three inclinometers.

2.5.2 Doppler Velocity Log (DVL)

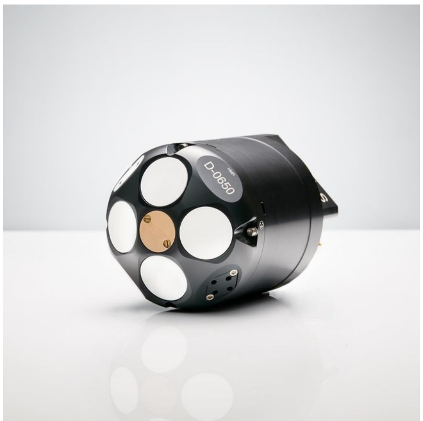


Figure 2.7: The implemented accoustic sensor - DVL1000. Courtesy:NORTEK [2016]

The DVL an acoustic sensor that estimates velocity relative to sea bottom. This is achieved by sending a long pulse along with the minimum of three acoustic beams in different directions. This produces translation velocity estimates that can be converted into the DVL-frame {d}, NORTEK [2019]. The DVL provides high accuracy measurements, and has low weight. Vortex NTNU borrow the DVL1000 from Nortek through a sponsorship. This DVL outputs a three dimensional velocity vector estimate relative to the ground in the frame of the DVL. The measurement is given by

$$\nu^d_{d/n} = R^d_b(\Theta_{bd}) R^b_m(\Theta_{bm})(\boldsymbol{\nu}^m_{m/n} + \boldsymbol{\omega}^m_{m/n} \times r^m_{dvl/m}) + w^d_{dvl} \tag{2.20}$$

$\boldsymbol{\nu}^d_{d/n}$ is the measured velocity of the DVL, $\boldsymbol{\nu}^m_{m/n}$ is the linear velocity of the imu with respect to NED, $\omega^m_{m/n}$ is angular velocity in {m} and $r^m_{dvl/m}$ is the position of the DVL relative to the IMU.

2.5.3 Pressure gauge

The pressure gauge measure the pressure and use this measurement to calculate the water depth. The measurement provided from the pressure gauge is given by the following equations:

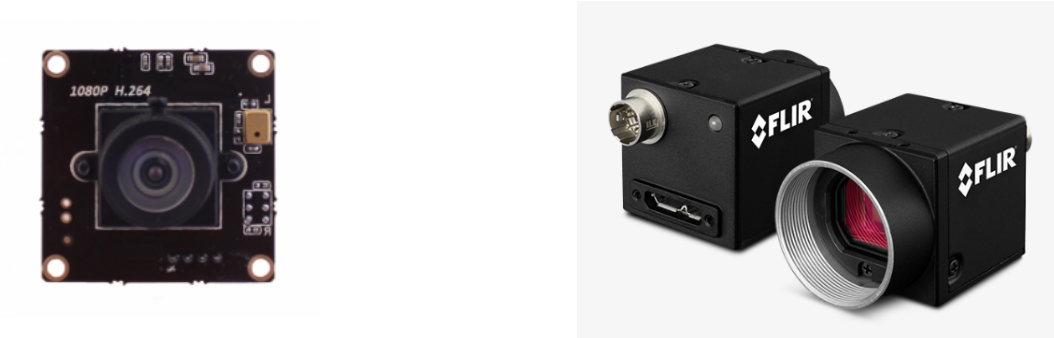
$$p_{pg} = p_{atm} + \rho g z^n_{pg/n} + w_{pg} \tag{2.21a}$$

$$z^n_{pg/n} = z^n_{m/n} + [0\ 0\ 1]\boldsymbol{R}^n_b(\boldsymbol{\Theta})\boldsymbol{r}^m_{pg/m}, \tag{2.21b}$$

where p_{pg} is measured pressure in the {pg} frame, p_{atm} is atmospheric preassure, $\rho g z^n_{pg/n}$ is static preassure and w_{pg} is the preassure gauge noise. $z^n_{pg/n}$ is the depth of the preassure gauge, and $z^n_{m/n}$ is the depth of the IMU, Dukan and Sørensen [2013]. The relevant preassure sensor is a part of the DVL - DVL1000 from Nortek.

2.5.4 Camera

On the AUV platform Manta, there are mounted two cameras, one front camera directed along positive x-axis in surge to observe objects ahead, and one ground camera directed down along negative heave axis to observe objects on the ground. The front camera - Low-Light HD USB Camera is delivered by BlueRobotic. The ground camera is a Blackfly USB3 delivered by Flir.



(a) Front camera. Courtesy:BlueRobotics [2019b]

(b) Ground camera. Courtesy:Flir [2019]

Figure 2.8: Front camera and bottom camera.

2.5.5 Acoustic system

Given acoustic signals sent out from an object to indicate the direction to that object . An acoustic system with four hydrophones on the AUV will then estimate the direction of the signal by triangulating the signal between the hydro phones. This is done by calculating the relative time difference between when signal was observed in the different hydrophones.

2.6 Robosub competition

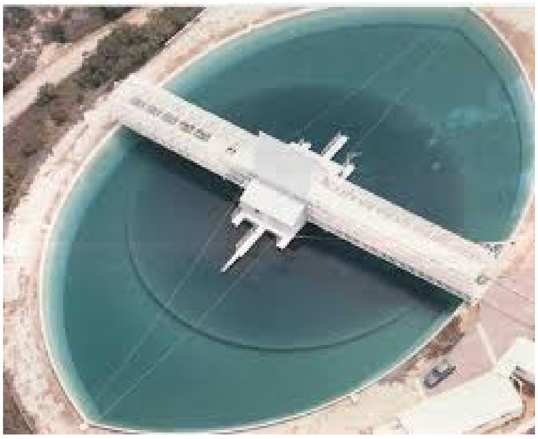


Figure 2.9: TRANSDEC facility. Courtesy:RoboSub [2019]

Robosub is a competition for students from all around the world where teams design and build their own Autonomous underwater vehicle (AUV).The underwater drones aims to navigate through through different tasks that should be performed autonomously. The tasks are designed with the intent to mimic ongoing research in the field of Autonomous Underwater Systems. The theme for this years competition is set to be Vampire, hence tasks are related to this. This entire section is mainly based on RoboSub [2019]

Environment and mission

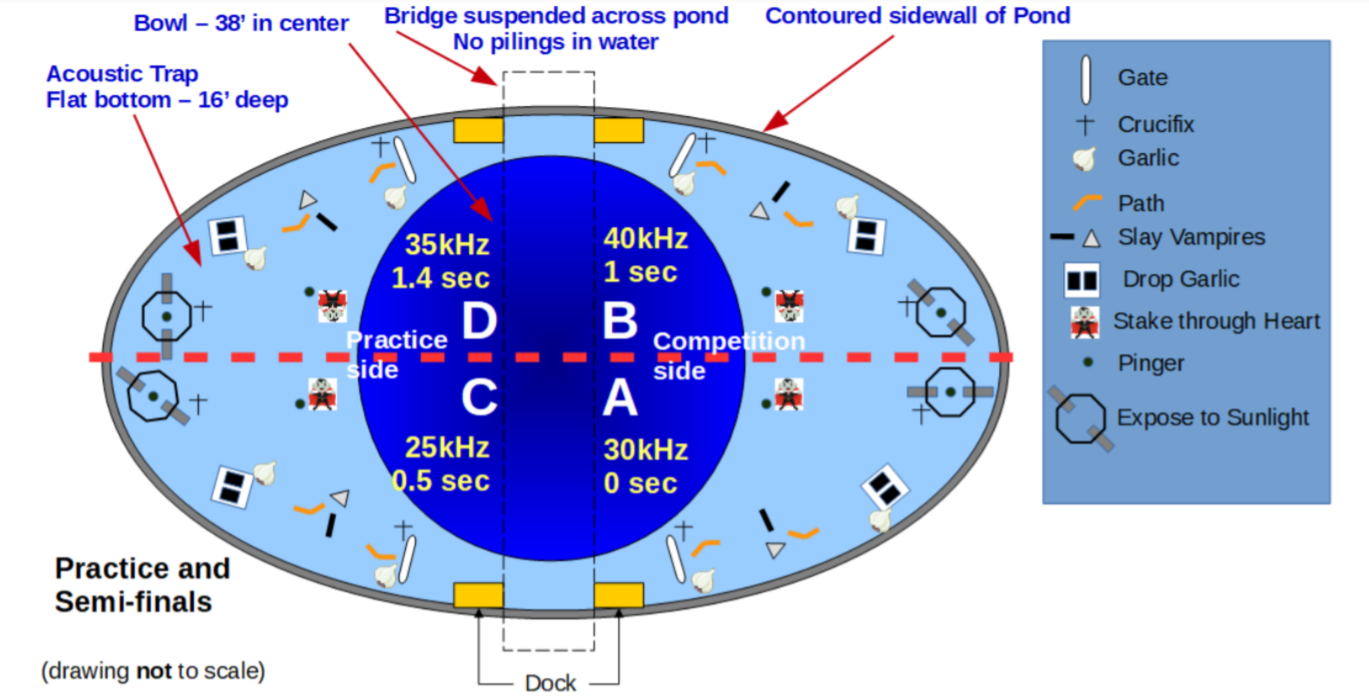

Bowl – 38' in center
Bridge suspended across pond
No pilings in water
Contoured sidewall of Pond
Acoustic Trap
Flat bottom – 16' deep
35kHz
1.4 sec
D
40kHz
1 sec
B
Practice side
Competition side
C
A
25kHz
0.5 sec
30kHz
0 sec
Practice and Semi-finals
(drawing **not** to scale)
Dock
Gate
Crucifix
Garlic
Path
Slay Vampires
Drop Garlic
Stake through Heart
Pinger
Expose to Sunlight

Figure 2.10: The task environment. The robot will operate in a quarter of the basin. Courtesy:RoboSub [2019]

The competition takes place at the Transducer Evaluation Center (TRANSDEC) facility in San Diego, California.

The mission consists of five sub-tasks. The mission in the semi-finals and final round will be the same, but the tasks will be located at different places in the basin, see figure 2.10.

Task one: Launching robot and enter the gate From the starting dock (yellow area in figure 2.10), the teams are allowed to self determine their initial heading. However, the teams will get additional points if they let a coin flip decide the initial heading. If the coin lands on heads the vehicle initial heading is perpendicular to the dock, and if the coin lands on tails the initial heading of the vehicle is parallel to the dock and pointing away from the gate, see figure 2.11.

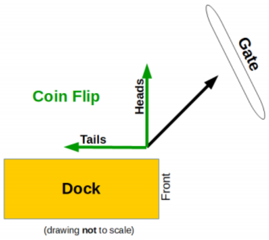

Gate
Coin Flip
Heads
Tails
Dock
Front
(drawing **not** to scale)

Figure 2.11: Initial heading if coin flip is chosen. Courtesy:RoboSub [2019]

The gate is a black pipe. It is buoyant, moored to the bottom and will float just below the surface. The vertical legs are orange. The vehicle is allowed to pass through the gate at any depth from the pool bottom to below the gate. The gate is divided into two sections, where the smallest sections make up

40 % of the gate length (and the largest section correspondingly makes up 60 %). More points will be handed out if the vehicle passes through the smallest section. For illustration with measurements, see figure 2.12.

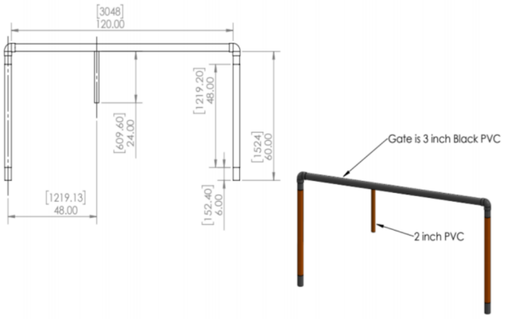


Figure 2.12: Computer-Aided Design (CAD) drawing of gate with measures.Specified in millimeters in the square brackets. Specified in inches outside the brackets. Courtesy:RoboSub [2019]

In addition, style points can be gained. For every 90 °change in orientation, the vehicle will gain points. Extra points will be handed out for up to eight 90 °changes. Returning to previous orientation after a 90 °will not give additional points.

Follow path markers Path markers are placed out in the environment according to figure **??**, represented as path. These path markers consist of two orange sections, where the second section is rotated ± 45 °on the first section, see figure 2.13

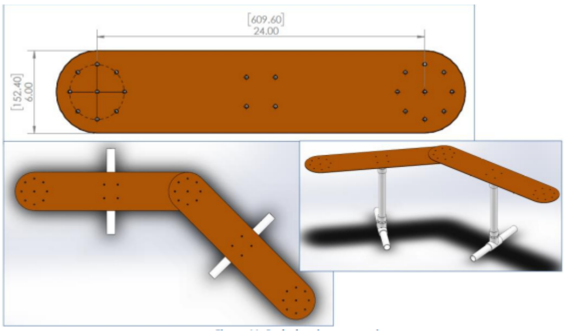


Figure 2.13: Path markers with dimensions. Specified in millimeters in the square brackets. Specified in inches outside the brackets. Courtesy:RoboSub [2019]

One path marker is placed just after task one, and another one is placed after task two. If the vehicle follow the path markers, the corresponding team will receive points.

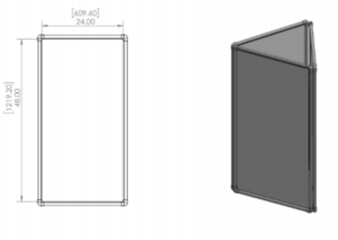


Figure 2.14: CAD drawing of buoys with dimensions. Specified in millimeters in the square brackets. Specified in inches outside the brackets. Courtesy Robosub RoboSub [2019]

Task two: touching buoys Two buoys are moored to the floor see figure 2.14. One of the buoys are two sided. The other buoy is three-sided The three-sided buoy will rotate between 1 and 5 Rounds Per Minute (RPM).

Points are awarded for for touching any buoy. Maximum points are awarded for touching the two-sided buoy and a pre-defined side of the three-sided buoy. Which side to touch should be determined before launching the AUV.

Task three: drop markers in bin Two bins are held together in a frame. There is always one bin open and one bin closed. the lever on the side can be moved to open the closed bin and correspondingly close the open bin. Points are awarded for leaving markers in the bin. Points are also awarded for moving the lever. Maximum points are given if the markers are put in the initially closed bin.

Task four: Shoot torpedoes The task consist of a vertical board divided into two sections. The upper section has two cutouts, whereas one section is open, and the other section is covered. There is a lever that can be moved to open one oval and close the other.

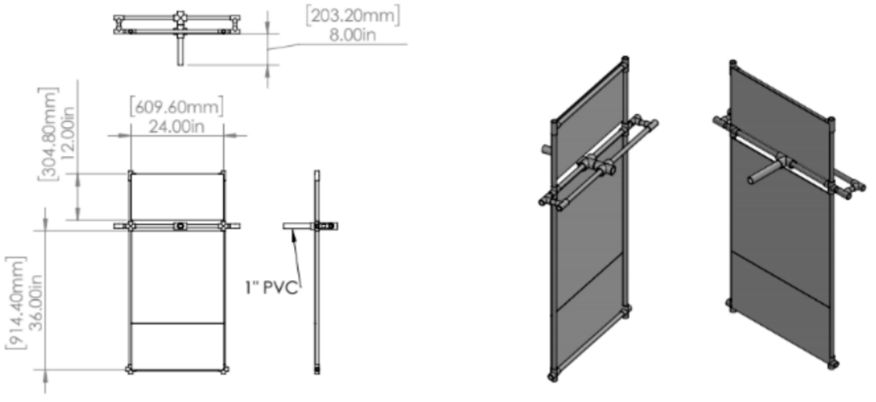


Figure 2.15: CAD drawing of the vertical board. oval-, and heart shaped targets are not marked on the drawing. Measures specified in millimeters in the square brackets. Specified in inches outside the brackets. Courtesy:RoboSub [2019]

In the bottom section, there is a heart-shaped cutout that is always open.
Points are awarded for firing torpedoes through the open oval and for firing torpedoes through the heart shape. Maximum amount of points are awarded for shooting one torpedo through the heart and move

the lever to open the initially closed oval and then shoot a torpedo through this oval shaped opening. An aucostic pinger is located close to the board so help navigate to the task.

Task five: Enter octagon On the surface an octagon-shaped shell is located. In the center of this octagon, an acoustic pinger is located to aid the AUV navigate to the task.

2.7 Vortex NTNU

Vortex NTNU is an independent student organisation at Norwegian University of Science and Technology (NTNU) aiming to compete in Robosub. The team is composed of students from different engineering disciplines from bachelor and master degree programs. Vortex NTNU was initially developing an Remotely operated vehicle (ROV) to compete in the Marine Advanced Technology Education (MATE) ROV competition. In 2019 Vortex NTNU is competing in the AUV competition RoboSub. This is the first time Vortex NTNU competes in an AUV competition. The same experimental platform-MANTA is used, but new sensor technology and software required for autonomous vehicles will be developed. See: NTNU [2019]

Organisational structure

Due to the drastic change from ROV to AUV,the organization needed to prioritize differently when it comes to competence and objectives for the project. See NTNU.

Hardware Hardware group is reponsible for design and developement of physical and electronic structure of the AUV.

Control Control group is responsible the control software of the AUV, making it able to solve tasks presented. This encompasses state machine, path and mission planning as well as designing controllers.

Perception Perception group is responsible for collecting data to complete the mission. Sensor fusion, acoustics, computer vision and SLAM are topics that concern the perception group.

Chapter 3

Problem formulation

3.1 Simulation Model

Due to the extent of the model derivation it was decided to dedicate a chapter to this. This section will summarize the main aspects of the model derivation and the assumptions made. For more complete explenations, see Chapter 4. The 6 DOF nonlinear marine craft equation of motion can be written as

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

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{g}(\boldsymbol{\eta}) = \boldsymbol{\tau} + \boldsymbol{\tau_{wind}} + \boldsymbol{\tau_{wave}}. \tag{3.1b}$$

Here, $\boldsymbol{M}$ is the total mass matrix, $\boldsymbol{C}(\boldsymbol{\nu})$ is the Coriolis centripetal matrix, $\boldsymbol{D}(\boldsymbol{\nu})$ is the damping matrix, $\boldsymbol{g}(\boldsymbol{\eta})$ are restoring forces, $\boldsymbol{\tau}$ are the control forces and moments, and $\boldsymbol{\tau_{wind}} + \boldsymbol{\tau_{wave}}$ are environmental forces. Since AUV operates in a basin without external forces, $\boldsymbol{\tau_{wind}}$ and $\boldsymbol{\tau_{wave}}$ can be disregarded. Furthermore, the AUV will operate in a *low-speed regime*, which is defined by operational speed within the interval $[0,\ U_{DP})$, where $U_{DP} = 1.5m/s$. In this speed range, a linear speed independent hydrodynamic model is valid (Fossen [2005]).

This results in a 6 DOF linear equation:

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}_{\theta}(\boldsymbol{\eta})\boldsymbol{\nu} \tag{3.2a}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{D}\boldsymbol{\nu} + \boldsymbol{G}\boldsymbol{\eta} = \boldsymbol{\tau} + \boldsymbol{b}. \tag{3.2b}$$

The uncertainties in the estimation methods of system parameters makes this model less suited as a test platform of the control system of the real AUV. However, as long as the dynamics are considered to be linear, the simulation model will work as a platform to test the effects of different tuning parameters in the control and guidance system. Further, it will be possible to test the robustness of the model based controller by changing model parameters.

3.2 Control Design Model

For the design of the *motion control system*, a reduced-order or simplified version of the simulation model, the Control Design Model (CDM) is used (Fossen [2011]). The AUV will be controlled in 4 DOF by two decoupled control designs. One of the controllers will be a maneuvering controller to follow a path with a speed assignment in the North-East plane, and the other controller will regulate the depth. This motivates two Control Design Models:

- **CDM for maneuvering controller:** 3DOF CDM representing the dynamics in surge, sway and yaw.
- **CDM for depth controller:** 1DOF CDM representing the heave dynamics.

3.2.1 CDM for maneuvering controller

The 3 DOF CDM assumes that the roll and pitch angles are fixed to zeros ($\theta \approx \phi \approx 0$). This is justified by the assumption that the AUV is highly metacentric stable (see Chapter 4). If this was not the case, this could be solved by additional controllers regulating θ and ϕ to zero. Furthermore, the depth is assumed a fixed. This is fulfilled due to the depth controller. The CDM is given by the following equations (Skjetne [2019b]) :

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

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{D}\boldsymbol{\nu} = \boldsymbol{\tau} + \boldsymbol{b}, \tag{3.3b}$$

where $\boldsymbol{\eta} = [N\ E\ \psi]^T \in \mathbb{R}^2 \times \mathcal{S}$, $\boldsymbol{\nu} = [u\ v\ r]^T \in \mathbb{R}^3$. Furthermore, we have that,

$$\boldsymbol{M} = \begin{bmatrix} M_{11} & M_{12} & M_{16} \\ M_{21} & M_{22} & M_{26} \\ M_{61} & M_{62} & M_{66} \end{bmatrix},$$

where M_{ij}, $\{i, j\} = \{1, 2, 6\}$ are extracted from the 6 DOF mass matrix

$$\boldsymbol{D} = \begin{bmatrix} D_{11} & D_{12} & D_{16} \\ D_{21} & D_{22} & D_{26} \\ D_{61} & D_{62} & D_{66} \end{bmatrix},$$

where D_{ij}, $\{i, j\} = \{1, 2, 6\}$ are extracted from the 6 DOF Damping matrix

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_{surge} \\ \tau_{sway} \\ \tau_{yaw} \end{bmatrix},$$

where $\boldsymbol{\tau}$ is the 3 DOF control output in body frame

$$\boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_6 \end{bmatrix},$$

where b is the bias due to unmodeled dynamics. For nummerical values of system matrices $\boldsymbol{M}$ and $\boldsymbol{D}$, see Chapter 4. As the objective is to control all three DOF, full actuation is required. This is fulfilled by the thrust allocation of the AUV, which is actuated in all 6 DOF.

3.2.2 CDM for depth control

For depth control, a 1 DOF CDM in heave is required. This CDM require that depth can be controlled independently of the other DOF. This is fulfilled since $\theta \approx \psi \approx 0$, and that heave motion is decoupled from the other DOF. The reulting CDM becomes:

$$M_{33}\dot{w} + D_{33}w = \tau_{heave} + b_3, \tag{3.4}$$

where M_{33} is the mass term in heave, D_{33} is the damping term, τ_{heave} is the control output in heave, w is the velocity in heave and b_3 is a bias term. Note that there is no stiffness term in this equation. This is due to neutral buoyancy.

3.3 Hardware

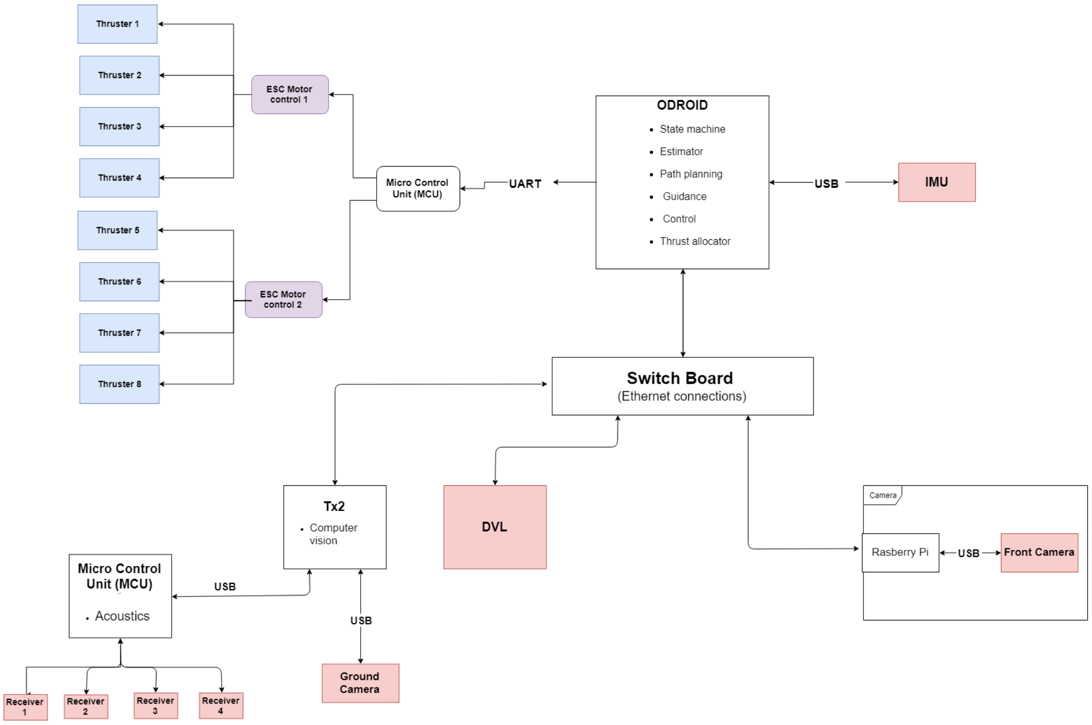

Thruster 1
Thruster 2
Thruster 3
Thruster 4
Thruster 5
Thruster 6
Thruster 7
Thruster 8
ESC Motor control 1
ESC Motor control 2
Micro Control Unit (MCU)
UART
ODROID
State machine
Estimator
Path planning
Guidance
Control
Thrust allocator
USB
IMU
Switch Board
(Ethernet connections)
Tx2
Computer vision
DVL
Camera
Rasberry Pi
USB
Front Camera
Micro Control Unit (MCU)
Acoustics
USB
USB
Ground Camera
Receiver 1
Receiver 2
Receiver 3
Receiver 4

Figure 3.1: Communication drawing of the robotic system. Sensors are marked in red, thrusters are marked in blue, motor controllers are marked in violet and MCU-, SBC and the switch board are marked in white. The Figure is developed in cooperation with other members of Vortex.

High level computation

The system is equipped with two Single Board Computer (SBC) dedicated to solve the high-level algorithm computations.
Odroid UX4 is a SBC with a powerful CPU. It is running on the operation system Ubuntu 16.04. The powerful CPU makes the unit fit to handle a variety of tasks in real time systems. The Odroid does the calculations on most of the modules related to the autonomous system. This includes the high level mission planning from the deliberative layer, path planning, path generation and guidance layer from the executive layer and low-level control algorithms and thrust allocation as well as state estimation from the reactive layer.
Tx2 Is a SBC with a 256-core NVIDIA Pascal Graphics Processing Unit (GPU). This powerful GPU makes Tx2 suitable for processing large amounts of data fast. This is why the computer vision algorithms run on this unit. The Tx2 also publish information from the acoustic system on the Robot Operating System (ROS) network. The image stream from the ground camera is encoded in the Tx2 before it is analysed in computer vision algorithms.

Low-level computers

MCU - Acoustics: The MCU is connected to the four hydrophones. The signals from these hydrophones are sampled at the MCU. These signals are then treated to estimate the direction of the acoustic signal.
Rasberry pi - Front camera: The Rasberry Pi encodes the image stream from the front camera and publish it on the ROS network.

ESC regulates the speed of the motors that drives the thrusters. There are two motor controllers connected to four thrusters each.

Thrusters

Figure 3.2: Thrusters mounted on Manta. Courtesy:BlueRobotics [2019a]

There are eight thrusters mounted on the platform. The thrusters are of the type *T200 BlueROV2*. Each thruster are connected to a voltage of 12 Volt.

Table 3.1: Maximum thrust - Forward and reverse. Courtesy:BlueRobotics [2019a]

Direction	**kg f**	**N**
Maximum Forward Thrust, 12V	3.55	34.8
Maximum Reverse Thrust, 12V	3.0 kg	29.4

Table 3.1 is showing that the maximum forward and reverse thrust for a T200 thruster.

The thrust configuration matrix is given by

$$\boldsymbol{T} = \begin{bmatrix} T_{surge} \\ T_{sway} \\ T_{heave} \\ T_{roll} \\ T_{pitch} \\ T_{yaw} \end{bmatrix} = \begin{bmatrix} 0.7 & 0.0 & 0.0 & -0.7 & -0.7 & 0.0 & 0.0 & 0.7 \\ -0.7 & 0.0 & 0.0 & -0.7 & 0.7 & 0.0 & 0.0 & 0.7 \\ 0.0 & 1.0 & 1.0 & 0.0 & 0.0 & 1.0 & 1.0 & 0.0 \\ 0.0 & 0.1 & 0.1 & 0.0 & 0.0 & -0.1 & -0.1 & 0.0 \\ 0.0 & -0.1 & 0.1 & 0.0 & 0.0 & 0.1 & -0.1 & 0.0 \\ -0.3 & -0.0 & -0.0 & 0.3 & -0.3 & 0.0 & 0.0 & 0.3 \end{bmatrix}$$

Where negative sign implies that the thruster is directed in the opposite direction of the force axis. The thrust configuration matrix, combined with Table 3.1 gives the maximum thrust in each direcion:

$$\boldsymbol{T}_{max,Forward} = \begin{bmatrix} T_{surge,max} \\ T_{sway,max} \\ T_{heave,max} \\ T_{roll,max} \\ T_{pitch,max} \\ T_{yaw,max} \end{bmatrix} = \begin{bmatrix} 90.8\text{N} \\ 90.8\text{N} \\ 139.2\text{N} \\ 15.4\text{NM} \\ 15.4\text{NM} \\ 37.2\text{NM} \end{bmatrix} \tag{3.5a}$$

$$\boldsymbol{T}_{max,Reverse} = \begin{bmatrix} T_{surge,max} \\ T_{sway,max} \\ T_{heave,max} \\ T_{roll,max} \\ T_{pitch,max} \\ T_{yaw,max} \end{bmatrix} = \begin{bmatrix} -90.8\text{N} \\ -90.8\text{N} \\ -117.6\text{N} \\ -15.4\text{NM} \\ -15.4\text{NM} \\ -37.2\text{NM} \end{bmatrix} \tag{3.5b}$$

3.4 Resulting autonomous system

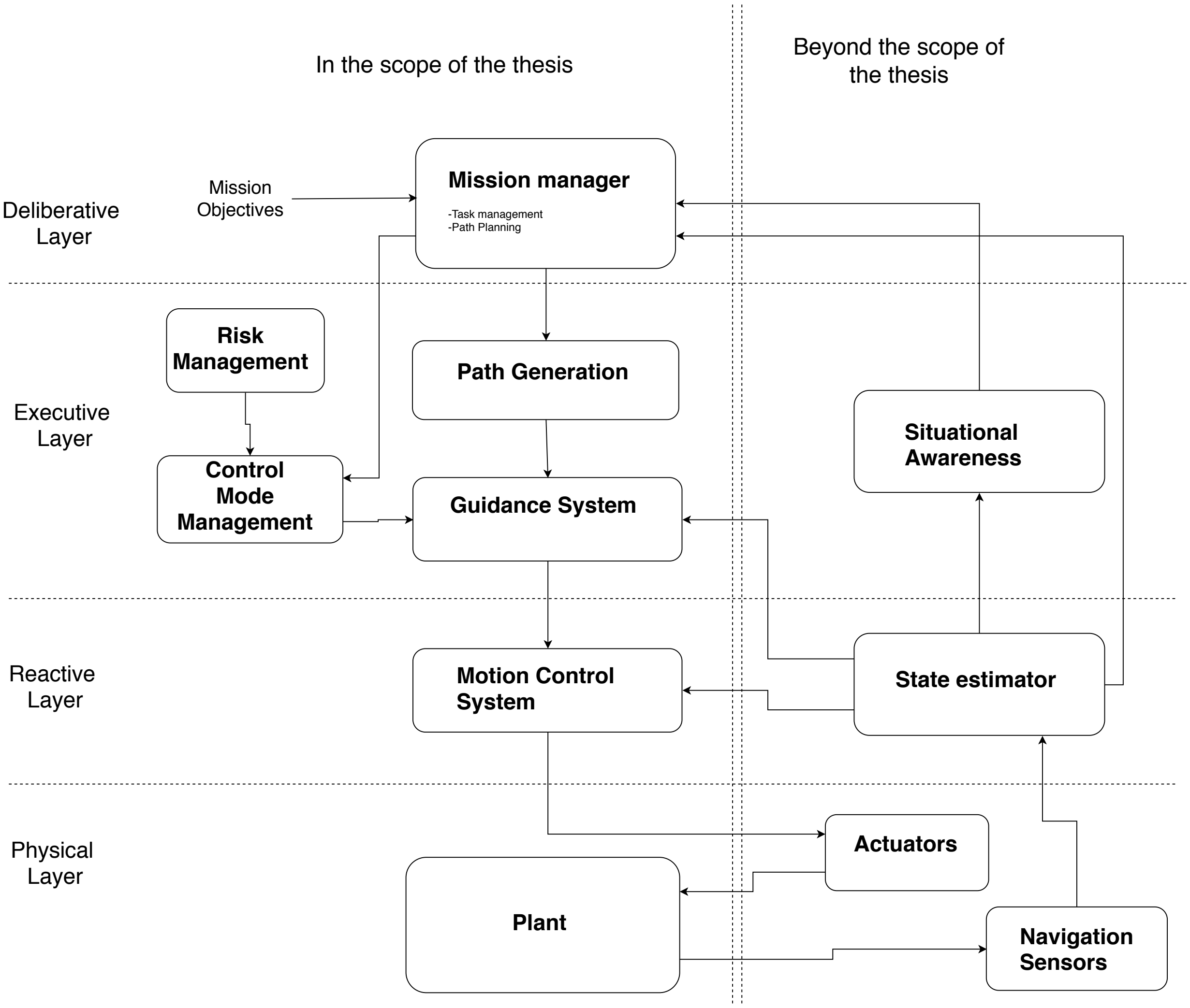

In the scope of the thesis
Beyond the scope of the thesis
Deliberative Layer
Executive Layer
Reactive Layer
Physical Layer
Mission Objectives
Mission manager
-Task management
-Path Planning
Risk Management
Path Generation
Control Mode Management
Guidance System
Situational Awareness
Motion Control System
State estimator
Actuators
Plant
Navigation Sensors

Figure 3.3: Graphical representation of the resulting autonomous system consistent with the three layered architecture presented in Section 2.1. The graph is divided by a double dotted line indicating the scope of the thesis. Directed arrows connecting the system modules indicate the direction of the signals.

Figure 3.3 show the resulting autonomous system. Included in the scope of the thesis are the following modules:

- **Supervisory system:** This system is divided into three modules distributed over two layers. These are The *Mission Manager*, which is included in the deliberative layer, *Risk Manager* and *Control Mode Manager*, both accosiated to the executive layer.

- **Path Generation:** Generate reference path for the guidance system. Part of the executive layer.
- **Guidance system:** Generating reference signals to the controller and calculate dynamics of path variable according to the *Maneuvering problem*. Included in the executive layer.
- **Motion control system:** Calculates thrust to ensure that control objective is fulfilled. Included in the Reactive layer.
- **Physical plant:** In this thesis, a linear model of the dynamics of the AUV is derived. The plant is in the physical layer.

3.5 Problem statement

3.5.1 Supervisory system

Mission manager

The mission manager represents the highest level of autonomy in the autonomous architecture structure. This module is to determine and schedule the overall objectives of the craft during operation. This includes

- Determining target way-points according to mission objectives, the location of the craft and perceived information
- Scheduling tasks

Path Planning Since paths are only planned one or a few way-points at a time, the purpose of the path planning method is simple. The path planning method should generate a list of way-points based on perceived targets and sub-targets that should be used by the path generation to generate smooth paths.

Risk Management system

In order to handle possible dangerous situations, a risk management system is incorporated into the supervisory system. This system should handle dynamic objects that might be at collision course with the craft. This is done by stopping the vessel and wait for the obstacle to go out of a predefined danger zone.

Control Mode Management

The control mode management system anticipate to manage changes in control objective. This is relevant when switching from dynamic positioning to maneuvering or vice versa.

3.5.2 Path Generation

The path generation algorithm takes way-points generated in the path planning as input to generate parametrized path segments between these way-points. By requiring bumpless transfers in the way-points where the switching of path segment happens, the path needs to be $\mathcal{C}^3$ continuous.The path is given by

$$\boldsymbol{p}_d(s) = [x_d(s)\ y_d(s)]^T,$$

where s is a continuous path parameter, $s \in [0,\ i)$, i is the number of path segments.

3.5.3 Maneuvering problem

The guidance and control problem should be defined as a maneuvering problem according to Skjetne [2005]. The problem is divided into a geometric task and a dynamic task:

Geometric task: Force the output of the system $\boldsymbol{\eta}(t) = [x\ y\ \psi]^T$ to converge to and follow a desired pose $\boldsymbol{\eta}_d(s) = [\boldsymbol{p}_d(s)\ \psi_d(s)]^T$ defined by parametrized path $\boldsymbol{p}_d(s) = [x_d(s)\ y_d(s)]^T$ and its tangent heading $\psi_d(s) = arctan\left(\frac{y_d^s(s)}{x_d^s(s)}\right)$:

$$\lim_{t\to\infty} |\boldsymbol{\eta}(t) - \boldsymbol{\eta}_d(\theta)| = 0 \tag{3.6}$$

Dynamic task: The dynamic task for this control problem is chosen to follow a speed assignment along the parametrized path. Therefore the objective is to force the speed of the path parameter $\dot{s}(t)$ to converge to and follow a speed assignment $\upsilon_s(s) = \frac{u_d(s)}{p_d^s(s)}$ along the path $|\boldsymbol{p}_d(s)|$. $u_d(s)$ is the desired speed along the path, and $|\boldsymbol{p}_d^s(s)|$ is the length of the tangent vector at a given point the path $\boldsymbol{p}_d(s)$. This way the desired speed $u_d(s)$ is normalized to the path speed $\dot{s}$

$$\lim_{t\to\infty} |\dot{s}(t) - \upsilon_s(s(t), t)| = 0 \tag{3.7}$$

$\upsilon_s(s(t), t)$ is the speed assignment given by

$$\upsilon_s(s(t), t) = \frac{u_d(t, s)}{|\boldsymbol{p}_d^s(s)|}$$

$u_d(t, s)$ is the desired speed that will be designed according to the path, whether the path segment is the first after a stop or last before a stop, and according to speed limits. $|p_d^s(s)|$ is the length of the tangent at any desired point of the path, and is included in the equation to scale the speed assignment in terms of the speed of the path parameter.

3.5.4 Depth control: Tracking problem

The most simple way of defining the control objective for depth control is as a regulation problem. This involves regulating the depth to a constant value. However, this can be problematic when large steps in desired value occurs, for example changing depth reference from 1 meter depth to 5 meters depth. One way of dealing with this is to send the reference value through a reference model:

$$\frac{z_d}{r}(s) = h_{lp}(s),$$

where z_d is reference sent to the depth controller, r is the desired steady state desired depth, $h_{lp}(s)$ is a low pass filter and s is the laplace variable. In this case, the control problem is redefined to a tracking problem, where the objective is to track a reference trajectory. By letting the reference model be of third order, reference trajectories can be generated in both position, velocity and acceleration. This motivates for a PID controller with acceleration feedback. Hence the objective is to find a control law τ_{heave} so that

$$\lim_{t\to\infty} |z(t) - z_d(t)| = 0 \tag{3.8}$$

Where $z(t)$ is the depth and $z_d(t)$ is the desired trajectory in depth.

3.5.5 Performance measures

The performance of the system will be measured according to three measures:

- Time: Minimize time will give points in the competition
- Path following - Following desired path with high accuracy gives extra points in some tasks (path marker f.eks) and important to avoid collisions in certain areas.
- Heading following: The heading must follow the tangent of the path since then the camera points along the path. This is important to detect hazards and targets.

Chapter 4

Simulation model

4.1 Dynamics

This section i based on Fossen [2011]. Dynamics is divided into two parts, *kinematics* which concerns the geometrical aspects of motion, and *kinetics* which is about the forces causing the motion.

4.1.1 Kinematics

For the motion of a marine craft, there are six DOF, which means that independent coordinates are required to determine the pose of the vessel. The three first coordinates and their time derivatives correspond to position in the three dimensional space and translational motion along these axis. The last three coordinates and their time derivatives represent orientation and rotational motion about the axis in three dimensional space.

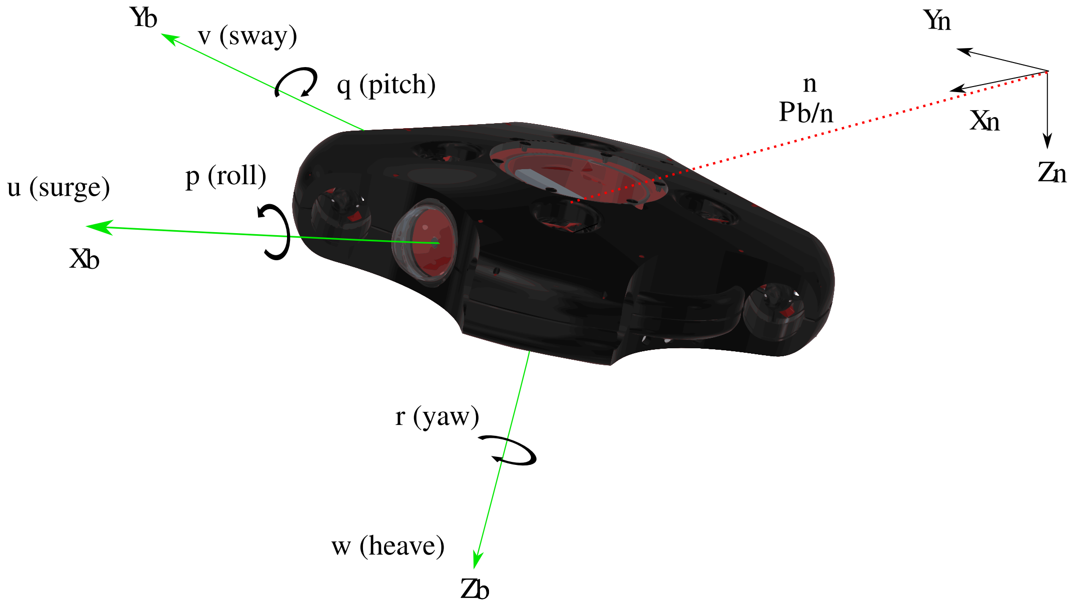


Figure 4.1: Manta with axis in NED frame (axis denoted with n subscript) and Body frame(axis denoted with b in subscript). Body-fixed linear velocity (u, v, w) and body-fixed angular-velocity (p, q, r) is marked on the drawing.

Note the following definitions:

$$\boldsymbol{p}^n_{b/n} = \begin{bmatrix} N \\ E \\ D \end{bmatrix} \in \mathbb{R}^3 \tag{4.1a}$$

$$\boldsymbol{\theta}_{nb} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \in \mathcal{S}^3 \tag{4.1b}$$

$$\boldsymbol{v}^b_{b/n} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \in \mathbb{R}^3 \tag{4.1c}$$

$$\boldsymbol{\omega}^b_{b/n} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \in \mathbb{R}^3 \tag{4.1d}$$

$$\boldsymbol{\epsilon}^p_{b/p} = \begin{bmatrix} s \\ e \end{bmatrix} \in \mathbb{R}^2 \tag{4.1e}$$

Terms explained:

- $p^n_{b/n}$ - NED position of the body frame
- $\boldsymbol{\theta}_{nb}$ - Euler angles between body frame and NED frame
- $\boldsymbol{v}^b_{b/n}$ - Linear velocity of body frame with respect to the NED frame, expressed in the body frame (Body-fixed linear velocity)
- $\boldsymbol{\omega}^b_{b/n}$ - Angular velocity of body frame with respect to the NED frame, expressed in the body frame (Body-fixed angular velocity)
- $\boldsymbol{\epsilon}^p_{b/p}$ - position of the body frame $\{b\}$ with respect to the path p expressed in the path frame $\{p\}$
- $\mathbb{R}^2$ - Euclidean space of two dimensions
- $\mathbb{R}^3$ - Euclidean space of three dimensions
- $\mathcal{S}^3$ - Manifold of a sphere

The resulting pose vector is

$$\boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{p}^n_{b/n} \\ \boldsymbol{\theta}_{nb} \end{bmatrix}$$

The resulting velocity vector is

$$\boldsymbol{\nu} = \begin{bmatrix} \boldsymbol{v}^b_{b/n} \\ \boldsymbol{\omega}^b_{b/n} \end{bmatrix}$$

Rotations

The relation between linear and angular velocities in body-frame $\boldsymbol{\nu}$ and NED frame can be expressed as

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}_\Theta(\boldsymbol{\eta})\boldsymbol{\nu} \tag{4.2}$$

Where the transformation matrix $\boldsymbol{J}_\theta(\boldsymbol{\eta})$ is expressed as:

$$\boldsymbol{J}_\Theta(\boldsymbol{\eta}) = \begin{bmatrix} \boldsymbol{R}^n_b(\boldsymbol{\Theta}_n b) & \boldsymbol{0}_{3x3} \\ \boldsymbol{0}_{3x3} & \boldsymbol{T}_\Theta(\boldsymbol{\Theta}_{nb}) \end{bmatrix} \tag{4.3}$$

Linear velocity transformation matrix $\boldsymbol{R}_b^n(\boldsymbol{\Theta}_{nb})$ is given by

$$\boldsymbol{R}_b^n(\boldsymbol{\Theta}_n b) = \boldsymbol{R}_{x,\phi}\boldsymbol{R}_{y,\theta}\boldsymbol{R}_{z,\psi} = \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi c\phi s\theta \\ s\psi c\theta & c\psi c\phi + s\phi s\theta s\psi & -c\psi s\phi + s\theta s\psi c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{4.4}$$

And the angular velocity transformation $\boldsymbol{T}_\theta(\boldsymbol{\Theta}_{nb})$ is given by

$$\boldsymbol{\Theta}_{nb}(\boldsymbol{\Theta}_n b) = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi/c\theta & c\phi/c\theta \end{bmatrix} \tag{4.5}$$

4.1.2 Kinetics

The 6 DOF marine craft equation of motion can be written as

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}_\theta(\boldsymbol{\eta})\boldsymbol{\nu} \tag{4.6a}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{g}(\boldsymbol{\eta}) = \boldsymbol{\tau} + \boldsymbol{\tau_{wind}} + \boldsymbol{\tau_{wave}} \tag{4.6b}$$

Where the mass matrix $\boldsymbol{M}$ is given by:

$$\boldsymbol{M} = \boldsymbol{M}_{RB} + \boldsymbol{M}_A$$

- $\boldsymbol{M}_{RB}$ - Rigid-Body System Inertia Matrix
- $\boldsymbol{M}_A$ - Added mass matrix

The Coriolis centripetal matrix $\boldsymbol{C}(\boldsymbol{\nu})$:

$$\boldsymbol{C}(\boldsymbol{\nu}) = \boldsymbol{C}_{RB}(\boldsymbol{\nu}) + \boldsymbol{C}_A(\boldsymbol{\nu})$$

- $\boldsymbol{C}_{RB}(\boldsymbol{\nu})$ - Rigid-Body Coriolis and Centripetal Matrix
- $\boldsymbol{C}_A(\boldsymbol{\nu})$ - Added-Mass Coriolis and Centripetal Matrix

Damping matrix $\boldsymbol{D}(\boldsymbol{\nu})$

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

- $\boldsymbol{D}$ - Linear damping matrix due to potential damping and skin friction
- $\boldsymbol{D}_n(\boldsymbol{\nu}_r)$ - Non-linear damping matrix due to quadratic damping and higher order terms

Hydrostatic forces $\boldsymbol{g}(\boldsymbol{\eta})$:

- Weight of the body $W = mg$, expressed in the NED frame as: $\boldsymbol{f}_b^n = -[0\ 0\ B]^T$
- Submerged buoyancy force $B = \rho g \Delta$ expressedd in NED frame as : $\boldsymbol{f}_g^n = [0\ 0\ W]^T$

Expressed in the body frame we get:

$$\begin{aligned} \boldsymbol{f}_g^b &= \boldsymbol{R}_b^n/\boldsymbol{\Theta}_{nb})^{-1}\boldsymbol{f}_g^n \\ \boldsymbol{f}_b^b &= \boldsymbol{R}_b^n/\boldsymbol{\Theta}_{nb})^{-1}\boldsymbol{f}_g^n \end{aligned}$$

The resulting restoring forces can then be expressed as:

$$\boldsymbol{g}(\boldsymbol{\eta}) = -\begin{bmatrix} \boldsymbol{f}_g^b + \boldsymbol{f}_b^b \\ \boldsymbol{r}_g^b \times \boldsymbol{f}_g^b + \boldsymbol{r}_b^b \times \boldsymbol{f}_b^b \end{bmatrix}$$

External forces:

- $\boldsymbol{\tau}$ - Control forces and moments
- $\boldsymbol{\tau}_{wave}$ - Wave forces and moments
- $\boldsymbol{\tau}_{wind}$ - Wind forces and moments

4.2 Assumptions and Simplifications

In the process of deriving a simulation model of the craft, it is reasonable to present the assumptions done in order to simplify the model in equation 4.6b.

Low speed: The AUV will operate in a *low-speed regime*, which is defined by operational speed within the interval $[0,\ U_{DP})$, where $U_{DP} = 1.5m/s$. In this speed range, a linear speed independent hydrodynamic model is valid. (Fossen [2005]).
Therefore, the following nonlinear terms are neglected:

- $\boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu}$
- $D_n(\boldsymbol{\nu}_r)$

Environmental forces: The operational environment is in a basin, hence there will be no environmental forces. Therefore the following terms will be omitted:

- $\boldsymbol{\tau_{wind}}$
- $\boldsymbol{\tau_{wave}}$

Geometry:

- **Symmetry:** The AUV is approximately symmetric about three axis, respectively xz, yz and xy. Therefore the mass matrix $\boldsymbol{M}$ is assumed to be symmetric (Fossen [2011]):

$$M = diag\{m_{11},\ m_{22},\ m_{33},\ m_{44},\ m_{55},\ m_{66}\}$$

- **Calculation of added mass:** In order to get a rough estimate of the added mass matrix $\boldsymbol{M_A}$ without complex computations, some simplifications of the geometry has to be done. The estimate is based on the geometry of an oblate spheroid.

4.3 Added mass estimate

THe geometry of the AUV is in a sence advanced and distinct. Therefore it is hard to find a good representation of the geometry among standardized geometries with known added mass. The ideal solution is probably to generate a geometry model to be analysed in nummerical added mass programs such as Wamit (WAM), however, this exceeds the scope of this thesis, and has still not been prioritized in the Vortex organization. In the process of estimating the added mass matrix of the AUV, the added mass of some known geometries will be compared.

4.3.1 Oblate spheroid

The inital estimate will be based on approximation the geometry of the AUV to an oblate spheroid. Roughly speaking, the geometry of the AUV have similarities to the geometry of an oblate spheroid. However, this approximation means that carvings in the real geometry compared to an oblate spheroid are neglected. This is important to be aware of in the estimation process.

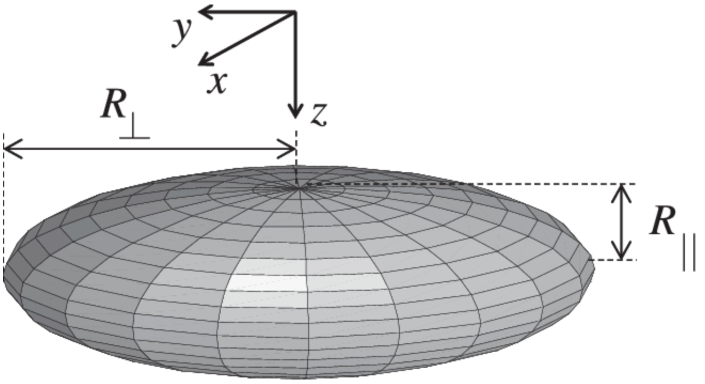


Figure 4.2: Illustration of an oblate spheroid. Courtesy:Grigorchuk and Karpov [2014]

Parameters For the vortex drone is given by

- $R_\perp = 36cm$
- $R_{||} = 8cm$

From now, $R_\perp$ is dentoted b, and $R_{||}$ is denoted a. In [Ed], values of zero frequency added mass coefficients are given in the following table, with the relationship $\lambda = \frac{b}{a}$ as a variable.

b/a	1.00	1.50	2.00	2.50	3.00	4.00	6.00	8.00	10.00	∞
$\mathrm{M}_{11}/\frac{4}{3}\rho\pi ab^2$	0.500	0.384	0.310	0.260	0.223	0.174	0.121	0.092	0.075	0.00
$\mathrm{M}_{33}/\frac{4}{3}\rho\pi ab^2$	0.500	0.803	1.118	1.428	1.742	2.379	3.642	4.915	6.183	∞
$\mathrm{M}_{44}/\frac{4}{15}\rho\pi ab^2(a^2+b^2)$	0.000	0.115	0.337	0.587	0.840	1.330	2.259	3.150	4.019	∞

Furthermore, $M_{11} = M_{22}$, $M_{44} = M_{55}$ and $M_{66} = 0$ In our case, $\lambda = 4.5$. By using the matlab function for the qubic spline interpolation method:$spline(x, y)$, we get the following results:

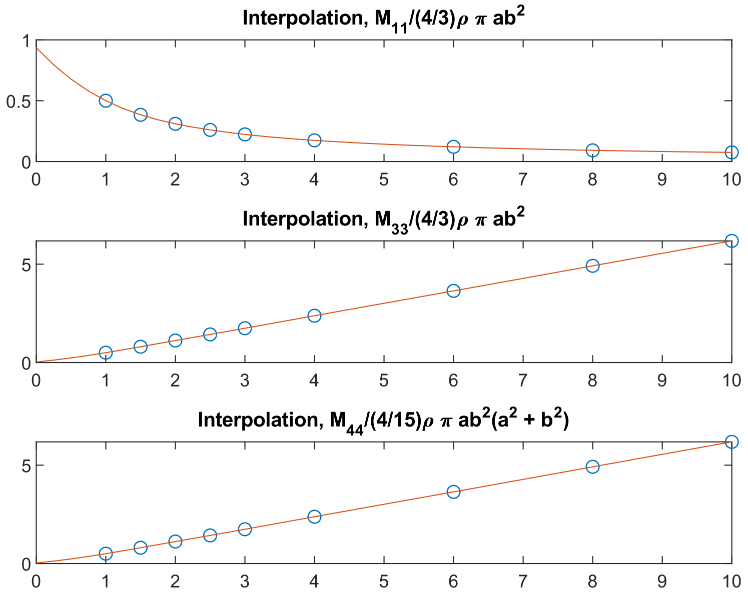


Figure 4.3: Matlab plot showing interpolation of table values of add

From this we get the following values

- $M_{11}/\frac{4}{3}\rho\pi ab^2 = 0.157$
- $M_{33}/\frac{4}{3}\rho\pi ab^2 = 2.696$
- $M_{44}/\frac{4}{15}\rho\pi ab^2(a^2+b^2) = 1.568$

By using that $\rho = 1000\frac{kg}{m^3}$, $a = 0.08m$ and $b = 0.36m$ as well as $M_{11} = M_{22}$, $M_{44} = M_{55}$ and $M_{66} = 0$ we get the following added mass matrix:

$$M_{A,\textit{oblate spheroid}} = \begin{bmatrix} 6.82 & 0 & 0 & 0 & 0 & 0 \\ 0 & 6.82 & 0 & 0 & 0 & 0 \\ 0 & 0 & 117.1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.852 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.852 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

These values matches the values calculated by using the method in Korotkin [2008] for added mass of an oblate spheroid. These values where $M_{A,\textit{Korotkin [2008]}} = diag\{7.46,\ 7.46,\ 118.2,\ 1.849,\ 1.849, 0\}$. In this method was a little less accurate as it required accuracy in the process of extracting coefficients from a graph.

By comparing the geometry of the AUV with the oblate spheroid, we can argue that the value of M_{33} represent an upper limit, while the value of M_{66} represent a lower limit. Furthermore, by comparing the geometry of the AUV with an oblate spheroid, lets assume that the values of $M_{11} = M_{22}$ and $M_{44} = M_{55}$ are decent estimates. Further we will investigate other geometries to estimate M_{33} and M_{66}

4.3.2 Rectangular octagon

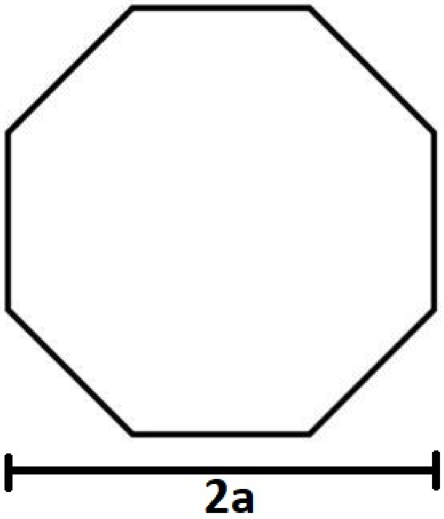


Figure 4.4: Rectangular octagon shape. The value 2a is the diameter. Adapted from Pettersen [2007]

In order to estimate the added mass in yaw: M_{66}, A rectangular octagon is used to approximate the geometry of the AUV. The formula of the 2 dimensional added mass (Added mass per length) in yaw for this geometry is given in Pettersen [2007] by the formula

$$M_{66,2D} = 0.055\pi\rho a^4$$

Extending this to three dimensions (without considering three dimensional flow effects) yields the formula

$$M_{66,3D} = 0.055\pi\rho a^4 b$$

Where b is the thickness of the auv. In our case, $a = 0.36m$ and $b = 0.16m$. Hence, we get

$$M_{66,3D} = 0.46kg * m^2$$

This is probably still an underestimate due to that the actual shape of the AUV have more carvings than an octagon, and therefore also accelerate more water in a yaw rotation due to a yaw movement. However, this difference is assumed small.

4.3.3 square

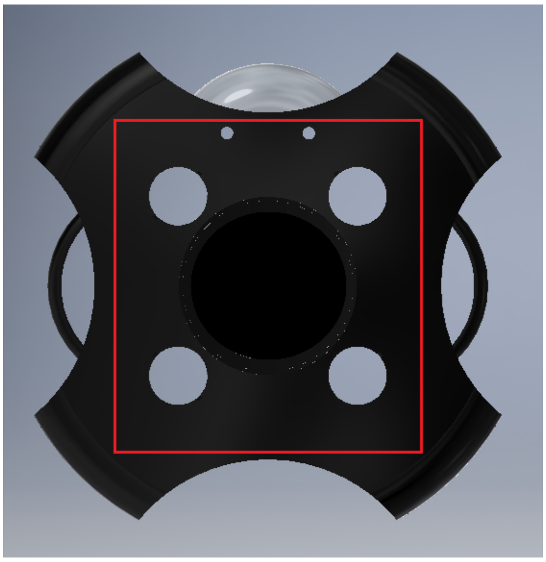


Figure 4.5: Rectangular shape. Underestimate of added mass in heave

In order to generate an underestimate of the added mass in heave M_{33} a square carved out of the actual geometry is used to represent the geometry. In addition, the added mass of four flat circular plates the representing the carved out area for thrusters will be subtracted from the added mass of the square to give the resulting estimate. According to Pettersen [2007], the formula for the added mass in heave of a square is given by:

$$M_{33,square} = \frac{0.579}{4}\rho\pi a^3$$

Where a is the cathetus of the square. Let $a = 0.42m$. Hence:

$$M_{33,square} = 33.7kg$$

The added mass in heave for a circular plate is given by (Pettersen [2007])

$$M_{33,circle} = \frac{\pi}{6}\rho a^3$$

Where a is the diameter, $a = 0.1m$. Hence

$$M_{33,circle} = 0.52kg$$

This yields the lower limit of the added mass in heave:

$$M_{33,\text{lower limit}} = 31.62kg$$

The fact that this estimate deviates a lot from the oblate-spheroid-estimate motivates to generate a third estimate of the added mass in heave. This time a the same geometry is used, but this time we let the diagonal of the AUV match the diagonal of the square:

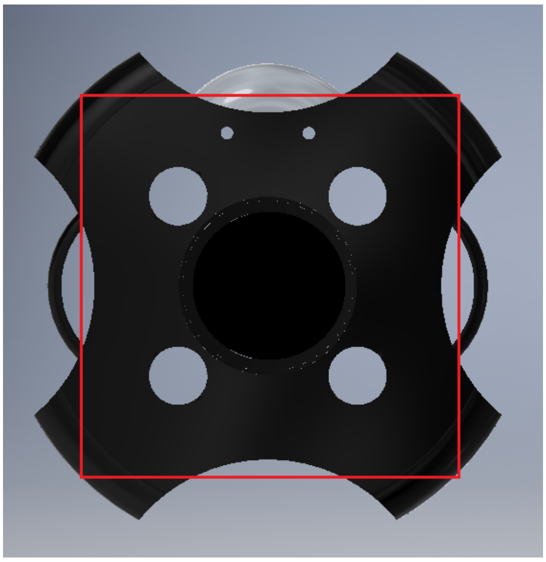


Figure 4.6: Rectangular shape. Added mass of a square plate in heave

Now, $a = 0.495m$, which yields

$$M_{33,square} = 55.15kg$$

If accounting for the carved out area for the tunnel thrusters, we get

$$M_{33} = 53.07kg$$

By comparing the square and the shape of the AUV in Figure 4.6, the last estimate of M_{33} is assumed to be a decent estimate. The resulting estimated matrix in added mass is

$$M_A = \begin{bmatrix} 6.82 & 0 & 0 & 0 & 0 & 0 \\ 0 & 6.82 & 0 & 0 & 0 & 0 \\ 0 & 0 & 53 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.852 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.852 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.46 \end{bmatrix}$$

4.4 Rigid-Body System Inertia Matrix

The Rigid-Body System Inertia Matrix $\boldsymbol{M}_{RB}$ is calculated in the CAD program Inventor. The resulting matrix is

$$\boldsymbol{M}_{RB} = \begin{bmatrix} 16.66 & 0 & 0 & 0 & 0 & 0 \\ 0 & 16.66 & 0 & 0 & 0 & 0 \\ 0 & 0 & 16.66 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.809 & 0.003 & 0.015 \\ 0 & 0 & 0 & 0.003 & 0.878 & 0.063 \\ 0 & 0 & 0 & 0.015 & 0.063 & 1.57 \end{bmatrix}$$

From this, we can see that all cross-terms are zero or almost zero. This is due to the symmetrical properties of the AUV.

This yields the total mass matrix

$$\boldsymbol{M} = \boldsymbol{M}_{RB} + \boldsymbol{M}_A = \begin{bmatrix} 23.48 & 0 & 0 & 0 & 0 & 0 \\ 0 & 23.48 & 0 & 0 & 0 & 0 \\ 0 & 0 & 69.66 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2.66 & 0.003 & 0.015 \\ 0 & 0 & 0 & 0.003 & 2.73 & 0.063 \\ 0 & 0 & 0 & 0.015 & 0.063 & 2.03 \end{bmatrix} \tag{4.7}$$

4.5 Linear Damping

In general, linear damping $\boldsymbol{D}$ consists of two terms, respectively potential damping $\boldsymbol{D}_P$ and possible skin friction $\boldsymbol{D}_V$, hence $\boldsymbol{D} = \boldsymbol{D}_P + \boldsymbol{D}_V$. Since there are no excitation forces present, there will be no potential damping ($\boldsymbol{D}_P = \boldsymbol{0}$) Fossen [2011]. Hence, the linear damping matrix $\boldsymbol{D} = \boldsymbol{D}_V$, whereas the viscous damping is approximated to $\boldsymbol{D}_V = diag\{B_{11v},\ B_{22v},\ B_{33v},\ B_{44v},\ B_{55v},\ B_{66v}\}$, where the estimates of $B_{iiv},\ i = 1, .., 6$ are given by Fossen [2011]:

$$\begin{aligned} B_{11,v} &= \frac{m + A_{11}(0)}{T_{surge}} \\ B_{22,v} &= \frac{m + A_{22}(0)}{T_{sway}} \\ B_{33,v} &= \frac{m + A_{33}(0)}{T_{heave}} \\ B_{44,v} &= 2\Delta\zeta_{roll}\omega_{roll}[I_{xx} + A_{44}(\omega_{roll})] \\ B_{55,v} &= 2\Delta\zeta_{pitch}\omega_{pitch}[I_{yy} + A_{55}(\omega_{pitch})] \\ B_{22,v} &= \frac{I_{zz} + A_{66}(0)}{T_{yaw}} \end{aligned}$$

There have not been made any experiences to estimate time constants and frequencies. During consultation with Thor I. Fossen, it was suggested that reasonable estimates for time constants in surge, sway, yaw and heave would be somewhere in the interval $1 \leq T_i \leq 5$, where i is surge, sway, heave and yaw. Further was a rough estimate for natural frequency in roll and pitch suggested to be $\omega_{roll} = \omega_{pitch} \approx 5Hz$, and a reasonable value for additional roll damping $\Delta\zeta_{roll} = \Delta\zeta_{pitch} = 0.2$. It is assumed that $T_{yaw} < T_{surge} (= T_{sway}) < T_{heave}$. A guess for these constants are respectively:

- $T_{yaw} = 1s$
- $T_{surge} = T_{sway} = 2s$
- $T_{heave} = 5s$

Since $A_{55}(\omega_{pitch})$ and $A_{44}(\omega_{roll})$ not are available, the values for $A_{44}(0)$ and $A_{55}(0)$ are used. This yields the linear damping matrix:

$$\boldsymbol{D} = \begin{bmatrix} 11.74 & 0 & 0 & 0 & 0 & 0 \\ 0 & 11.74 & 0 & 0 & 0 & 0 \\ 0 & 0 & 13.93 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5.32 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5.46 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2.03 \end{bmatrix} \tag{4.8}$$

4.6 Restoring forces

The value of $\boldsymbol{r}_g \simeq [0\ 0\ 0]^T$, while $\boldsymbol{r}_b \simeq [0\ 0\ \ -a]^T$ where a is a positive constant. This constant has not been detected properly, but from experiments, it seems like the AUV is highly metacentric stable. Therefore we will assume that θ and ϕ are small, hence

$$\begin{aligned} cos(b) &\simeq 1 \\ sin(b) &\simeq b \end{aligned}$$

where b represent θ and ϕ. The AUV is designed to be neutrally buoyant, hence the weight of the AUV equals the bouyancy force $B = W$. This leads to the resulting restoring forces:

$$\boldsymbol{g}(\boldsymbol{\eta}) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ (z_g - z_b)W\phi \\ (z_g - z_b)W\theta \\ 0 \end{bmatrix}$$

Where the weight $W = mg = 163.4[N]$ and $(z_g - z_b) = a$. In the resulting linear equation this term will be expressed as:

$$\boldsymbol{g}(\boldsymbol{\eta}) = \boldsymbol{G}\boldsymbol{\eta}$$

Where $\boldsymbol{G} = [0,\ 0,\ 0,\ (z_g - z_b)W,\ (z_g - z_b)W,\ 0]^T$

4.7 Resulting model

The resulting linear model is given by the equation

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}_\theta(\boldsymbol{\eta})\boldsymbol{\nu} \tag{4.9a}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{D}\boldsymbol{\nu} + \boldsymbol{G}\boldsymbol{\eta} = \boldsymbol{\tau} + \boldsymbol{b} \tag{4.9b}$$

Values for $\boldsymbol{M}$, $\boldsymbol{D}$ and $\boldsymbol{G}$ are given above, while $\boldsymbol{b}$ is a bias that could be estimated.

Chapter 5

Supervisory system

The Supervisory system is the system that control the behavior of the entire system. It is composed of three parallel-running systems, respectively:

- **Mission managment system** that represents the deliberate layer in the autonomous structure.
- **Risk management system** is responsible for safe operation.
- **Control mode system** controls the control modes according to current mission mission objective and risk management system

The design of the supervisory system is based on the principles of the Harel Statechart Harel [1987]. This is a state machine with extended capabilities that makes the state machine more suited to handle complex systems.

5.1 State machines and Harel Statechart

The state machine control the behavior of the system through states. Each state commands a certain behavior of the system, either set at the *entry* or *exit* of the state or *during* the state. The state machine can change from one state to another through *transitions*. In order for a transition to happen, the corresponding *condition* must be valid.

A complex system consisting of many behaviors will be chaotic and unstructured if arranged in ’flat’, unstratified fashion. The Harel Statechart introduces three important elements dealing with hierarchy, parallelism and communication. These elements makes the statecharts compact and expressive - making it possible to present complex composition of behaviors compact, well structured and modular Harel [1987].

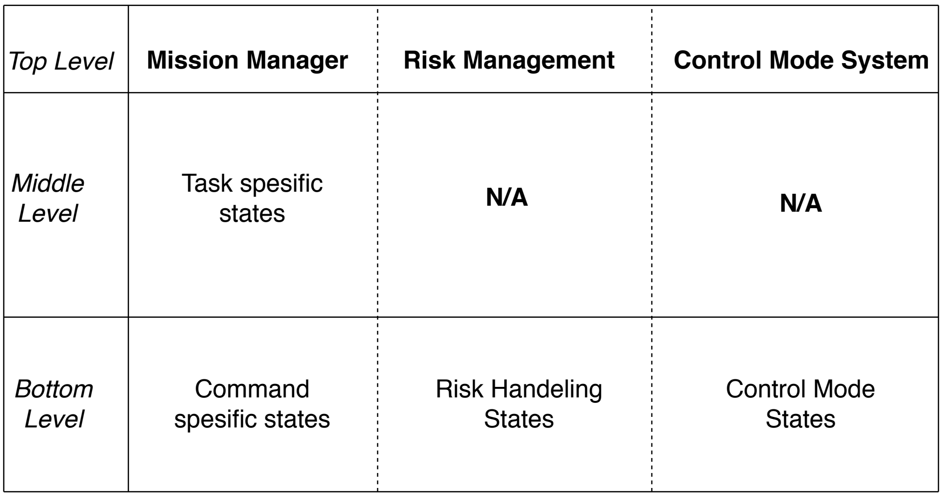
Hierarchy: By clustering related states into superstates abstraction and modularization of larger models are emphasised.

Parallelism: By introducing parallelism, sub-states contained in a so called AND state are allowed to operate simultaneously. Parallel states are the opposite of exclusive states, which are states where only one of the state are active. Exclusive states are found within each of the parallel states.

Communication: Communication concerns a broadcast mechanism for communication between concurrent components. This allows parallel states to interact with each other and still keep a well structured system.

5.2 Structure

At the *top level* of the hierarchy, there are three parallel superstates that are always active during operation. These are States are *Mission Management*, *Risk Management* and *Control Mode Management*. These states correspond to the three main components in the supervisory system.



Top Level	**Mission Manager**	**Risk Management**	**Control Mode System**
Middle Level	Task spesific states	**N/A**	**N/A**
Bottom Level	Command spesific states	Risk Handeling States	Control Mode States

Figure 5.1: Hierachy of the supervisory system. The dotted line between the tree superstates indicates that these states are parallel

5.2.1 Mission manager

The mission manager represents the deliberate layer in the autonomous structure. It determines the desired behavior of the system in the form of deciding the current mission objective and giving instructions to lower level components of the system according to this objective.

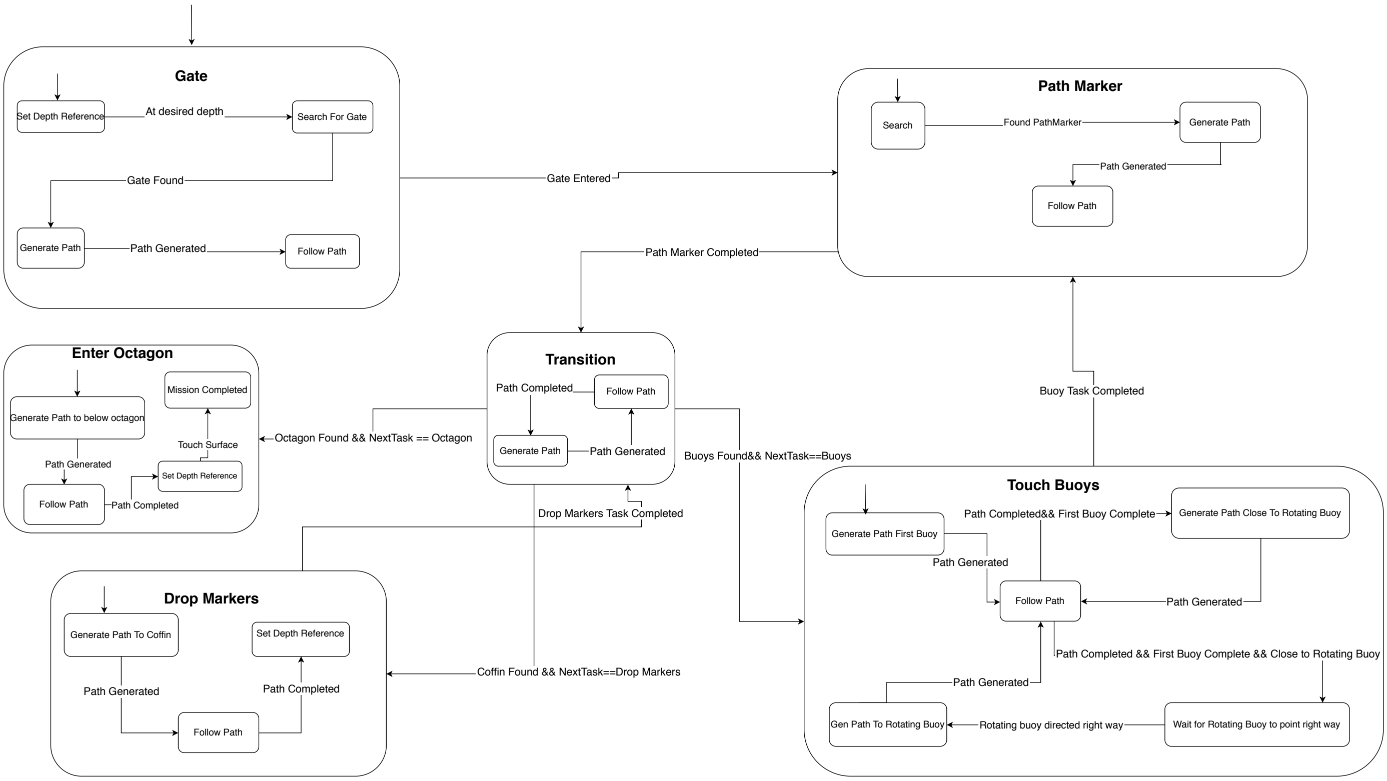

Gate
Set Depth Reference
At desired depth
Search For Gate
Gate Found
Generate Path
Path Generated
Follow Path
Gate Entered
Path Marker
Search
Found PathMarker
Generate Path
Path Generated
Follow Path
Path Marker Completed
Enter Octagon
Generate Path to below octagon
Mission Completed
Touch Surface
Path Generated
Set Depth Reference
Follow Path
Path Completed
Transition
Path Completed
Follow Path
Generate Path
Path Generated
Octagon Found && NextTask == Octagon
Buoys Found&& NextTask==Buoys
Buoy Task Completed
Drop Markers Task Completed
Drop Markers
Generate Path To Coffin
Set Depth Reference
Path Generated
Path Completed
Follow Path
Coffin Found && NextTask==Drop Markers
Touch Buoys
Generate Path First Buoy
Path Completed&& First Buoy Complete
Generate Path Close To Rotating Buoy
Path Generated
Follow Path
Path Generated
Path Completed && First Buoy Complete && Close to Rotating Buoy
Path Generated
Gen Path To Rotating Buoy
Rotating buoy directed right way
Wait for Rotating Buoy to point right way

Figure 5.2: Graphical representation of the mission manager. Boxes represents states. Arrows indicate the direction of the transitions the statechart. Arrows without origin indicates default states. Conditions for transitions are found at the arrows between the states

The mission manager is divided into six substates. Each of these substates aimes to perform on spesific task. These *task-spesific states* are also structured as superstates. Their children states are the lowest level of states in the mission manager hierarchy. In these states, references and commands are generated for lower level modules of the autonomous system architecture, therefore refered to as *command spesific states*.

Clustering of tasks

The task specific states are clustered according to the tasks and environment presented in Section 2.6. Each task have a corresponding state at the middle layer of the statechart hierarchy, respectively "Enter Gate", Touch Buoys", "Drop Markers" and "Enter Octagon". Two more task-specific states are added to this layer to make sure the AUV is able to navigate tasks mentioned above. These are the "Path Marker" state and "Transition" state. Based on a desired direction (either found from acousics or by following the tangent of the last segment of the path marker) paths are generated in that direction until the desired target is found.
Commands and references are to be generated at the bottom layer of the statechart hierarchy. These states are chosen according to sub-tasks that needs to be done in order to carry out each task successfully. Roughly speaking, there are four types of command specific states, with local variations depending on the superstate. These types are:

- **Set Depth Reference:** Making sure AUV operates at a certain depth by sending a depth reference to the depth reference model. A Dynamic positioning command is broadcasted the *Control Mode*

Management to make sure that the craft do not maneuver while settling at desired depth.

- **Search For target:** In the case of not knowing the position of the target, references in heading will be generated in order to make the front camera perceive more of the environment
- **Generate Path:** This is the state where the **path planner** is active. Based on perceived targets, the path planner will generate way-points to approach the target.
- **Follow Path:** When a plan is generated, a maneuvering command is sent to the *Control Mode Management*.

Scheduling tasks

The task schedule can be found by following the transition arrows in Figure 5.2. The order of the tasks corresponding to *task-specific states* is scheduled according to how hints are placed along the path. Further, sub-tasks are scheduled in a reasonable sequence to fulfill the target objectives.
The transitions between task-specific states are dependent on criterion's related to the completion of tasks. In the special case of the transition state, the criterion's are different. In this case, for a state to be entered, it must be classified as the next state, and then it must be detected. For the front camera, the detection zone is assumed to be five meters ahead and 20 degrees to each side relative to the surge axis, which points in the same direction as the front camera. With the bottom fixed camera, the detection zone is five meters radius down.

5.2.2 Risk Manager

The risk management system is designed to deal with dynamic obstacles that appear and then disappears after a while. The systems respond to obstacles that moves closer than two meters from the AUV, in a zone of 20 degrees to each side of the front camera. The response is to send a command to stop the vessel to *Control mode management*.

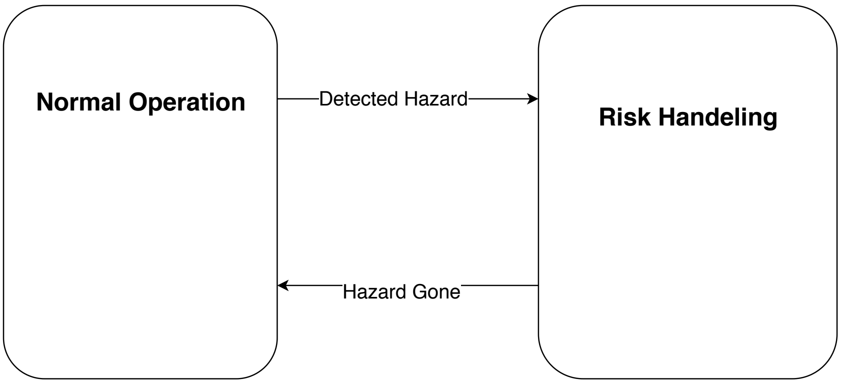


Figure 5.3: Risk Management system. Boxes represents states. Arrows indicate the direction of the transitions the statechart. Arrows without origin indicates default states. Conditions for transitions are found at the arrows between the states

The Risk management system is less complex than the Mission management system, and there is no need for more than one sub-layer with exclusive states. These are the states are namely **Normal Operation** and **Risk Handling**. In normal operation mode, the risk management system has no affect on the operation. When the risk handling state is active, the current operation according to the Mission manager is interrupted. all processes are stopped, and stop command is sent to the Control Mode Management. When the hazard is gone, the process that ran before the interruption is continued.

Control Mode Management

The control mode management system send signals to the guidance system according to the current control mode.

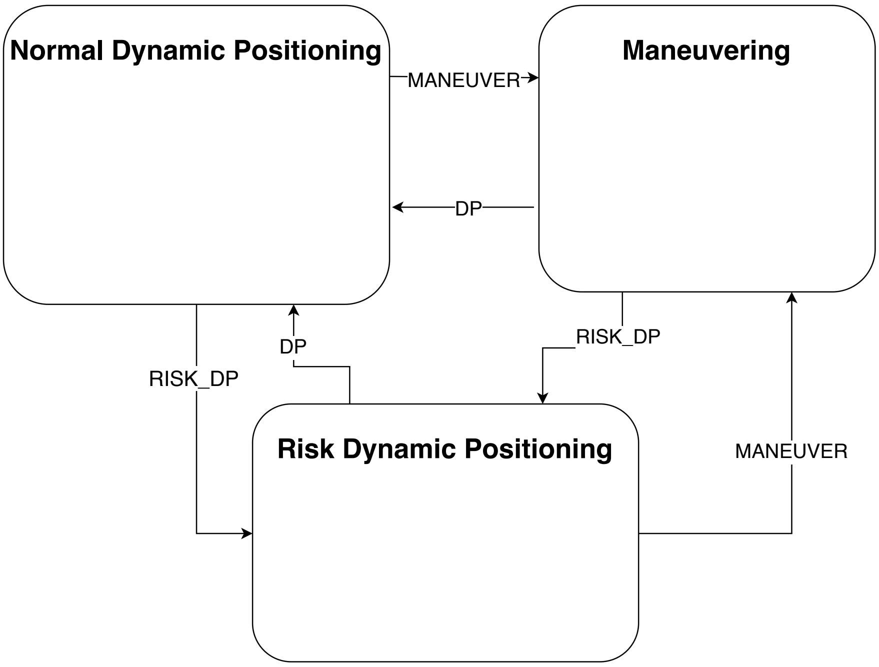


Figure 5.4: Control Mode Management system. Boxes represents states. Arrows indicate the direction of the transitions the statechart. Arrows without origin indicates default states. Transitions are triggered by event. These events are found at the arrows between the states

Normal Dynamic Positioning stop the path variable θ instantaneously. This mode is suited for situations where the craft already is at zero speed, typically after a path following is completed or at initial state. This mode is waiting for or searching for next maneuvering task. **Risk Dynamic Position** is triggered when dynamic obstacles close to the vessel are observed. The vessel can be at maneuvering state when this happens. For the vessel to follow the reference perfectly in this situation, the magnitude of the speed assignment must decrease gradually. This technique is used in this state. **Maneuvering** indicates that the craft is suppose to follow a path.
The modes are controlled by commands in the Mission management system and Risk assignment system through broadcasted events. These events are indicated at the transition arrows. These events are "MANEUVER", "DP" and "RISK_DP".

Chapter 6

Path Generation, Guidance and Control Design

The design of the path generation, guidance law and the control law is in this case a highly coupled task. For the controller to avoid bumpless transfers in the way-points there are certain requirements in degree of continuity in the generated path.

By limiting the problem to follow the path $\boldsymbol{p}_d(s)$ and not a heading reference along the track, the path needs to be $\mathcal{C}^2$ continuous as the backstepping controller demands references up to second derivative $\boldsymbol{p}_d^{s^2}$. By additionally require that the heading should follow the desired path, the path needs to be $\mathcal{C}^3$ continuous since the control law in this case require references $\boldsymbol{\eta}_d^{s^2} = [\boldsymbol{p}_d^{s^2} \ \psi_d^{s^2}]^T$, where $\psi_d^{s^2}$ is dependent on $\boldsymbol{p}_d^{s^3}$. Intuitively, the control law design also affects the guidance law design since the guidance law needs to provide reference signals to the control law.

Furthermore, tuning parameters such as λ in the path generation which affects the curvature of the path, and speed assignment and update law in guidance block affects the performance of the controller. Each of the modules path generation, guidance and controller will be investigated in this chapter. The design of the dynamic assignment will also be reviewed in this chapter. The dynamic assignment can be seen as a part of the guidance system, but will be investigated individually.

6.1 Path Generation

The purpose of the path generation is to generate smooth, parametrized path segments $\boldsymbol{p}_d(s) = col(x_d(s), y_d(s))$. For the backstepping controller to avoid bumpless transitions in the way-point transfers, the the path needs to be $\mathcal{C}^3$ continuous. It is not desirable to have higher degree of continuity due to that uncontrollable shapes that may occur with increasingly degree of continuity, thus the path segments are chosen to be $\mathcal{C}^3$ continuous.

6.1.1 $\mathcal{C}^r$ path generated from way-points

In order to generate a $\mathcal{C}^r$ path, a sufficiently differential curve is needed. This can be created by using interpolation techniques and splines to generate curves that goes through the way-points. The overall desired path $p_d(\theta)$ is divided into n sub paths $p_{d,i}(\theta)$ between the way-points. These paths are expressed as a polynomial in θ of a certain order, and the sub-paths are concatenated at the way-points to assemble a full path. To ensure that the path is sufficiently differentiable at the way-points, the order of the path must be sufficiently high. For a curve in $\mathbb{R}^2$, with $\mathcal{I} = \{1, ..., n\}$ path segments. The full path is $p_d(\theta) = col(x_d(\theta), y_d(\theta)), \theta \in [0, n)$, while the the sub-paths are denoted $p_{d,i}(\theta) = col(x_{d,i}, y_{d,i}), i \in \mathcal{I}$. $p_i = col(x_i, y_i), i \in \mathcal{I} \cup \{n+1\}$ are the way-points. The differential requirement $p_d(\theta) \in \mathcal{C}^r$ can then by formulated as follows:

$$\begin{array}{ll}
\lim_{\theta\nearrow i-1} x_{d,i-1} = \lim_{\theta\searrow i-1} x_{d,i-1} & \lim_{\theta\nearrow i-1} y_{d,i-1} = \lim_{\theta\searrow i-1} y_{d,i-1} \\
\lim_{\theta\nearrow i-1} x^{\theta}_{d,i-1} = \lim_{\theta\searrow i-1} x^{\theta}_{d,i-1} & \lim_{\theta\nearrow i-1} y^{\theta}_{d,i-1} = \lim_{\theta\searrow i-1} y^{\theta}_{d,i-1} \\
\vdots & \\
\lim_{\theta\nearrow i-1} x^{\theta^r}_{d,i-1} = \lim_{\theta\searrow i-1} x^{\theta^r}_{d,i-1} & \lim_{\theta\nearrow i-1} y^{\theta^r}_{d,i-1} = \lim_{\theta\searrow i-1} y^{\theta^r}_{d,i-1}
\end{array}$$

Where the sub-paths are defined as

$$\begin{aligned}
x_{d,i}(\theta) &= a_{k,i}\theta^k + a_{1,i}\theta + a_{0,i} \\
y_{d,i}(\theta) &= b_{k,i}\theta^k + b_{1,i}\theta + b_{0,i}
\end{aligned} \tag{6.1}$$

Where the coefficients $\{a_{j,i}, b_{j,i}\}$ are to be determined. There are, for each sub-path $(2k+1)$ unknown coefficients to be determined, which means $(2k+1) * n$ unknown coefficients for the entire path. The scope of this thesis concern online path generation, hence it is interesting to consider the generation of one sub-path at a time.

6.1.2 Step-wize generation of $\mathcal{C}^3$ path

This section is based on Skjetne [2019b]. Step-wize path generation is relevant in the case of online path generation where the next way-point is determined real-time. The path is required to be $\mathcal{C}^3$ in order to ensure smooth, bumpless transitions when switching sub-path. Given a path variable for the full path $s \in [0, n)$, where n is the number of sub-paths. The local path variable for the sub-path is given by $\theta = s - \lfloor s \rfloor \in [0, 1)$. $i = \lfloor s \rfloor + 1 \in \mathcal{I}$, where i identifies the active sub-path.

The path-generation technique generates a polynomial between current way-point $p_{0,i}$ and target way-point $p_{t,i}$. At the first way-point $p_{d,0}(0)$, the unit tangent is $p^{\theta}_{d,0}(0) = \frac{p_{t,0}-p_{0,0}}{|p_{t,0}-p_{0,0}|}$. i indicates the line segment number, and θ is the local path parameter. The notation for the unit tangent vector in the point $p_d(i, \theta) = p_{d,i}(\theta)$ is $T_{\theta,i}$

To make sure that the requirement for $\mathcal{C}^3$ continuity at the way-points is taken into account, derivatives up to third derivative of the path segments connected by the way-points must be equal:

- $p_{d,i-1}(1) = p_{d,i}(0)$
- $p^{\theta}_{d,i-1}(1) = p^{\theta}_{d,i}(0)$
- $p^{\theta^2}_{d,i-1}(1) = p^{\theta^2}_{d,i}(0)$
- $p^{\theta^3}_{d,i-1}(1) = p^{\theta^3}_{d,i}(0)$

$\mathcal{C}^1$ continuity implies that the tangent at the previous target way-point and the next starting way-point is the same. This is obvious as it is the same way-point. This means

$$T_{1,i-1} = T_{0,i}$$

The tangent at the target way-point for path segment i is set to the slope between $p_{0,i}$ and $p_{t,i}$ times λ:

$$p_{d,i}(1) = \frac{p_{t,i} - p_{0,i}}{|p_{t,i} - p_{0,i}|}\lambda = \lambda T_{1,i}$$

$\lambda > 0$ is a tuning parameter, where the radius of curvature is proportional to increasing λ.

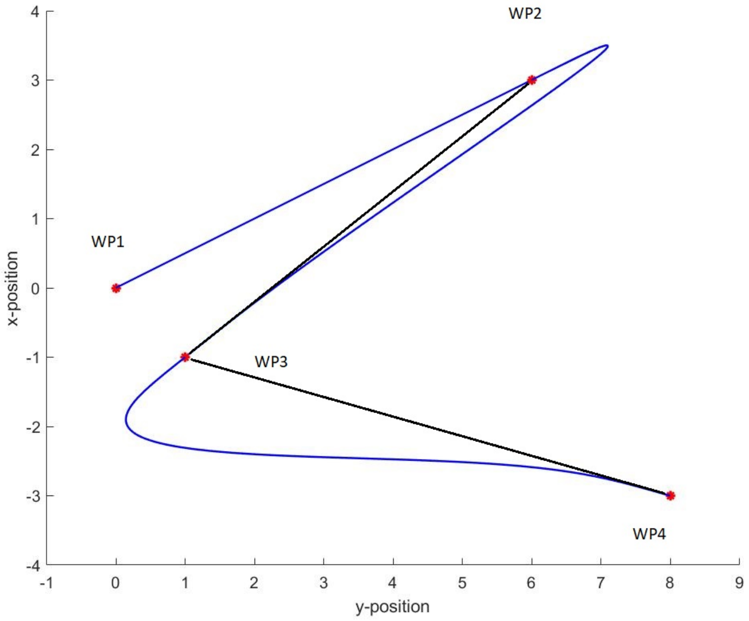


Figure 6.1: Online path generation from Matlab. The blue line is the path, the black lines are tangent vector between two vectors. The red dots are way-points, denoted as WP{i}, where i indicates in what sequence the way-points are entered. The curvature is somewhat exaggerated to make the boundary conditions in $p^{\theta}_{d,i}(\theta)$ more visible.

Hence, the boundary conditions are:

- $p^{\theta}_{d,0}(0) = \lambda \frac{p_{t,0} - p_{0,0}}{|p_{t,0} - p_{0,0}|} = \lambda T_{0,0}$
- $p^{\theta}_{d,i}(0) = \frac{p_{i-1,t} - p_{i-1,0}}{|p_{i-1,t} - p_{i-1,0}|} \lambda = \lambda T_{0,i}$
- $p_{d,i}(1) = \lambda \frac{p_{t,i} - p_{0,i}}{|p_{t,i} - p_{0,i}|} = \lambda T_{1,i}$
- $p_{t,i-1} = p_{0,i}$

Consider the polynomial of order k:

$$
\begin{aligned}
x_{d,i}(\theta) &= a_{k,i}\theta^k + \ldots + a_{3,i}\theta^3 + a_{2,i}\theta^2 + a_{1,i}\theta + a_{0,i} \\
y_{d,i}(\theta) &= b_{k,i}\theta^k + \ldots + b_{3,i}\theta^3 + b_{2,i}\theta^2 + b_{1,i}\theta + b_{0,i} \\
\\
x^{\theta}_{d,i}(\theta) &= k a_{k,i}\theta^{k-1} + \ldots + 3a_{3,i}\theta^2 + 2a_{2,i}\theta + a_{1,i} \\
y^{\theta}_{d,i}(\theta) &= k b_{k,i}\theta^{k-1} + \ldots + 3b_{3,i}\theta^2 + 2b_{2,i}\theta + b_{1,i} \\
\\
x^{\theta^2}_{d,i}(\theta) &= k(k-1)a_{k,i}\theta^{k-2} + \ldots + 6a_{3,i}\theta + 2a_{2,i} \\
y^{\theta^2}_{d,i}(\theta) &= k(k-1)b_{k,i}\theta^{k-2} + \ldots + 6b_{3,i}\theta^2 + 2b_{2,i} \\
\\
x^{\theta^3}_{d,i}(\theta) &= k(k-1)(k-2)a_{k,i}\theta^{k-3} + \ldots + 6a_{3,i} \\
y^{\theta^3}_{d,i}(\theta) &= k(k-1)(k-2)b_{k,i}\theta^{k-3} + \ldots + 6b_{3,i} \\
&\vdots
\end{aligned}
\tag{6.2}
$$

In order to solve the polynomial equations for $(a_{j,i}, b_{j,i})$ and get a $\mathcal{C}^3$ path, boundary conditions are used. $\mathcal{C}^0$: Continuity at the way-points for path segment i gives:

$$p_{d,i}(0) = p_{0,i} \qquad\qquad p_{d,i}(1) = p_{t,i}$$

$$\begin{array}{ll} x_{d,i}(0) = a_{0,i} = x_{0,i} & x_{d,i}(1) = a_{k,i} + \ldots + a_{1,i} + a_{0,i} = x_{t,i} \\ y_{d,i}(0) = b_{0,i} = y_{0,i} & y_{d,i}(1) = b_{k,i} + \ldots + b_{1,i} + b_{0,i} = y_{t,i} \end{array}$$

$\mathcal{C}^1$: Continuity in the slopes at the way-points for path segment i gives:

$$p^{\theta}_{d,i}(0) = \lambda T_{0,i} \qquad\qquad p^{\theta}_{d,i}(1) = \lambda \frac{p_{t,i} - p_{0,i}}{|p_{t,i} - p_{0,i}|}$$

$$\begin{array}{ll} x^{\theta}_{d,i}(0) = a_{1,i} = \lambda T_{0x,i} & x^{\theta}_{d,i}(1) = k a_{k,i} + \ldots + 2a_{2,i} + a_{1,i} = \lambda \frac{x_{t,i} - x_{0,i}}{|p_{t,i} - p_{0,i}|} \\ y^{\theta}_{d,i}(0) = b_{1,i} = \lambda T_{0y,i} & y^{\theta}_{d,i}(1) = k b_{k,i} + \ldots + 2b_{2,i} + b_{1,i} = \lambda \frac{y_{t,i} - y_{0,i}}{|p_{t,i} - p_{0,i}|} \end{array}$$

$\mathcal{C}^j$: Continuity in the j derivative for $j \geq 2$, setting the derivative equal to zero at the way-points:

$$p^{\theta^j}_{d,i}(0) = 0 \qquad\qquad p^{\theta^j}_{d,i}(1) = 0$$

$$\begin{array}{ll} x^{\theta^2}_{d,i}(0) = 2a_{2,i} = 0 & x^{\theta^2}_{d,i}(1) = k(k-1)a_{k,i} + \ldots + 6a_{3,i} + 2a_{2,i} = 0 \\ y^{\theta^2}_{d,i}(0) = 2b_{2,i} = 0 & y^{\theta^2}_{d,i}(1) = k(k-1)b_{k,i} + \ldots + 6b_{3,i} + 2b_{2,i} = 0 \\ x^{\theta^3}_{d,i}(0) = 6a_{3,i} = 0 & x^{\theta^3}_{d,i}(1) = k(k-1)(k-2)a_{k,i} + \ldots + 6a_{3,i} = 0 \\ y^{\theta^3}_{d,i}(0) = 6b_{3,i} = 0 & y^{\theta^3}_{d,i}(1) = k(k-1)(k-2)b_{k,i} + \ldots + 6b_{3,i} = 0 \\ \vdots & \end{array} \tag{6.3}$$

In the case of a $\mathcal{C}^3$ continuous path, the second and third derivatives of the path $p_{d,i}(\theta)$ will be set to zero in the transition points between sub-paths. There will be no restrictions on continuity of derivatives of order higher than three.

6.2 Guidance

Given a path $\boldsymbol{p}_d(s)$ and a speed profile $u_d(t)$, the guidance law generates references signals for the control law, and the dynamics of the path variable s. Hence the **guidance law function** is to:

- Calulate the dynamics of the path variable s:

  $$\dot{s} = \upsilon_s(t,s) + \omega(\boldsymbol{\eta}, s), \quad s(t_0) = 0 \tag{6.4}$$

  ω is a feedback term chosen as a unit-tangent update law to ensure convergence of the along-track error:

  $$\omega = \mu \frac{(\boldsymbol{\eta}_d^s)^T}{|\boldsymbol{\eta}_d^s|}(\boldsymbol{\eta} - \boldsymbol{\eta}_d) \tag{6.5}$$

  Where μ is a positive constant

- Generate the output signals for the control law

$$\upsilon_s = \upsilon_{s,i}(t,\theta), \quad \upsilon_s^t = \upsilon_{s,i}^t(t,\theta), \quad \upsilon_s^s = \upsilon_{s,i}^\theta(t,\theta) \tag{6.6}$$

$$\psi_d = \psi_d(i,\theta), \quad \psi_d^s = \psi_d^\theta(i,\theta), \quad \psi_d^{s^2} = \psi_d^{\theta^2}(i,\theta) \tag{6.7}$$

$$\boldsymbol{p}_d = p_d(i,\theta), \quad \boldsymbol{p}_d^s = p_d^\theta(i,\theta), \quad \boldsymbol{p}_d^{s^2} = p_d^{\theta^2}(i,\theta) \tag{6.8}$$

$$\boldsymbol{\eta}_d = [\boldsymbol{p}_d\ \psi_d]^T, \quad \boldsymbol{\eta}_d^s = [\boldsymbol{p}_d^s\ \psi_d^s]^T, \quad \boldsymbol{\eta}_d^{s^2} = [\boldsymbol{p}_d^{s^2}\ \psi_d^{s^2}]^T \tag{6.9}$$

6.3 Dynamic assignment

This section is based on Skjetne [2019a] The dynamic assignment should be designed according to the dynamic limitations of the vessel.
Consider the functions $h_\nu(\beta)$ and $h_a(\beta)$ so that

$$\bar{u}_\beta = h_\nu(\beta) \tag{6.10a}$$

$$\bar{a}_\beta = h_a(\beta) \tag{6.10b}$$

Where $\bar{u}_\beta$ is the speed limit and $\bar{a}_\beta$ is the acceleration limit as a function of the Crab angle β. Let the speed assignment be defined by

$$\upsilon_s(t,s) = \sigma(t,s)\frac{u_d(s)}{|p_d^s(s)|}$$

Where $\sigma(t,s) : \mathbb{R} \to \{0,1\}$ Is an activation to start or stop the motion of the path variable $s(t)$. For a vessel that starts with zero speed at way point one and stops at way point two, $u_d(s)$ is defined by

$$u_d(s) = \begin{cases} 0 & s < -\lambda \\ \bar{u}_\beta(tanh(k_\beta(s+\lambda))) & s \in [-\lambda, \frac{1-\lambda}{2}) \\ \bar{u}_\beta(tanh(k_\beta(1+\lambda-s))) & s \in [\frac{1-\lambda}{2}, \infty) \end{cases}$$

Where k_β is a β dependent gain that will affect the acceleration of the vessel. The purpose of $\lambda, 0 < \lambda << 1$ is to shift the parametrization of the path parameter to avoid that the path parameter gets stuck in $s = 0$. We have that

$$\dot{u}_d = u_d^t + u_d^s\dot{s}$$

Where $u_d^t = 0$. Let

$$\dot{s} = \upsilon_s = \sigma \frac{u_d}{|p_d^s|}$$

With $\sigma = 1$ for a moving vessel. Knowing that $\frac{d}{dx} tanh(x) = 1 - tanh^2(x)$, we get

$$u_d^s = \begin{cases} 0 & s < -\lambda \\ +\bar{u}_\beta k_\beta (1 - tanh^2(k_\beta (s+\lambda))) & s \in [-\lambda, \frac{1-\lambda}{2}) \\ -\bar{u}_\beta k_\beta (1 - tanh^2(k_\beta (1+\lambda - s))) & s \in [\frac{1-\lambda}{2}, \infty) \end{cases}$$

Hence, the acceleration is given by

$$\dot{u}_d = u_d^s \frac{u_d}{|p_d^s|}$$

And the acceleration limit is given by

$$\bar{a}_\beta = k_\beta \frac{\bar{u}_\beta^2}{|p_d^s|}$$

Which yields the expression for the gain k_β

$$k_\beta = \frac{|p_d^s|}{\bar{u}_\beta^2} \bar{a}_\beta$$

Since there is no current present, $\beta_d = 0$. Hence $\chi_d = \psi_d$. We have that the desired heading ψ_d is the tangent along the desired pat p_d. If the vessel moves along p_d, then the crab angle $\beta(t)$ is given by

$$\beta(t) = \chi_d(s) - \psi(t) = \psi_d(s) - \psi(t)$$

Therefore, the speed and acceleration limitations, respectively $\bar{u}_\beta = \bar{u}_\beta(t) = h_v(\beta(t))$ and $\bar{a}_\beta = \bar{a}_\beta(t) = h_a(\beta(t))$ are time dependent. However, if we assume that the speed and acceleration limits are uniform about the yaw axis, these limits can be considered constants. This assumption is reasonable considering the symmetrical geometry and thrust configuration of the AUV.

6.4 Backstepping control desing

The control design is based on Skjetne [2019b]. The backstepping control design implemented in simulations is a model based control design for a fully actuated vessel in 3 DOF. Since the model is linearized about zero speed, the control design is applicable for low speed operations.

6.4.1 Control objective

The control objective is developed according to the maneuvering problem formulation presented in 3.5.3:

Geometric task: Force the output of the system to converge to and follow a desired pose $\boldsymbol{\eta}_d(\theta) = [\boldsymbol{p}_d(\theta) \; \psi_d(\theta)]^T$

$$\lim_{t \to \infty} |\boldsymbol{\eta}(t) - \boldsymbol{\eta}_d(\theta)| = 0 \tag{6.11}$$

Dynamic task: The objective is to force the speed of the parametrized path $\dot{\theta}(t)$ to converge to and follow a speed assignment $\upsilon_s(\theta) = \frac{u_d(\theta)}{p_d^\theta(\theta)}$ along the parametrized path $\boldsymbol{p_d(\theta)}$.

$$\lim_{t\to\infty} |\dot{\theta}(t) - \upsilon_s(\theta)| = 0 \tag{6.12}$$

6.4.2 Control design

The control design is based on Skjetne [2019b]. The control design is a recursive design using control Lyapunov functions (CLF) in order to guarantee Global asymptotic stability (GAS) for the system according to the control objectives. Consider the 3 DOF control design model presented in Chapter 3:

$$\begin{aligned}\dot{\boldsymbol{\eta}} &= \boldsymbol{R}_z(\psi)\boldsymbol{\nu}\\ \boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{D}\boldsymbol{\nu} &= \boldsymbol{\tau} + \boldsymbol{b}\end{aligned}$$

Integrator augmentation technique A robust way of dealing with the induced bias $\boldsymbol{b}$ due to unmodeled dynamics is to include its estimate $\hat{\boldsymbol{b}}$ in a feedback term in the control law $\boldsymbol{\tau}$. In the absence of a bias estimate $\hat{\boldsymbol{b}}$ it is need for an alternative way of dealing with the unmodeled dynamics. By introducing an integral term in the control law, the bias can be compensated.
From the geometric task, we have the error dynamics in body frame $\boldsymbol{e} = \boldsymbol{R}^T(\psi)[\eta(t) - \eta_d(\theta)]$. We have that

$$\begin{aligned}\dot{\boldsymbol{e}} &= \frac{d}{dt}\left(\boldsymbol{R}^T(\psi)[\boldsymbol{\eta}(t) - \boldsymbol{\eta}_d(\theta)]\right)\\ &= \dot{\boldsymbol{R}}^T(\psi)[\boldsymbol{\eta}(t) - \boldsymbol{\eta_d}(\theta)] + \boldsymbol{R}^T(\psi)[\dot{\boldsymbol{\eta}}(t) - \boldsymbol{\eta}_d^s(\theta)\dot{s}]\\ &= -r\boldsymbol{S}\boldsymbol{R}^T(\psi)[\boldsymbol{\eta}(t) - \boldsymbol{\eta}_d(\theta)] + \boldsymbol{\nu} - \boldsymbol{R}^T(\psi)\boldsymbol{\eta}_d^s\dot{s}\\ &= -r\boldsymbol{S}\boldsymbol{e} + \boldsymbol{\nu} - \boldsymbol{R}^T(\psi)\boldsymbol{\eta}_d^s\dot{\boldsymbol{s}}\end{aligned}$$

Where $\boldsymbol{S} = \begin{bmatrix}0 & -1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}$ and r is the yaw rate. s is the global path parameter

$$s = \lfloor i \rfloor + \theta$$

Where i is the indexation of the i^{th} path segment and $\theta \in [0, 1)$ is the local path parameter. This yields the error dynamics

$$\dot{\boldsymbol{e}} = -r\boldsymbol{S}\boldsymbol{e} + \boldsymbol{\nu} - \boldsymbol{R}^T(\psi)\boldsymbol{\eta}_d^s\dot{\boldsymbol{s}} \tag{6.13a}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{D}\boldsymbol{\nu} = \boldsymbol{\tau} + \boldsymbol{b} \tag{6.13b}$$

Introducing the new state vector $\boldsymbol{z} = [\boldsymbol{z_1}\ \boldsymbol{z_2}\ \boldsymbol{z_3}]^{\boldsymbol{T}}$. $\boldsymbol{z}$ is a transformation of the state vector $\boldsymbol{x}$. A virtual controller $\boldsymbol{\alpha}_i$ is introduced at each step to stabilize the dynamics of the variable $\boldsymbol{z}_i$. This is not the case for the last step as the resulting control law does the stabilizing at this stage. The CLF at each step i is given by V_i . The path speed is defined as

$$\dot{s} = \omega + \upsilon_s$$

Where υ_s is the speed assignment and ω is an update law to ensure that the along-track error converges to zero. This term will be designed after step two to ensure that it only acts in the output space of $\boldsymbol{\eta}(t)$. Furthermore, note that Young's inequality is defined as

$$\boldsymbol{a}^{\boldsymbol{T}}\boldsymbol{b} \leq \kappa\boldsymbol{a}^{\boldsymbol{T}}\boldsymbol{a} + \frac{1}{4\kappa}\boldsymbol{b}^{\boldsymbol{T}}\boldsymbol{b}$$

Step 1: Let the output to control be the error dynamics $\boldsymbol{e} = \boldsymbol{R}^T(\psi)[\boldsymbol{\eta}(t) - \boldsymbol{\eta}_d(\theta)]$. Hence, we have that $\boldsymbol{z}_1 = \boldsymbol{e}$:

$$
\begin{aligned}
\boldsymbol{z}_1 &= \boldsymbol{e} \\
\dot{\boldsymbol{z}}_1 &= \dot{\boldsymbol{e}} = -r\boldsymbol{S}\boldsymbol{z}_1 + \boldsymbol{\nu} - \boldsymbol{R}^T(\psi)\boldsymbol{\eta}_d^s\dot{s}, \quad \text{Now define: } \boldsymbol{z}_2 = \boldsymbol{\nu} - \boldsymbol{\alpha}_1 \\
\dot{\boldsymbol{z}}_1 &= -r\boldsymbol{S}\boldsymbol{z}_1 + \boldsymbol{z}_2 + \boldsymbol{\alpha}_1 - \boldsymbol{R}^T(\psi)\boldsymbol{\eta}_d^s(\omega + \upsilon_s), \quad \text{Define } V_1 \text{ as:} \\
V_1 &= \frac{1}{2}\boldsymbol{z}_1^T\boldsymbol{z}_1 \\
\dot{V}_1 &= \boldsymbol{z}_1^T\dot{\boldsymbol{z}}_1 \\
\dot{V}_1 &= -r\boldsymbol{z}_1^T\boldsymbol{S}\boldsymbol{z}_1 + \boldsymbol{z}_1^T[\boldsymbol{z}_2 + \boldsymbol{\alpha}_1 - \boldsymbol{R}^T(\psi)\boldsymbol{\eta}_d^s(\omega + \upsilon_s)], \quad \text{Where the term } -r\boldsymbol{z}_1^T\boldsymbol{S}\boldsymbol{z}_1 = 0 \\
\dot{V}_1 &\leq \frac{1}{4\kappa}\boldsymbol{z}_2^T\boldsymbol{z}_2 + \boldsymbol{z}_1^T[\kappa\boldsymbol{z}_1 + \boldsymbol{\alpha}_1 - \boldsymbol{R}^T(\psi)\boldsymbol{\eta}_d^s(\omega + \upsilon_s)] \\
& \text{Choosing the stabilizing function } \boldsymbol{\alpha}_1 \text{ as:} \\
\boldsymbol{\alpha}_1 &= \boldsymbol{R}^T(\psi)\boldsymbol{\eta}_d^s\upsilon_s - \kappa\boldsymbol{z}_1 - \boldsymbol{K}_1\boldsymbol{z}_1 \quad \text{Yields:} \\
\dot{V}_1 &\leq \frac{1}{4\kappa}\boldsymbol{z}_2^T\boldsymbol{z}_2 - \boldsymbol{z}_1^T(K_1 + \kappa\boldsymbol{I})\boldsymbol{z}_1 - \boldsymbol{R}^T(\psi)\boldsymbol{\eta}_d^s\omega
\end{aligned}
$$

Where $\boldsymbol{K}_1 = diag(k_i),\ i = 1,\ 2,\ 3,\quad k_i > 0$. and $\kappa > 0$. Let $\tilde{\boldsymbol{K}}_1 = (\boldsymbol{K}_1 + \kappa\boldsymbol{I})$.

Design of maneuvering update law : The update law ω will be designed at this stage to ensure that it ony acts on the output space. We have that:

$$\dot{s} = \omega + \upsilon_s(t)$$

There are several ways of designing ω, whereas two will be considered. First, recall the time derivative of the CLF $\dot{V}_2$:

$$\dot{V}_1 \leq \frac{1}{4\kappa}\boldsymbol{z}_2^T\boldsymbol{z}_2 - \boldsymbol{z}_1^T(K_1 + \kappa\boldsymbol{I})\boldsymbol{z}_1 - \boldsymbol{R}^T(\psi)\boldsymbol{\eta}_d^s\omega$$

Where we design the ω according to the term:

$$\boldsymbol{\rho}\omega = -\boldsymbol{z}_2^T\boldsymbol{R}^T(\psi)\eta_d^s = V_1^s$$

- **Tracking update law:** If we chose

  $$\omega = 0$$

  We get that

  $$\boldsymbol{\rho}\omega = 0$$

  And

  $$\dot{s} = \upsilon_s$$

  Hence the update law ω will have no effect on the path variable speed $\dot{s}$ and the path speed will track the speed assignment υ_s, hence the name "Tracking update law".

- **Unit-tangent gradient update law:**

  $$\omega = -\mu\frac{\rho}{|\boldsymbol{\eta}_d^s|} = \mu\frac{\boldsymbol{\eta}_d^{s^T}}{|\boldsymbol{\eta}_d^s|}\boldsymbol{R}(\psi)\boldsymbol{z}_2$$

  Where $\frac{\boldsymbol{\eta}_d^{s^T}}{|\eta_d^s|}$ represent the unit tangent of the path. μ is a tuning parameter.

This design yields:

$$\dot{s} = \upsilon_s + \mu \frac{\boldsymbol{\eta}_d^{s^T}}{|\eta_d^s|} \boldsymbol{R}(\psi) \boldsymbol{z}_2$$

And

$$\boldsymbol{\rho}\omega = -\mu \boldsymbol{z}_2^T \boldsymbol{R}^T(\psi) \boldsymbol{\eta}_d^s \frac{\boldsymbol{\eta}_d^{s^T}}{|\eta_d^s|} \boldsymbol{R}(\psi) \boldsymbol{z}_2$$

Which is negative semi-definite. This guarantees convergence of the along track error to zero.

Since the update law is already chosen, the derivative of $\boldsymbol{\alpha}_1$ needs to be cancelled directly at the next step. Hence we need to calculate the value of $\dot{\boldsymbol{\alpha}}_1$

$$\begin{aligned}
\boldsymbol{\alpha}_1 &= \boldsymbol{R}^T(\psi)\boldsymbol{\eta}_d^s \upsilon_s - \tilde{\boldsymbol{K}}_1 \boldsymbol{z}_1 \\
\dot{\boldsymbol{\alpha}}_1 &= \boldsymbol{\sigma} + \boldsymbol{\alpha}_1^s \dot{s} \\
\dot{\boldsymbol{\alpha}}_1 &= \dot{\boldsymbol{R}}^T(\psi)\boldsymbol{\eta}_d^s \upsilon_s + \boldsymbol{R}^T(\psi)\boldsymbol{\eta}_d^{s^2} \dot{s}\upsilon_s + \boldsymbol{R}^T(\psi)\boldsymbol{\eta}_d^s \upsilon_s^t + \boldsymbol{R}^T(\psi)\boldsymbol{\eta}_d^s \upsilon_s^s \dot{s} - \tilde{\boldsymbol{K}}_1 \dot{\boldsymbol{z}}_1 \\
\dot{\boldsymbol{z}}_1 &= -r\boldsymbol{S}\boldsymbol{z}_1 + \boldsymbol{\nu} - \boldsymbol{R}^T(\psi)\boldsymbol{\eta}_d^s \dot{s} \\
\dot{\boldsymbol{\alpha}}_1 &= -r\boldsymbol{S}\boldsymbol{R}^T(\psi)\boldsymbol{\eta}_d^s \upsilon_s + \boldsymbol{R}^T(\psi)\boldsymbol{\eta}_d^s \upsilon_s^t + r\tilde{\boldsymbol{K}}_1 \boldsymbol{S}\boldsymbol{z}_1 - \tilde{\boldsymbol{K}}_1 \boldsymbol{\nu} + \\
&\quad [\boldsymbol{R}^T(\psi)\boldsymbol{\eta}_d^{s^2} \upsilon_s + \boldsymbol{R}^T(\psi)\boldsymbol{\eta}_d^s \upsilon_s^s + \tilde{\boldsymbol{K}}_1 \boldsymbol{R}^T(\psi)\boldsymbol{\eta}_d^s]\dot{s} \\
\boldsymbol{\sigma} &= -r\boldsymbol{S}\boldsymbol{R}^T(\psi)\boldsymbol{\eta}_d^s \upsilon_s + \boldsymbol{R}^T(\psi)\boldsymbol{\eta}_d^s \upsilon_s^t + r\tilde{\boldsymbol{K}}_1 \boldsymbol{S}\boldsymbol{z}_1 - \tilde{\boldsymbol{K}}_1 \boldsymbol{\nu} \\
\boldsymbol{\alpha}_1^s &= \boldsymbol{R}^T(\psi)\boldsymbol{\eta}_d^{s^2} \upsilon_s + \boldsymbol{R}^T(\psi)\boldsymbol{\eta}_d^s \upsilon_s^s + \tilde{\boldsymbol{K}}_1 \boldsymbol{R}^T(\psi)\boldsymbol{\eta}_d^s
\end{aligned}$$

Step 2: Recall that $\boldsymbol{z}_2 = \boldsymbol{\nu} - \boldsymbol{\alpha}_1$. Now, introduce the integral action state $\boldsymbol{\xi}$ to compensate for the bias $\boldsymbol{b}$. Hence we have the system:

$$\dot{\boldsymbol{z}}_1 = -r\boldsymbol{S}\boldsymbol{z}_1 + \boldsymbol{z}_2 + \boldsymbol{\alpha}_1 - \boldsymbol{R}^T(\psi)\boldsymbol{\eta}_d^s \dot{s} \tag{6.14a}$$

$$\dot{\boldsymbol{\xi}} = \boldsymbol{z}_2 \tag{6.14b}$$

$$\boldsymbol{M}\dot{\boldsymbol{z}}_2 = -\boldsymbol{D}\boldsymbol{\nu} + \boldsymbol{\tau} + \boldsymbol{b} - \boldsymbol{M}\dot{\boldsymbol{\alpha}}_1 \tag{6.14c}$$

$$
\begin{aligned}
\dot{\boldsymbol{\xi}} &= \boldsymbol{z}_2 \\
\dot{\boldsymbol{z}}_2 &= \dot{\boldsymbol{\nu}} - \dot{\boldsymbol{\alpha}}_1 \\
\boldsymbol{M}\dot{\boldsymbol{z}}_2 &= \boldsymbol{M}\dot{\boldsymbol{\nu}} - \boldsymbol{M}\dot{\boldsymbol{\alpha}}_1 \\
\boldsymbol{M}\dot{\boldsymbol{z}}_2 &= -\boldsymbol{D}\boldsymbol{\nu} + \boldsymbol{\tau} + \boldsymbol{b} - \boldsymbol{M}\dot{\boldsymbol{\alpha}}_1 \quad \text{Let the integral action error state } \tilde{\boldsymbol{\xi}} \text{ be:} \\
\tilde{\boldsymbol{\xi}} &= \boldsymbol{\xi} - \boldsymbol{K}_i^{-1}\boldsymbol{b} \quad \text{Where } \boldsymbol{K}_i = diag(k_i),\ k_i > 0,\ i = 1,2,3 \text{ is the integral gain matrix} \\
V_2 &= V_1 + \frac{1}{2}\boldsymbol{z}_2^T\boldsymbol{M}\boldsymbol{z}_2 + \frac{1}{2}\tilde{\boldsymbol{\xi}}^T\boldsymbol{K}_i\tilde{\boldsymbol{\xi}} \\
\dot{V}_2 &= \dot{V}_1 + \boldsymbol{z}_2^T\boldsymbol{M}\dot{\boldsymbol{z}}_2 + \tilde{\boldsymbol{\xi}}^T\boldsymbol{K}_i\dot{\boldsymbol{\xi}} \\
\dot{V}_2 &\leq \frac{1}{4\kappa}\boldsymbol{z}_2^T\boldsymbol{z}_2 - \boldsymbol{z}_1^T\tilde{\boldsymbol{K}}_1\boldsymbol{z}_1 + \boldsymbol{\rho}\omega + \boldsymbol{z}_2^T[-\boldsymbol{D}(\boldsymbol{z}_2 + \boldsymbol{\alpha}_1) + \boldsymbol{\tau} + \boldsymbol{b} - \boldsymbol{M}\dot{\boldsymbol{\alpha}}_1] + \tilde{\boldsymbol{\xi}}^T\boldsymbol{K}_i\boldsymbol{z}_2 \\
& \text{Using that } \tilde{\boldsymbol{\xi}}^T\boldsymbol{K}_i\boldsymbol{z}_2 = \boldsymbol{z}_2^T\boldsymbol{K}_i\tilde{\boldsymbol{\xi}} \text{ and } \tilde{\boldsymbol{\xi}} = \boldsymbol{\xi} - \boldsymbol{K}_i^{-1}\boldsymbol{b} \text{ yields:} \\
\dot{V}_2 &\leq \frac{1}{4\kappa}\boldsymbol{z}_2^T\boldsymbol{z}_2 - \boldsymbol{z}_1^T\tilde{\boldsymbol{K}}_1\boldsymbol{z}_1 + \boldsymbol{\rho}\omega + \boldsymbol{z}_2^T[-\boldsymbol{D}(\boldsymbol{z}_2 + \boldsymbol{\alpha}_1) + \boldsymbol{\tau} + \boldsymbol{b} - \boldsymbol{b} + \boldsymbol{K}_i\boldsymbol{\xi} - \boldsymbol{M}\dot{\boldsymbol{\alpha}}_1] \\
&= \frac{1}{4\kappa}\boldsymbol{z}_2^T\boldsymbol{z}_2 - \boldsymbol{z}_1^T\tilde{\boldsymbol{K}}_1\boldsymbol{z}_1 + \boldsymbol{\rho}\omega + \boldsymbol{z}_2^T[-\boldsymbol{D}(\boldsymbol{z}_2 + \boldsymbol{\alpha}_1) + \boldsymbol{\tau} + \boldsymbol{K}_i\boldsymbol{\xi} - \boldsymbol{M}\dot{\boldsymbol{\alpha}}_1] \\
\boldsymbol{\tau} &= -\tilde{\boldsymbol{K}}_2\boldsymbol{z}_2 - \boldsymbol{K}_i\boldsymbol{\xi} + \boldsymbol{D}\boldsymbol{\alpha}_1 + \boldsymbol{M}\dot{\boldsymbol{\alpha}}_1 \\
& \text{Where } \tilde{\boldsymbol{K}}_2 = (\boldsymbol{K}_2 + \frac{1}{4\kappa}),\ \boldsymbol{K}_2 = diag(k_i),\ i = 1,2,3\ K_i > 0. \text{ Hence:} \\
\dot{V}_2 &\leq -\boldsymbol{z}_1^T\tilde{\boldsymbol{K}}_1\boldsymbol{z}_1 - \boldsymbol{z}_2^T\boldsymbol{K}_2\boldsymbol{z}_2
\end{aligned}
$$

Since the derivative of the accumulated lyapunov function

$$\dot{V}_2 \leq -\boldsymbol{z}_1^T\tilde{\boldsymbol{K}}_1\boldsymbol{z}_1 - \boldsymbol{z}_2^T\boldsymbol{K}_2\boldsymbol{z}_2 < 0$$

The control law is proved to be asymptotic stable within the region where the dynamic model is valid. To summarize, the resulting **control law function** is given by

$$
\begin{aligned}
\boldsymbol{z}_1 &= \boldsymbol{R}(\psi)^T[\boldsymbol{\eta} - \boldsymbol{\eta}_d] && (6.15) \\
\boldsymbol{\alpha}_1 &= -(\boldsymbol{K}_1 + \kappa\boldsymbol{I})\boldsymbol{z}_1 + \boldsymbol{R}(\psi)^T\boldsymbol{\eta}_d^s\upsilon_s && (6.16) \\
\boldsymbol{z}_2 &= \boldsymbol{\nu} - \boldsymbol{\alpha}_1 && (6.17) \\
\boldsymbol{\sigma}_1 &= r(\boldsymbol{K}_1 + \kappa I)\boldsymbol{S}\boldsymbol{z}_1 - (\boldsymbol{K}_1 + \kappa\boldsymbol{I})\boldsymbol{\nu}) - r\boldsymbol{S}\boldsymbol{R}(\psi)^T\boldsymbol{\eta}_d^s\upsilon_s + \boldsymbol{R}(\psi)^T\boldsymbol{\eta}_d^s\upsilon_s^t && (6.18) \\
\boldsymbol{\alpha}_1^s &= (\boldsymbol{K}_1 + \kappa\boldsymbol{I})\boldsymbol{R}(\psi)^T\boldsymbol{\eta}_d^s + \boldsymbol{R}(\psi)^T\boldsymbol{\eta}_d^{s^2}\upsilon_s + \boldsymbol{R}(\psi)^T\boldsymbol{\eta}_d^s\upsilon_s^s && (6.19)
\end{aligned}
$$

With the Control law:

$$
\begin{aligned}
\dot{s} &= \upsilon_s + \omega && (6.20) \\
\dot{\boldsymbol{\xi}} &= \boldsymbol{z}_2 && (6.21) \\
\boldsymbol{\tau} &= -\tilde{\boldsymbol{K}}_2\boldsymbol{z}_2 - \boldsymbol{K}_i\boldsymbol{\xi} + \boldsymbol{D}\boldsymbol{\alpha}_1 + \boldsymbol{M}\dot{\boldsymbol{\alpha}}_1 && (6.22)
\end{aligned}
$$

Chapter 7

Results and discussion

This chapter will present the results obtained through simulations performed by the autonomous system presented in chapter 5 and chapter 6, applied to the simulation model derived in chapter 4.
Performance is evaluated according performance criterions presented in Section 3.5.5. To make the simulations more realistic, the desired body control forces $\boldsymbol{\tau}_d$ are filtered by a first order low pass filter $h_{lp} = \frac{1}{Ts+1}$ to simulate thruster dynamics. The time constant T is set to $T = 0.4$, which is a conservative estimate of the delay between desired control force $\boldsymbol{\tau}_d$ and the control signal $\boldsymbol{\tau}_{d,filtered}$ that are applied on the simulation model. Furthermore, saturation elements have been added to all body forces. The saturation limits are beyond the calcuated max thrust limits in Section 3.3 by good margin.

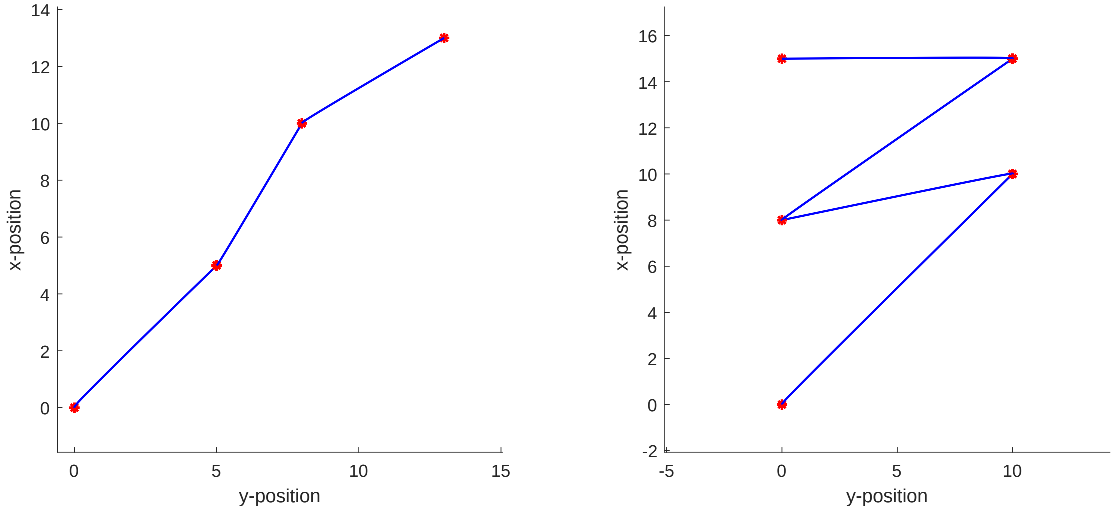
7.1 Parameter study

Both in the guidance system and path generation system there are tuning parameters. In order to improve system performance, it is interesting to see how these parameters individually affect the performance.

The tuning parameters are

- $u_{d,max}$ - Speed assignment limit
- $\dot{u}_{d,max}$ - Acceleration assignment limit
- μ - Gain for the unit gradient update law
- λ - Path parameter to determine curvature

In addition, the dynamic assignment can be manipulated by commanding stops and starts during operation.



(a) Path A: Simple path test

(b) Path B: Demanding path test

Figure 7.1: Paths used in the case study for the dynamic task.

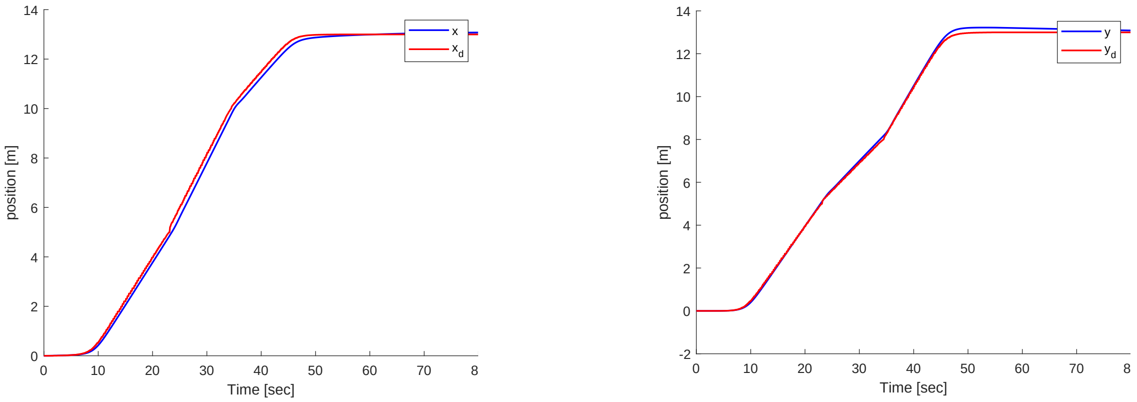
The paths that will be used in the case study are presented in Figure 7.1.The path in Figure 7.1a is a simple test with small curvature and little change of direction in the way-point transitions. The path in Figure 7.1b on the other hand consist of more demanding transitions due to high curvature. It is chosen to

7.1.1 Case 1: Path A

As a staring point, the tuning parameters are set to

- $u_{d,max} = 0.5$
- $\dot{u}_{d,max} = 0.2$
- $\mu = 0$
- $\lambda = 0.35$

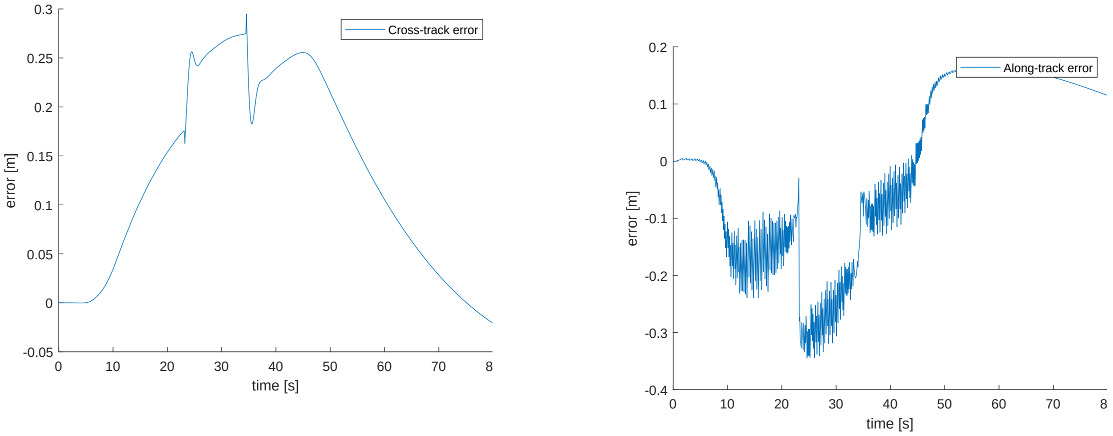
With these inital parameter values, we get the following simulation results for path A:



(a) Desired and measured x position

(b) Desired and measured y position

Figure 7.2: Tracking performance of position references.

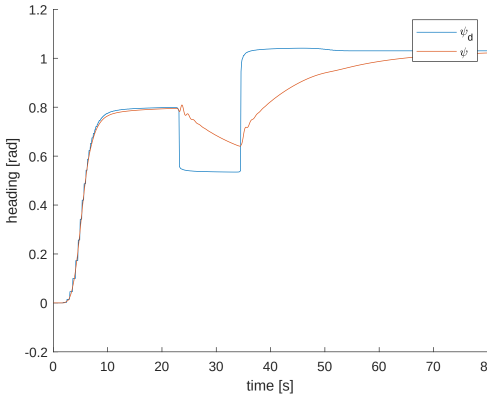


(a) Cross-track error

(b) Along-track error

Figure 7.3: Cross-track and along track error.

From Figure 7.2 and 7.5 shows that the path tracking is quite satisfactory. The oscillations in the along-track is due to the discrete change in reference value which is a consequence of that the path variable s is updated discrete in the guidance law.



(a) Desired and measured heading

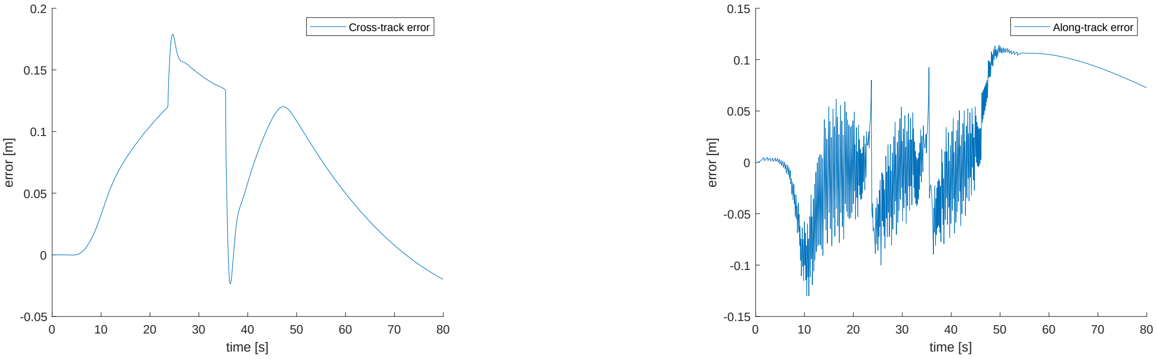
(b) Desired and measured speed

Figure 7.4: Performance of heading and speed control.

Figure 7.4a shows that the heading initially follows its reference trajectory perfectly, but in the way-point transitions, the reference heading changes to fast.

Tuning μ:

By setting $\mu \neq 0$, the *unit gradient update law* is introduced. In Section 6.4, it was shown that this update law guarantees convergence of the along-track error to zero. By letting $\mu = 0.02$, and other parameters be constant, we get the results as follows:



(a) Cross-track error.

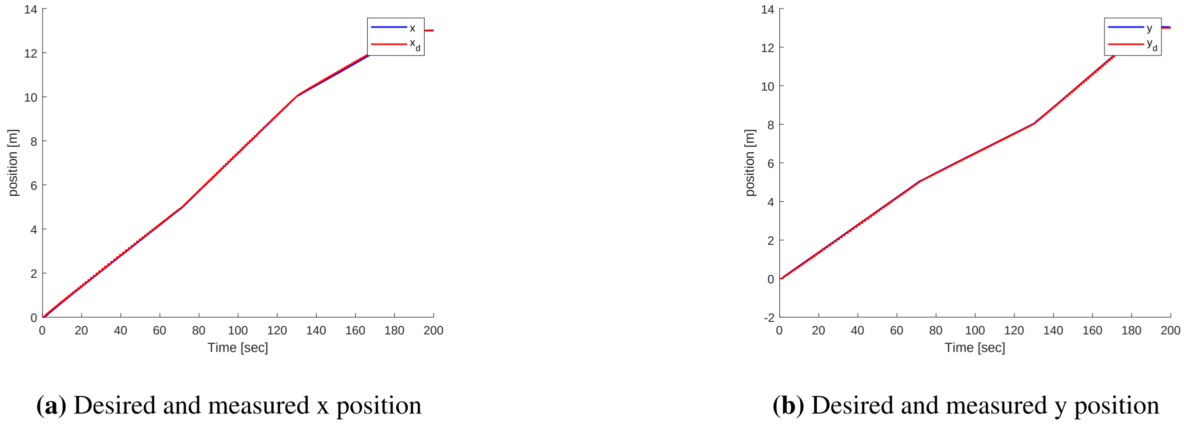
(b) Along-track error.

Figure 7.5: Cross-track and along track error.

As shown in Figure 7.5b, the along-track error oscillates about zero, as anticipated. In addition, we see that the cross-track error also decreases. This makes sense since the cancellation of the along-track error means that a "spatial delay" between $\boldsymbol{p}_d(s,t)$ and $\boldsymbol{p}(t)$ is cancelled. The unit gradient update law handles the along-track error. This isolates the controller task to minimize the cross-track error., and performs the path following task better.

Tuning $u_{d,max}$

As illustrated above, the controller cannot handle such rapid changes in heading reference as required in the way-point transitions. One solution is to demand a lower speed u_d. To illustrate the isolated effect of reducing desired speed, we set the tuning parameter $\mu = 0$. This way, the tracking problem is reintroduced. Choosing $u_{d,max} = 0.1$ gives us the following results:

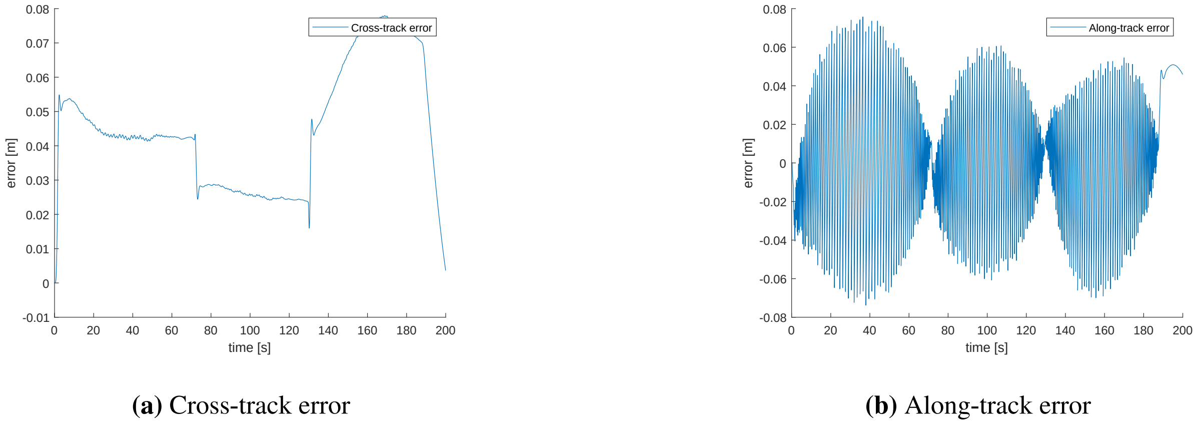


(a) Desired and measured x position

(b) Desired and measured y position

Figure 7.6: Performance tracking of position references

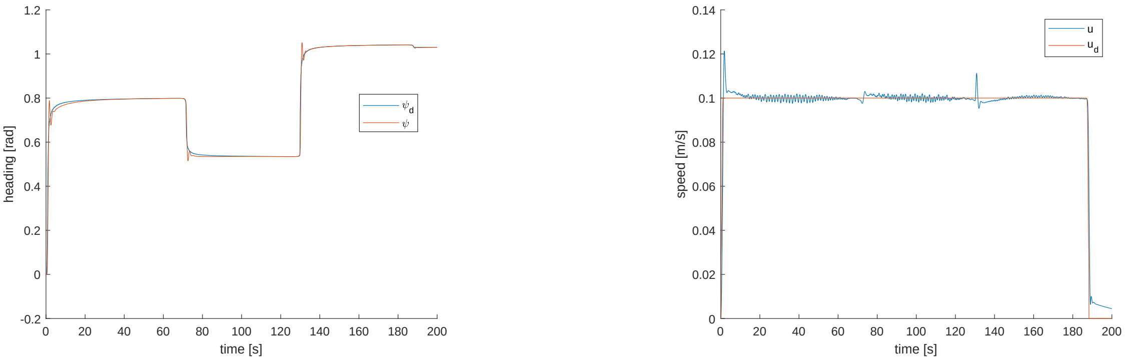
When considering a tracking problem ($\mu = 0$), changing the speed assignment is equivalent to tuning a reference filter. By comparing Figure 7.2 and Figure 7.6, this is evident. The reference trajectories in x and y position is more aggressive in Figure 7.2 where $u_{d,max} = 0.5$, than in Figure7.6, where $u_{d,max} = 0.1$.



(a) Cross-track error

(b) Along-track error

Figure 7.7: Cross-track and along track error.



(a) Desired and measured heading

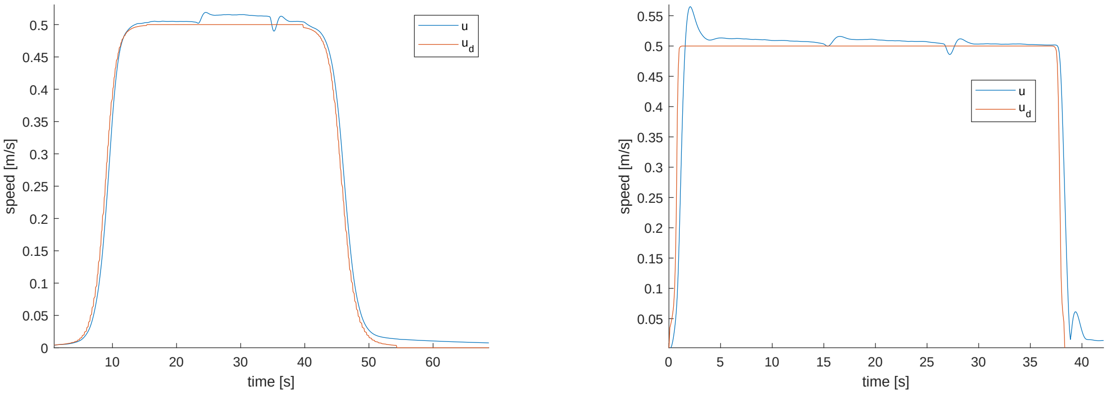
(b) Desired and measured speed

Figure 7.8: Heading and speed.

Both Figure 7.7 and Figure 7.8 emphasizes how trajectory tracking is improved by reducing u_d. The reference trajectories are made more feasible.
However, by reducing commanded speed raises the operation time considerably. Compared With a maximum commanded speed $u_{d,max} = 0.5m/s$, it takes five times longer to complete this path with $u_{d,max} = 0.1m/s$.

Tuning $\dot{u}_{d,max}$

By changing $\dot{u}_{d,max}$ and letting rest of the parameters be as the initial values, we get the following result:



(a) Desired and measured speed with $\dot{u}_{d,max} = 0.1m/s^2$

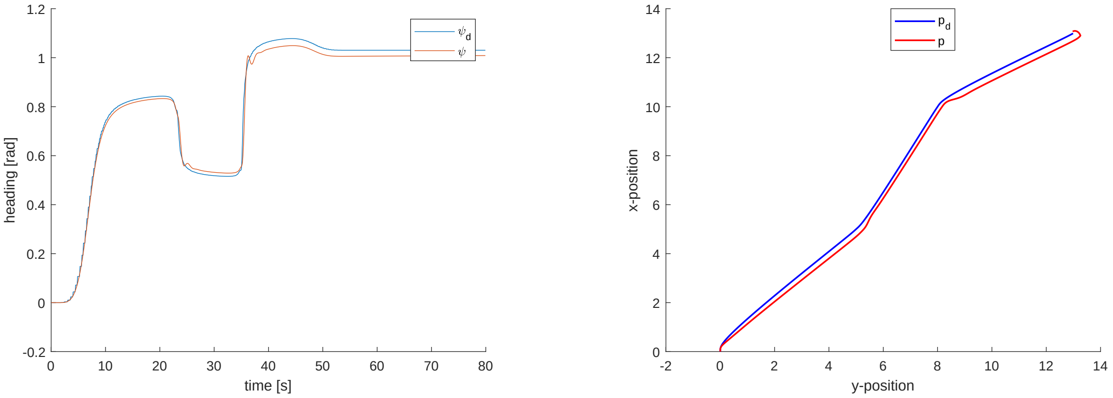
(b) Desired and measured speed with $\dot{u}_{d,max} = 1.0m/s^2$

Figure 7.9: Tuning $\dot{u}_{d,max}$

In Figure 7.9a the speed follows its desired value to a higher degree than in Figure 7.9b in the start and stop phase. This motivates a mild tuning of $\dot{u}_{d,max}$ in cases where it is specifically important that the craft behaves as desired in the start or stop phase of an operation.

Tuning path parameter λ

Keeping the initial value of the rest of the tuning parameters, and tuning λ gives the following results for $\lambda = 1.5$



(a) Desired and measured heading

(b) Desired and measured values of the path with $\lambda = 1.5$

Figure 7.10: Effect of the tuning parameter λ.

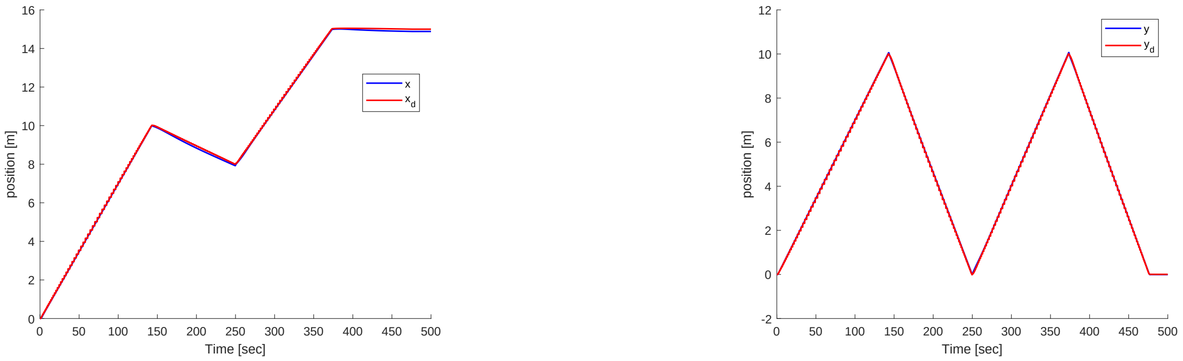
Furthermore, the performance of the path following was approximately the same with cross-track error in same range as for the initial simulation.

7.1.2 Case 2: Path B

This case is far more challenging for the system. In order to generate feasible trajectories in heading so that the security requirement of pointing the front camera along the desired path, the time requirements are violated. This point is illustrated in the following simulation scenario.

- $u_{d,max} = 0.1 m/s$
- $\dot{u}_{d,max} = 0.09 m/s^2$
- $\mu = 0.02$
- $\lambda = 0.35$

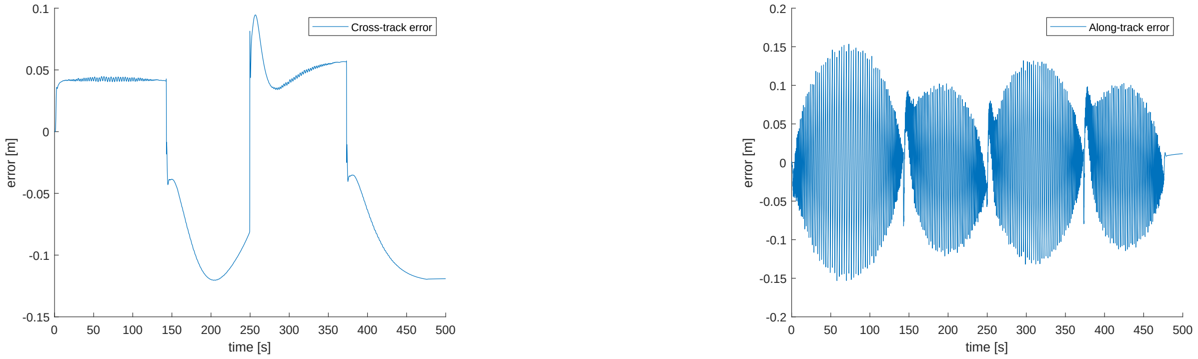
This value of $u_{d,max} = 0.1 m/s$ is considered to be at a minimum according to the time requirement. The simulation results as as follows:



(a) Desired and measured x position.

(b) Desired and measured y position.

Figure 7.11: Tracking performance of position references.

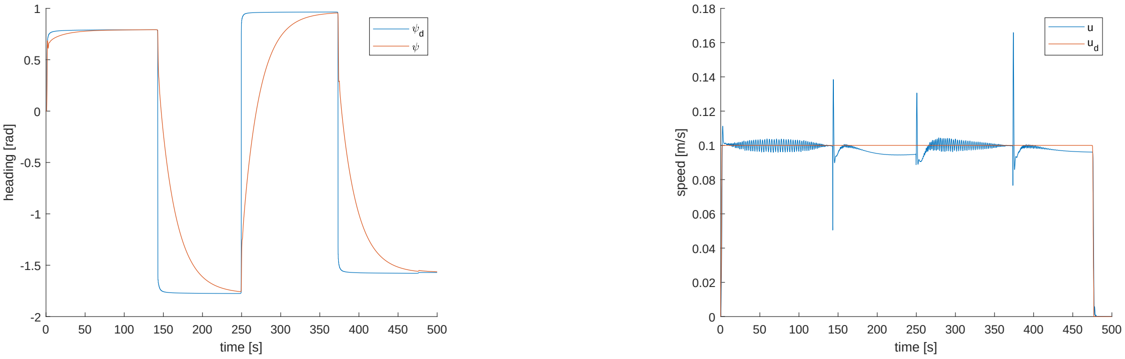


(a) Cross-track error.

(b) Along-track error.

Figure 7.12: Cross-track and along track error.

Figure 7.11 and Figure 7.12 indicates that the path following is desirable. However, it is worth mentioning that the amplitude of noise in the along-track error is particularly high. This leads to more noisy speed measurements as well.



(a) Desired and measured heading.

(b) Desired and measured speed.

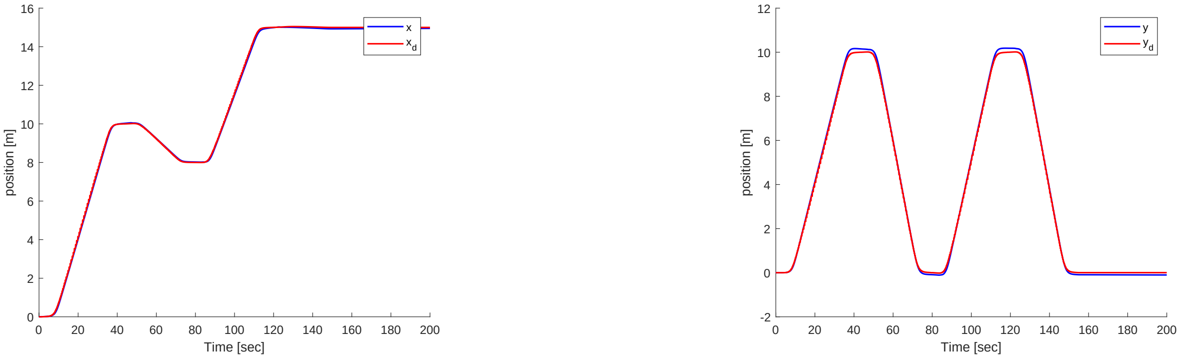
Figure 7.13: Performance of heading and speed control.

Figure 7.13a Shows the performance of tracking of desired heading. This is not satisfactory.

Varying u_d during transit

The purposed solution to the problem presented in the case above is to vary the desired speed along the path by stopping or slowing down at path segment transitions. Following parameter values where used in simulation:

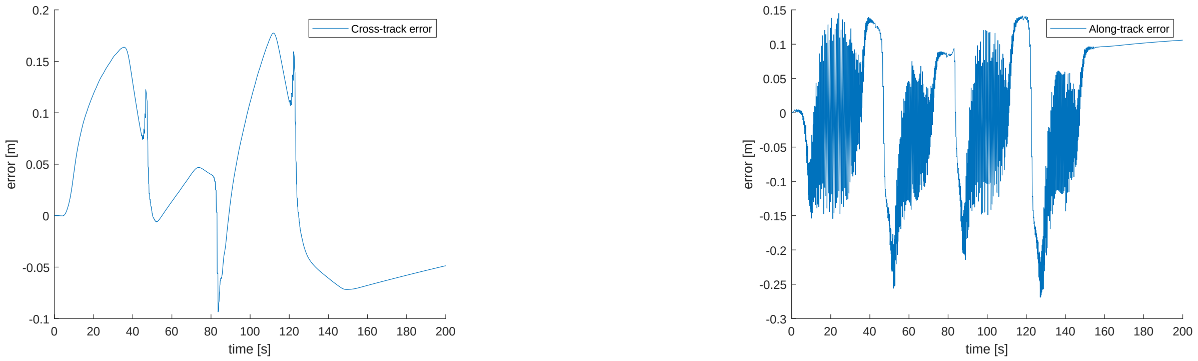
- $u_{d,max} = 0.5 m/s$
- $\dot{u}_{d,max} = 0.1 m/s^2$
- $\mu = 0.03$



(a) Desired and measured x position.

(b) Desired and measured y position.

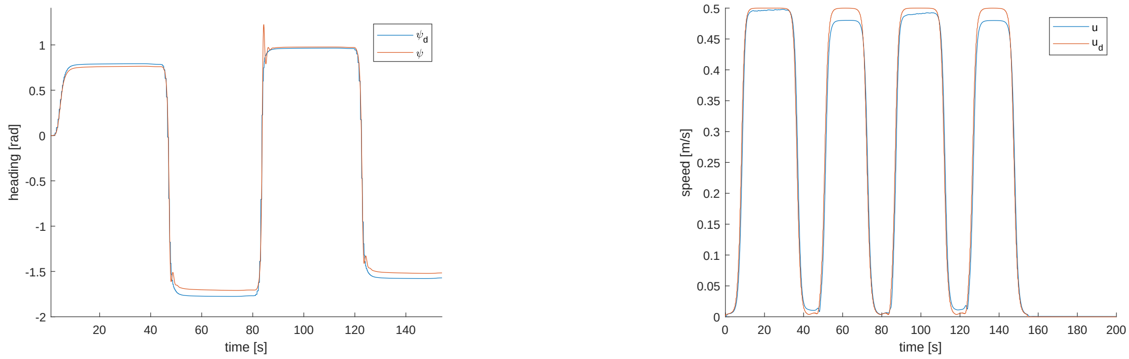
Figure 7.14: Tracking performance of position references.



(a) Cross-track error.

(b) Along-track error.

Figure 7.15: Cross-track and along track error.



(a) Desired and measured heading.

(b) Desired and measured speed.

Figure 7.16: Heading and speed.

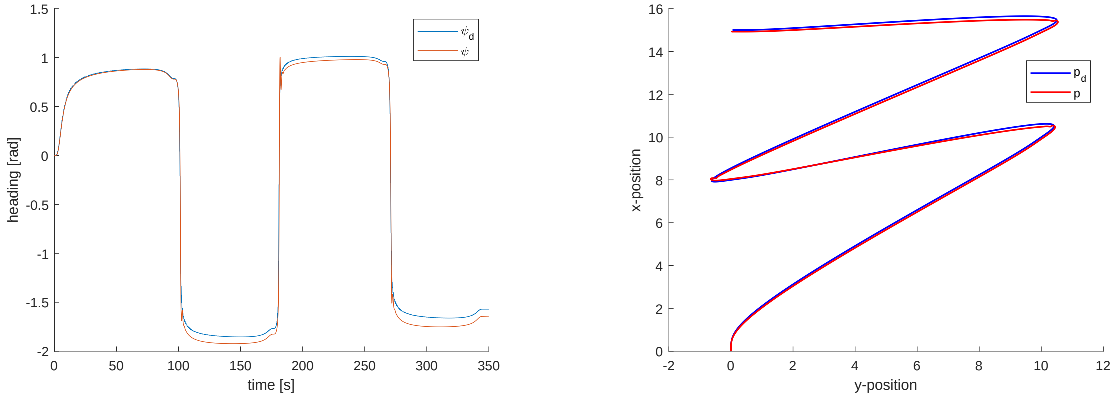
Despite high speed, the results of trajectory following in both position and heading where satisfactory. Furthermore, despite three long stops, the total operation time was less than three times the times the operation time in the previous simulation.

Tuning path parameter λ

With the following values of the parameters:

- $u_{d,max} = 0.15m/s$
- $\dot{u}_{d,max} = 0.1m/s^2$
- $\mu = 0.03$
- $\lambda = 5$

gives the following simulation results



(a) Desired and measured heading

(b) Desired and measured values of the path with $\lambda = 5$.

Figure 7.17

Which are considered satisfactory in terms of path following and trajectory tracking for the heading.

7.1.3 Remarks

This section has shown how powerful the methodology introduced by the Maneuvering Problem Skjetne [2005] is in terms of designing and tuning the dynamic assignment independently of the geometry of the path. The dynamic assignment can continuously change according to the current performance measure, which makes the system very flexible in complex operations.
Simulations with tuning the path parameter λ have given great results in terms of solving performance requirement problems related to reference following for the heading. However, it is important to highlight a major drawback with high values of lambda. If the way-points are to close, and the path parameter too high, the path generated will contain undesirable shapes. This is demonstrated in the following figure:

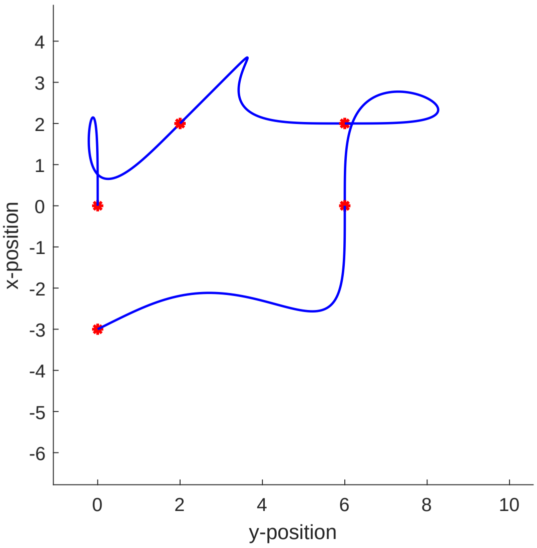


Figure 7.18: Undesirable path. $\lambda = 10$.

For this reason, it is necessary to test the value of λ for sequences of way-points that could be realistic according to a given operation.

7.2 Controller robustness against modelling errors

The control design model for the maneuvering controller is based on the simulation model derived in Chapter 4. This derivation was based on many assumptions and approximations leaving room for uncertainties in the model. Thus, it is interesting to evaluate the controllers robustness against modelling errors. The following parameter values are chosen, and will be constant in this section:

- $u_{d,max} = 0.2$
- $\dot{u}_{d,max} = 0.05$
- $\mu = 0.02$
- $\lambda = 0.8$

Furthermore, the test path chosen for this section is illustrated in the following figure:

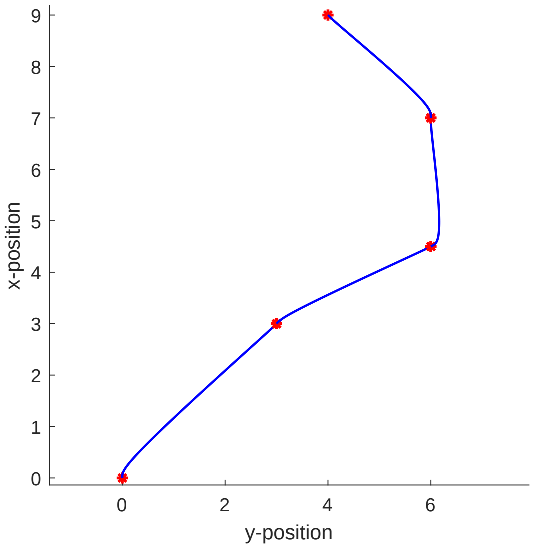
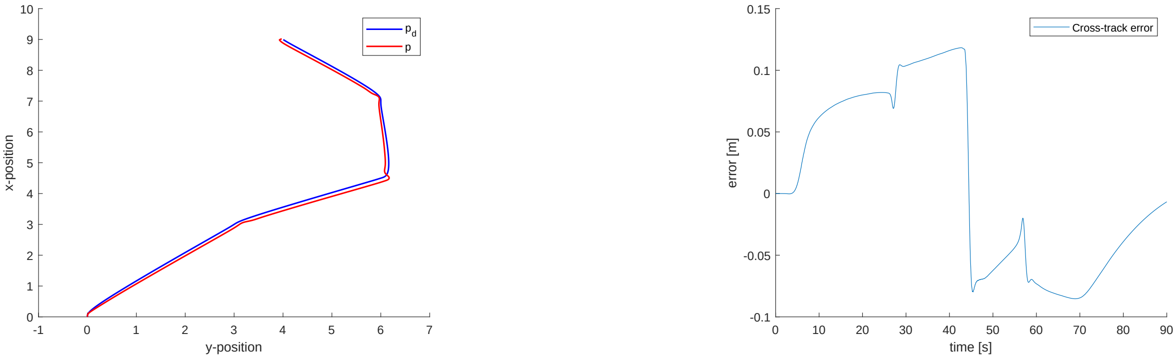


Figure 7.19: Test path.

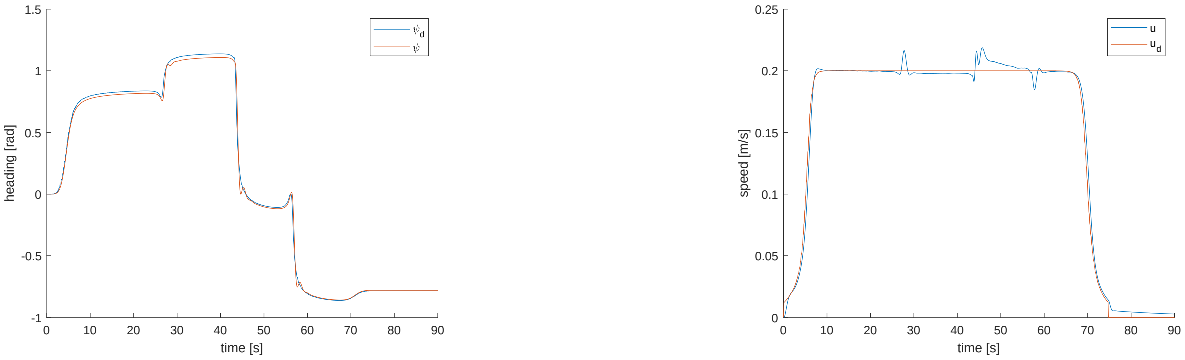
The path and the parameters where chosen to so that the system will perform well in the case of a perfect control design model. That is, a control design model that correspond perfect to the simulation model, which was the case in the previous section. The simulation results in that case are:



(a) Desired and measured path.

(b) Cross-track error.

Figure 7.20: Tracking performance of position references



(a) Desired and measured heading.

(b) Desired and measured speed.

Figure 7.21

7.2.1 Simulating model error

The model errors are simulated the following way. The control design model is provided with $\boldsymbol{M}_{estim}$ amd $\boldsymbol{D}_{estim}$.Then, the simulation model is calculated the following way:

$$\boldsymbol{M}_{sim} = \boldsymbol{M}_{estim} \pm \alpha \boldsymbol{M}_{estim} \tag{7.1a}$$

$$\boldsymbol{D}_{sim} = \boldsymbol{D}_{estim} \pm \alpha \boldsymbol{D}_{estim}, \tag{7.1b}$$

where $\boldsymbol{M}_{sim}$ and $\boldsymbol{D}_{sim}$ are the matrices used in the simulation model, $\boldsymbol{M}_{estim}$ and $\boldsymbol{D}_{estim}$ are the estimated matrices from Chapter 4 and α is a factor proportional to the deviation between control design model and simulation model. By producing the error in this manner, the assumption of diagonal matrices are maintained.

It is assumed that the correct model is within fifty percent of the estimated values. Roughly, this means that $\alpha \leq 0.5$.

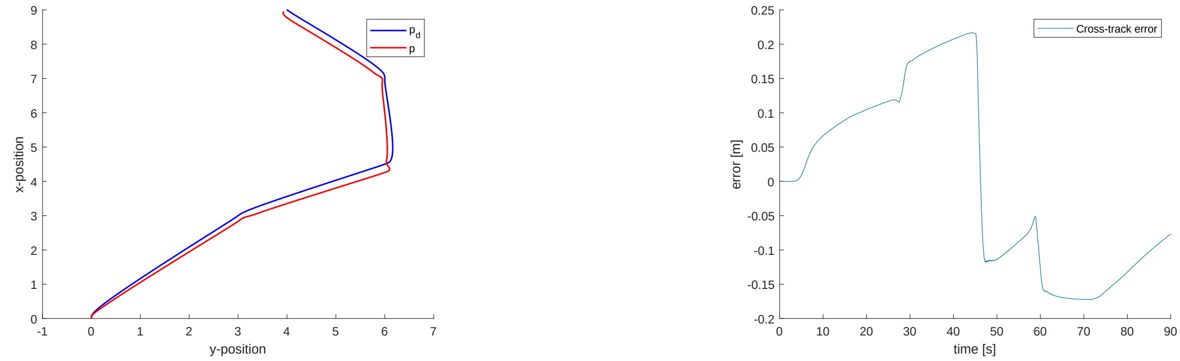
7.2.2 Underestimate of pyhsical model, $\alpha = 0.5$

An underestimate is produced by letting the values of the simulation model be larger than the values of the control design model. Letting the matrix elements in the simulation model be 50 percent larger yields:

$$\boldsymbol{M}_{sim} = \boldsymbol{M}_{estim} + 0.5\boldsymbol{M}_{estim} \tag{7.2a}$$

$$\boldsymbol{D}_{sim} = \boldsymbol{D}_{estim} + 0.5\boldsymbol{D}_{estim}. \tag{7.2b}$$

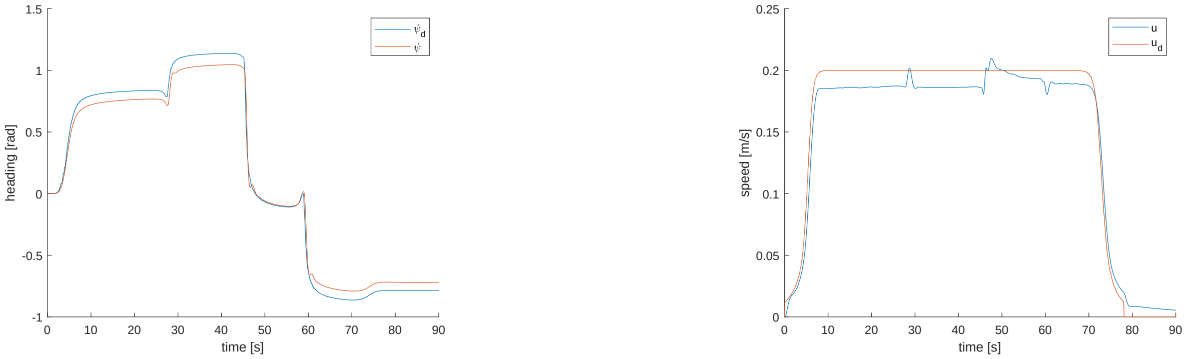
This yields the following simulation results:



(a) Desired and measured path

(b) Cross-track error

Figure 7.22: Performance tracking of position references



(a) Desired and measured heading

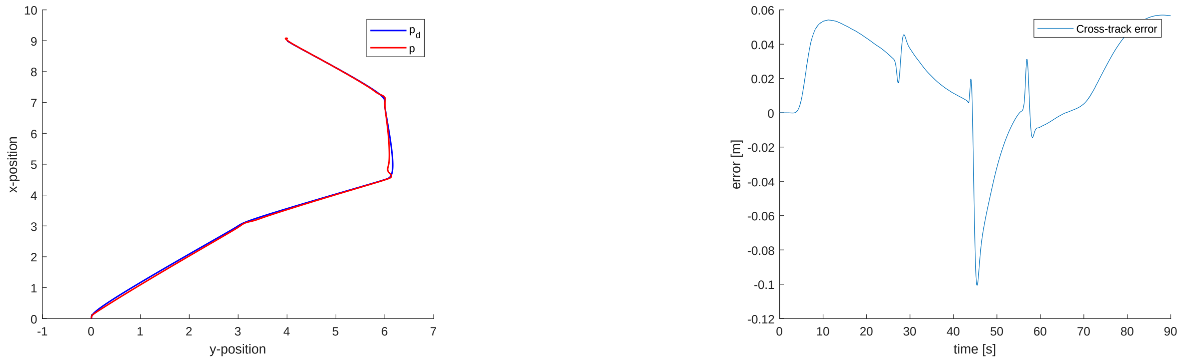
(b) Desired and measured speed

Figure 7.23

From these results, we can see that the performance clearly is worse than if the control design model correspond 100 percent to the simulation model. However, the performance is still acceptable. That is, the cross-track error is within a range of 0.25 meters, and the deviation between ψ and ψ_d does not exceeds more than approximately six degrees.

Integral action

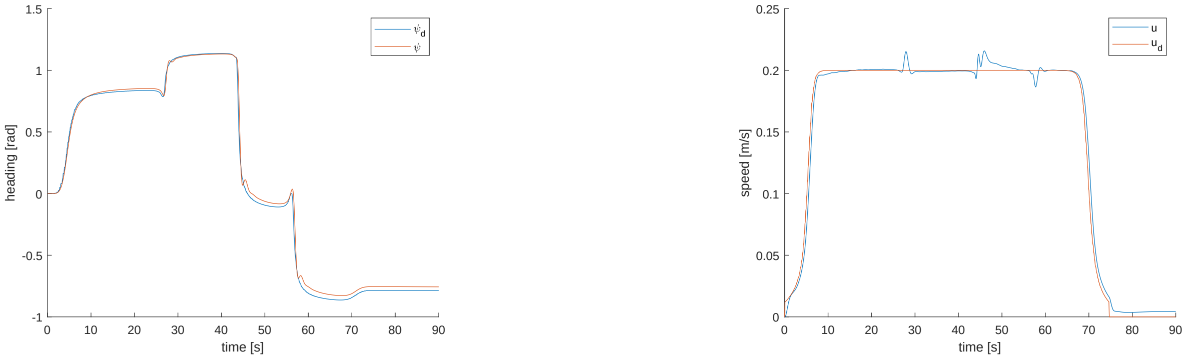
Until this point the backstepping maneuvering controller was to be considered a PD-controller. We now aims to cancel the unmodeled dynamics by introducing integral action. This was in the derivation of the controller in Section 6.4. The simulation results are:



(a) Desired and measured path

(b) Cross-track error

Figure 7.24: Performance tracking of position references



(a) Desired and measured heading

(b) Desired and measured speed

Figure 7.25

The results after introducing integral action are satisfactorily.

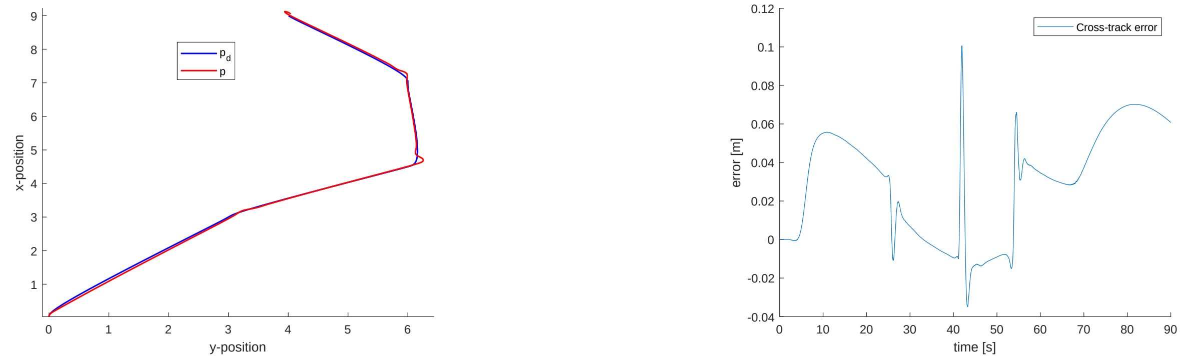
7.2.3 Overestimate of physical model, $\alpha = 0.5$

Overestimate of the physical model can be simulated my making the values in the simulation model smaller than the control design model. In this case, the matrix elements in the damping and mass matrices of the simulation model are 50 percent smaller than the estimated model:

$$\boldsymbol{M}_{sim} = \boldsymbol{M}_{estim} - 0.5\boldsymbol{M}_{estim} \tag{7.3a}$$

$$\boldsymbol{D}_{sim} = \boldsymbol{D}_{estim} - 0.5\boldsymbol{D}_{estim}. \tag{7.3b}$$

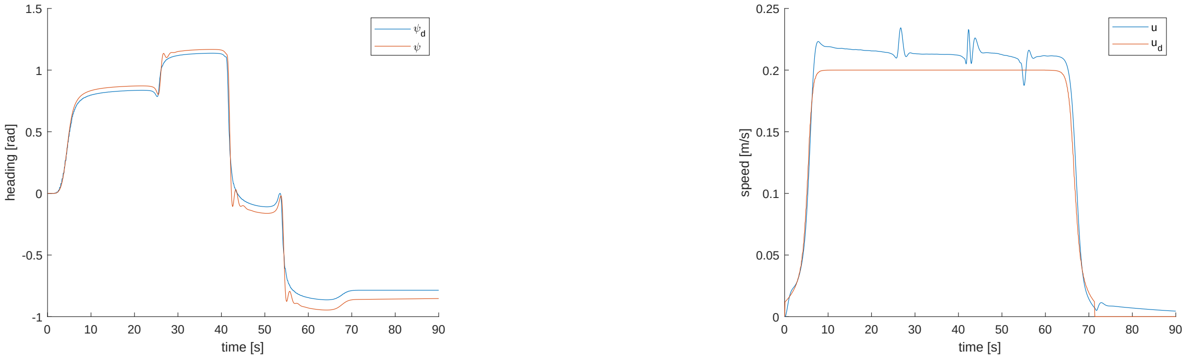
Simulation results are:



(a) Desired and measured path

(b) Cross-track error

Figure 7.26: Performance tracking of position references

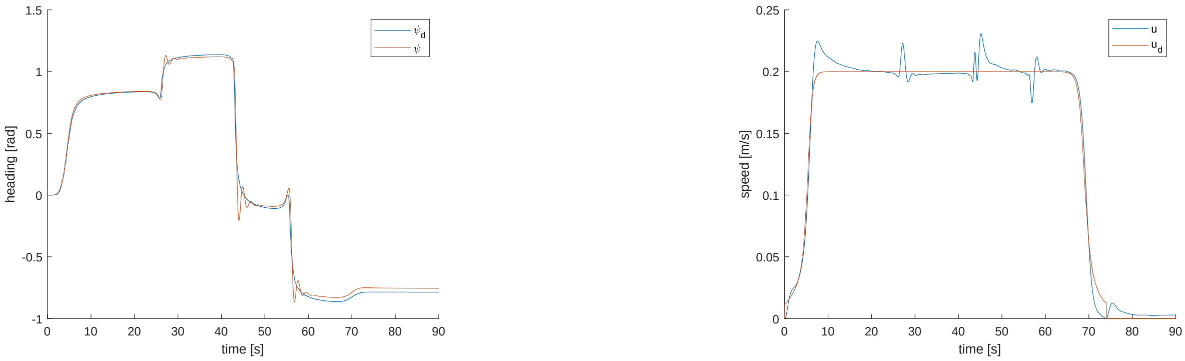


(a) Desired and measured heading

(b) Desired and measured speed

Figure 7.27

The behavior of this system is more oscillatory than the in the case of underestimating the physical properties. This is most evident in the heading and speed plots. The reason for this is intuitive. Overestimating the physical properties is equivalent to aggressive tuning of the control system, which can cause oscillatory behavior. Introducing integral action will help cancelling the gap between desired and measured values. However it will *not* counteract the oscillatory behaviour, if anything, make matters worse. This is best illustrated in the two following plots



(a) Desired and measured heading

(b) Desired and measured speed

Figure 7.28: Speed and heading plots after introducing integral action

By comparison of Figure 7.28 and Figure 7.27, we see that the integral action contributes to even more oscillatory behavior.

7.2.4 Remarks

From the results of this section, one can easily argue that it is better to underestimate the physical properties of the plant. This is the safest solution in terms of stability, and it is also easy to compensate the unmodeled dynamics through integral action.

7.3 Resulting system

The purpose of simulating the entire system is to show that the missiom manager is able to schedule tasks according to percieved information. All plots in this section is from the same run.

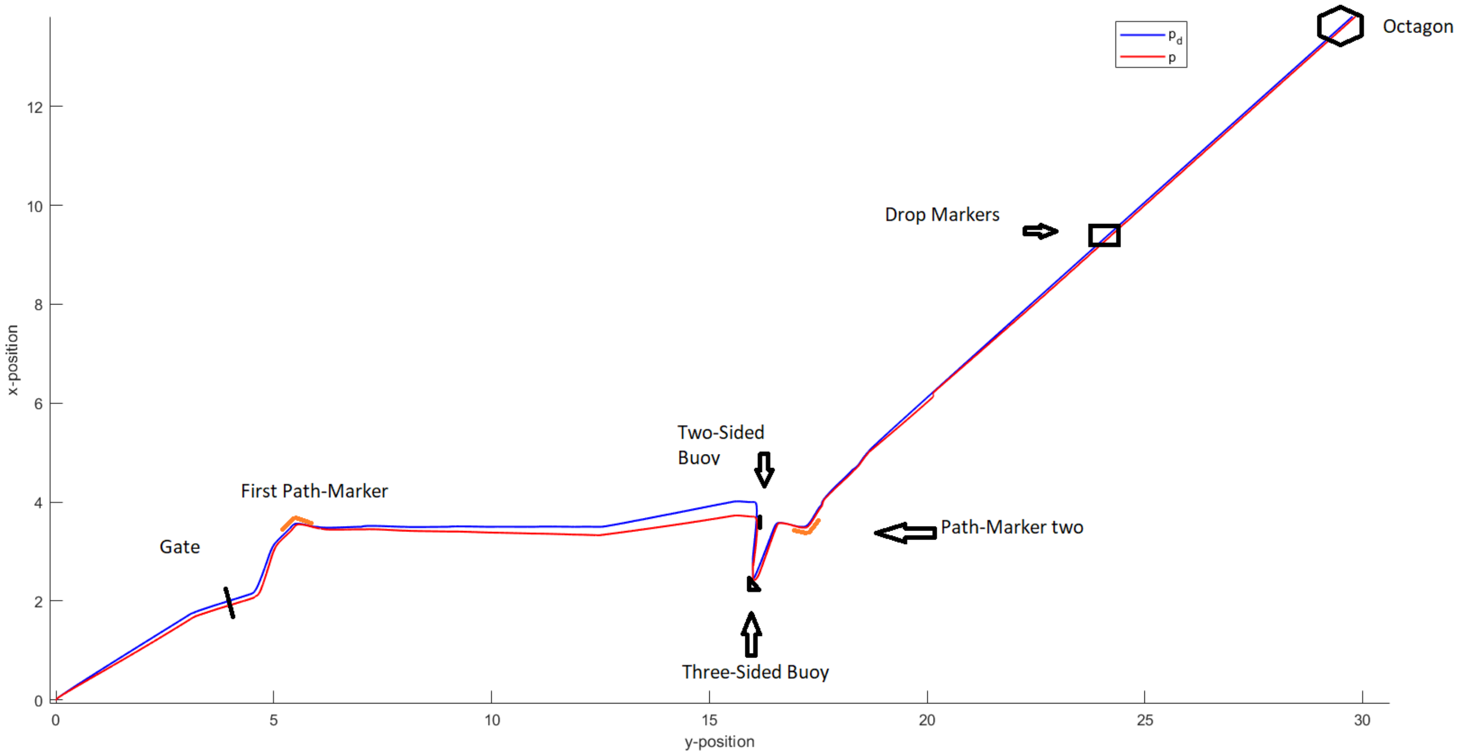
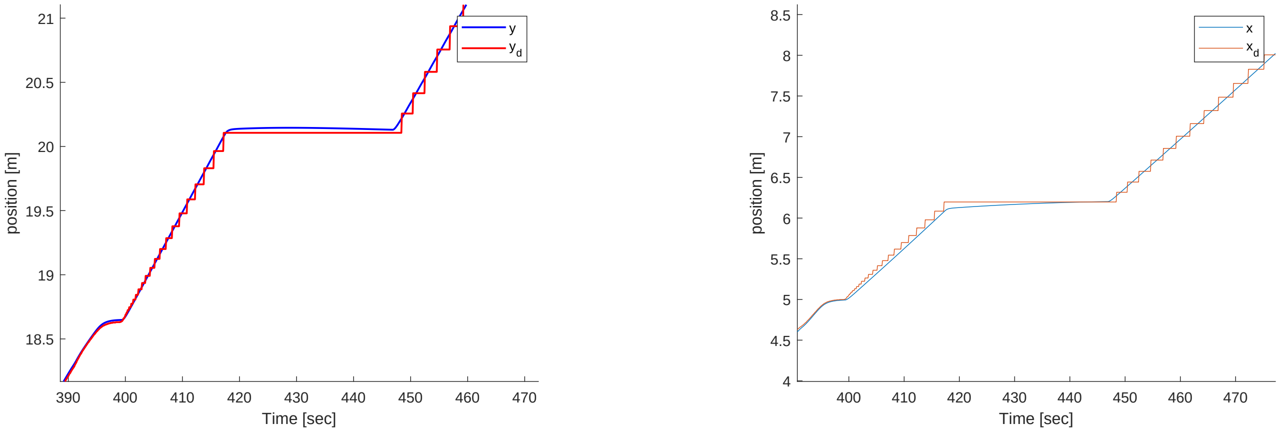


Figure 7.29: Entire Run. The tasks are indicated in the plot

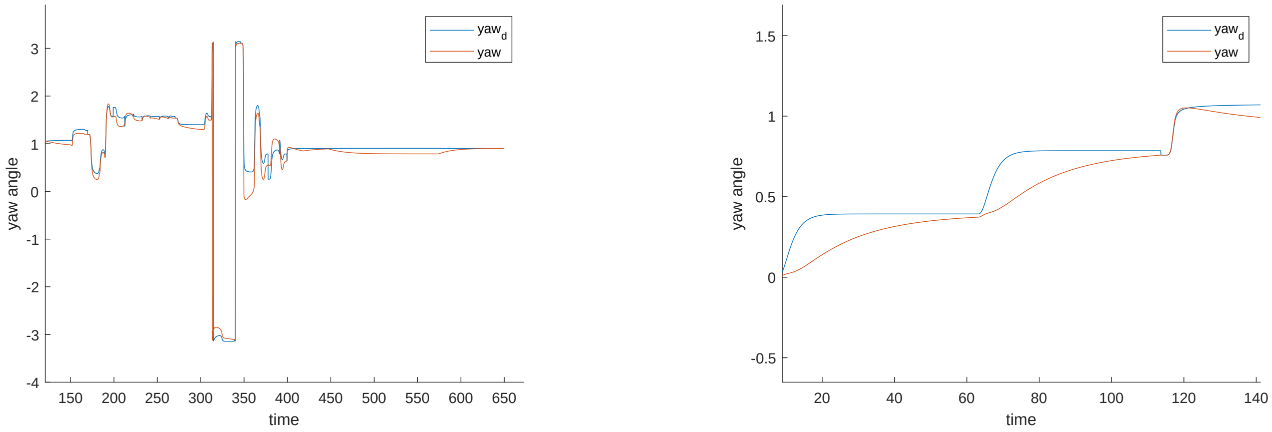


(a) y position

(b) x position

Figure 7.30: Crafts approach when dynamical object is detected

In the figure above, we can see how the risk system instantaneously reacts to dynamic hazards detected. The craft slows down controllable, and waits on the spot until the hazard is gone.

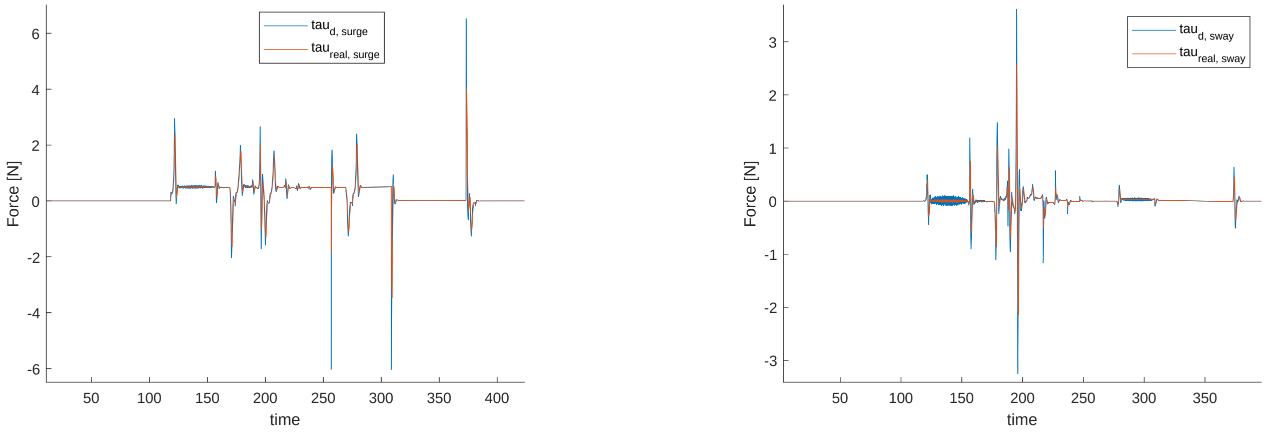


(a) Heading entire run

(b) Searching for gate

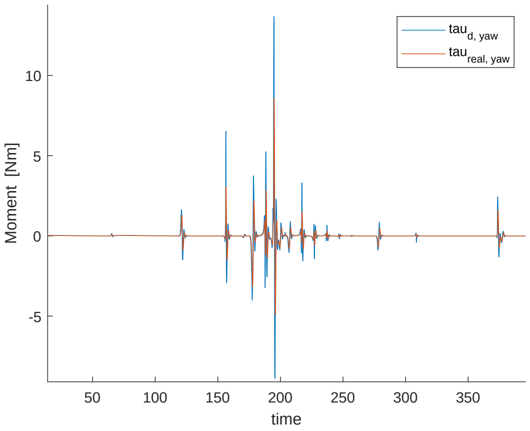
Figure 7.31: Heading performance

Figure 7.31b shows how the craft generates new heading references as long as it searches for the gate. When the gate is found, the run begins.



(a) controller body force in surge

(b) controller body force in sway



(c) controller body force in yaw

Figure 7.32: Crafts approach when dynamical object is detected

Body control Forces. The signals with highest amplitude and highest frequency are the control output from the controller, while the other signal is filtered through a low pass filter simulating the thruster dynamics.

Overall, we can see that the performance is not optimal. This can be improved through the design of dynamic assignment, tuning of the path curvature parameter λ path generation as shown in Section 7.1 and by tuning controller gains. By following the methodology provided in Section 7.1, designing a system with high performance with respect to all performance measures presented in Section 3.5.5 is a managable task.

Chapter 8

Concluding remarks

8.1 Conclusion

This thesis has presented the main components of an AUV related to control and supervision of the system. The relevant system components are the Mission managment system, Path Generation module, and a guidance and control system based on the maneuvering problem Skjetne [2005]. A simulation model of the Vortex drone is derived, and used as platform for testing and developement of the autonomous system.

The mission management system works well in simulations. That is, it provides the rest of the system with necessary commands and references according to mission objectives and perceived information. The hierarchical structure of the mission states provides a clear structure which makes it easy to implement new behaviors in terms of sub tasks when the structure gets large and easy to debug. However, logic related to recovery modes such as if the system fails a submission and will either continue towards next submission or try to complete current mission is not implemented.

The path generation provides smooth references and ensures bumpless transfers in transitions between way-points. In terms of performance measures presented in Section 3.5.5, the most challenging task is to balance the between high performance in following desired heading and minimizing operation time. The best solution to this was found in a combination between generating paths with relative low curvature and designing a flexible dynamic assignment, where the speed of the path variable is determined by the curvature of the path when considering a craft in transit.

By simulating modeling errors it was shown that the controller is robust against modeling errors. By providing 50 percent modeling error, both by simulating an overestimation and underestimation of 50 percent in both mass matrix and damping matrix, the craft still performed relatively satisfactorily. By introducing integral action, error between desired and measured state as a consequence of modeling error was to a large extent cancelled. An important remark is that overestimating the vessel dynamics is more problematic than the underestimating since overestimating can lead to oscillatory behavior. This is also something the integral action cannot compensate.

8.2 Further Work

In this thesis the foundation has been laid for a platform in Simulink for testing and development for an autonomous system. An more accurate simulation model of the Vortex drone could make this platform even more valuable. The platform could then be used for tuning the motion control system that is used on the real robot.

Furthermore, by doing some adjustments, the mission manager can be implemented in on the real drone by utilizing the interface between Simulink and ROS.

The guidance and control system developed in this thesis can be implemented on the drone. These system modules are implemented in the Matlab language, and will have to be translated to python or C++ to be compatible with ROS.

Bibliography

Wamit homepage.

BlueRobotics. T200 bluerov2 spare. `https://bluerobotics.com/store/thrusters/t100-t200-thrusters/t200-bluerov2-spare-r1/`, 2019a. Accessed: 01.04.2019.

BlueRobotics. Low-Light HD USB Camera. `https://www.bluerobotics.com/store/sensors-sonars-cameras/cameras/cam-usb-low-light-r1/?fbclid=IwAR2iV_O_uKl058lnTTqI_3R00pQu40oFc8Moyn_1N71cy3rDQV0xNnY9WT0`, April 2019b.

F. Dukan and A. J. Sørensen. Integration filter for aps, dvl, imu and pressure gauge for underwater vehicles. *IFAC Proceedings Volumes*, 46(33):280–285, 2013.

C. E. Brennen (Ed). Internet book on fluid dynamics. `http://brennen.caltech.edu/fluidbook/`, October 2006. Accessed: 26.03.2019.

Flir. Blackfly usb3. `https://www.flir.com/products/blackfly-usb3?model=BFLY-U3-23S6C-C`, 2019. Accessed: 17.04.2019.

T. I. Fossen. A NONLINEAR UNIFIED STATE-SPACE MODEL FOR SHIP MANEUVERING AND CONTROL IN A SEAWAY. *Journal of Bifurcation and Chaos*, 15(09), 9 2005.

T. I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley Sons, Ltd, West Sussex, United Kingdom, 2011.

E. Gat. On three-layer architectures. In D. Kortenkamp, R. P. Bonnasso, , and R. Murphy (Eds), editors, *Artificial Intelligence and Mobile Robots*, pages 195 – 210. MIT Press, 1998.

N. I. Grigorchuk and V. G. Karpov. Light induced nucleation of metallic nanoparticles with frequency controlled shapes. *Applied Physics Letters*, 105:223103, 12 2014. doi: 10.1063/1.4902120.

D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8: 231–274, June 1987. doi: 10.1016/0167-6423(87)90035-9.

A. Havnegjerde. Remote Control and Path Following for the ReVolt Model Ship. Msc thesis, Norwegian University of Science and Technology, 2018.

A. I. Korotkin. *Added Masses of Ship Structures*. 2008.

A. M. Lekkas. *Guidance and Path-Planning Systems for Autonomous Vehicles*. phD thesis, Norwegian University of Science and Technology, 2014.

M. Ludvigsen and A. J. Sørensen. Towards integrated autonomous underwater operations for ocean mapping and monitoring. *Annual Reviews in Control*, 42:145–157, 01 2016.

A. Moreno. Agent architectures. `https://www.slideshare.net/ToniMorenoURV/agent-architectures-3181662`, 2010. Accessed: 08.07.2019.

NORTEK. Dvl1000 technical spesifications, 2016. https://www.nortekgroup.com/products/dvl-1000-300m.

NORTEK. New to subsea navigation? `https://www.nortekgroup.com/insight/nortek-wiki/new-to-subsea-navigation`, 2019. Accessed 31.03.2019.

Vortex NTNU.

Vortex NTNU. Vortex ntnu homepage: about. `www.vortexntnu.no:about`, 2019. Accessed: 14.05.2019.

B. Pettersen. *Kompendium, Marin Teknikk 3 - Hydrodynamikk*. Akademika forlag, Gløshaugen, Trondheim, January 2007.

RoboSub. 22st annual international robosub competition : Mission and scoring. `https://www.robonation.org/sites/default/files/2019%20RoboSub%20Mission%20and%20Scoring_v1.0.pdf`, April 2019. Accessed: 30.04.2019.

J. L. Sanchez-Lopez, R. A. Suarez Fernandez, H. Bavle, C. Sampedro, M. Molina, J. Pestana, and P. Campoy. Aerostack: An architecture and open-source software framework for aerial robotics. pages 332–341, 06 2016.

B. Siciliano and O.Khatib. *Springer Handbook of Robotics*. Springer-Verlag.

R. Skjetne. *The Maneuvering Problem*. phD thesis, Norwegian University of Science and Technology, March 2005.

Roger Skjetne. Notes on: Guidance and maneuvering control design for stationkeeping. unpublished, May 2019a.

Roger Skjetne. Maneuvering control design of a low-speed fully-actuated vessel with stepwise path generation. unpublished, April 2019b.

H. Trentelman. Linear quadratic optimal control. In J. Baillieul and T. Samad, editors, *Encyclopedia of Systems and Control*, pages 644 – 649. Springer, London, 2015.