Design Specification

Remotely Operated Underwater Vehicle

Version 1.0

Author: Marcus Homelius
Date: October 19, 2017

Status

Reviewed	Marcus Homelius	2017-10-19
Approved	Jonas Linder	2017-10-19

Course name:	Automatic Control Project Course	E-mail:	tsrt10rov2017@gmail.com
Project group:	ROV2017	Document responsible:	Marcus Homelius
Course code:	TSRT10	Author's E-mail:	marho949@student.liu.se
Project:	Remotely Operated Underwater Vehicle	Document name:	DesignSpecification.pdf

Project Identity

Group E-mail:	tsrt10rov2017@gmail.com
Homepage:	http://www.isy.liu.se/edu/projekt/tsrt10/2017/rov/
Orderer:	Jonas Linder, Linköping University **Phone:** +46 13 28 28 04 **E-mail:** jonas.linder@liu.se
Customer:	Rikard Hagman, Combine Control Systems AB **Phone:** +46 72 964 70 59 **E-mail:** rikard.hagman@combine.se
Course Responsible:	Daniel Axehill, Linköping University **Phone:** +46 13 28 40 42 **E-mail:** daniel@isy.liu.se
Project Manager:	Amanda Andersson **Phone:** +46 76 843 40 79 **E-mail:** amaan181@student.liu.se
Advisors:	Kristoffer Bergman, Linköping University **Phone:** +46 73 847 31 51 **E-mail:** kristoffer.bergman@liu.se

Group Members

Name	Responsibility	Phone	E-mail (@student.liu.se)
Amanda Andersson	Project manager	+46 76 843 40 79	amaan181
Marcus Homelius	Documentation	+46 70 245 90 96	marho949
George Jajji	Design	+46 70 790 16 17	geoja551
Martin Johannesson	Hardware	+46 76 949 79 69	marjo790
Mattias Mucherie	Information	+46 76 238 53 06	matmu715
Fredrik Nilsson	Software	+46 73 575 49 11	freni169
Anton Nordlöf	Modelling & Simulation	+46 76 145 04 90	antno848
Alaa Saeed	Tests	+46 76 207 57 86	alasa433

Document History

Version	Date	Changes made	Sign	Reviewer(s)
0.1	2017-09-18	First draft.	All	MH
0.2	2017-09-27	First revision.	All	MH, FN, AA
0.3	2017-09-29	Second revision.	All	MH
0.4	2017-10-02	Third revision.	All	MH
0.5	2017-10-08	Fourth revision.	All	MH, GJ
0.6	2017-10-10	Fifth revision.	All	MH
0.7	2017-10-12	Sixth revision.	All	MH, MM
0.8	2017-10-19	Seventh revision.	All	AA
1.0	2017-10-19	First version.	All	AA

Contents

Notations

CG Center of Gravity

EKF Extended Kalman Filter

ESC Electronic Speed Controller

GUI Graphical User Interface

HIL Hardware In the Loop

IMU Inertial Measurement Unit

I/O Input/Output

LCS Local Coordinate System

LQR Linear-Quadratic Regulator

MPC Model Predictive Control

NED North-East-Down

PCS Pool Coordinate System

PF Particle Filter

PWM Pulse Width Modulation

ROV Remotely Operated Underwater Vehicle

ROS Robot Operating System

SCS Sonar Coordinate System

SIL Software In the Loop

1 Introduction

Autonomous vehicles that can carry out missions at sea, in the air and on land without an operator are increasingly sought after in both civilian and military applications. Surveillance, rescue operations, mapping, repairs or tactical missions could be examples of tasks for such vehicles.

This project is a continued development of a remotely operated underwater vehicle (ROV). This document contains a detailed description of the ROV and suggestions of the continued development.

1.1 Purpose

The purpose of this project is to develop autonomous behaviors for the ROV, which include positioning in a known environment and trajectory following. With the new installation of the sensors, the model of the ROV also needs to be updated accordingly.

1.2 Goal

The goal with the project is to develop an autonomous ROV that can follow a given trajectory. Another goal is to develop a robust system to control the ROV and create autonomous behaviors in a pool environment.

The long term goal is to further develop the ROV to be able to 3D-map an unknown environment or search for interesting objects. Therefore it is important that the ROV can easily be further developed after this project ends.

1.3 Use

When the project is finished, the result will be used to further develop the ROV. Ultimately, the goal is to have a fully autonomous vehicle. The ROV will also be used as a test platform for development of control systems for underwater vehicles.

2 System Overview

This section describes the system regarding both hardware and software. The overall system consists of the ROV and a PC, here referred to as workstation, which are connected via an Ethernet cable. A schematic diagram of the system can be seen in Figure 1. Another diagram that shows the flow of information can be seen in Figure 2.

2.1 Hardware Components

The system consists of two main components: the ROV and the workstation to which it is connected. The ROV is composed by multiple components:

- An acrylic chassis.
- An acrylic tube.
- Six electronic speed controllers (ESC) - 30A AfroESC flashed with Blue Robotics linearising firmware.

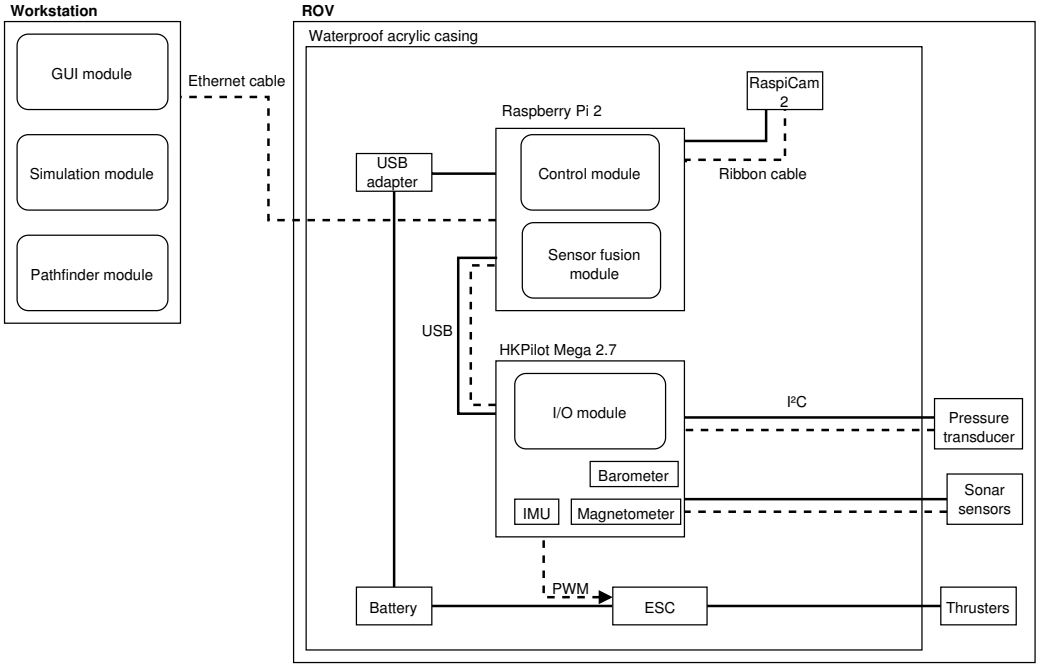

Figure 1: An overview of the system and its subsystems. The sharp-edged boxes represent hardware components while the soft-edged boxes represent software modules. The solid lines represent channels from where the components are powered while the dotted lines represent signal channels.

- Six Blue Robotics T200 thrusters.
- A Raspberry Pi 2 Model B single board computer with a Raspberry Pi camera module v.2.
- A HKPilot Mega 2.7 which is based on Ardupilot and has the following sensors:
  - Magnetometer - HMC5883L.
  - Barometer - MS5611-01BA.
  - Inertial measurement unit (IMU) - MPU6000.
- Three sonar senors - HRXL-MaxSonar- WR-MB7360.
- An external pressure sensor - MS5837-30BA.
- A battery - Turnigy 5000mAh 4S 25C Lipo Pack.
- A HobbyKing LiPo to USB Charging Adapter.

The workstation is a Lenovo T430 running Ubuntu and ROS. The ROV and the workstation are connected through an Ethernet cable.

2.2 Software Components

ROS is used as the main framework for writing robot software because of its ability to communicate with the help of nodes. Nodes are essentially programs that can run

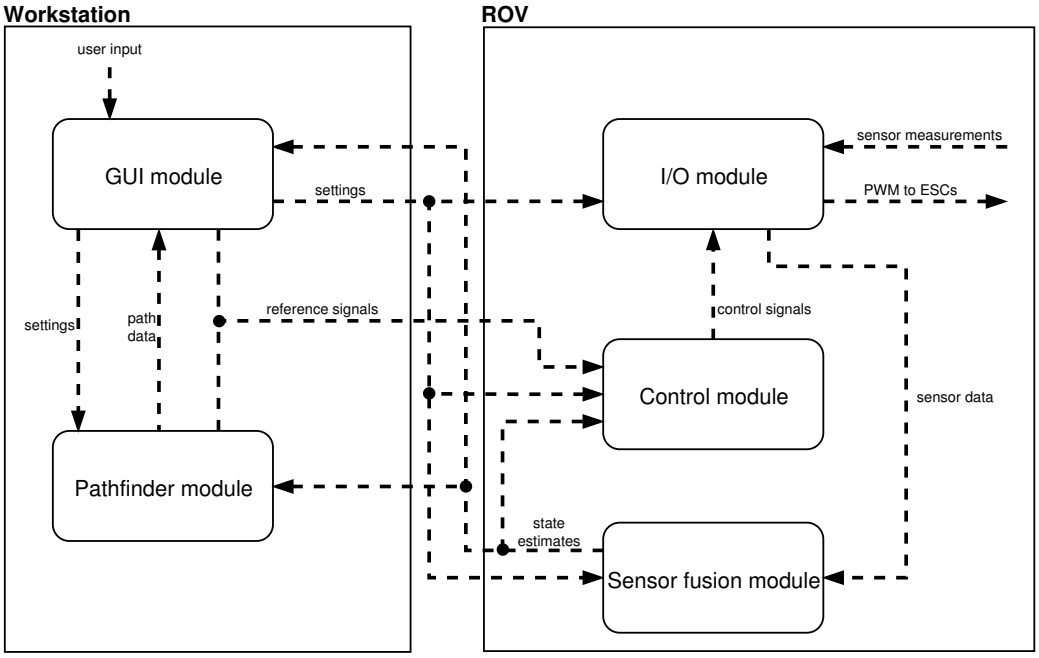

Figure 2: An overview of the system and what kind of data that are transmitted between different modules.

independently from each other. These nodes, which will run on both the ROV and the workstation, can communicate with each other with the help of topics. Topics are channels that nodes can send and receive messages from, by either publish or subscribe to the topic.

The nodes that are currently running on the ROV are listed below.

- **roscore** is a master node. It is used to tell each node where they can find each other. It must run in order for ROS nodes to communicate.
- **raspicam_node** is a camera node for the raspicam. This node is preexisting from previous projects and will not be used this year.
- **bluerov_arduino** is a node which encapsulates the HKPilot where the sensors and the ESCs are connected.
- **controller** is a node that handles the implementation of the controllers for the ROV.

The nodes currently running on the workstation are listed below.

- **heartbeat** is a node that makes sure that there is a connection between the ROV and the HKPilot.
- **teleop_xbox** is a node that handles XBOX-controller input.
- **joy** is a node that handles the OS USB inputs.
- **rqt** is a node that handles the GUI for controlling the ROV.
- **sensorfusion** is a node that handles the sensor fusion.

2.3 System Modules

As can be seen in Figure 1, there are six modules in the system. The control module and the I/O module are located on the ROV itself, while the GUI module and the simulation module are located on the workstation. The sensor fusion module is located at the workstation, but during this project it will be moved to the ROV's on-board computer. This is a step towards the long term goal of making the ROV completely autonomous. The pathfinder module is not preexisting and will be developed on the workstation.

The simulation module is located on the workstation, or compatible computer, and it consists of a simulation environment in MATLAB and Simulink. The main purpose of this module is to reduce the time needed to test the ROV in a water tank. If the module is well integrated, it will ease further test and development. The simulation module will also be used to perform Hardware In the Loop (HIL) and Software In the Loop (SIL). More information about HIL and SIL can be found in Section 8.1 and 8.2, respectively.

The sensor module will be moved to the ROV. This module consists of implementations of filters and state estimators. It takes raw data from the sensors as input and outputs state estimates.

The control module runs on the on-board computer. This module uses the state estimates from the sensor fusion module and the reference module from the user and outputs a control signal. This control signal is sent to the I/O module which sends a corresponding PWM signal to the ESCs.

The GUI module is located on the workstation. It will be used by the operators to change settings on the ROV and set reference signals. The module is based on graphical interactive objects, such as windows, buttons, plots, etc. The module will contain multiple functions, such as graphical representation of the ROV's position and orientation and ability to set reference point's for the pathfinder module.

The I/O module is located on the ROV and mainly consists of the on-board computer and the HKPilot. The on-board computer runs a rosserial node and functions as master while the HKPilot runs a rosserial-arduino node and functions as a sensor reader that sends readings to the master via serial communications. The sensors attached to the I/O module are described in Section 9.

The pathfinder module will be located on the workstation. This module takes a starting position and an ending position from the GUI and calculates the shortest path while taking into account any obstacles that might be in the way.

2.4 Exclusions

The ROV is expected to operate in a swimming pool and it is therefore presumed that the water is calm and and clear. There will not be any random debris floating around which could interfere with the sonar sensors. If the ROV is used in another environment, it is not guaranteed that it will work as intended.

2.5 Design Philosophy

The software should be module based to ease the swapping or removal of a module if necessary. It will also make it easier for the development team to work in parallel.

The software should be well written and well commented. All code associated with the ROS interface should follow the ROS C++ code standard. The other code not associated with the ROS should follow the Google C++ code standard.

3 New Hardware

The hardware is similar to last years project [6]. However, three sonar sensors will be installed to help determine the position of the ROV relative the pool.

3.1 Sonar Sensors

The ROV will be equipped with three sonars and two solution have been considered. One solution is presented in Figure 3. Two sonars can be mounted on the sides of the ROV, pointing outwards, and the third sonar on the front, pointing forward, to receive as much data as possible in the xy-plane. The depth of the ROV in the pool can be determined with the help of the pressure sensor.

The other solution is to mount one sonar underneath the ROV, pointing down, as can be seen in Figure 4. With this setup, a more accurate determination of the depth of the ROV can be acquired. However, because of the introduction of measurements in the z-direction, the readings from the xy-plane will give less information. This can lead to less accurate determination of the ROV's position in the xy-plane.

More information of how the sonar senors are used with sensor fusion can be found in Section 9.

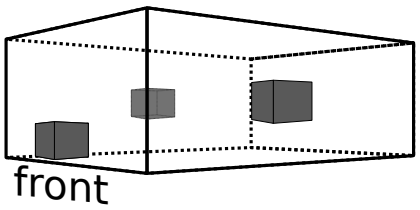

Figure 3: One solution of the placement of the sonars. The view is from the left side. They grey boxes represent the sonar sensors while the transparent box represent the ROV.

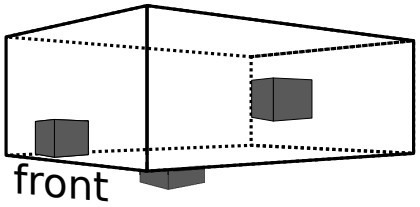

Figure 4: Another solution of the placement of the sonars. The view is from the left side. They grey boxes represent the sonar sensors while the transparent box represent the ROV.

4 Communication

As mentioned in Section 2.2, the communication between the different modules will be done with ROS topics and ROS nodes. Communication between the ROS nodes works by subscribing and publishing messages to different topics. Each node connected to the ROS master have access to the information from the topics which it subscribe to. Therefore, the publisher does not need to know which nodes that will use the available information. Most messages used in last year's project [6] will be continued to be used. The message topics to be used and their description are shown in Tables 1 and 2. Table 2 also shows new topics, that will be created.

4.1 Dynamic Reconfigure

The available sensor fusion node and the control node allows source code variables to be changed while running the program. This is made by a feature from ROS called Dynamic Reconfiguration. The variables marked by dynamic reconfigurable are shown in the GUI and can be changed from the interface during the run-time.

Table 1: The messages and topics for the system (1/2). Long topic name is divided into several rows with '-'.

Message type	Topic	Subscriber	Publisher	Description
std_msgs: Bool	/rovio/enable_thrusters	controller, bluerov_arduino	teleop_xbox	Signals for enabling/disabling the thrusters.
std_msgs: UInt16MultiArray	/rovio/thrusters	bluerov_arduino	controller	Control signals to the thrusters.
std_msgs: Float64MultiArray	/reference	-	controller	The reference signals used by the controller.
std_msgs: Float32MultiArray	/rovio/imu/data	sensorfusion	bluerov_arduino	Accelerometer and gyroscope measurement from the IMU.
std_msgs: Float32	/rovio/water_pressure/data	sensorfusion	bluerov_arduino	Water pressure measurements from the HKPilot.
std_msgs: Float32MultiArray	/rovio/magnetometer/data	sensorfusion	bluerov_arduino	Magnetometer measurements from the HKPilot.
std_msgs: Bool	/sensor_fusion/restart	sensorfusion	-	Command from the user to restart the filter. Sent from a console interface.
std_msgs: Bool	/sensor_fusion/calibrate/mag	sensorfusion	-	Command from the user to calibrate the sensors. Sent from a console interface.
std_msgs: Bool	/rovio/heartbeat	bluerov_arduino	heartbeat_rovio	Sent continuously to that the communication between workstation and ROV works.
std_msgs: Float64MultiArray	/sensor_fusion/states	controller, pathfinder	sensorfusion	Estimated states for the ROV.

Table 2: The messages and topics for the system (2/2). Long topic name is divided into several rows with '-'.

Message type	Topic	Subscriber	Publisher	Description
geometry_msgs: Twist	/cmd-_vel	controller	teleop_xbox	Scaled user input from the XBOX controller.
std_msgs: Float32MultiArray	/controller/params	controller	User (script), setController-Params.sh	Controller parameters.
std_msgs: Float32MultiArray	/controller/cmd_ref	controller	User (script), setReferences.sh and setPath.sh	Topic for reading references published by user via script.
sensor_msgs: Joy	/joy	teleop_xbox	joy_node	Input from XBOX controller.
std_msgs: Bool	/rovio/magneto-meter/calibrate-_offsets	bluerov_arduino	-	Command to calibrate the magnetometer. Sent from a console interface.
std_msgs: Bool	/rovio/gyro/calib-rate_offsets	bluerov_arduino	-	Command to calibrate the gyroscope. Sent from a console interface.
std_msgs: Bool	/rovio/accelerome-ter/calibrate-_offsets	bluerov_arduino	-	Command to calibrate the accelerometer. Sent from a console interface.
std_msgs: Float32	/rovio/water_pre-ssure/sample_time	bluerov_arduino	-	Sets the sample time for the water pressure sensor.
std_msgs: Float32	/rovio/magneto-meter/sample_time	bluerov_arduino	-	Sets the sample time for the magnetometer.
std_msgs: Float32	/rovio/imu/sam-ple_time	bluerov_arduino	-	Sets the sample time for the IMU.
std_msgs: Float32MultiArray	/rovio/sonar/data	sensorfusion	bluerov_arduino	Measurements from the sonar sensors.
std_msgs: Float64MultiArray	/pathfinder/goal-_points	pathfinder	GUI	The goal position in the known environment sent from the GUI.
std_msgs: Float64MultiArray	/pathfinder/path	controller, GUI	pathfinder	The estimated trajectory from the A* algorithm.

5 Coordinate Systems

Two coordinate systems will be used, Local- and Pool-coordinate systems. The coordinate systems are described below. Figure 5 illustrates how the coordinate systems can be located in the pool. Both systems are right-handed coordinate system. There is also other coordinate systems, the sonar coordinate systems, which are described in Section 5.3.

5.1 Local Coordinate System

The Local Coordinate System (LCS) is fixed in the ROV, with its origin assumed to be located in the ROV's center of gravity (CG). The x-axis points forward and the y-axis points to starboard. Perpendicular to the x- and y-axis is the z-axis, that then will point down provided that the ROV is placed horizontal. The sensors will always rotate with

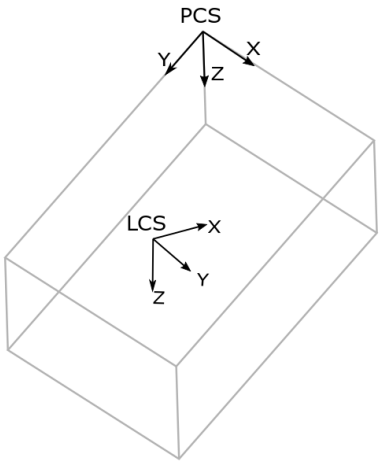

Figure 5: An illustration of the pool- and local coordinate systems in the pool.

this coordinate system.

5.2 Pool Coordinate System

The Pool Coordinate System (PCS), is an initial coordinate system with origin in one upper corner in the pool and the x-y-axes along the sides of the pool. This will also be a corner of the known environment. The xy-plane aligns with the water surface and perpendicular to it, the z-axis points downwards. This coordinate system will be used to set the reference trajectory for the ROV and measure its position.

5.3 Sonar Coordinate System

The Sonar Coordinate Systems (SCS), are coordinate systems introduced since the sonars will not be mounted exactly in the axes of the LCS. When the positions of the sonars have been decided, the transformation from each of the three SCS to LCS is to be calculated.

6 Modelling

A dynamic model is used to describe how the ROV is affected by external forces. The model is derived by physical laws but since the reality is complex some assumptions are made to simplify the model. With a simplified model, parameters can be estimated by taking measurements while conducting tests on the real system.

Since the model is not changed from earlier projects, the following sections contain many assumptions and equations made by the previous master's thesis done on the ROV in 2016. Some of the theory has been left out but can be read in [1].

The submersible with six degrees of freedom is described by

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}_{\boldsymbol{\Theta}}(\boldsymbol{\eta})\boldsymbol{\nu}, \tag{1a}$$

$$\boldsymbol{\tau} = \boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{g}(\boldsymbol{\eta}). \tag{1b}$$

The generalized forces acting on the ROV caused by the thrusters and the environmental disturbances are described by the vector $\boldsymbol{\tau}$. The vector $\boldsymbol{\eta}$ are the generalized positions and $\boldsymbol{\nu}$ is the generalized velocities used to describe the motion of the ROV. The matrices $\boldsymbol{M}$, $\boldsymbol{C}(\boldsymbol{\nu})$, $\boldsymbol{D}(\boldsymbol{\nu})$ and the vector $\boldsymbol{g}(\boldsymbol{\eta})$, describe mass and moment of inertia, Coriolis effect, damping forces, gravity and buoyancy effect on the ROV, respectively. The notations used to describe the parameters are summarized in the Table 3.

6.1 Notations

The notations used to describe an underwater vehicle with six degrees of freedom is

$$\begin{aligned} \boldsymbol{\eta} &\triangleq \begin{bmatrix} x & y & z & \phi & \theta & \psi \end{bmatrix}^T, \\ \boldsymbol{\nu} &\triangleq \begin{bmatrix} u & v & w & p & q & r \end{bmatrix}^T. \end{aligned} \tag{2}$$

The vector $\boldsymbol{\eta}$ describes the position and attitude in PCS and $\boldsymbol{\nu}$ is a vector that contain linear and angular velocities along and about the LCS. The naming convention of (2), is

$$\boldsymbol{\eta}_1 \triangleq \boldsymbol{p} \triangleq \begin{bmatrix} x & y & z \end{bmatrix}^T, \tag{3a}$$

$$\boldsymbol{\eta}_2 \triangleq \boldsymbol{\Theta} \triangleq \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T, \tag{3b}$$

$$\boldsymbol{\nu_1} \triangleq \boldsymbol{v} \triangleq \begin{bmatrix} u & v & w \end{bmatrix}^T, \tag{3c}$$

$$\boldsymbol{\nu_2} \triangleq \boldsymbol{\omega} \triangleq \begin{bmatrix} p & q & r \end{bmatrix}^T. \tag{3d}$$

Note that $\boldsymbol{\eta}$ are states in the PCS, while $\boldsymbol{\nu}$ are states in the LCS.

Table 3: The parameters used in the ROV model.

Parameter	**Value/Source**	**Description**
m	measured [kg]	The total mass of the ROV (without sonars).
g	9.82 [m/s^2]	Gravitational constant.
ρ	1000 [kg/m^3]	Density of water.
l_{x1}	0.19 [m]	Distance from thruster 1 to CG in x-direction.
l_{y1}	0.11	Distance from thruster 1 to CG in y-direction.
l_{x2}	0.19	Distance from thruster 2 to CG in x-direction.
l_{y2}	0.11	Distance from thruster 2 to CG in y-direction.
l_{y3}	0.11	Distance from thruster 3 to CG in y-direction.
l_{y4}	0.11	Distance from thruster 4 to CG in y-direction.
l_{x5}	0.11	Distance from thruster 5 to CG in x-direction.
l_{z6}	0.17	Distance from thruster 6 to CG in z-direction.
Z_b	-0.0420 [M]	Distance from CG to CB.
V	Measured [m^3]	Displaced volume.
B	Computed [N]	The ROV's buoyancy (B = $\rho g V$).
W	Computed [N]	Force of gravity ($W = mg$).
X_u	Estimated [kg/s]	Linear damping coefficient due to translation in the x-direction.
$X_{u\|u\|}$	Estimated [kg/m]	Quadratic damping coefficient due to translation in the x-direction.
Y_v	Estimated [kg/s]	Linear damping coefficient due to translation in the y-direction.
$Y_{v\|v}$	Estimated [kg/m]	Quadratic damping coefficient due to translation in the y-direction.
Z_w	Estimated [kg/s]	Linear damping coefficient due to translation in the z-direction.
$Z_{w\|w\|}$	Estimated [kg/m]	Quadratic damping coefficient due to translation in the z-direction.
K_p	Estimated [kg m^2/s]	Linear damping coefficient due to rotation in water about the x-axis.
$K_{p\|p}$	Estimated [kg m^2]	Quadratic damping coefficient due to rotation in water about the x-axis
M_q	Estimated [kg m^2/s]	Linear damping coefficient due to rotation in water about the y-axis.
$M_{q\|q\|}$	Estimated [kg m^2]	Quadratic damping coefficient due to rotation in water about the y-axis.
N_r	Estimated [kg m^2/s]	Linear damping coefficient due to rotation in water about the z-axis.
$Nr\|r\|$	Estimated [kg m^2]	Quadratic damping coefficient due to rotation in water about the z-axis.
A_p	Estimated [kg m^2]	Inertia and increased inertia around the x-axis.
B_q	Estimated [kg m^2]	Inertia and increased inertia around the y-axis.
C_r	Estimated [kg m^2]	Inertia and increased inertia around the z-axis.
D_u	Estimated [kg]	Mass and mass added due to translation in the x-direction.
E_v	Estimated [kg]	Mass and mass added due to translation in the y-direction.
F_w	Estimated [kg]	Mass and mass added due to translation in the z-direction.

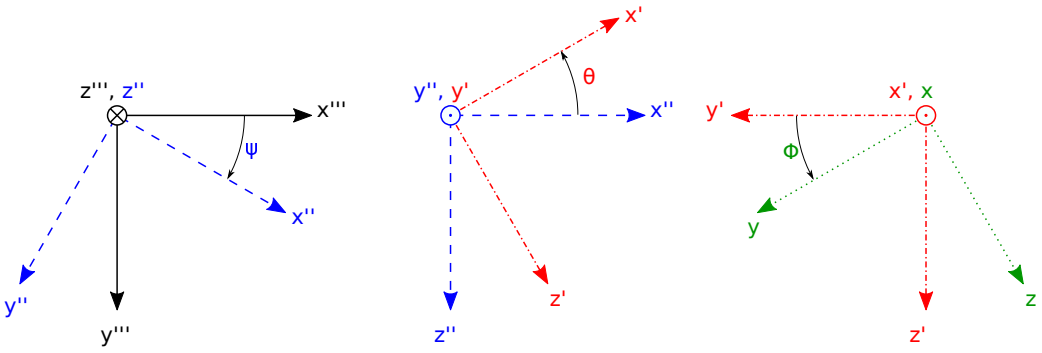

Figure 6: Illustration of transformation of Euler angles from LCS to PCS.

6.2 Euler Angles

The orientation of the ROV is described with Euler angles. The ROV's rotations in yaw, pitch and roll are represented by the angles ψ, θ and ϕ, respectively. An illustration of the rotations is shown in Figure 6. The rotations are done in a predefined order according to the zyx-convention.

With a rotation matrix, one can rotate a vector from one coordinate system into another. With the given rotation convention, the rotation matrix $\boldsymbol{R}_{\mathrm{L}}^{\mathrm{P}}(\boldsymbol{\Theta})$, i.e. rotation from LCS to PCS, is given by

$$\boldsymbol{R}_{\mathrm{L}}^{\mathrm{P}}(\boldsymbol{\Theta}) = \begin{bmatrix} C_\theta C_\psi & -C_\theta S_\psi + S_\phi S_\theta C_\psi & -S_\theta S_\psi + C_\phi S_\theta C_\psi \\ C_\theta S_\psi & C_\theta C_\psi + S_\phi S_\theta S_\psi & -S_\theta C_\psi + C_\phi S_\theta S_\psi \\ -S_\theta & S_\phi C_\theta & C_\phi C_\theta \end{bmatrix}, \tag{4}$$

where $C_{(\cdot)}$ and $S_{(\cdot)}$ are shorthand notations for $\cos(\cdot)$ and $\sin(\cdot)$, respectively. It also has the property of being orthogonal, i.e. $\boldsymbol{R}_{\mathrm{L}}^{\mathrm{P}}(\boldsymbol{\Theta})^T = \boldsymbol{R}_{\mathrm{L}}^{\mathrm{P}}(\boldsymbol{\Theta})^{-1}$. This also means that $\boldsymbol{R}_{\mathrm{P}}^{\mathrm{L}}(\boldsymbol{\Theta}) = \boldsymbol{R}_{\mathrm{L}}^{\mathrm{P}}(\boldsymbol{\Theta})^T$. For notational convenience, we introduce the notation

$$\boldsymbol{R}(\boldsymbol{\Theta}) \triangleq \boldsymbol{R}_{\mathrm{L}}^{\mathrm{P}}(\boldsymbol{\Theta}). \tag{5}$$

The transformation from angular velocity to Euler angles is

$$\dot{\boldsymbol{\Theta}} = \boldsymbol{T}_{\boldsymbol{\Theta}}(\boldsymbol{\Theta})\boldsymbol{\omega}, \tag{6}$$

where

$$\boldsymbol{T}_{\boldsymbol{\Theta}}(\boldsymbol{\Theta}) = \begin{bmatrix} 1 & S_\phi T_\theta & C_\phi T_\theta \\ 0 & C_\phi & -S_\psi \\ 0 & S_\phi / C_\theta & C_\phi / C_\theta \end{bmatrix}, \tag{7}$$

and $T_{(\cdot)}$ is shorthand notations for $\tan(\cdot)$.

6.3 Dynamics

The dynamics of the ROV can be divided into kinematics and kinetics. Kinematics describes the motion of a body due to geometrical aspects. Kinetics describes the motion due to external forces.

6.3.1 Kinematics

The kinematic state equations can be expressed as

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}_{\boldsymbol{\Theta}}(\boldsymbol{\eta})\boldsymbol{\nu} \Longleftrightarrow \begin{bmatrix} \dot{\boldsymbol{p}} \\ \dot{\boldsymbol{\Theta}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}(\boldsymbol{\Theta}) & \boldsymbol{0}_{3\times 3} \\ \boldsymbol{0}_{3\times 3} & \boldsymbol{T}_{\boldsymbol{\Theta}}(\boldsymbol{\Theta}) \end{bmatrix} \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{\omega} \end{bmatrix}, \tag{8}$$

see Section 6.2 for more details.

6.3.2 Rigid-Body Kinetics

The Euler angles formulation is used to express the rigid-body kinetics of the ROV. The kinetics can be modelled as

$$\boldsymbol{\tau}_{RB} = \boldsymbol{M}_{RB}\dot{\boldsymbol{\nu}} + \boldsymbol{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu}, \tag{9}$$

were the forces and moments acting on the rigid-body are collected in $\boldsymbol{\tau}_{RB}$. The matrix $\boldsymbol{M}_{RB}$ is the inertia matrix of the rigid-body and $\boldsymbol{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu}$ is a vector that describe the centripetal and Coriolis effect. By assuming that the ROV's center of origin and its center of gravity coincides, the matrix describing inertia can be simplified to

$$\boldsymbol{M}_{RB} = \begin{bmatrix} m\boldsymbol{I}_{3\times 3} & \boldsymbol{0}_{3\times 3} \\ \boldsymbol{0}_{3\times 3} & \boldsymbol{I}_g \end{bmatrix}, \tag{10}$$

were $\boldsymbol{I}_g$ is the matrix describing the inertia about the CG of the ROV. Another simplification is made with the Coriolis forces; here the ROV is assumed to have three planes of symmetry in order to eliminate some cross terms and gives the vector

$$\boldsymbol{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu} = \begin{bmatrix} m(qw - rv) \\ m(ru - pw) \\ m(pv - qu) \\ qr(I_y - I_z) \\ rp(I_z - I_x) \\ qp(I_x - I_y) \end{bmatrix}, \tag{11}$$

which describes the Coriolis and centripetal forces to the submersible caused by its mass.

6.4 Hydrodynamics

When the ROV is operating in water, it is exposed to hydrodynamic forces. These external forces can be modelled as

$$\boldsymbol{\tau}_{\text{Dyn}} = -\boldsymbol{M}_A\dot{\boldsymbol{\nu}} - \boldsymbol{C}_A(\boldsymbol{\nu})\boldsymbol{\nu} - \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu}, \tag{12}$$

where $\boldsymbol{M}_A$ contains the model for added mass and moment of inertia, $\boldsymbol{C}_A$ contains the model for Coriolis and centripetal effects and the $\boldsymbol{D}$ matrix contains the linear and quadratic dampening effects. The added mass and moment of inertia is defined as

$$\boldsymbol{M}_A = -\begin{bmatrix} X_{\dot{u}} & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{\dot{v}} & 0 & 0 & 0 & 0 \\ 0 & 0 & Z_{\dot{w}} & 0 & 0 & 0 \\ 0 & 0 & 0 & K_{\dot{p}} & 0 & 0 \\ 0 & 0 & 0 & 0 & M_{\dot{q}} & 0 \\ 0 & 0 & 0 & 0 & 0 & N_{\dot{r}} \end{bmatrix} \tag{13}$$

under the assumption that the ROV is moving slowly in the water. The matrix containing Coriolis and centripetal effects from the added mass and moment of inertia multiplied with $\boldsymbol{\nu}$ equals to the vector

$$\boldsymbol{C}_A(\boldsymbol{\nu})\boldsymbol{\nu} = \begin{bmatrix} Y_{\dot{v}}vr - Z_{\dot{w}}wq \\ Z_{\dot{w}}p - X_{\dot{u}}ur \\ X_{\dot{u}}uq - Y_{\dot{v}}vp \\ (Y_{\dot{v}} - Z_{\dot{w}})vw + (M_{\dot{q}} - N_{\dot{r}})qr \\ (Z_{\dot{w}} - X_{\dot{u}})uw + (N_{\dot{r}} - K_{\dot{p}})pr \\ (X_{\dot{u}} - Y_{\dot{v}})uv + (K_{\dot{p}} - M_{\dot{q}})pq \end{bmatrix}, \tag{14}$$

assuming that the ROV has three planes of symmetry and is moving slowly in the water. The hydrodynamic dampening that are acting on the submersible are skin friction, potential and vortex shedding dampening. The dampening is modelled by the vector

$$\boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} = -\begin{bmatrix} (X_u + X_{u|u|}|u|)u \\ (Y_v + Y_{v|v|}|v|)v \\ (Z_w + Z_{w|w|}|w|)w \\ (K_p + K_{p|p|}|p|)p \\ (M_q + M_{q|q|}|q|)q \\ (N_r + N_{r|r|}|r|)r \end{bmatrix}, \tag{15}$$

assuming that the dampening is decoupled and that the ROV is symmetrical in the xz-plane.

6.5 Hydrostatics

The forces and moments caused by the earths gravitational pull and the ROV's buoyancy force in water are modelled as

$$\boldsymbol{\tau}_{\text{Stat}} = -\boldsymbol{g}(\boldsymbol{\eta}), \tag{16}$$

The magnitude of the buoyancy forces are equal to the weight of the displaced water. For the fully submerged ROV, that volume will be equal to the submersibles total volume. Those effects are described in the vector

$$\boldsymbol{g}(\boldsymbol{\eta}) = \begin{bmatrix} (W - B)\sin\theta \\ -(W - B)\cos\theta\sin\phi \\ -(W - B)\cos\theta\cos\phi \\ -z_B B\cos\theta\sin\phi \\ -z_B B\sin\theta \\ 0 \end{bmatrix}, \tag{17}$$

where z_B is the distance from the center of gravity to the center of buoyancy, B = $\rho g V$ and W = mg. If $B = W$ the ROV has a neutral buoyancy. To have a roll and pitch stable submersible it is important that $z_B < 0$.

6.6 Actuators

The six thrusters generate forces and torques along and about the LCS. These forces can be modelled as the vector $\boldsymbol{\tau}_{Act}$. The matrix $\boldsymbol{T}$ describes the placement of the actuators and $\boldsymbol{f}(\boldsymbol{u})$ is a look-up table from control signal to force in Newton. The relation is

$$\boldsymbol{\tau}_{Act} \triangleq \boldsymbol{T}\boldsymbol{f}(\boldsymbol{u}), \tag{18}$$

where

$$\boldsymbol{\tau}_{Act} = \begin{bmatrix} \tau_u \\ \tau_v \\ \tau_w \\ \tau_p \\ \tau_q \\ \tau_r \end{bmatrix}, \quad \boldsymbol{T} = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ -1 & -1 & 0 & 0 & -1 & 0 \\ l_{y1} & -l_{y2} & 0 & 0 & 0 & l_{z6} \\ l_{x1} & l_{x2} & 0 & 0 & -l_{x5} & 0 \\ 0 & 0 & l_{y3} & -l_{y4} & 0 & 0 \end{bmatrix}$$

and

$$\boldsymbol{f}(\boldsymbol{u}) = \begin{bmatrix} f(u_1) \\ f(u_2) \\ f(u_3) \\ f(u_4) \\ f(u_5) \\ f(u_6) \end{bmatrix}.$$

6.7 Model in Component Form

The expressions for the accelerations of the ROV all six degrees of freedom, $\dot{\boldsymbol{\nu}}$, can be split into the following components

$$\dot{u} = \frac{1}{m - X_{\dot{u}}}\Big(\tau_u + (X_u + X_{u|u|}|u|)u + (B - W)\sin\theta + m(rv - qw) - Y_{\dot{v}}rv + Z_{\dot{w}}qw\Big), \tag{19a}$$

$$\dot{v} = \frac{1}{m - Y_{\dot{v}}}\Big(- \tau_v + (Y_v + Y_{v|v|}|v|)v - (B - W)\cos\theta\sin\phi + m(pw - ru) + X_{\dot{u}}ru - Z_{\dot{w}}pw\Big), \tag{19b}$$

$$\dot{w} = \frac{1}{m - Z_{\dot{w}}}\Big(\tau_w + (Z_w + Z_{w|w}|w|)w - (B - W)\cos\theta\cos\phi + m(qu - pv) - X_{\dot{u}}qu + Y_{\dot{v}}pv\Big), \tag{19c}$$

$$\dot{p} = \frac{1}{I_x - K_{\dot{p}}}\Big(\tau_p + (K_p + K_{p|p|}|p|)p + Bz_B\cos\theta\sin\phi + qr(I_y - I_z + N_{\dot{r}} - M_{\dot{p}}) + vw(Z_{\dot{w}} - Y_{\dot{v}})\Big), \tag{19d}$$

$$\dot{q} = \frac{1}{I_y - M_{\dot{q}}}\Big(\quad \tau_q + (M_q + M_{q|q|}|q|)q + Bz_B\sin\theta + pr(K_{\dot{p}} - N_{\dot{r}} + I_z - I_x) - Z_{\dot{w}}uw + X_{\dot{u}}uw\Big), \tag{19e}$$

$$\dot{r} = \frac{1}{I_z - N_{\dot{r}}}\Big(\tau_r + (N_r + N_{r|r|}|r|)r + pq(M_{\dot{q}} - K_{\dot{p}} + I_x - I_y) + uv(Y_{\dot{v}} - X_{\dot{u}})\Big), \tag{19f}$$

6.8 Simplified Model Structures

Estimation of the parameters relating to the cross-terms in the model (19) is difficult due to the sensor setup. These cross terms will be negligible since the ROV will operate at low speed and the data gathering experiments will be designed to minimize them. The

tests will be conducted so that only excitation is done in one degree of freedom at a time. With simplified denominators, $A_p = I_x - K_{\dot{p}}$, $B_q = I_y - M_{\dot{q}}$, $C_r = I_z - N_{\dot{r}}$, $D_u = m - X_{\dot{u}}$, $E_v = m - Y_{\dot{v}}$, and $F_w = m - Z_{\dot{w}}$, the simplified model can be described as

$$\dot{u} = \frac{1}{D_u}\Big(\tau_u + (X_u + X_{u|u|}|u|)u\Big), \tag{20a}$$

$$\dot{v} = \frac{1}{E_v}\Big(\tau_v + (Y_v + Y_{v|v|}|v|)v\Big), \tag{20b}$$

$$\dot{w} = \frac{1}{F_w}\Big(\tau_w + (Z_w + Z_{w|w|}|w|)w - (B - W)\Big), \tag{20c}$$

$$\dot{p} = \frac{1}{A_p}\Big(\tau_p + (K_p + K_{p|p|}|p|)p + Bz_B \sin\phi\Big), \tag{20d}$$

$$\dot{q} = \frac{1}{B_q}\Big(\tau_q + (M_q + M_{q|q|}|q|)q + Bz_B \sin\theta\Big), \tag{20e}$$

$$\dot{r} = \frac{1}{C_r}\Big(\tau_r + (N_r + N_{r|r|}|r|)r\Big). \tag{20f}$$

7 Parameter Estimation

Since the system's dynamics are not known to 100 %, white box modeling is difficult and some parameter estimation is needed to complete the model for the ROV. Some of the parameters, such as the mass of the ROV and its displaced volume can be measured but other parameters needs to be estimated using experiments. One alternative is to estimate them using a gray-box model.

In gray-box modeling, partial theoretical structure is used combined with data gathering to estimate the unknown parameters. If, for example, the model is described by $y = Au^3 + \sqrt{u - B}$ were u is the input signal and y is the output signal, the gray-box model will try to estimate the unknown parameters A and B as well as possible.

One drawback with gray-box modeling is that the solver might get stuck in a local minimum and not progress to the true solution. This can be avoided by setting good initial guesses of the parameters and by constraining the parameters as much as possible, which minimizes the number of local minimums. The full list of parameters that need to be estimated are represented in Table 3.

7.1 Data Collection

To be able to estimate the parameters under the assumptions made in the simplified models (20), data points with relevant data needs to be collected. The simplified model requires that the tests are done with excitation only in one DOF at the time.

Telegraph signals will be used as the main control signals to excite the system. It is important that the telegraph signal is implemented with a low enough switch probability that a majority of the signals are within the systems bandwidth. Earlier test made by the 2016 project, found that the true physical system was slower than initially thought and their tests were to high in frequency to give realistic valued parameters.

With the implementation of the sonar system to the ROV, actual measurements can be made to estimate the linear velocities in two dimensions. Sonar data is advantageous to integration of the accelerometer readings, which has been shown by earlier projects to give erroneous readings when speed was kept constant.

7.2 Estimation of Simplified Model Structure

Using the decoupled models shown in (20), it is possible to limit the model to only one degree of freedom. The telegraph signals used to only excite one DOF at a time is

$$\boldsymbol{\tau}_{Act} = \begin{bmatrix} \lambda & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

for velocity excitation in the LCS x-direction,

$$\boldsymbol{\tau}_{Act} = \begin{bmatrix} 0 & \lambda & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

for velocity excitation in the LCS y-direction etc. Here, λ is the chosen amplitude of the telegraph signal. It should be changed binary, with low enough switch probability in order to excite the system within its bandwidth.

When estimating the model parameters it is important to use the direct output of the ROV's sensor measurements and not the estimated states. This is because the sensor fusion model could insert extra dynamic which would effect the estimates of the parameters. To avoid this problem, the modelling will be done without the estimated states from the sensor fusion node. By not using the estimated states from the sensor fusion module, it will not interfere with the estimated model, effectively eliminating one source of potential errors.

8 Simulation

Since all real testing that is made on the ROV includes having access to a pool and the necessary preparations for a dive, the project will benefit from a simulation tool. Having a working simulation tool will speed up the development of new software for the ROV without having to dive more than necessary. The simulator will be built in MATLAB/Simulink and include an extension with a node in ROS. This enables the simulator to receive and send data directly to the nodes on the ROV. The simulated states will be shown in a real time GUI where a 3D model of the ROV will be displayed.

8.1 Hardware In the Loop

With hardware in the loop, the simulation uses the existing sensor fusion and controllers implemented on the ROV. The simulated data is fed to the sensor fusion node in ROS which calculates the ROV's current states which are then fed to the onboard controllers. The controllers then produces control signals to the simulation environment. The simulation updates the outgoing sensor data to the sensor fusion node. In Figure 7, the information flow of HIL is shown. This is an effective way of developing the controllers and testing the sensor fusion since there is no need to create a virtual replica of the already implemented systems on the ROV. Due to the sensor data being simulated and the control signals are sent to the simulation environment, the HKPilot doesn't need to be powered up. Simulations can then be executed without the risk of damaging the actuators of the ROV. Since the HKPilot is not powered up, the heartbeat node of the HKPilot need to be implemented in the simulation.

8.2 Software In the Loop

With software in the loop it is possible to implement some modules in MATLAB/Simulink and run the sensor fusion and control module outside of MATLAB/Simulink. The sensor fusion module and the control module will have the same implementation as when they are running on the ROV, but will be running on another Linux computer instead. This is shown in Figure 8. The benefit of SIL is that the software that will be used in the

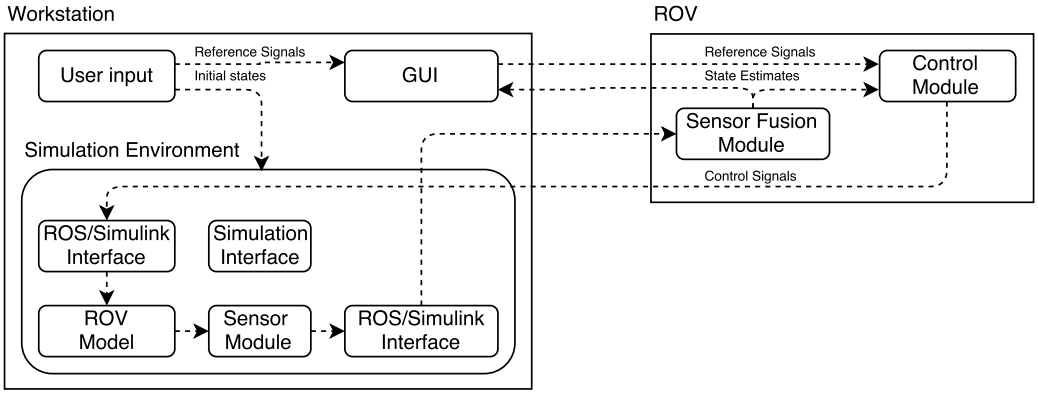

Figure 7: Structure of HIL simulations. Arrows represent information flow. Blocks with sharp corners represent hardware. Blocks with rounded corners represent software.

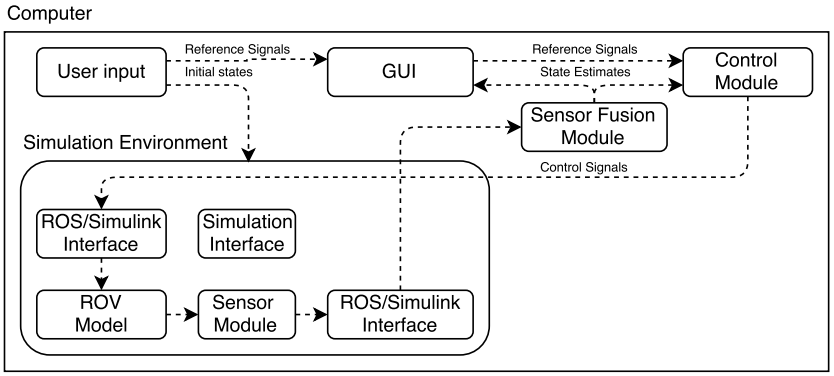

Figure 8: Structure of SIL implemented both in MATLAB/Simulink and on a computer. Arrows represent information flow. Blocks with rounded corners represent software.

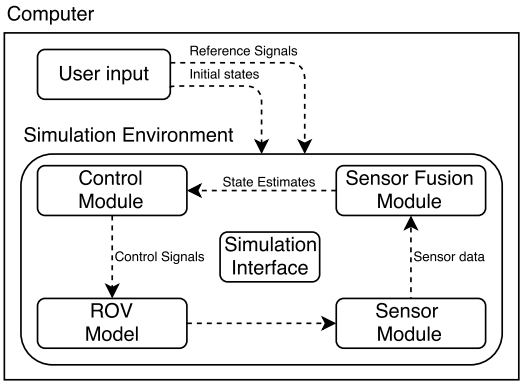

Figure 9: Structure of SIL implemented in MATLAB/Simulink. Arrows represent information flow. Blocks with rounded corners represent software.

ROV is a part of the simulation. The sensor fusion module and the control module will communicate to MATLAB/Simulink with ROS.

8.3 Pure Software Simulation

With pure software simulation, the modules are directly implemented in MATLAB/Simulink. Pure software simulation can be done before the communication with ROS is implemented and without having the ROV avaliable. The pure software simulation will facilitate parallel development of the control module and the sensor fusion module, since one of the modules does not need that the other one is fully functional. The modules can be implemented in MATLAB/Simulink, as shown in Figure 9.

8.4 Sensor Module

The sensor module in the simulator produces the measurements that the sensors would have measured, given a certain state of the ROV. The sensor module takes the simulated states from the model of the ROV as input to produce the expected measurements. The measurement equations defined in Section 9.4 will be implemented in the sensor module.

Some additive noise will be added to the output as well.

The sonar sensors need information of the pool environment in order to simulate measurements. The measurement equation for the sonars can be found in Section 9.4.4. For simplicity, the environment will be modeled so the ROV cannot see the bottom or the surface of the pool. In practice, this means that the geometry model is simplified and the distance given by the sensor module will approach infinity as the sensor is turned towards the bottom or the surface.

8.5 Pool Environment

The pool environment is known and will be represented by a 2D grid map using ROS packages and libraries. The resolution of the grid is a design parameter that can be adjusted. The modules that demands knowledge about the environment are the sensor fusion module, the GUI module and the pathfinder module. The sensor fusion module must know the environment in order to make position estimation and the pathfinder module must know the environment in order to calculate a trajectory route. The GUI module uses the the pool environment to visualize it to the user.

8.6 Simulation of Sensor Fusion System

The sensor fusion system can either be implemented directly in MATLAB/Simulink, or be located externally as a ROS node. The purpose of having a simple sensor fusion implementation directly in MATLAB/Simulink is that the simulation can be used immediately. This would facilitate parallel development of, for example, the control module and the sensor fusion module.

8.7 Simulation of Control System

As mention in Section 8.6, a simple control implementation directly in Simulink would facilitate parallel development. The idea of simulating the control system will be the same as for the sensor fusion system. The control system can either be implemented directly in Simulink, or with ROS to an external program or device.

8.8 Simulation Interface

The simulation interface in MATLAB/Simulink enables the user to interact with the models that are implemented in the simulation environment. In the simulation interface it should be possible to change parameters in the models. It will provide the user an opportunity to display and compare data during simulation and after simulation with stored data. The interface will also provide a 3D visualization of ROV and the states.

9 Sensor Fusion

This section will describe how the sensor fusion module will work on the ROV. A new requirement for this year's design is to move the sensor fusion module from the workstation to the Raspberry Pi. This is a step into approaching the long term goal of the project to achieve a fully autonomous operating vehicle.

The main purpose of the sensor fusion module is to estimate the state of the ROV, including position, attitude, linear- and angular velocities. This is done through signal processing of the measured signals from the different sensors mounted on the ROV.

9.1 Filter Choice

Since the ROV shall be able to position itself in a know environment, there are several possible solutions. Last year's project [6], used two extended Kalman filters, one *position filter* for estimation of the position and linear velocities and one *attitude filter* for estimation of attitude, angular velocities, bias for angular velocities and depth. The argument for using two filters instead of one, was that the states in each filter was separated and it required less computations to update two smaller filters. This year, new sensors for positioning the ROV will be mounted. Therefore a discussion is necessary about how these new measurements shall be incorporated into the current sensor fusion model.

9.1.1 Extended Kalman Filter

A Kalman filter uses models of the system and the measurements in order to estimate the states. If there are nonlinear models, the extended Kalman filter (EKF) is a way of approaching the problem. The models in (21) describe the time discrete motion model and measurement model,

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{v}_k \\ \boldsymbol{y}_k &= \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{e}_k \end{aligned} \tag{21}$$

where $\boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k)$ is the time discrete motion model, $\boldsymbol{v}_k$ is the process noise with the Gaussian distribution $v_k \sim N(0, \boldsymbol{Q}_k)$, $\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{u}_k)$ is the measurement model and e_k is the measurement noise with the Gaussian distribution $\boldsymbol{e}_k \sim N(0, \boldsymbol{R}_k)$.

The Algorithm to perform with an EKF is found in [3] and listed as Algorithm 1. It is divided into two steps: a measurement update and a time update.

Algorithm 1 Extended Kalman filter

Initialize the filter with $\hat{\boldsymbol{x}}_{1|0} = \boldsymbol{x}_0$ and $\boldsymbol{P}_{1|0} = \boldsymbol{P}_0$.

1. *Measurement update:*

$$\text{Innovation covariance: } \boldsymbol{S}_k = \boldsymbol{R}_k + \boldsymbol{H}_k^T \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k$$
$$\text{Kalman gain: } \boldsymbol{K}_k = \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k^T \boldsymbol{S}_k^{-1}$$
$$\text{Innovation: } \boldsymbol{\epsilon}_k = \boldsymbol{y}_k - \boldsymbol{h}(\hat{\boldsymbol{x}}_{k|k-1})$$

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k \boldsymbol{\epsilon}_k \tag{22a}$$

$$\boldsymbol{P}_{k|k} = \boldsymbol{P}_{k|k-1} - \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k^T \boldsymbol{S}_k^{-1} \boldsymbol{H}_k \tag{22b}$$

2. *Time update:*

$$\hat{\boldsymbol{x}}_{k+1|k} = \boldsymbol{f}(\hat{\boldsymbol{x}}_{k|k}) \tag{22c}$$

$$\boldsymbol{P}_{k+1|k} = \boldsymbol{Q}_k + \boldsymbol{F}_k \boldsymbol{P}_{k|k} \boldsymbol{F}_k^T \tag{22d}$$

Here, $\boldsymbol{H}_k = \boldsymbol{h}'(\hat{\boldsymbol{x}}_{k|k-1})$ and $\boldsymbol{F}_k = \boldsymbol{f}'(\hat{\boldsymbol{x}}_{k|k-1})$.

One drawback of using an EKF is the assumption of the noise distribution. The filter assumes Gaussian noise and if it is not, some problems may arise. Although, it is generally a good assumption to use a Gaussian distribution for the noise. So it might more be seen as a limitation rather than a drawback.

9.1.2 Particle Filter

The particle filter (PF) estimates the posterior distribution, $p(x_k|y_{1:k})$, where x_k is the current state and $y_{1:k}$ is the measurements $\{y_1, y_2, ..., y_k\}$ up to time k. One advantage with a particle filter is the ability to handle other kind of noise distributions. The cost of this, is often complex computations in a particle filter if, for instance, many particles are used. Algorithm 2 from [3] describes in general how to perform these steps.

Algorithm 2 Particle filter

Choose the number of particles, N, and initialize with $x_1^i \sim p_{x_0}$, $i = 1, 2, ..., N$, and $w_{1|0}^i = 1/N$.

1. *Measurement update:* For $i = 1, 2, ..., N$, calculate the weights

$$w_{k|k}^i = \frac{1}{c_k} w_{k|k-1}^i p(y_k|x_k^i), \tag{23a}$$

where c_k is a normalization constant given by

$$c_k = \sum_{i=1}^{N} w_{k|k-1}^i p(y_k|x_k^i).$$

2. *Estimation:* The filtering density is

$$\hat{p}(x_{1:k}|y_{1:k}) = \sum_{i=1}^{N} w_{k|k}^i \delta(x_{1:k} - x_{1:k}^i), \tag{23b}$$

and

$$\hat{x}_{1:k} \approx \sum_{i=1}^{N} w_{k|k}^i x_{1:k}^i. \tag{23c}$$

3. *Resampling:* Take N samples from the set $\{x_{1:k}^i\}_{i=1}^N$ where the probability of taking sample i, is $w_{k|k}^i = 1/N$.

4. *Time update:* Generate predictions,

$$x_{k+1}^i \sim q(x_{k+1}|x_k^i, y_{k+1}), \tag{23d}$$

and change the weights as

$$w_{k+1|k}^i = w_{k|k}^i \frac{p(x_{k+1}^i|x_k^i)}{q(x_{k+1}^i|x_k^i, y_{k+1})}. \tag{23e}$$

9.2 Additional Notation

A new notation, for the bias in the gyroscope is defined like

$$\boldsymbol{b}^{\text{gyro}} \triangleq \begin{bmatrix} b_p & b_q & b_r \end{bmatrix}^T, \tag{24}$$

where the components are the bias for each angular direction.

9.3 Motion Model

Last year's project [6] derived an advanced model for describing the motion of the ROV. It is possible to use it as a motion model for the ROV, instead of a basic constant velocity model which was used last year. This model should better describe the dynamic of the ROV and could improve the state estimates. In continuous time, the new model is

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\tau}_{Act}) + \boldsymbol{v}, \tag{25}$$

where

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{\eta} \\ \boldsymbol{\nu} \\ \boldsymbol{b}^{\text{gyro}} \end{bmatrix},$$
$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\tau}_{Act}) = \begin{bmatrix} \boldsymbol{J}_{\boldsymbol{\Theta}}(\boldsymbol{\eta})\boldsymbol{\nu} \\ \boldsymbol{G}_{\boldsymbol{\nu}}(\boldsymbol{\Theta}, \boldsymbol{\nu}, \boldsymbol{\tau}_{Act}) \\ 0 \end{bmatrix},$$

and $\boldsymbol{G}_{\boldsymbol{\nu}}(\boldsymbol{\Theta}, \boldsymbol{\nu}, \boldsymbol{\tau}_{Act})$ is given by the right hand side of (20). Using Euler forward,

$$\dot{\boldsymbol{x}} = \frac{\boldsymbol{x}_{k+1} - \boldsymbol{x}_k}{T_s} \Longleftrightarrow \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + T_s \left(f(\boldsymbol{x}, \boldsymbol{\tau}_{Act}) + \boldsymbol{v} \right),$$

where T_s is the sample period, results in the time discrete model

$$\begin{aligned} \boldsymbol{x}_{k+1} = \begin{bmatrix} \boldsymbol{\eta}_{k+1} \\ \boldsymbol{\nu}_{k+1} \\ \boldsymbol{b}^{\text{gyro}}_{k+1} \end{bmatrix} &= \begin{bmatrix} \boldsymbol{I}_{6\times6}\boldsymbol{\eta}_k + T_s \boldsymbol{J}_{\boldsymbol{\Theta}}(\boldsymbol{\eta}_k)\boldsymbol{\nu}_k \\ \boldsymbol{I}_{6\times6}\boldsymbol{\nu}_k + T_s \boldsymbol{G}_{\boldsymbol{\nu}}(\boldsymbol{\Theta}_k, \boldsymbol{\nu}_k, \boldsymbol{\tau}_{Act_k}) \\ \boldsymbol{b}^{\text{gyro}}_k \end{bmatrix} \\ &+ \begin{bmatrix} T_s\boldsymbol{I}_{6\times6} & 0 & 0 \\ 0 & T_s\boldsymbol{I}_{6\times6} & 0 \\ 0 & 0 & T_s\boldsymbol{I}_{3\times3} \end{bmatrix} \begin{bmatrix} \boldsymbol{v}^{\eta} \\ \boldsymbol{v}^{\nu} \\ \boldsymbol{v}^{\text{gyro}} \end{bmatrix}. \end{aligned} \tag{26}$$

The current time discrete motion model, used by the sensor fusion module is

$$\begin{bmatrix} \boldsymbol{p}_{k+1} \\ \boldsymbol{v}_{k+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}_{3\times3} & T_s\boldsymbol{I}_{3\times3} \\ 0 & \boldsymbol{I}_{3\times3} \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_k \\ \boldsymbol{v}_k \end{bmatrix} + \begin{bmatrix} \frac{T_s^2}{2}\boldsymbol{I}_{3\times3} & 0 \\ 0 & T_s\boldsymbol{I}_{3\times3} \end{bmatrix} \begin{bmatrix} \boldsymbol{v}^p \\ \boldsymbol{v}^v \end{bmatrix} \tag{27}$$

and

$$\begin{bmatrix} \boldsymbol{q}_{k+1} \\ \boldsymbol{\omega}_{k+1} \\ \boldsymbol{b}_{k+1}^{\text{gyro}} \\ d_{k+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}_{4\times4} + \frac{1}{2}T_s\boldsymbol{S}(\boldsymbol{\omega}) & \frac{T_s^2}{2}\bar{\boldsymbol{S}}(\boldsymbol{q}) & 0 & 0 \\ 0 & \boldsymbol{I}_{3\times3} & 0 & 0 \\ 0 & 0 & \boldsymbol{I}_{3\times3} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{q}_k \\ \boldsymbol{\omega}_k \\ \boldsymbol{b}_k^{gyro} \\ d_k \end{bmatrix} + \begin{bmatrix} \frac{T_s^3}{4}\bar{\boldsymbol{S}}(\boldsymbol{q}) & 0 & 0 & 0 \\ 0 & T_s\boldsymbol{I}_{3\times3} & 0 & 0 \\ 0 & 0 & T_s\boldsymbol{I}_{3\times3} & 0 \\ 0 & 0 & 0 & T_s \end{bmatrix} \begin{bmatrix} \boldsymbol{v}^q \\ \boldsymbol{v}^\omega \\ \boldsymbol{v}^{\text{gyro}} \\ v^d \end{bmatrix}, \tag{28}$$

where $\boldsymbol{q}$ is the quaternion representation of the Euler angles and

$$\boldsymbol{S}(\boldsymbol{\omega}) = \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & -r & q \\ q & r & 0 & -p \\ r & -q & p & 0 \end{bmatrix} \text{ and } \bar{\boldsymbol{S}}(\boldsymbol{q}) = \begin{bmatrix} -\epsilon_1 & -\epsilon_2 & -\epsilon_3 \\ -\eta & \epsilon_3 & -\epsilon_2 \\ -\epsilon_3 & \eta & \epsilon_1 \\ \epsilon_2 & -\epsilon_1 & \eta \end{bmatrix}.$$

One major change is that the quaternion representation is changed to only use Euler angles. Since the ROV should not operate near angles of $\pm 90°$, the gimbal lock should not be an issue. Although, this is something to have in mind in the proceeding of the project.

9.4 Sensors and Measurements

The ROV has several sensors to use in the data fusion, including the new sonar sensors. In order to incorporate the sensors into the sensor fusion module, measurement equations for all sensors have to be constructed. The measurement equations must be expressed in the states used in the model.

9.4.1 IMU Measurements

The IMU has one gyroscope and one accelerometer. These measurements can be used to estimate the states of the attitude and angular velocities.

The measurement equation for the gyroscope is assumed to be

$$\boldsymbol{y}^{\text{gyro}} = \boldsymbol{\omega} + \boldsymbol{b}^{\text{gyro}} + \boldsymbol{e}^{\text{gyro}}, \tag{29}$$

where $\boldsymbol{y}^{\text{gyro}}$ is the measurement from the gyro, $\boldsymbol{\omega}$ is the angular velocities in LCS, $\boldsymbol{b}^{\text{gyro}}$ is the bias in the gyro and $\boldsymbol{e}^{\text{gyro}} \sim N(0, \boldsymbol{R}^{\text{gyro}})$ is the Gaussian measurement noise.

The measurement equation for the accelerometer is assumed to be

$$\boldsymbol{y}^{\text{acc}} = \boldsymbol{R}(\boldsymbol{\Theta})^T \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} + \boldsymbol{e}^{\text{acc}}, \tag{30}$$

where $\boldsymbol{y}^{\text{acc}}$ is the measurement from the accelerometer, g is the gravitational acceleration and $\boldsymbol{e}^{\text{acc}} \sim N(0, \boldsymbol{R}^{\text{acc}})$ is the Gaussian measurement noise. The displacement of the accelerometer, compared to the ROV's CG, is neglected.

9.4.2 Magnetometer Measurements

The magnetometer measures the strength of the magnetic field at a point in space. It can thus measure the direction of the earth's magnetic field relative to the ROV. This can be useful information to the sensor fusion module when it tries to estimate the ROV's attitude.

The measurement equation is assumed to be

$$\boldsymbol{y}^{\text{mag}} = \boldsymbol{R}(\boldsymbol{\Theta})^T \begin{bmatrix} \sqrt{m_x^2 + m_y^2} \\ 0 \\ m_z \end{bmatrix} + \boldsymbol{e}^{\text{mag}}, \tag{31}$$

where $\boldsymbol{y}^{\text{mag}}$ is the measured magnetic field, m_x, m_y, m_z is the measured magnetic field when the ROV is aligned with the PCS at start up and $e^{\text{mag}} \sim N(0, \boldsymbol{R}^{\text{mag}})$ is the Gaussian measurement noise.

9.4.3 Pressure Measurements

A pressure sensor and a barometer are mounted on the ROV. The barometer measures the atmospheric pressure inside the ROV and the pressure sensor measures the pressure in the water. This can be used to estimate the z-position of the ROV. Since the pressure sensor is positioned in the stern of the ROV, the attitude of the ROV has to be taken into account when estimating the z-position of the center of gravity.

The measurement equation is assumed to be given by

$$y^{\text{pressure}} = \rho g \left(z + \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \boldsymbol{R}(\boldsymbol{\Theta}) \begin{bmatrix} x_{\text{offset}}^{\text{pressure}} \\ 0 \\ 0 \end{bmatrix} \right) + p^{\text{air}} + e^{\text{pressure}}, \tag{32}$$

where y^{pressure} is the measured pressure from the sensor, ρ is the density of the water, g is the gravitational acceleration, z is the z-component of the position $\boldsymbol{p}$, $x_{\text{offset}}^{\text{pressure}}$ is the offset in x-direction of the pressure sensors position, p^{air} is the pressure at the surface and $e^{\text{pressure}} \sim N(0, R^{\text{pressure}})$ is the Gaussian measurement noise.

9.4.4 Sonar Measurements

Three sonar sensors will be mounted on the ROV. By transmitting a sound wave at a specific frequency and listening for that sound wave to bounce back, they can measure the distance to an object. These sonar sensors can therefore be used to estimate the ROV's relative position in a pool when the ROV's attitude is taken into account. One possible solution discussed in Section 3.1, is to position the sonars in the xy-plane of the PCS. The measurement equations are design for this configuration, but can be reconstructed to fit the other possible solution of a mounting the sonars in all three, x, y and z, directions. If the sonars are not mounted in the axis of the LCS, one have to compensate for this when using the measurements. This is the purpose of the SCS, which is described in Section 5.3.

The measurement equation can generalized be written in the form

$$\boldsymbol{y}^{\text{sonar}} = \boldsymbol{h}(\boldsymbol{\eta}) + \boldsymbol{e}^{\text{sonar}}. \tag{33}$$

It is assumed that the measurements are in the LCS since the transformation from SCS to LCS is straightforward. In order to simplify (33), one can think of it as three separate

equations; one equation for each sonar sensor. The measurement of one sonar sensor y_i^{sonar} is the measured distance to the wall from one of the sonar sensor i. It is here assumed that the three sensors are mounted in the xy-plane. Then $i = 1$ corresponds to the sonar in the x-direction, $i = 2$ is the sonar in positive y-direction and $i = 3$ is the sonar in negative y-direction. See Section 3.1 for a discussion about different sonar placements. The function $h_i(\boldsymbol{\eta})$ describes what the ROV should measure, given a certain state $\boldsymbol{\eta}$. The function is

$$h(\boldsymbol{\eta}) = \boldsymbol{\delta} R_z(\boldsymbol{\Theta})^T \left(\boldsymbol{p}_s - \boldsymbol{p}\right),$$

where

$$\boldsymbol{\delta} = \begin{cases} \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, & i = 1 \\ \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}, & i = 2, 3 \end{cases}$$

and

$$\boldsymbol{R}_z(\boldsymbol{\Theta}) = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The coordinate $\boldsymbol{p}_s$, is given by the minimum Δ hitting an object or wall, in the function

$$\boldsymbol{p}_s = \boldsymbol{p} + R(\boldsymbol{\Theta})\boldsymbol{p}_\Delta,$$

where

$$\boldsymbol{p}_\Delta = \begin{cases} \begin{bmatrix} \Delta & 0 & 0 \end{bmatrix}^T, & i = 1 \\ \begin{bmatrix} 0 & \Delta & 0 \end{bmatrix}^T, & i = 2 \\ -\begin{bmatrix} 0 & \Delta & 0 \end{bmatrix}^T, & i = 3 \end{cases}.$$

One illustration of a sonar measurement can be seen in Figure 10.

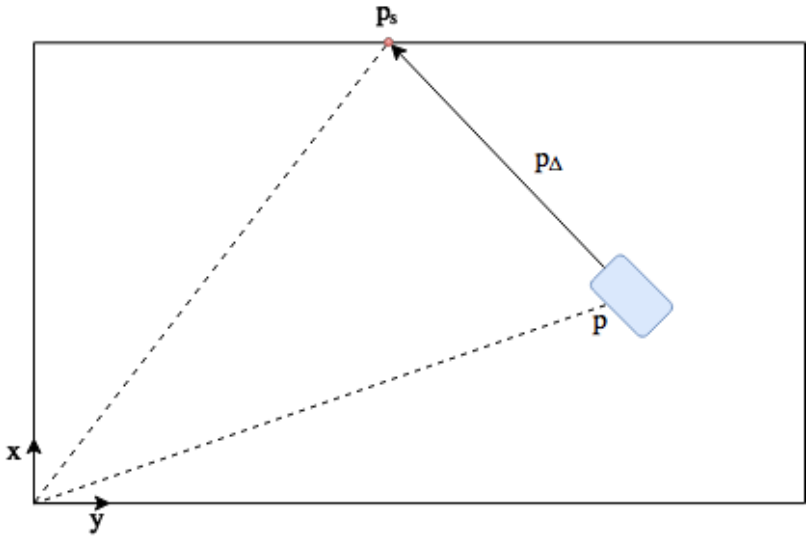

Figure 10: An overview of a sonar measurement in the x-direction in the LCS, with sonar y_1^{sonar}. The figure seen from above and illustrates a 2D view.

10 Graphical User Interface

The existing Graphical User Interface (GUI) is created with *rqt* [5], which is a framework for GUI development for ROS. The framework is based on graphical interactive objects such as windows, buttons, plots, etc. The module will contain multiple functions such as graphical representation of the ROV's orientation.

10.1 Existing Graphical User Interface

The existing GUI allows the user two ways of operating the ROV. The ROV can be operated from the GUI or with a XBOX-controller. A number of other parameters can be set and changed from the GUI. The GUI makes it possible to log and record data from the run. After the data has been recorded and logged it is sent to ROS topics. The data can then be plotted in the GUI or in the terminal. The GUI shows a continuous plot that can plot different sensor states and reference data, of the operators choice.

10.2 Further Development on the Graphical User Interface

Expanding the GUI and adding further functionality is possible since the GUI is *rqt* based. Preferably, existing plugins shall be used, however, if no plugin satisfy the need, new plugins could be developed.

The GUI shall be able to provide reference points to the pathfinder module. More information on the pathfinder module can be found in Section 11.

On the GUI, a map of the known environment should be shown and also give a graphical view of the ROV's current estimated location on the map.

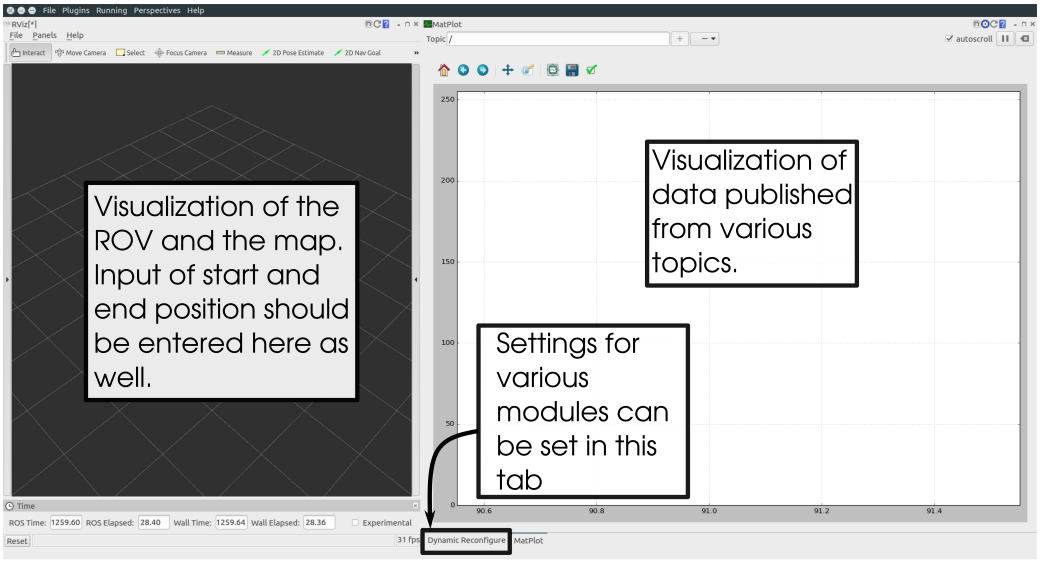

Figure 11: A preview of the GUI for the system

10.2.1 Reference Point Generation

The user could give an input reference trajectory in multiple ways. The inputs will be sent to the pathfinder module that will calculate a reference trajectory.

One way could be to set the reference points from the plugin Dynamic Reconfigure in *rqt*. Another way could be to use the visualization tool *RViz* and set the reference points on a map.

10.2.2 Visualization of Trajectory

The visualization of the reference trajectory could be set in *RViz*. Using this tool could allow the user to also visualize what the ROV is doing with a short delay.

Visualization of inputs, outputs and other messages can be read from the plot plugin in *rqt*.

10.2.3 GUI preview

In Figure 11, a preview of the GUI can be seen. On the left-hand side, a visualization of the map with the ROV and its path should be seen. On the right-hand side, there are two tabs: Dynamic Reconfigure and MatPlot. In the Dynamic Reconfigure tab, settings for the controller and the sensor fusion module could be set. In the MatPlot tab, data from different signals could be visualized in a plot.

11 Pathfinder

The pathfinder module is a new module that will be developed during this project. In the GUI, a starting and an end position will be set by the user. These positions are given in the xy-plane and the path will only be generated in 2D without taking into account the

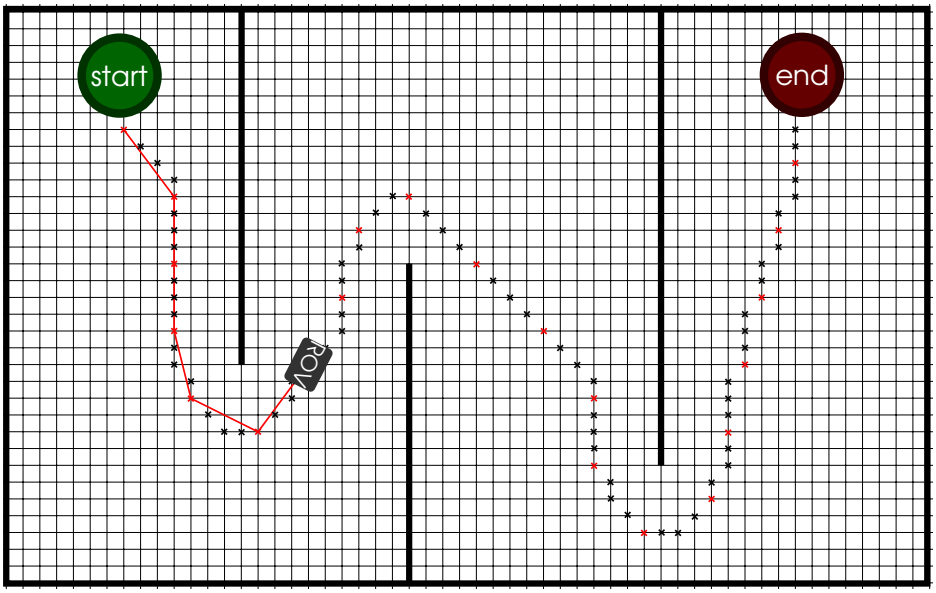

Figure 12: An illustration of the pathfinder. The map is represented by a grid. The crosses represents the path generated by the pathfinding algorithm. The red crosses represents the simplified path after the original path has been divided into multiple points. The ROV will advance to these positions and the reference position will be updated to the next red cross after the ROV has reached the current reference position. The red line represents the traveled path.

z-position. The purpose of the pathfinder node is to create a path between these positions while taking into account the pool environment as well as any obstacles that might exist in the map. These obstacles are virtual and are only introduced for the pathfinding and GUI node to generate a path in a more complicated environment. The path will be divided in multiple points and the pathfinder node will output these points to the controller node.

An illustration of the pathfinder can be seen in Figure 12.

11.1 Path Generation

The map of the pool environment will be designed as a grid and then an algorithm will be used to find the shortest path. The grid map can be interpreted as multiple nodes connected to each other when building the pathfinding algorithm. There are multiple search algorithms that can be implemented.

The simplest algorithm would be to use the breadth first search algorithm. This algorithm runs through every neighbouring node that it has without accounting for any path cost.

A common algorithm for pathfinding is Dijkstra's algorithm. Instead of calculating all possible paths, Dijkstra's algorithm prioritizes which paths to calculate by favoring lower cost paths. To discourage the ROV to run into obstacles, higher costs can be assigned to the obstacles or the nodes of the obstacles can be removed. The nodes neighbouring the obstacles can also have a higher cost or be removed to ensure that the ROV does not navigate too close to the obstacles [4].

A* is a modified version of Dijkstra's algorithm that is optimized towards a destination. It prioritizes the nodes that are leading towards the end point. It does so by introducing a heuristic function. Instead of minimizing $f(n) = g(n)$, where n is the last node, $g(n)$

is the accumulated cost of the path from the start node, the A* algorithm minimizes $f(n) = g(n) + h(n)$, where $h(n)$ is the heuristic function that estimated the cost to move from n to the end position [4].

Since the A* algorithm is an improved version of Dijkstra's algorithm and both algorithms are similar to implement, A* is the obvious algorithm to choose.

12 Control System

In this project, the ROV will be controllable in three ways.

- It shall be possible to specify and maintain the ROV's attitude and position.
- It shall be possible to specify and maintain the ROV's linear and angular velocity.
- The ROV shall be able to follow a reference trajectory.

For the ROV to be able to follow a trajectory requires the position controller and velocity controller to be well designed and functional.

12.1 Various Control Methods

In this section, different control methods are presented. These methods have been considered for use in the ROV.

12.1.1 Linear-Quadratic Regulator

To improve the control system a linear-quadratic regulator (LQR) can be implemented. A linear state feedback is calculated by solving an infinite horizon optimization problem offline. The control signals' task is to minimize a cost function. The cost function represents a trade-off between using large control signals and using small deviations from the reference point. One of the disadvantage of LQR is the difficulty of tuning the parameters, especially when the system is expected to be able to operate near its physical boundaries.

12.1.2 Model Predictive Control

MPC is a predictive control method, which means that the applied control signal is calculated by predicting the future behaviour of the system using a dynamic model. A MPC can take into account constraints and limitations in the control process. It works similarly as a LQR controller but with one fundamental different, it allows the possibility to handle constrains. For this method the control problem is translated to an optimization problem which needs to be solved online for each time the control node shall calculate a new control signal. A disadvantage of a MPC is that it requires a lot of computing.

12.1.3 Cascade Control

If there are different states in a system, subject to different dynamics, a cascade control structure can be a good solution to compute the control signal, see Figure 13. In this project, cascade control structure can be used to control angles and positions. The outer loop will use an angular reference as input, and give a reference signal for the inner loop as output, which is used to control the angular velocity.

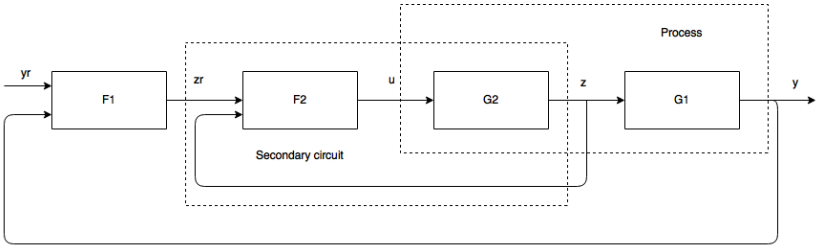

Figure 13: An illustration of a Cascade control structure.

12.1.4 Feedback Linearization

The feedback linearization is a common method usually used to control nonlinear systems. The main aim is to transfer a nonlinear system into a linear system through a change of variables and appropriate control signal. A nonlinear system can, for example, have the following state model:

$$\dot{x} = f(x) + g(x)u, \tag{34}$$

$$y = h(x). \tag{35}$$

There are two different variants of the feedback linearization, input-output linearization and state-space linearization. The first one is about developing an input

$$u = a(x) + b(x)w,$$

to linearize the map between the new input w and the output y. In the second method, the state-space linearization, the linearization is based on linearizing the map between the new input w and the entire vector of state variables x.

12.2 General Structure

The main elements used for the general structure for controlling the ROV are a method to pass state references, a controller to calculate control signals, measurement signals and an observer to compute estimated states in the feedback loop.

The state references $\boldsymbol{x}_{ref}$ and state estimates $\hat{\boldsymbol{x}}$ enter the controller as inputs, and the controller computes a vector of control signals, $\boldsymbol{u}$. The control signals $\boldsymbol{u}$ together with Actuator Dynamics gives $\boldsymbol{\tau}_{Act}$ which is a vector of the forces and torques acting on the ROV from the actuators. In addition, the ROV is subject to external disturbances $\boldsymbol{\tau}_{dist}$ and also forces and torques from the actuators. The measurement signals $\boldsymbol{y}$ is fed back through an observer. The observer is used to estimate the ROV's state estimates which are used in the controller.

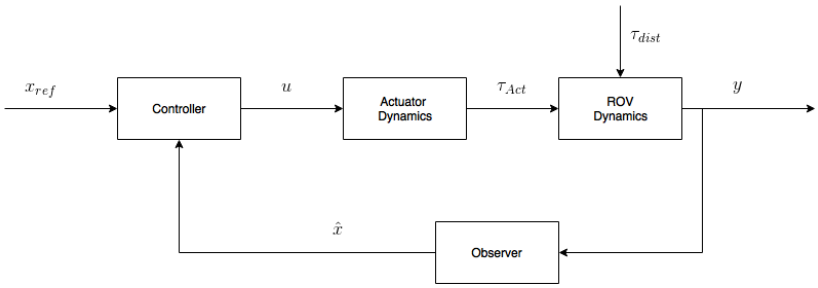

Figure 14: An illustration of the general structure of the feedback control system.

12.3 The Inverse Thrust Geometry Matrix Look-up Table

Decoupling in the system can be achieved by calculating the inverse thrust geometry matrix $\boldsymbol{T}^{-1}$. The controller's job is to calculate the desired forces and torques $\boldsymbol{\tau}_{Act}$ which together with the the look-up table from force to control signal and the inverse of the thrust geometry matrix, computes the control signals.

$$\boldsymbol{u_v} = \boldsymbol{f}^{-1}(\boldsymbol{T}^{-1}\boldsymbol{\tau}_{Act}) \tag{36}$$

12.4 Decoupling in the Local Coordinate System

To be able to transform the underwater vehicle's dynamics to a linear system the model defined in (1) needs to be reconsidered. The vector $\boldsymbol{a^b}$ contains the body-fixed desired acceleration

$$\dot{\boldsymbol{\nu}} = \boldsymbol{a^b}. \tag{37}$$

The model in (1) and the approach discribed in [2] are used to obtain the decoupling in the LCS. The model in (1) contains the non-linearities $\boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu}$, $\boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu}$ and $\boldsymbol{g}(\boldsymbol{\eta})$. To obtain a linear system, these need to be cancelled out. The body-fixed desired acceleration and the nonlinearities in the system gives the forces and torques that are demanded. This can be achieved by using a feedback linearization using state estimates in the following model,

$$\boldsymbol{\tau} = \boldsymbol{M}\boldsymbol{a^b} + \boldsymbol{G}^{-1}(\hat{\boldsymbol{\eta}}, \hat{\boldsymbol{\nu}}), \tag{38}$$

$$\boldsymbol{G}^{-1}(\hat{\boldsymbol{\eta}}, \hat{\boldsymbol{\nu}}) = \boldsymbol{C}(\hat{\boldsymbol{\nu}})\hat{\boldsymbol{\nu}} + \boldsymbol{D}(\hat{\boldsymbol{\nu}})\hat{\boldsymbol{\nu}} + \boldsymbol{g}(\hat{\boldsymbol{\eta}}), \tag{39}$$

and a control structure as seen in Figure 15.

Here, the commanded acceleration $\boldsymbol{a^b}$ can be chosen, for instance, by pole placement or linear quadratic optimal control theory.

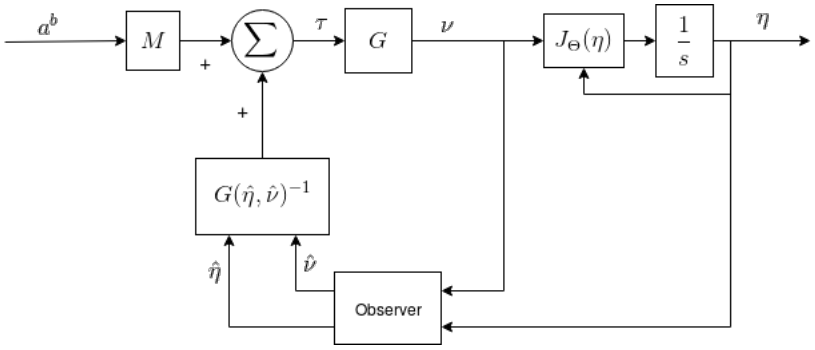

Figure 15: Nonlinear decoupling in the velocity control.

12.5 Decoupling in the Pool Coordinate System

Similar to Section 12.4, the position and attitude control can also be linearly decoupled in the pool coordinate system. Taking in consideration the model (1), consider

$$\ddot{\boldsymbol{\eta}} = \boldsymbol{a^p}, \tag{40}$$

where $\boldsymbol{a^p}$ is the desired acceleration in the PCS. Differentiating Equation (1a) with respect to time gives

$$\ddot{\boldsymbol{\eta}} = \dot{\boldsymbol{J}}_{\boldsymbol{\Theta}}(\boldsymbol{\eta})\boldsymbol{\nu} + \boldsymbol{J}_{\boldsymbol{\Theta}}(\boldsymbol{\eta})\dot{\boldsymbol{\nu}}, \tag{41}$$

which also can be written as

$$\dot{\boldsymbol{\nu}} = \boldsymbol{J}_{\boldsymbol{\Theta}}^{-1}(\boldsymbol{\eta})[\ddot{\boldsymbol{\eta}} - \dot{\boldsymbol{J}}_{\boldsymbol{\Theta}}(\boldsymbol{\eta})\boldsymbol{\nu}]. \tag{42}$$

The nonlinear control law (38), inserted in (1b) gives

$$\boldsymbol{M}(\dot{\boldsymbol{\nu}} - \boldsymbol{a^b}) = \boldsymbol{M}\boldsymbol{J}_{\boldsymbol{\Theta}}^{-1}(\boldsymbol{\eta})[\ddot{\boldsymbol{\eta}} - \dot{\boldsymbol{J}}_{\boldsymbol{\Theta}}(\boldsymbol{\eta})\boldsymbol{\nu} - \boldsymbol{J}_{\boldsymbol{\Theta}}(\boldsymbol{\eta})\boldsymbol{a^b}] = 0. \tag{43}$$

If $\boldsymbol{a^p}$ is chosen as

$$\boldsymbol{a^p} = \dot{\boldsymbol{J}}_{\boldsymbol{\Theta}}(\boldsymbol{\eta})\boldsymbol{\nu} + \boldsymbol{J}_{\boldsymbol{\Theta}}(\boldsymbol{\eta})\boldsymbol{a^b}, \tag{44}$$

then (43) can be written

$$\boldsymbol{M}^*(\ddot{\boldsymbol{\eta}} - \boldsymbol{a^p}) = 0, \tag{45}$$

where $\boldsymbol{M}^* = \boldsymbol{J}_{\boldsymbol{\Theta}}^{-T}(\boldsymbol{\eta})\boldsymbol{M}\boldsymbol{J}_{\boldsymbol{\Theta}}^{-1}(\boldsymbol{\eta}) > 0$. From Equation (44) it can be seen that

$$\boldsymbol{a^b} = \boldsymbol{J}_{\boldsymbol{\Theta}}^{-1}(\boldsymbol{\eta})[\boldsymbol{a^p} - \dot{\boldsymbol{J}}_{\boldsymbol{\Theta}}(\boldsymbol{\eta})\boldsymbol{\nu}], \tag{46}$$

where the desired acceleration $\boldsymbol{a^p}$ can, for instance, be chosen by pole placement with feed-forward acceleration or by using linear quadratic linear control theory similarly to Section 12.4.

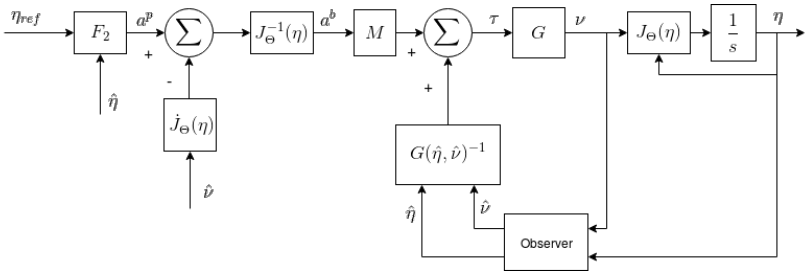

Figure 16: Decoupling in the position and attitude control.

12.6 Combined Velocity Controller

The reference $\boldsymbol{\nu}_{ref}$ has the reference signals of the linear and angular velocities along and about LCS. Feedback linearization is used as explained in Section 12.4, and the accelerations in LCS $\boldsymbol{a^b}$ can be computed, for instance, by pole placement or linear quadratic optimal control theory. This gives the desired forces and torques $\boldsymbol{\tau}$. The desired thruster forces can then be calculated by multiplication with the inverse of the thrust geometry matrix $\boldsymbol{T}^{-1}$. The thruster forces with the look-up table from force to control signal $\boldsymbol{f}^{-1}(\cdot)$ give the control signals. Figure 17 represents the controller structure used to control the linear and angular velocities along and about the LCS, where the controller block F_1 contains a linear controller.

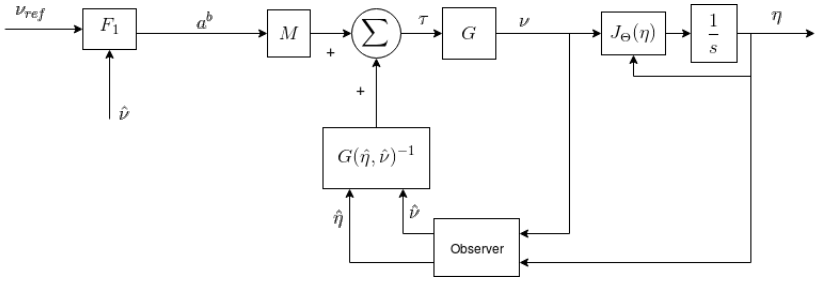

Figure 17: Combined velocity controller and decoupling in LCS.

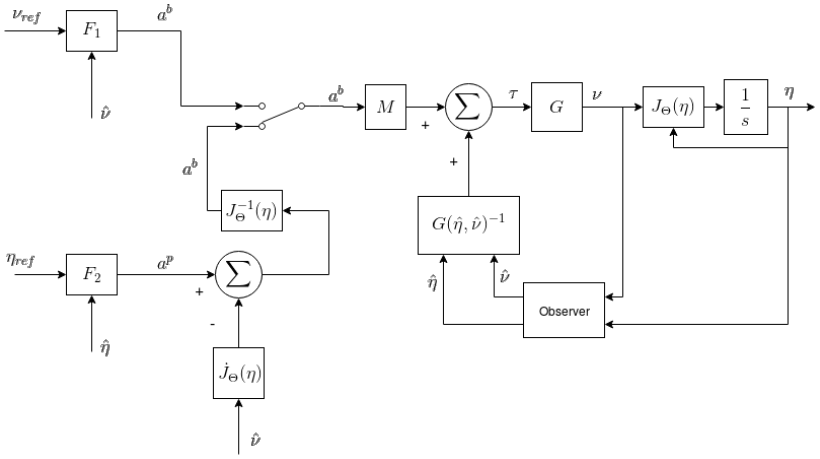

Figure 18: A possible structure for both position and attitude and velocity controllers.

12.7 Position and Attitude Controller

The positional and angular references $\boldsymbol{\eta}_{ref}$ are first given in PCS coordinates (position) and Euler angles (attitude). A model based controller for example a LQR treats these references and outputs linear references in PCS and Euler angle time derivative references. An alternative is by using a feedback linearization as seen in Section 12.5 and described in [2].

Figure 18 shows a possible structure for both the velocity controller and position and attitude controller. The latter receives desired references $\boldsymbol{\eta}_{ref}$ and is fed position estimates $\hat{\boldsymbol{\eta}}$ and enter the controller F_2. The controller F_2 threats the signals and yields an $\boldsymbol{a^p}$ which is transformed in terms of $\boldsymbol{a^b}$ as shown in Section 12.5. The commanded acceleration $\boldsymbol{a^b}$ enters the combined velocity controller which then delivers the necessary velocity $\boldsymbol{\nu}$ to reach a certain position and lastly outputs the position $\boldsymbol{\eta}$.
This structure contains a switch, which can be used to switch between signals from trajectory references, $\boldsymbol{\eta}_{ref}$ or direct inputs of $\boldsymbol{\nu}_{ref}$. The latter is in case it is desired to control the ROV manually, for instance, with a XBOX controller, where the joystick inputs could act as $\boldsymbol{\nu}_{ref}$ references to the velocity controller.

12.8 Trajectory Following

As described in Section 11, the trajectory is multiple positions after each other sent from the pathfinder node. The first position will be sent as a reference position to the position controller and when the position has been achieved, the reference position should be updated to the next position in the trajectory and sent to the position controller. The process is repeated until the ROV has reached it's final destination. The distance between ROV and the reference position before the reference position is updated is a design parameter that can be adjusted.

12.9 Modes of Operation

Two main modes of operation shall be implemented in the ROV. The functionality of the first mode is to control the ROV with an XBOX controller. For linear and angular maneuvering, this mode will use the combined velocity controller. References of the linear

and angular velocities in LCS will be controlled by the operator.

The second mode activates the position and velocity controller. References for the position will be in PCS and references for the angle will be sent as Euler angles.

References

[1] Adam Aili and Erik Ekelund. *Model-Based Design, Development and Control of an Underwater Vehicle*. MSc Thesis - LiTH-ISY-EX–16/4979–SE, Sweden: Linköping University, 2016.

[2] Thor Fossen. *Handbook of marine craft hydrodynamics and motion control*. Chichester, West Sussex, U.K. ; Hoboken N.J. : Wiley, 2011., 2011.

[3] Fredrik Gustafsson. *Statistical Sensor Fusion*. Studentlitteratur, 2012.

[4] Redblobgames. Introduction to A*
. http://theory.stanford.edu/ amitp/GameProgramming/AStarComparison.html. Accessed: 2017-10-18.

[5] ROS.org. rqt: a Qt-based framework for GUI development for ROS. http://wiki.ros.org/rqt. Accessed: 2017-09-20.

[6] Niklas Sundholm. *Technical Documentation, Remotely Operated Underwater Vehicle*. 2016.