AALBORG UNIVERSITY
STUDENT REPORT

Bachelor Thesis

Parameter Estimation and Model-based Control of an ROV with Imaging for Object Detection

Students:
Emil Már Einarsson
Thomas Thuesen Enevoldsen

Supervisors:
Zhenyu Yang
Simon Pedersen

June 11, 2017

AALBORG UNIVERSITY

STUDENT REPORT

School of Information and Communication Technology
Niels Bohrs Vej 8
DK-6700 Esbjerg
http://sict.aau.dk

Title:
Parameter Estimation and Model-based Control of an ROV with Imaging for Object Detection

Theme:
Scientific Theme

Project Period:
Spring Semester 2017

Project Group:
ED6-1-F17

Participant(s):
Emil Már Einarsson
Thomas Thuesen Enevoldsen

Supervisor(s):
Zhenyu Yang
Simon Pedersen

Page Numbers: 94

Date of Completion:
June 11, 2017

Abstract:

This bachelor thesis focuses on implementing semi-autonomous capabilities on an existing ROV platform. During the report the creation of the mathematical model, parameter estimation and model-based control of an ROV is explained. The platform used is the VideoRay Pro 4 using the on-board IMU, a multi-beam sonar and HD camera for acquiring environmental and positional data. Based on the dynamics of the ROV, a general mathematical model is created describing the movement in 3 degrees of freedom, these being surge, heave and yaw. Extensive parameter estimation is performed to determine the various parameters needed for the mathematical model. The control system is MIMO and uses modern control in order to create an optimal controller using LQR, where due to lack of sensory information available a state estimator is utilised. The developed linear controller is simulated on the nonlinear system to more accurately simulate the actual system dynamics. The secondary focus is obtaining positional data using computer vision, to detect and navigate with respect to a predefined object. The CV is created and implemented using EmguCV and C#.

Contents

Preface

The project entitled *Parameter Estimation and Model-based Control of an ROV with Imaging for Object Detection* was made by two students from the Electronics and Computer Engineering Programme at Aalborg University Esbjerg, for the P6 bachelor thesis during the sixth semester. All of the software written for the project will be submitted as a digital appendix.

We would like to thank our supervisor and professor Zhenyu Yang for suggesting the use of VideoRay platform and for supervision and guidance throughout the project.

We would also like to thank Simon Pedersen our for his exceptional supervision and guidance throughout the entire project, especially with the insight into new concepts and methods related to the development and implementation of modern control theory.

Finally we would like thank Auraskolen Hjerting, Esbjerg and MacArtney A/S for providing access to their respective swimming pools and test environment to conduct crucial experiments and the final implementation.

Aalborg University, June 11, 2017.

Thomas Thuesen Enevoldsen
<tten14@student.aau.dk>

Emil Már Einarsson
<eeinar14@student.aau.dk>

Acronyms

SNAME	The Society of Naval Architects and Marine Engineers
MIMO	Multiple Input Multiple Output
SISO	Single Input Single Output
AUV	Autonomous Underwater Vehicle
UUV	Underwater Unmanned Vehicle
ROV	Remotely Operated Vehicle
ROTV	Remotely Operated Towed Vehicle
LTI	Linear time-invariant
SI	Single Input
MI	Multiple Input
SDK	Software Development Kit
GUI	Graphical User Interface
CV	Computer Vision
DOF	Degrees of freedom
AUX	Auxiliary
COM	Communication port, Serial port
BRG	Blue Red Green
RGB	Red Green Blue
HSV	Hue Saturation Value
LQR	Linear Quadratic Regulator
WCF	Windows Communication Foundation
USB	Universal Serial Bus

Chapter 1

Introduction

1.1 Introduction

The ocean is a vital part of everyday life on earth, since it influences weather and regulates temperature that in the end is required for all living creatures and organisms. In the past, the ocean surface has been used to explore the different continents and it currently transports 90% of the world's trade [1] [2]. Even though 70% of the planet's surface is covered in water, 95% of the 70% remains unexplored [3]. Due to the large amount of unknown, there is a motivation to map these unknown areas and the possibility to gain new knowledge within the many sciences associated with the ocean. Besides exploring the unexplored, the ocean is also home to many off-shore operations. These range from wind power farms to drilling, where the inspection and maintenance of the sub-sea parts of the installations requires divers or advanced underwater equipment, such as underwater vehicles.

Underwater vehicles are an important tool in order to move the exploration from the surface to the depths of the ocean. These underwater vehicles come in many shapes and sizes, and serve many different purposes. There are different classifications and types of underwater vehicles, which include remotely operated vehicles (ROV), remotely operated towed vehicles (ROTV), autonomous underwater vehicles (AUV) and unmanned underwater vehicles (UUV) [4]. ROVs are tethered vehicles, where it is possible for the operator to communicate and also power the vehicle from a separate location. AUVs are the tether-less counterparts, where the vehicle is fully autonomous in terms of navigation and movement, and are typically pre-programmed on a mission to mission basis. These different underwater vehicles are used frequently in all areas of the offshore and underwater industry, in order to either perform inspections and maintenance or to explore various known and unknown environments. These unmanned underwater vehicles are often favoured, due to their ability to perform tasks and operate in conditions hazardous to humans, especially since deep sea diving performed by humans is time consuming and requires careful execution, due to the health issues related to the high underwater pressures and unsafe environments [5].

Exploration and maintenance ROVs are typically equipped with high-definition cameras and lighting in order to view their surroundings, so that the operator can navigate based on the video feed and collect visual data. Vehicles interested in mapping the seabed commonly use sonar technology, typically multi-beam sonar. Depending on the vehicle type or classification they are also equipped with grippers and other tools, so that it is possible to interact with the environment. This is especially important when performing maintenance tasks on offshore equipment or taking samples of deep-sea material. In the offshore industry both the working class and inspection class ROVs are highly desirable, since the working class ROVs are capable of performing repairs, installations and also general maintaince of the deep-sea equipment, and the inspection class ROVs use their cameras and lighting systems to perform inspections of things such as deep sea piping and constructions. An example of a current exploration team who utilizes ROVs and ROTVs is the Nautilus Live [6] ocean exploration team. They use various underwater vehicles to explore the deep sea with main focus being biology, geology and archaeology using their arsenal of UUVs.

1.2 ROV Classification

As mentioned in the previous section, there are many classes of underwater vehicles. A general hierachy for the classification of underwater vehicles can be seen in the chart shown in figure 1.1.

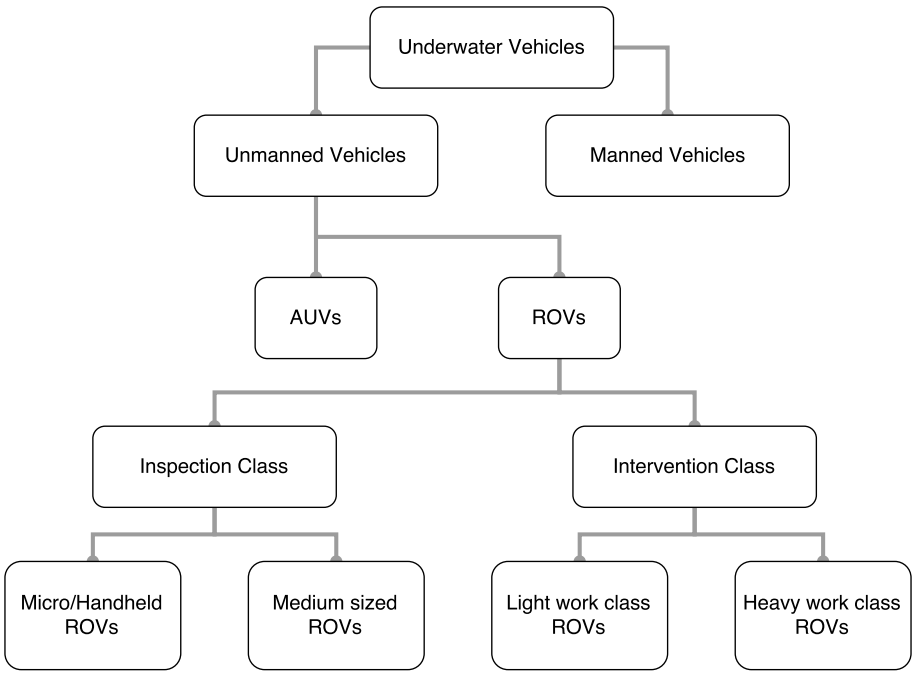


Figure 1.1: Classification of UUVs [7].

Underwater vehicles are often classified based on what they can do and also based on their operating objective. As an example, adding a tool to an inspection class ROV

would change it to an intervention class. There are broad uses for all of the different ROV classes, where the AUVs typically are programmed based on the mission at hand. Therefore, there is an interest in closing the gap between the functionality and capabilities of the ROVs and AUVs [7].

1.3 Notation

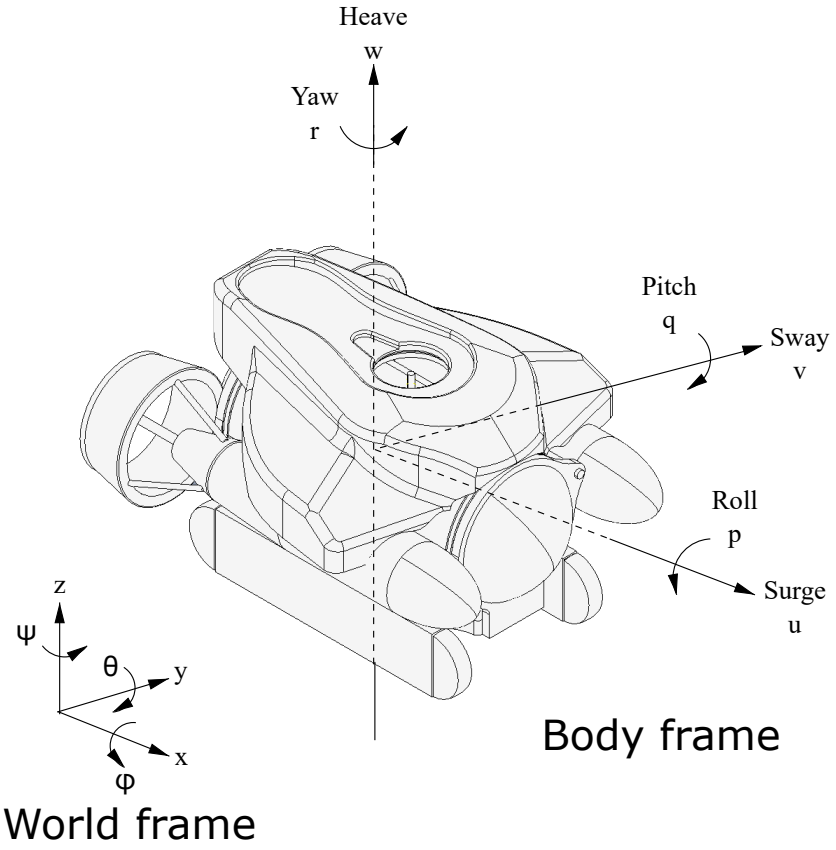


Figure 1.2: The ROV's reference frames and notation [8].

Different forms of notation will be utilized throughout the remainder of the report in order to describe the position, movements and dynamics of the ROV. Figure 1.2 shows the representation of two reference frames, where the world frame denotes the position in terms of x, y and z coordinates and angles in terms of ϕ, θ and ψ. The body frame includes the linear and angular velocities and these will be used when modelling the dynamics of the system. The mathematical modelling will use the notation for the movement, forces and moments shown in table 1.1. Whenever a derivative is used it will be in the form of dot notation, as an example, u is the surge velocity and $\dot{u}$ the associated acceleration.

DOF	Movement Notation	Forces and moments	Linear and angular velocities	Positions and Euler angles
1	Motions in x direction (surge)	**X**	u	x
2	Motions in y direction (sway)	**Y**	v	y
3	Motions in z direction (heave)	**Z**	w	z
4	Rotation about the x axis (roll)	**K**	p	ϕ
5	Rotation about the y axis (pitch)	**M**	q	θ
6	Rotation about the z axis (yaw)	**N**	r	ψ

Table 1.1: Notation for marine vessels [9].

Movement in the given 6 degrees of freedom will from here on out be specified in terms of surge, sway and heave for the linear motion and roll, pitch and yaw for the rotational motion.

Chapter 2

Problem Description

2.1 Problem Description

Key qualities desired from autonomous vehicles are often navigation, object detection and tracking. Autonomous underwater vehicles open up many possibilities in terms of exploration, maintenance and so forth. This project will focus on closing the gap between ROVs and AUVs, by attempting to create and implement the autonomous capability of detecting an object navigating based on the sensory information given on an existing ROV platform. The main objective of the developed system is to implement the ability to identify, track and navigate towards specified objects autonomously, which as previously mentioned are essential qualities of an autonomous vehicle. The main control system will require the creation of a mathematical model of the given ROV, which will be used when developing the required controllers needed in order to identify the object and thereafter navigate. The navigation controller will consist of two parts, the first being the control of the position of the physical system, where the desired position will be in terms of the world frame, such as the current depth and heading, in the three dimensional coordinate space with respect to the magnetic north and the sea-level. The second part of the navigation controller will be for the navigation towards an identified object, where the required control inputs will be calculated based on the location of the object with respect to the ROVs body frame. The ROV used for this project will be the VideoRay Pro 4, which is a commercial and industry grade ROV and therefore requires no assembly or programming for regular manual operation using the available software. The ROV kit supports a sonar attachment, which in this case is the Teledyne BlueView P450-45-I sonar. Using the given equipment, the VideoRay can be manually controlled and is fully controllable for surge, heave and yaw movement. In order to create these movements the ROV is equipped with 3 thrusters, a top-side thruster with 3 a bladed propeller for heave motion, and two powerful rear thrusters with 3 blades for surge and yaw movement. A more in-depth overview and technical description can be found in chapter 3.

In order to accomplish the object detection, environmental and visual data is necessary. In this case images from the sonar and/or camera, will be image processed in order to identify the given objects and their position relative to the ROVs body frame. The strengths and weaknesses of using either the sonar and/or camera for object detection will be discussed during chapter 4.

2.2 Problem Delimitation

To ensure that the project meets the desired scope, specific success criteria are formed which will be evaluated during the final stages of the report. The following list of requirements are the desired success criteria, where the focus will be on reaching the given goals.

- Create a 3DOF mathematical model for the VideoRay
- Identify and estimate the model parameters
- Design controllers for world frame and body navigation
- Detect and identify a predefined object in water
- Implement a navigation system with minimizing the distance towards the detected object

The creation of the mathematical model will be based on the given ROVs dynamics and its capabilities. Based on the formulated mathematical model, the model parameters will be estimated through experimentation and then validated using the recorded data. The controller design process will focus on optimally controlling the desired heading and depth with respect to the world frame and also the optimal control of approaching the detected object depending on the distances with respect to the body frame. In order to detect and identify the object under water, different technologies will be reviewed and discussed before choosing the appropriate approach. The ultimate goal will be developing and implementing the above mentioned points to bring autonomous capabilities to an existing ROV platform.

Chapter 3

Physical Platform

3.1 Sonar

3.1.1 Theoretical Background

SONAR stands for SOund Navigation And Ranging, in order to maintain consistency with other acronyms such as RADAR, which stands for RAdio Detecting And Ranging [10]. Sonar uses the physics of sound propagation to navigate, to communicate or to detect objects in water. The technology is divided into two subsections, these being active and passive sonar. Passive sonar listens to a sound waves generated from external sources by the use of hydrophones, whereas active sonar generates its own sound source and listens to its echoes [11].

As mentioned previously, sonar uses sound propagation as the basis of its technology. A sound wave loses energy in the form of heat while colliding with atoms in the medium it is travelling through. Water is more dense than air and therefore waves travel faster in water but will not travel as far. The speed of sound in air at $20°C$ is 343 m/s, water at $25°C$ 1493 m/s and in sea water at $25°C$ 1533 m/s [12]. The travel distance and through what medium sound waves can travel also depends on its frequency. A wavelength of 20 kHz (20000 cps) has a wavelength of 1.7 cm, compared to a wavelength of 20 Hz (20 cps) that has a wavelength of 1.7 m. This effect is noticeable with music when played in another room, where the lower frequencies from the bass is most notable and the middle frequencies, but the higher frequencies will not make it through the given barriers, in this case a wall that has a width wider than the given wavelength. In seawater the acoustic abortion is also frequency dependent in the way that higher frequencies will be absorbed faster [11] [13]. A wave travelling in water will have a spherical spread from its source and since the signal strength varies and changes greatly through the medium it is travelling through the Decibel scale (dB) is used to indicate signal strength. The main effects on sound waves travelling through water is the salinity, temperature and depth. Since we primarily will be working in shallow water the effects of depth can be ignored, where in deeper waters such as the ocean the effects of what ocean

layer the signal is travelling through the characteristics such as temperature vary greatly [11].

In this project, the sonar will be used for determining the distance to an object or range to an object and heading angle with respect to the object. In order to do so, a short pulse of duration T_p is transmitted from the source and the equation for the distance or range between the source and the object can be seen in equation 3.1, where R is the distance from the object, c is the speed of sound in the given medium and τ is the time delay. [11].

$$R = \frac{c\tau}{2} \tag{3.1}$$

The equation is divided by two since the sound has to travel to the object and return. For the calculations to be correct, sound speed must be known. The sound speed, as mentioned previously, is affected by depth, temperature and salinity. To have precises readings all three factors need to be known and based on them you can calculate the speed of the sound in the water seen in equation 3.2 [11].

$$\delta R = \frac{cT_p}{2} \tag{3.2}$$

Equation 3.2 is the calculation of the range resolution, which yields the minimum required delay between two echoes in order to be detectable.

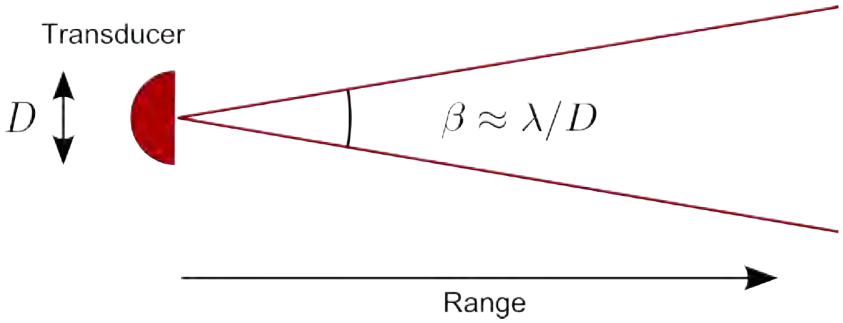


Figure 3.1: Beamwidth [11].

In order to get the direction of the echo there are two requirements. First, the antenna or source is large compared to the wavelength. The field of view from the speaker can be seen in figure 3.1 where D is the diameter of the speaker, λ is the wavelength and β is the beamwidth. The second important part is the grouping of hydrophones or transducers into an array shown in figures 3.2 and 3.3.

$$\theta = sin^{-1}\frac{c\delta t}{L_1} \tag{3.3}$$

To calculate the angle of the object, θ, equation 3.3 is utilized [11]. δt is the time difference of arrival and L_1 the distance between the receivers in the array, as shown on figure 3.2.

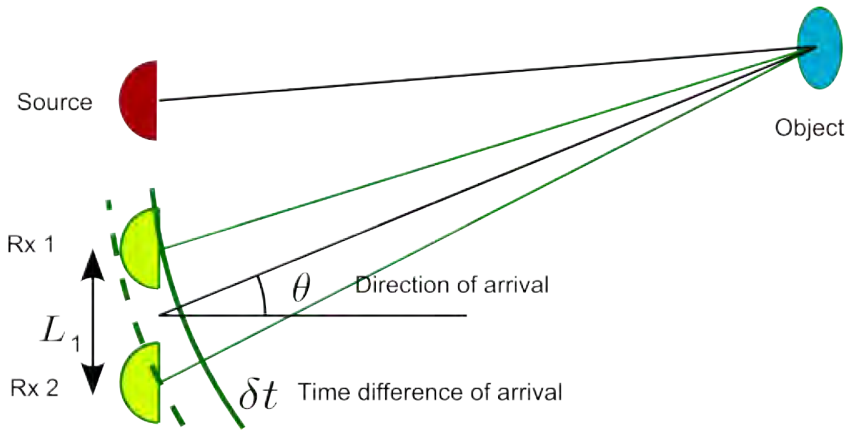


Figure 3.2: Directional array [11].

Imaging sonars require all the criteria mentioned previously and an implementation of an imaging sonar can be seen in figure 3.3. The resolution is influenced by the different characteristics of the sonar, for example a source with a fixed frequency but a higher amplitude will have a greater field of view, but travel a shorter distance. If a detected object is close to the device the image will be clearer because of the limited amount of external noise. The same frequency but at a lower amplitude will have a narrower field of view, but can travel a further distance [11] [13].

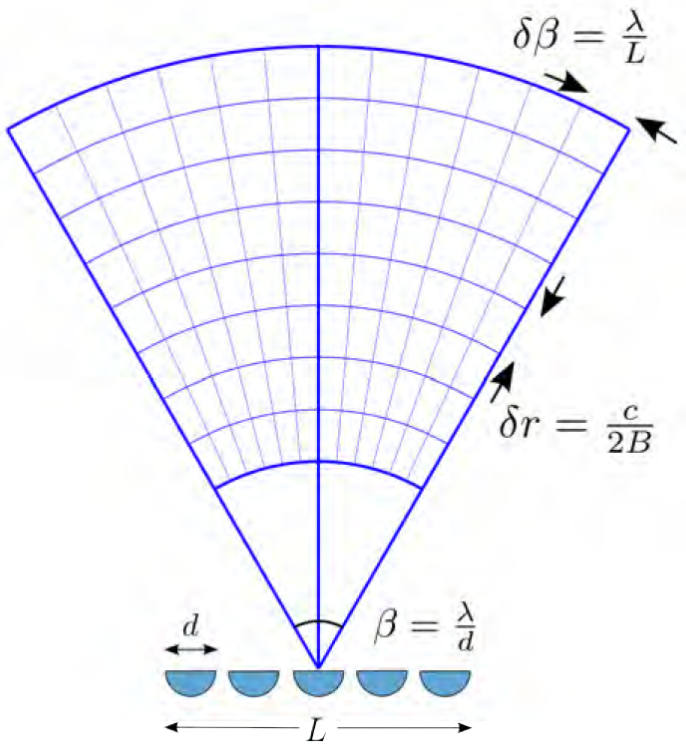


Figure 3.3: Imaging sonar [11].

It is then necessary to perform signal processing on the signal returning to the sonar before the image can be generated, where the unwanted external sounds, backscatter and other disturbances are filtered out. Multibeam sonars are used where they will be forward facing on the vehicle and send out a signal with a steady time delay with multiple beams; they are mostly used for sea floor mapping [11].

The reason for choosing sound waves for navigation, location and mapping underwater is mainly because they are one of the only usable sources that can transport information over large distances in water. For an example the radio frequencies from GPS and so forth are dampened completely by the water within a very short distance. There are still many things that can affect the sound waves and their quality while travelling as previously mentioned. These include salinity, depth, temperature, other sources, backscatter and many more. If the same energy is given from the source a sound wave can only travel a set number of wavelengths, therefore a lower frequency soundwave will travel farther than a high frequency one and a lower amplitude will also travel farther compared to a higher amplitude. The relationship between amplitude and frequency is therefore important for the wavelength characteristics [13].

3.1.2 Blueview P450-45-I

The sonar that is available is a Teledyne BlueView sonar, the P450-45-I which can be seen in figure 3.4 and specifications in table 3.1. It is a multibeam, 2D imaging sonar that has field of view of 45°. It has an operating range from 4m up to 175m, which can be adjusted based on the current need. It comes pre-supported and fully attachable to the VideoRay Pro 4 but requires a standalone program to run and view the data [14]. This particular sonar has quite a large operating range when used for object detection and tracking, which may cause some issues due to the minimum range of 4m. To communicate with the sonar the Ethernet cable is used on the Base station. A walkthrough guide for the setup is described in the manual of the sonar.

Figure 3.4: BlueView P450-45-I sonar [15].

Sonar	**P450-45-I**
Operating Frequency	450 kHz
Update Rate	Up to 12 Hz
Field-of-View	45deg
Max Range	300 m
Optimum Range	4 - 175 m
Beam Width	1deg x 10deg
Number of Beams	256
Beam Spacing	0.18deg
Range Resolution	2.0 in
Interface	
Supply Voltage	12 - 48 VDC
Power Consumption (24VDC)	30 W - 15W (Min - Max Range)
Connectivity	Ethernet/VDSL
Mechanical	
Weight in Air	5.7 lbs
Weight in Water	1.4 lbs
Depth Rating	1000 m
Size L x W (max OD)	9.6 x 6.9 in

Table 3.1: Table of sonar specifications [14].

3.2 VideoRay PRO 4

VideoRay is a producer of underwater ROVs and make small sized inspection class ROVs, which can be equipped with additional equipment such as a sonar, additional camera and/or a gripper. This project will be using the VideoRay PRO 4 as seen in figure 3.5. It is currently their newest commercially released ROV in the PRO series.

Figure 3.5: VideoRay Pro 4 with BlueView sonar [16].

3.2.1 Specifications

The following specifications are taken from the official VideoRay PRO 4 datasheet [17].

Physical Specifications

- Weighs 6.1kg (With Full Ballast)
- 37.5cm x 28.9cm x 22.3cm
- Buoyancy Controllable with Ballast

Performance Specifications

- Maximum Depth 305m
- Maximum Forward Speed 4.2 knots (2.16m/s)

Instrumentation

- Camera, High Resolution - NTSC Format.
- Lighting, 2 LED arrays 7200 lumen total
- 3D-Tilt Compensated Compass
- Leak Indicator
- System Voltage, Accelerometer, External Temperature
- Internal Temperature, Depth Sensor, MEMS Gyro

3.2.2 Additional Equipment

VideoRay provides different options for control setups, but the one available to us is in the form of pelican case equipped with a top-side computer, which can be seen in figure 3.6a. Specifically the control setup is called the VideoRay 4 PRO 4 plus BASE ROV system. The system comes with a sonar ready system, a powerful laptop for the cockpit, communication electronics and video adapters all secured in pelican case [17]. The system also comes with a hand held controller seen in figure 3.6b, which can be used in order to operate the vehicle from the top-side computer.

(a) VideoRay Plus Base [17].

(b) Included controller [18].

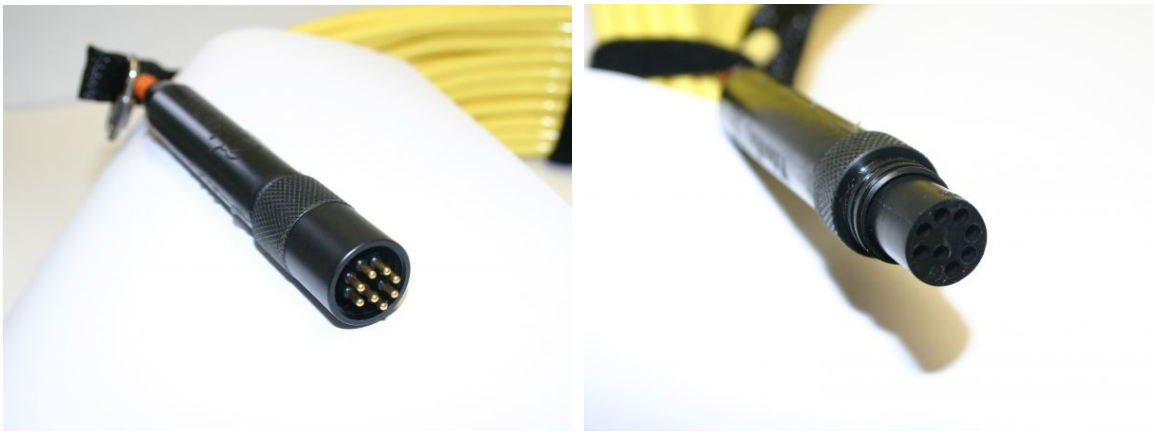
Figure 3.6: Base and handheld remote.

The system can provide the user with a set of environmental data, such as current heading using the compass, body angles from the gyroscope, depth based on pressure, linear and angular acceleration, internal humidity and temperature, external temperature and a video stream. Depending on the measurement quality the environmental data can be used to navigate and for feedback to control the ROV.

3.2.3 Tether

The tether VideoRay uses is rated up to 45kg and is neutrally buoyant. The system available includes two tethers, where one is a 75m long neutrally buoyant tether

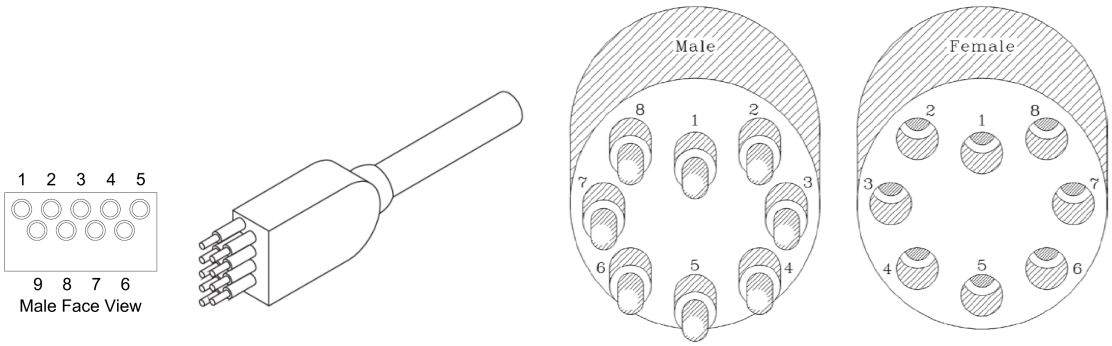
and the other one is a 40m performance neutrally buoyant tether. It is possible to upgrade to a Tether Deployment System (TDS), which means the tether comes with a management case designed to store and deploy the tether while keeping it protected and clean. The tether uses the RS-485 protocol to communicate with the ROV, and the top-side computer converts it to USB in order to communicate with the cockpit software. The tether functions as the communication line to the ROV and at the same time provides the power, which is in this case 74VDC. The tether can be seen in figure 3.7.



(a) Tether male end. **(b)** Tether female end.

Figure 3.7: VideoRay tether [19].

The pinout of the ROV tether can be seen in figure 3.8 and their functions in table 3.2. The tether uses Subconn connectors that are made specially for underwater connections. The VideoRay tether is made specifically for the VideoRay system and has the connectors required for the system with additional pins for an external plugin using the AUX pins.



(a) Subcon connector pinout [20]. **(b)** VideoRay tether pinout [21].

Figure 3.8: Motors and propellers.

Pin No.	Function	Pin No.	Function
1	Video −	6	AUX −
2	Video +	7	Internal CAN +
3	74 VDC +	8	Internal CAN −
4	AUX +	9	NC
5	Ground		

Table 3.2: Table of the tether pinout [21].

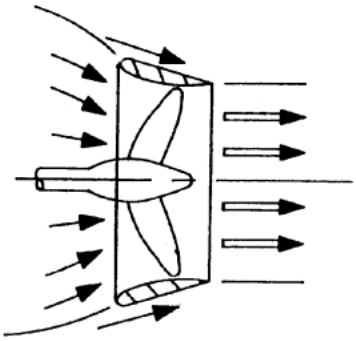
Pins 1 and 2 are used for the video feed of the main camera. Pins 3 and 5 provide the system with a 48 VDC from the topside to the ROV. Pins 4 and 6 are unused ports that the user can use for addons. Pins 7 and 8 are used for communication to the controls of the ROV.

3.2.4 Thrusters

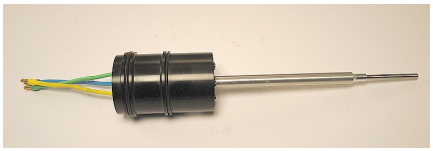
Figure 3.9: VideoRay thruster [22].

VideoRay uses their own custom made brushless thrusters. They are high RPM thrusters that provide 9.5kg thrust to a 4.5kg vehicle, which relates to a maximum speed of 2.1 m/s. The thrusters are shaped like torpedos beacause of their high RPM and speed, in order to increase their movement in water. During heavy duty use cycles the thrusters will generate a lot of heat, therefore they are fitted with an extended metal shaft inside the tubing that gives the motor more surface to cool down, and this can be seen in figure 3.10b. The propellers are 3 bladed and 90 mm in diameter, and the nozzle on the propellers is similar to a Kort Nozzle. The Kort Nozzle is shaped like a wing on a plane and uses the available fluid dynamics to make the water flow faster through the inner side of the nozzle. This results in saving energy needed for the propeller rotation, see figure 3.10a [23]. The nozzle

also functions as protection for the propeller blades, and this is beneficial when navigating unknown or hazardous enviroments [22] [24].



(a) Kort Nozzel [23].



(b) VideoRay Thruster shaft [25].

Figure 3.10: Kort nozzle and shaft.

3.3 Interfacing

In order to create and eventually implement the object detection and navigation system, it is necessary to communicate with and control the ROV. Using the SDK supplied by VideoRay, it is possible to create a custom control panel in order to interact with the ROV and implement the proposed control system. The SDK is for developing a .NET host, which is written in C#, and this means a program running on the topside computer. Online documentation of the namespaces and classes contained in the SDK (vrlib.dll) and simple examples are available on the VideoRay website [26] [27].

3.3.1 VideoRay Communication

By modifying the sample code provided, it is possible to establish a connection to the ROV from any given host computer. This requires the installation of a serial emulator driver that enables the USB compatibility since the control box requires the translation between USB and serial. Establishing a connection to the ROV is done completely automatically using the communication namespace, since the current configuration automatically scans the COM ports in order to detect the connected ROV. In order to ensure that the communication between the program and ROV is exited correctly whenever the program is ended or an error occurs, an event handler is created for the application exit, where the connection to ROV is terminated correctly. Figure 3.11 shows the main functions of the custom interface created for the ROV, with a specific emphasis on establishing a connection to the ROV and data logging.

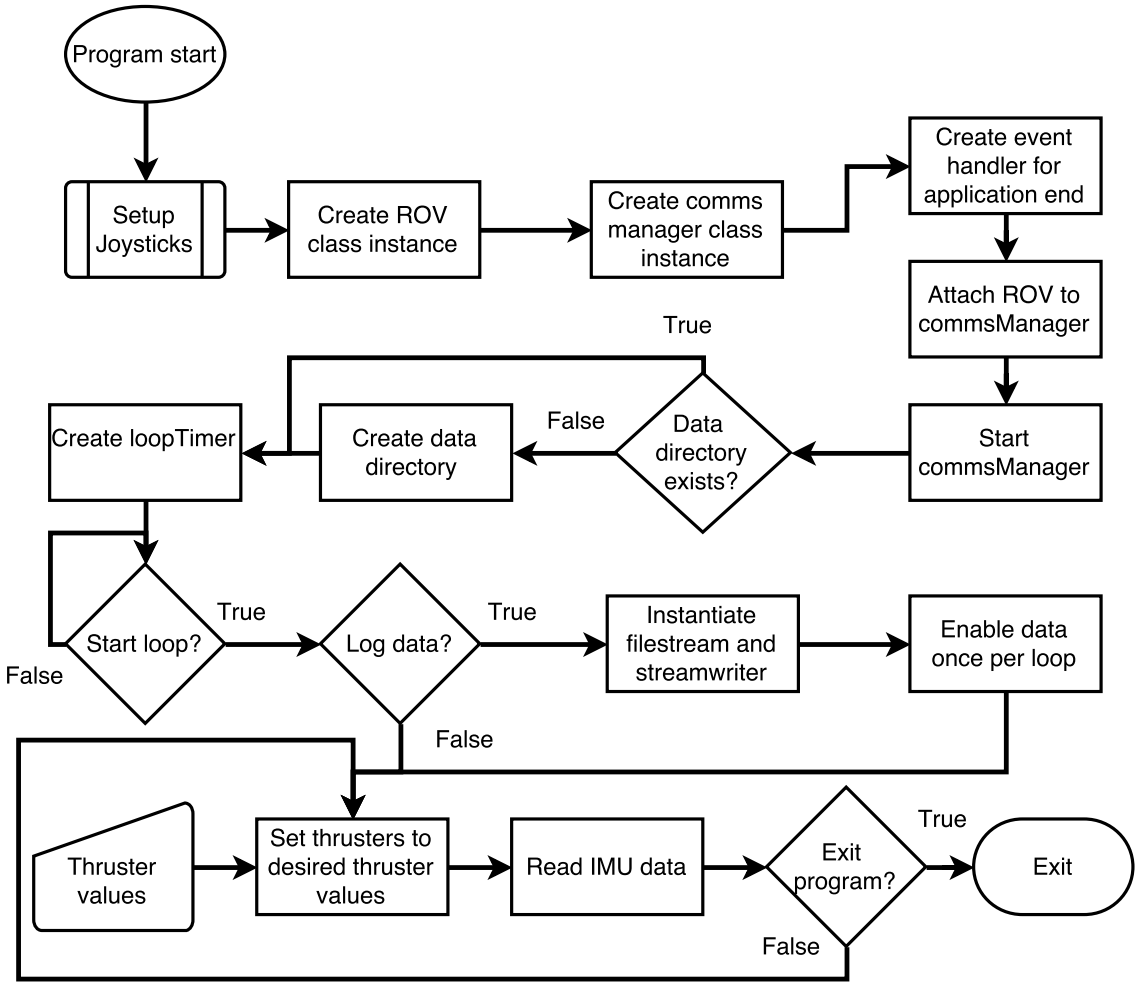


Figure 3.11: Flow chart for manual control of the ROV.

The main loop of the custom control panel features a timer with an adjustable period, in order to adjust the sampling period or operation period required during the given experiment or control scenario. During the parameter estimation for the mathematical model, individual control over the given thrusters and data logging is required. The relevant sensory data is accessed though the calling separate commands for each of the given available data, where the angle data needs to be reduced by a factor of 10. All of the interaction with the ROV occurs during the lower loop of flow chart from figure 3.11, where the loop iterates every set period and checks for a exit flag. Each time the loop performs an iteration the desired sensory data is read and logged, if enabled, to a text file in a Matlab compatible format, in order to be manipulated offline. The file handling is done using the filestream and streamwriter in C#.

There are two options for controlling the thrusters: each thruster can be accessed individually and actuated based on given inputs between the interval of -100 to 100, or the surge and yaw movement can be controlled using a single function call that couples the two rear thrusters and takes the input values for both the surge and yaw movement simultaneously.

3.3.2 Joystick Implementation

In order to maintain easy-to-use human control of the ROV, joystick input is also created made to emulate the functionality of the original cockpit. This is done using SlimDX [28], which is an open source framework used to create DirectX applications, where in this case the DirectInput namespace is used in order to read the analog values of the sticks and the boolean states of the buttons. Documentation and sample code is available on the SlimDX website and repository [29] [30]. The joystick of choice is the one shown in figure 3.6b and was included alongside the control box for the ROV.

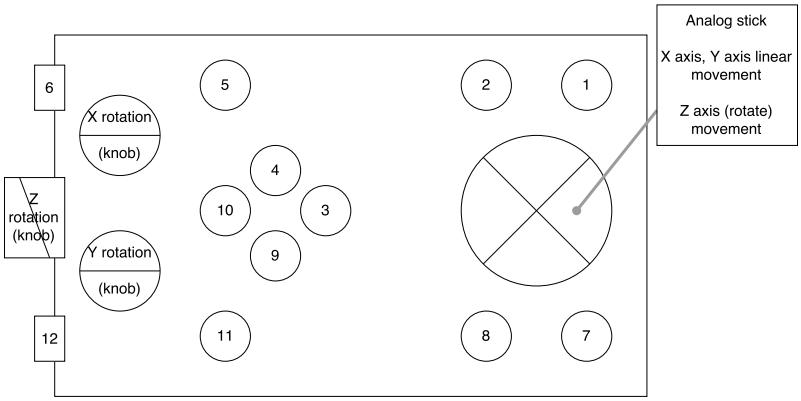


Figure 3.12: Input mapping of the joystick.

The joystick layout as recognized by Windows can be seen in figure 3.12. It has three knobs that can be rotated to alter xyz rotation values, 12 buttons with boolean values and a single stick with output as 3-dimensional Cartesian coordinates. The main objective of creating the joystick support is to map the stick to the surge and yaw movement and rotary input to the heave motion. This will allow the operator on the top side to position the ROV at a given location before initiating the given autonomous feature or experiment.

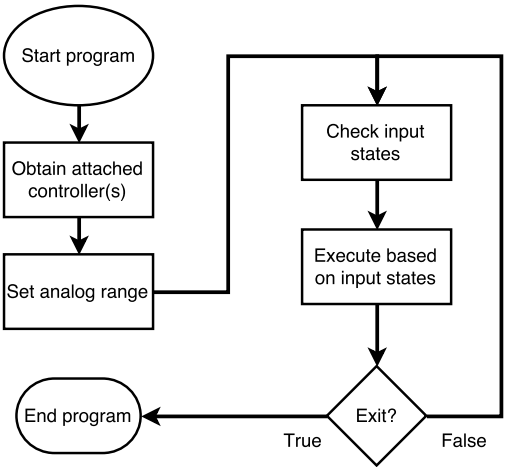
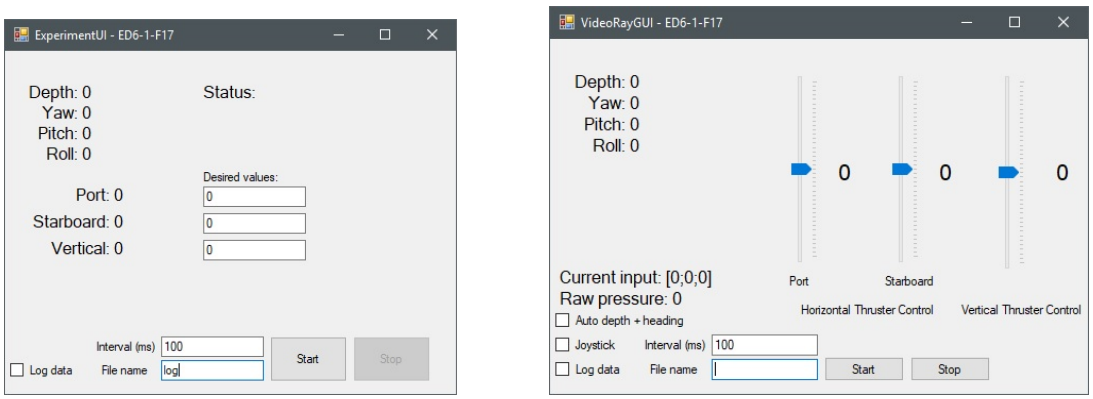


Figure 3.13: Flowchart for establishing connection to the joystick.

When the program is executed, it checks for available hardware devices and initializes them according to the desired value ranges for the inputs. Once the connected joysticks are initialized it is possible to poll the various states of the different input values and utilize them for the control of the ROV. In this case the x and y axis values from the stick are used to control the surge and yaw motion, whereas the z rotation is used for the heave motion. When interfacing with the buttons an array of boolean values is created in order to keep track of the different states. It is important to note that the button values are shifted in terms of naming, where button 1 on figure 3.12 is read using *buttons[0]*, due to the entry convention of the array.

3.3.3 Custom GUI



(a) Experiment UI. **(b)** Control UI.

Figure 3.14: Two custom interfaces created for the ROV.

In order to perform the desired experiments and to implement the final control solution, two interfaces for the ROV have been created, as shown in figure 3.14. Figure 3.14a is designed to be used during the parameter estimation where the thruster input values can be updated live during the experiments while logging the data at a specified interval. The interface displays relevant information from the IMU and other information such as the thruster values. Figure 3.14b is the main interface for the controller implementation, where the joystick input, controller or manual input can be selected as desired. This interface also has the capability of setting a custom interval, which in this case is for the entire control loop. The "current input" label displays the calculated control inputs from the system as they are applied to the system. If more sensory or calculated values information is needed the interfaces can easily be modified in order to display this information. The two interfaces also log the data with a specified delimiter which is compatible with Matlab. In order to retrieve the sensory information related to the object detection, a separate program will be utilised where the data will be broadcasted and used by the interface shown in figure 3.14b. This will be covered during the next chapter.

Chapter 4

Object Detection

4.1 Object Detection

The object to be identified will be of a known size and shape. For simplicity a red buoy will be used as the object to be identified. The edges of a square will be noticeable for the sonar and therefore a metal bottom frame was made to hold down the buoy. As such there are two shapes available to be identified: the red buoy and metal frame. For close range object detection using the camera will be necessary, since the Blueview P450 sonar does not work at a range under 4 meters. Therefore, a computer vision (CV) software can also be applied using the camera mounted on the ROV. Since the buoy is red, it will be easier to identify by using the camera, which will be sufficient as a proof of concept for object identification.

(a) Red buoy.

(b) Bottom frame.

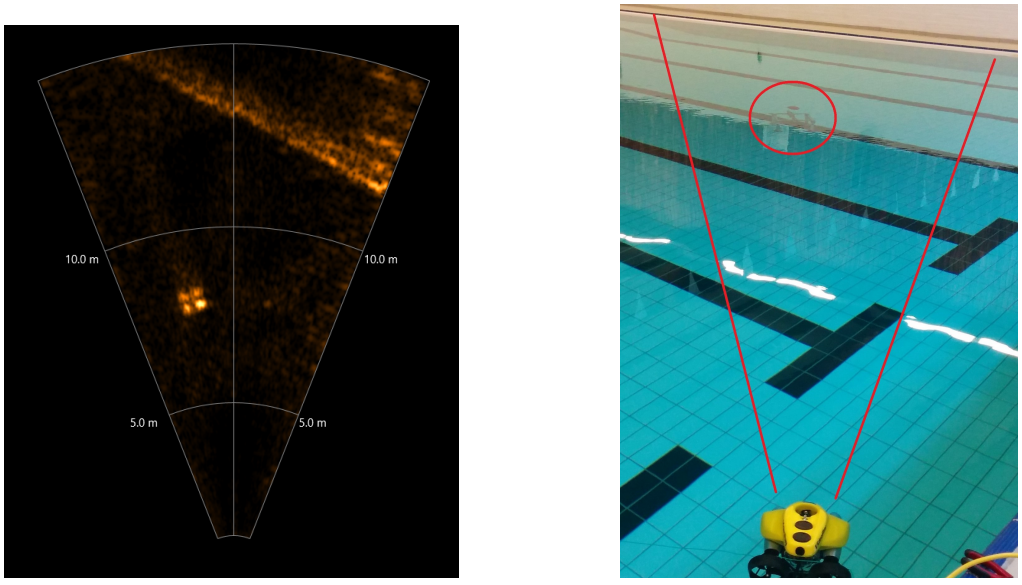
Figure 4.1: Object for detection.

A controlled testing facility will be used during the testing and implementation of

the sonar and computer vision system, where the pool size is required to be within the operating range of the sonar. The buoy used for the vision code and the frame built for the sonar test can be seen if figures 4.1. Imaging can be categorised as either passive or active. Active imaging is when the system provides its own energy to detect items, an example being sonar and radar. Passive imaging uses the energy provided naturally. By moving from using the sonar based system to a video based one we move from an active system to a passive one [31].

4.2 Sonar

Using sonar as the main vision has both advantages and disadvantages. The sonar is not affected by the lack of lighting in the water and has an extensive range. However it has a long minimum working range- in our specific sonar it is 4 meters. To be able to apply image processing on the data, either the data stream has to be intercepted or a custom software must be written using the SDK from BlueView. The SDK is not open-source, and, as stated by BlueView, they recommend an expert knowledge in programming due to the lack of documentation for the SDK. Therefore there are many steps in order to obtain reliable data from the sonar. In figure 4.2a the data generated by the BlueView sonar using the ProViewer software is shown where it detects the metal frame from the previously mentioned figure 4.1b. A top side picture can be seen in figure 4.2b where the buoy and the platform have been marked by a red circle.



(a) Sonar image using copper image filter. **(b)** Top side picture of the sonar setup.

Figure 4.2: Setup for sonar test.

After performing multiple experiments, the sonar returns accurate positional data, when the image is analysed by a human using the given software from BlueView.

The sonar also accurately calculates the range to the object, in this case it was placed 8 meters away, and this distance was confirmed using a measuring tape. Due to the limited time of the project and the scope, it was decided only to test the sonar and verify the retrieved data using the supplied software. Since retrieving the information from the sonar needed for the control system is currently unavailable, the focus will be shifted to creating a proof of concept using the built in forward facing camera and applying computer vision for object detection.

4.3 Computer Vision

To be able to use the analog video feed from the VideoRay base station, it is necessary to use a capture card to create the conversion to a digital picture. However, the capture card built into the base system failed due to a hardware fault and therefore another solution was needed. The system base station had an additional analog output that was used to supply the base station's screen with a live video feed. The video capture card used as an alternative was a Terratec G1 capture card; luckily this device supported the available analog signal and successfully converted it to a digital signal for processing on the computer.

Figure 4.3: Terratec G1 capture card [32].

In order to perform the computer vision it was decided to use the OpenCV library. The OpenCV library has more than 2500 optimized algorithms, including object detection, colour detection, movement tracking and so fourth. It has an active community with more than 47 thousand people and it is estimated to have been downloaded more than 14 million times. Since the VideoRay SDK uses C# it was decided to have the computer vision code running in the same language, in order for the for the sensory information retrieved from the computer vision to be compatible with the controller. OpenCV is built for C/C++, Java and Python [33]. To be able to use the OpenCV functionality, it was necessary to use EmguCV, which is a .Net wrapper that works on a cross platform, such as on C#, VB, etc. EmguCV allows the program to call OpenCV functions [34]. It is not possible to call all functions since

there is a difference in syntax between the given language and OpenCV. Therefore it was decided to use an older but stable version of the EmguCV, specifically the 3.0.0.alpha version, in order to ensure that the desired functionality was available.

As previously mentioned, for proof of concept a red buoy was chosen as the object to be detected. The reason for the colour is that there are typically not many red objects to be found in water, making it unique. Red also sits at the far left of the electromagnetic spectrum seen in figure 4.4. Since waves with the same energy but a bigger wavelength can travel further, as mentioned before in the sonar section 3.1, it is also easier to detect red at a distance. An added benefit of using a buoy its easily-detectable shape, which in this case is a circle. This makes the image processing and required function calls simpler when creating the implementation, since there are functions pre-written for circle detection.

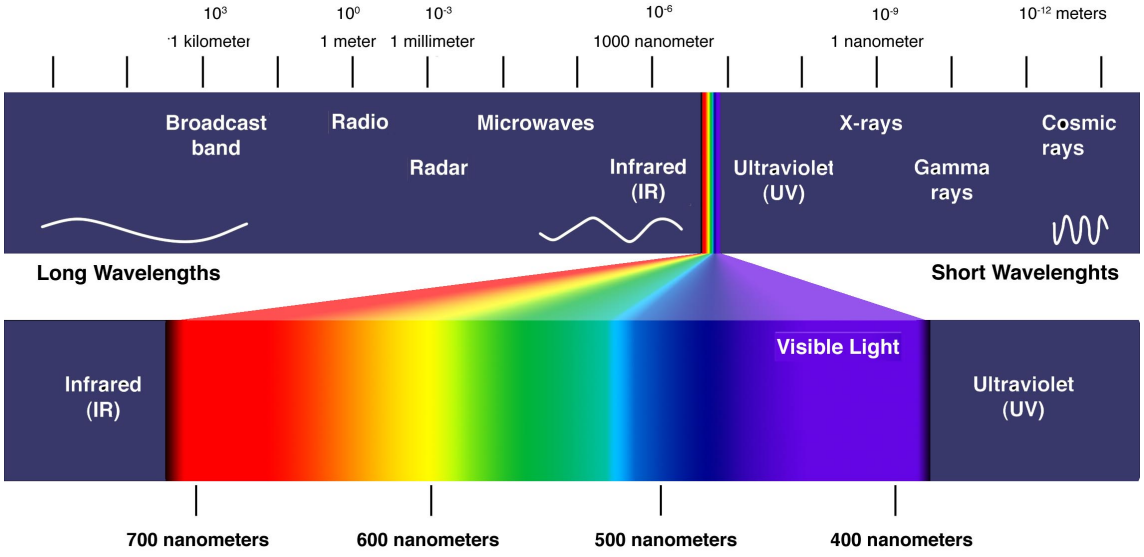


Figure 4.4: Electromagnetic spectrum [35].

Since the VideoRay only has a single camera and no external references attached to it, estimating the position of the detected object with respect to the ROV requires pre-defined knowledge of the object and pixel-to-distance relationship between the object and the ROV. In order for the single camera system to work, it is required to know the radius of the object being searched for and detected. If the number of pixels that the given radius occupies at a set distances is known, it is possible to make a curve fit of the recorded data for the relationship between pixels in the object's radius and the given distance from the ROV to the object, as seen in equation 4.1 and in figure 4.5, where d_u is distance to object in centimetres, r_{mp} is measured pixel radius of the object seen by the camera.

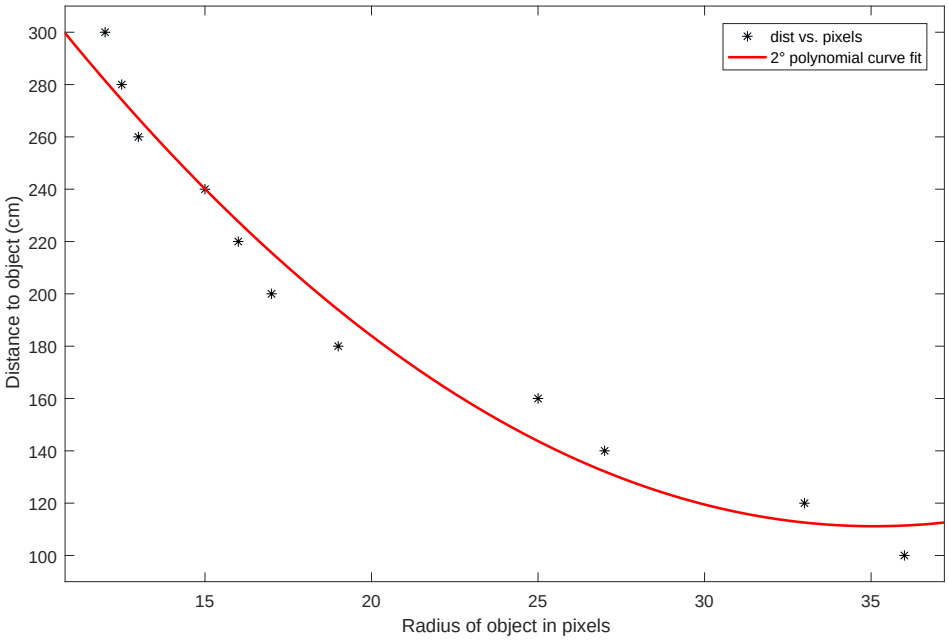


Figure 4.5: Pixel radius to distance in centimetres.

$$d_u = 0.3193{r_{mp}}^2 - 22.4126r_{mp} + 504.4667 \tag{4.1}$$

From equation 4.1 the measured pixels are inserted and the distance to the object from the ROV is obtained. Since the distance to the object is now known, it is possible to calculate the angle and difference in depth with respect to the body frame of the ROV. To obtain the angle the centre of the camera image is set to be the point (0,0), as in a Cartesian coordinate system. The pixel distance between the center of the image at (0,0) and the coordinate location of the off-set detected object, is the difference in coordinate values between the centre and the current coordinate of the detected object's center. That distance is then used alongside the distance from the ROV in order to obtain the hypotenuse in the formed triangle. The distance from the center of the image to detected object is then obtained using the Pythagorean theorem, shown in equation 4.2, where d_{cp} is distance to centre in pixels.

$$d_{cp} = \sqrt{x^2 + y^2} \tag{4.2}$$

Since equation 4.2 returns the distance in pixels, it has to be converted to centimetres. This conversion is done using equation 4.3, where d_c is the calculated distance to centre in centimetres and r_{ob} is the pre-measured physical radius of the given object.

$$d_c = \frac{d_{cp}}{r_{mp}} r_{ob} \tag{4.3}$$

Two of the sides of a triangle are now known in centimetres, these being d_c and d_u. It is then possible to calculate the angle between the forward facing direction and

to the object, shown in equation 4.4.

$$a_r = atan\left(\frac{d_c}{d_u}\right) \tag{4.4}$$

To get the depth of the object with respect to the object, d_w, similar methods are applied as previously mentioned. Compared to finding the heading, the distance in the y-coordinates are analysed from 0 or d_{wp}, divide that by the object radius in pixels or r_{mp} and multiple by the object's known radius or r_{ob} seen in equation 4.5.

$$d_w = \frac{d_{wp}}{r_{mp}} r_{ob} \tag{4.5}$$

In figures 4.6 the distance, depth and angle can be visualized. The distance to centre from the object is denoted as d_c, the distance to the object from the ROV as d_u, the final distance d_o is not used in our case since only one camera is available and the polynomial made for the distance will always assume that the distance seen on the camera is the same as looking at it from the centre of the camera. d_w is the depth in relation to the ROV so it can be either positive or negative and a_r is the angle between the ROV and the object.

(a) Depth distance. **(b)** Object distance and angle.

Figure 4.6: Distances and angles with respect to the ROV.

4.3.1 Computer Vision Code

The flowchart for the main loop of the created computer vision program can be seen in figure 4.7, where the process of the detecting and processing the images are explained.

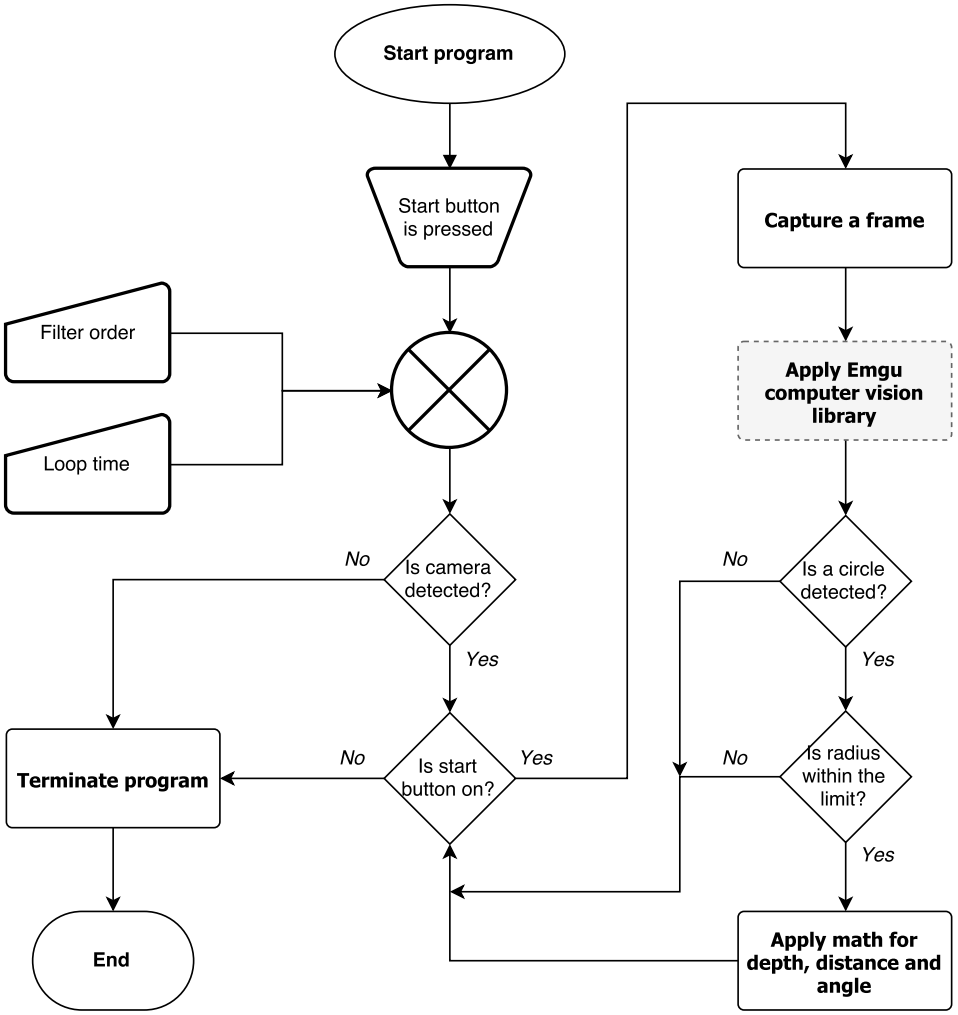


Figure 4.7: Flowchart of the main vision code.

In order to implement the object detection and distance measurements both EmguCV library functions and relevant math functions are used to calculate or obtain the desired values. The actual size of the object, in this case the buoy, limits the detection range based on the size of its radius. The size of the radius determines the maximum distance from which the object can be seen, since the larger the buoy, the greater the distance from which it is measurably visible by the computer. A larger buoy will also have a greater minimum distance from which it can be detected because a larger buoy will cover the entire image at a greater distance compared to a smaller buoy. With the given buoy for this project, the minimum working distance was 1m and the maximum 3m, with its radius being 10 cm.

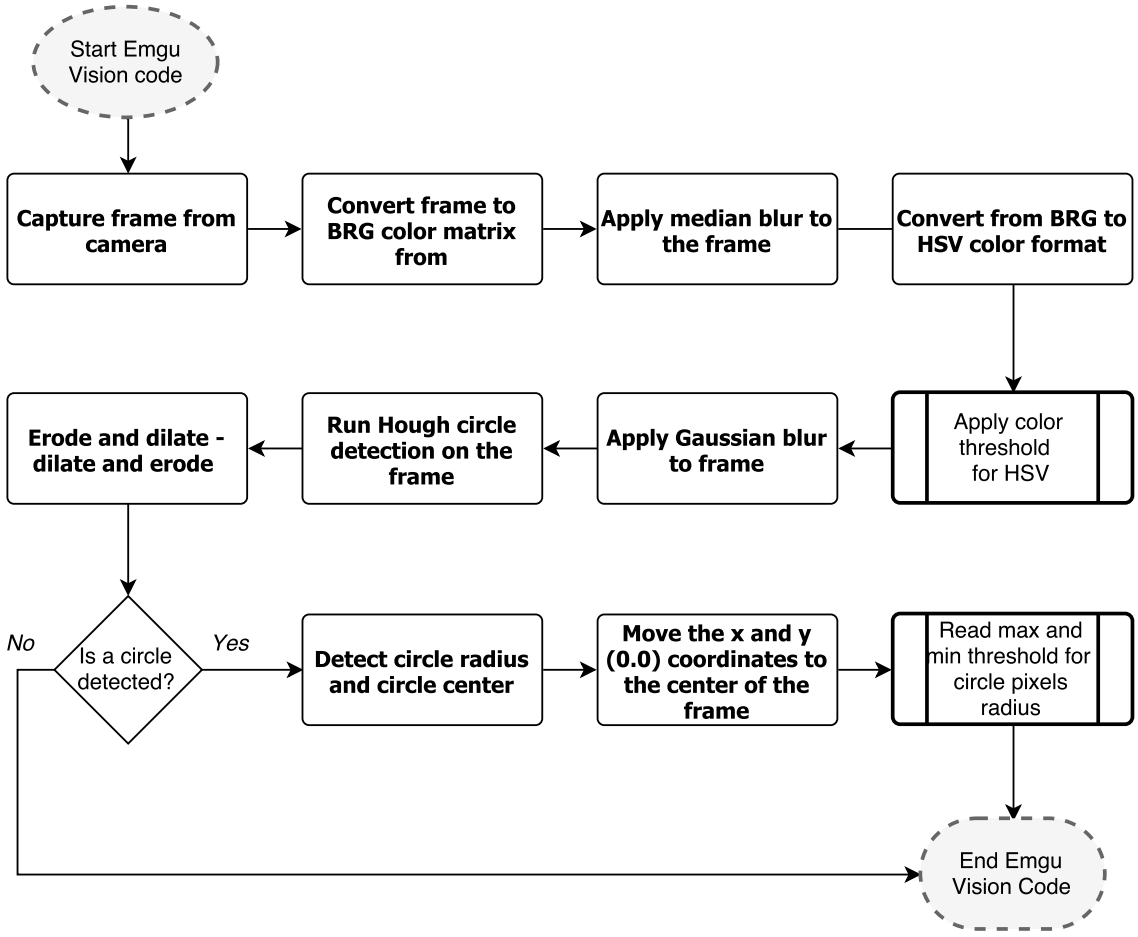
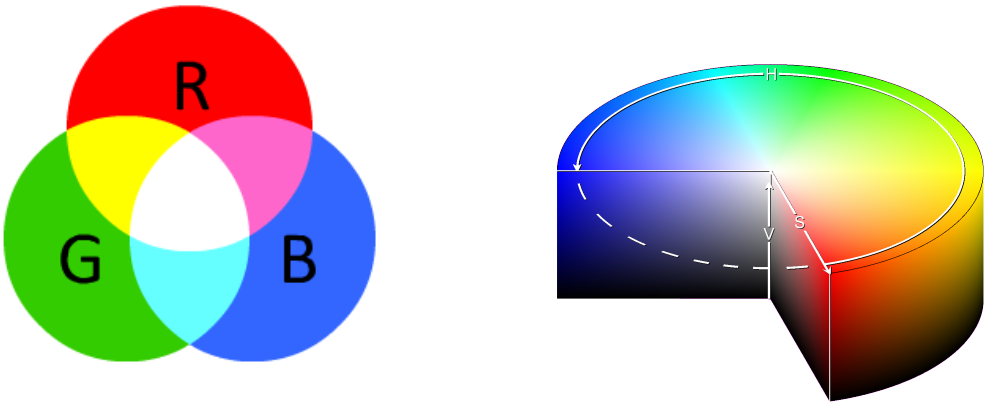


Figure 4.8: Flowchart of Emgu vision code.

The flowchart for the grey sub-block from figure 4.7, for applying the Emgu computer vision library, can be seen in figure 4.8. Built-in functions for image processing that the Emgu library offers are utilized in order to process the captured images. The code starts by checking if there is a usable camera for the software. If so, the program will start. It will continously capture each single frame of the video and store the images in a three dimensional matrix. The first frame is stored in a BRG format (blue, red and green). The reason for OpenCV to use BRG instead of RGB is just because camera manufacturers traditionally used BRG instead of RGB [36]. The programme then converts the frame in the BRG colour spectrum to the HSV spectrum (hue, saturation and value). The reason to shift the color spectrum from BRG to HSV can be seen in figure 4.9. HSV separates the image intensity from the color information. That is very important to be able to separate the colors in a given picture and not be affected by shadow or certain lighting conditions.



(a) BRG color spectrum [37]. **(b)** HSV cylinder [38].

Figure 4.9: BRG and HSV color spectrum.

Also when using the HSV format it is only required to set one of the values, which in this case is setting the hue to the desired colour, which is a certain shade of red. The hue values for the different shades of red are shown in figure 4.10. In EmguCV and OpenCV the implementation of the HSV cylinder has the hue range from 0 to 180, where saturation and value have a range from 0 to 255. The colour red sits in between the values of 0-10 and 160-180 in the hue range. During the dry testing, meaning identifying the buoy on land, the settings for the HSV values were the following: hue 155-8, saturation 33-255 and value 128-159. Changing the saturation and the value settings intensifies the change seen on the screen. Else there would be a perfectly circular buoy visible in the HSV threshold frame but for the sake of showing how different functions help to detect circles, the settings where changed to only show a partial circle. The buoy as seen through the camera and in HSV format is shown in figure 4.11.

Figure 4.10: Hue of the HSV [38].

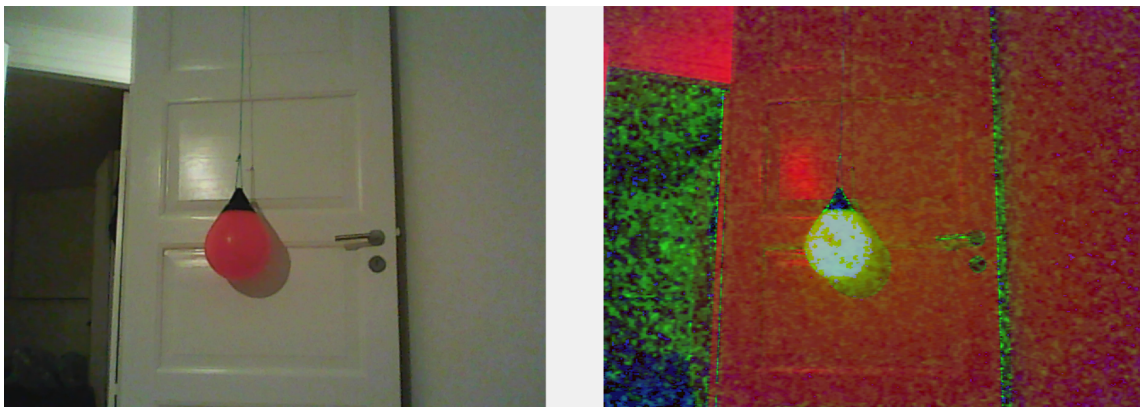
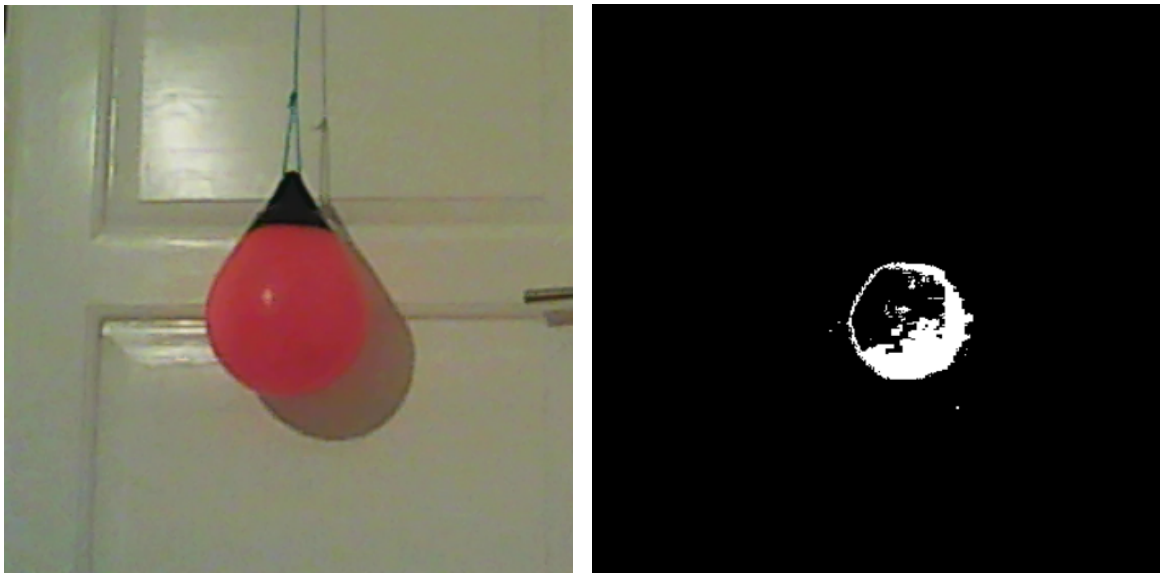


Figure 4.11: The buoy seen through the web camera and in the HSV spectrum.

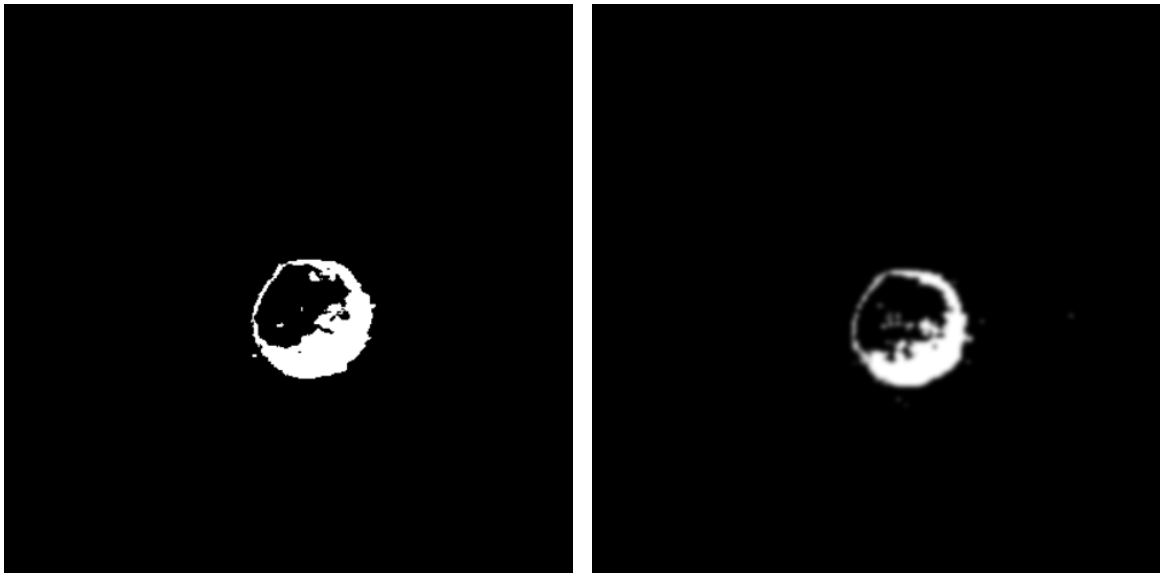
The next step is to apply a threshold function to the now HSV image using the given EmguCV functions. The threshold function makes all the elements in the picture that are not within the desired values appear black and the ones that are within the threshold white, this can be seen figure 4.12b where it has been applied to the red colour of the buoy. Since the colour red is at both ends of the hue value spectrum, a workaround is made by creating three separate matrices. The first one stores the lower limits of the HSV threshold, the second one the higher limits of the HSV threshold and the third one is used to check if the original image is within either one of those thresholds. If they are then a 1 is written in the third matrix for that pixel and it will show as white pixel, but if it is outside of the limit a 0 is put in it and it will be a black pixel. This will make a matrix that has only black and white pixels. The threshold limits can be extended to include a weighting scheme if necessary. A threshold frame looking at the buoy can be seen in figure 4.12b and the raw image in figure 4.12a; here the colour spectrum has been decreased so it is easier to see the changes made by different functions built into EmguCV.



(a) The raw buoy image. **(b)** Applying the threshold function.

Figure 4.12: Change from original capture to HSV.

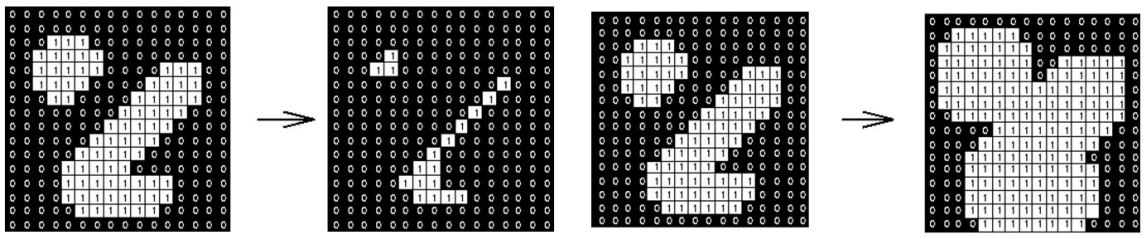
To get rid of false positives and sharpen edges, blur effects are applied. In this case a Median blur is applied to the original image and a Gaussian blur is applied to the threshold frame. Different intensities of the Gaussian blurring can be set in the built in function. A comparison between the different blurring can be seen in figures 4.13. During the implementation both of the blurring effects are applied.



(a) Median blur applied. **(b)** Gaussian 9 blur applied.

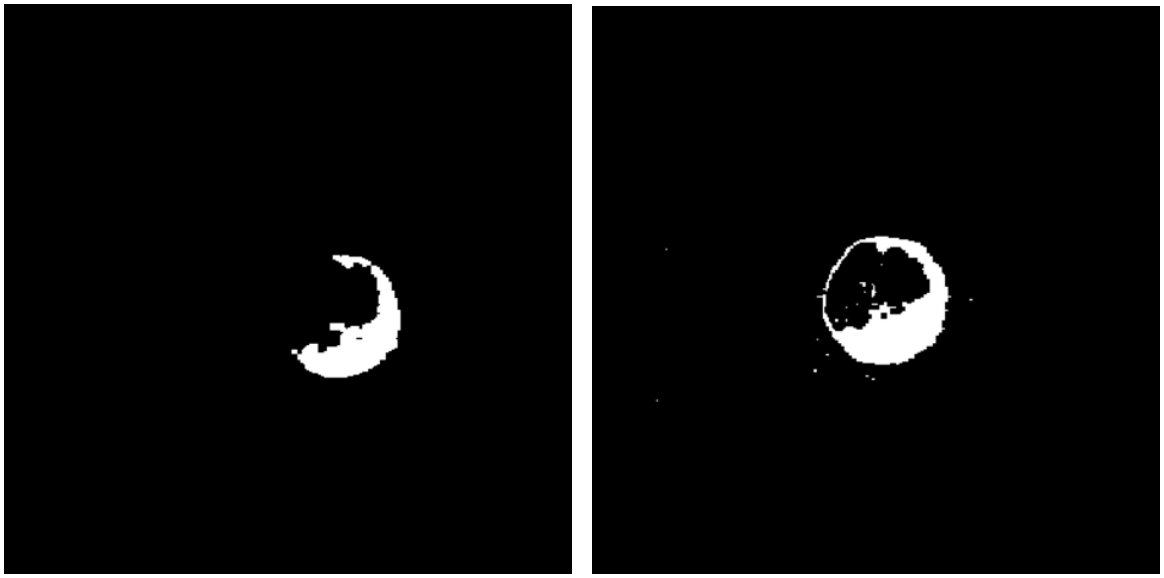
Figure 4.13: Comparison of blurring.

Combining the blurs with the set thresholds and weightings, it is possible to obtain the desired red color and create a smoother circle on the image, which is crucial when attempting to identify circular objects. This is done by applying mathematical morphology on the threshold image, which in this case are only erosion and dilation. Both erosion and dilation will take in the binary image, this being the matrix containing the pixels data in the form of 1's and 0's, and a structuring element. Our structing element is just adding a single pixle to the limits of the image that is within the threshold and applying them together. When applying erosion the outer limits of the set of pixels will be erased and in dilation the opposite will be done, where pixels will be added to the limits. [39]. A comparison between these two functions can be seen in 4.14. First erosion is used and then dilation to remove small objects in the background, then dilation is applied and erosion to fill in small holes in the foreground. Comparison between the morphs can be seen in figure 4.15b. Both morphs are applied during the implementation.



(a) Erosion [40] **(b)** Dilation [41]

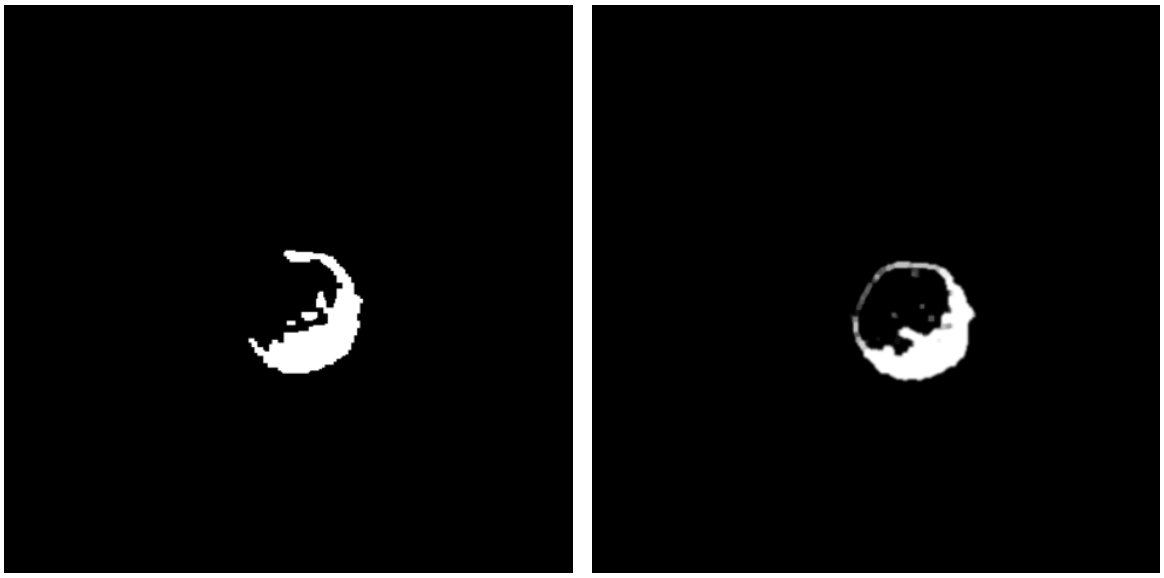
Figure 4.14: Mathematical Morphology.



(a) Erosion then dilations. **(b)** Dilation then erosions.

Figure 4.15: Morphology.

A comparison of using the full morphology method and then also applying both the blurring effects on the image can be seen in figure 4.16.



(a) Applying full morphology. **(b)** Applying blurring and morphology.

Figure 4.16: Comparison of the full setup.

In order to detect the circles in the frame a pre built function is called, which is named Hough transform circle. The benefits of the Hough transform circle is that it does not need to see a full circle in order to detect one. So, if the image only contains a partial circle because of shadows or other external factors, the Hough transform will still be able to detect it. Hough transform circle also uses edge detection that

finds all edges in the frame. That is done by using mathematical morphology where either dilate or erosion is used, then the new image is compared to the old one. The new pixels, or the difference between the frames, will show the edges of the objects in the frame. From that picture the Hough transform circle will check if the object detected fits the criteria of having a certain radius, centre point and a set angle between dots. It helps if the radius is known, but if not the Hough transform will start with radius 1 and work up to the upper limit set by the user [42]. The Hough transform circles function will return the number of circles found within the given frame, the radius of the circles and their centre points in terms of x and y coordinates. There will be a set limitation in the code for the maximum and minimum distance that the buoy is detected. If the circles are outside those limitations then they will be ignored. If the circle is within the specified limits, the math required to compute the distance between the buoy and ROV will be applied as demonstrated previously, using the relevant trigonometry operations and the returned coordinate values. Due to uncertainties, the circle radius can fluctuate in size between the captured frames so a moving average filter is implemented. The order is adjustable through the created GUI, but a minimum order of 4 is recommended in order to obtain suitable averaging. The moving average filter then stores the last n-order of measurements in an array, sums all the array entries and divides by the order. The moving average filter shifts the data in the array and adds the newest recorded value to the front. The GUI made for the computer vision setup, where the settings for the HSV have been set to detect the full buoy and all of the previous methods are applied, can be seen in figure 4.17.

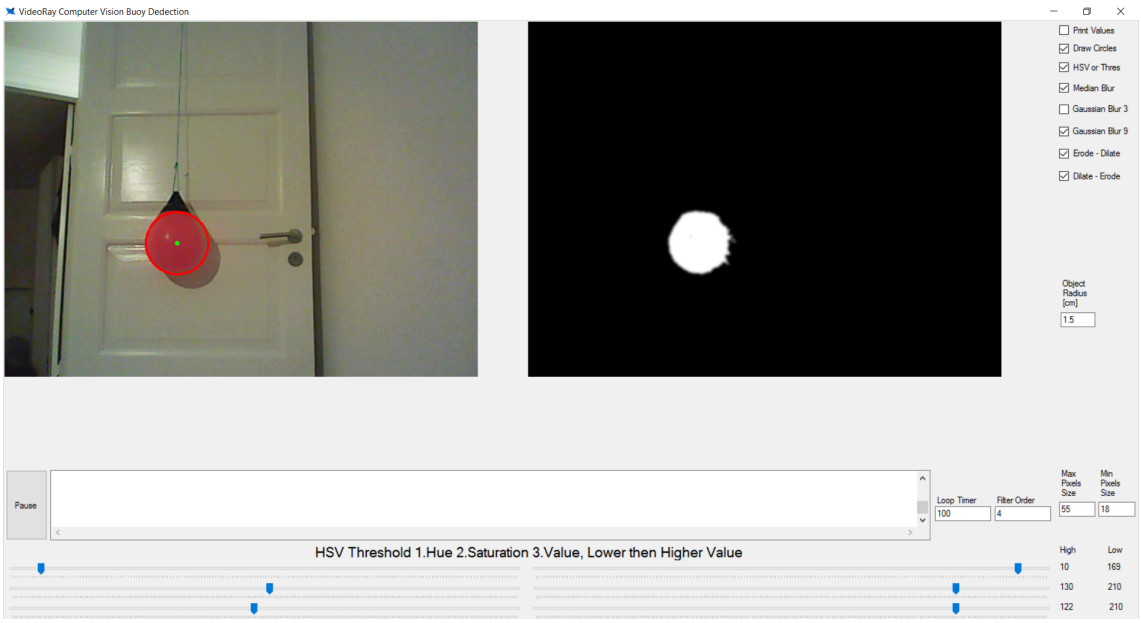


Figure 4.17: Screenshot of the vision code GUI.

4.3.2 WCF Service

Since the current implementation runs using two seperate programs, one for the interaction with the ROV and one for the computer vision, a method was needed to transfer data from the computer vision program into the controller program. This is done using the Windows Communication Foundation (WCF). WCF is a service-oriented architecture (SOA), meaning that when needed to either send or retrieve data the service can be used and the user will be called the client of the service. The service can be called asynchronously or synchronously. The benefits of using it asynchronously is that the service is not locked up when one program is using it, which would be the case if it is called synchronously. Messages can be sent from one service endpoint to another or can be hosted on the server. The service stores the functions on what will be called and applied to the data sent to the server from the client. In our case the computer vision program sends information about the angle, distance and depth from the object with respect to the ROV. The data is stored as a double on the server and when the controller requires an updated value it can call for the data. This is just one possible approach in order to get the data from one program to the other. The data could be sent from one client to the other without having to be stored at the server side. The reason for doing it this way is the difference in how fast the programs run, since the controller for the ROV runs at a set frequency but the computer vision program runs at its maximum capabilities. A benefit of using WCF is that it allows the two programs to be run on seperate computers if necessary. The WCF can be run locally and also over the web [43].

Chapter 5

Mathematical Modelling

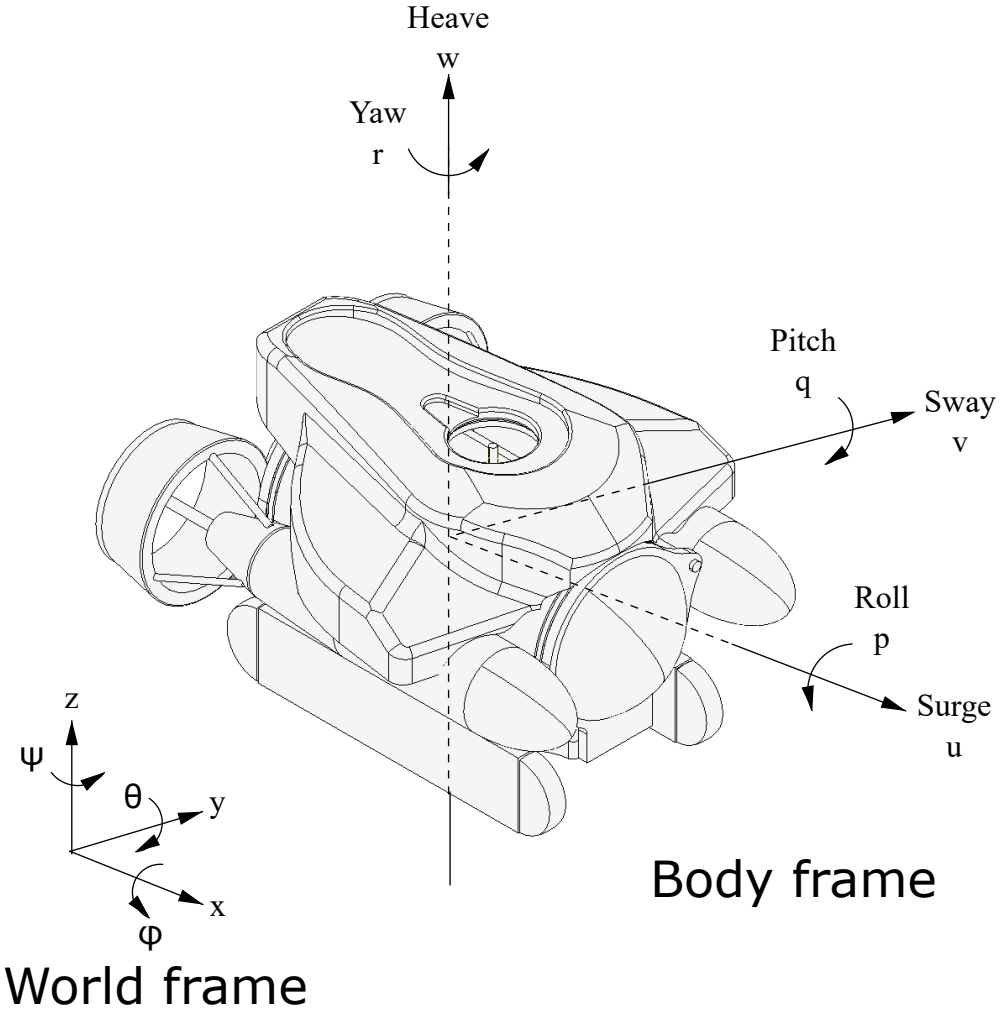


Figure 5.1: The ROV's reference frames and notation [8].

5.1 Thruster Dynamics

The three thrusters create the desired thrust forces and torques required for movement and directional control. Depending on their placement and configuration, in terms of propeller blades and general thruster characteristics, the movement of the ROV is defined. In this case the ROV is directly controllable in the surge, heave and yaw direction. The ROV's thrusters uses three fixed pitch propellers, where the generated thrust is dependant on the advance speed u_a and propeller rate n. This can be described as follows in equation 5.1 [44].

$$F(u_a, n) = \rho D^4(\alpha_1 + \alpha_2 \frac{u_a}{nD})n|n| \tag{5.1}$$

Here D is the propeller diameter, ρ is the water density and α_1, α_2 are constants based on the given propeller characteristics. By neglecting the advance speed and assuming a linear relationship between the propeller rate and control input, equation 5.2 is obtained. [8]

$$F_t(u_t) = a_t|u_t|u_t + b_t u_t \tag{5.2}$$

The given input thruster percentages will be denoted as u_t for the top thruster percentage, u_r for the combined rear thruster actuation and u_y for the combined yaw thruster actuation.

5.2 Hydrodynamic Drag

When a vehicle is in motion under water it experiences some hydrodynamic drag forces. A possible way of modelling this drag force for linear motion is shown in equation 5.3, where $\boldsymbol{X}$ denotes surge motion with its associated velocity u.

$$\boldsymbol{X}_{u|u|}(u|u|) = 0.5\rho C_d u^2 A \tag{5.3}$$

The behaviour of the drag in equation 5.3 is determined by the characteristics of the moving body but also the environment. The environment dependant coefficients are ρ, which is the temperature dependant fluid density, C_d which is the drag coefficient calculated based on the shape of the vehicle and A which is the surface area. Due to the formulation of equation 5.3, it purely models the drag force as quadratic, but at lower speeds the drag has linear behaviour. Also, when linearising equation 5.3 around and operating point of 0, the whole term becomes zero. Therefore, for low enough velocities the the drag force can be modelled based on the second and third terms of the Taylor expansion, which results in having seperate linear and quadratic quantities [44] [45]. Since the drag force is 0 when the velocity is 0 the first term of the Taylor series is neglected.

$$F_d^u = a_d u^2 + b_d u \tag{5.4}$$

Equation 5.4 works since it then captures the desired behaviour of the hydrodynamic drag force as mentioned previously with the linear behaviour at lower velocities,

where the notation F_d^u means the drag force for the surge motion. When linearised this form also represents the drag as a linear function, compared to resulting in 0, with an operating point of 0. The coefficients a_d and b_d in equation 5.4 contain similar environmental characteristics as described in equation 5.3, but in this case the individual characteristics are inseparable from each other and are estimated as a single coefficient.

$$F_d^u = a_d u|u| + b_d u \tag{5.5}$$

In order to utilize equation 5.4 for bi-directional movement, the squared velocity must be modified in order to take the sign of the velocity into consideration seen in equation 5.5.

5.3 Buoyancy

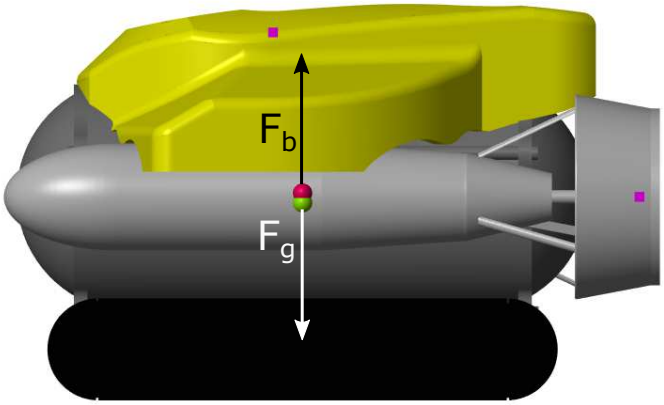


Figure 5.2: Free body diagram for the buoyancy [8].

The buoyancy force describes the ROVs ability to float in the water. This force depends on the volume of the object and also the density of the medium, as shown in equation 5.6.

$$F_b = \rho V g \tag{5.6}$$

Besides the buoyancy force, the gravitational force also acts on the ROV and this force is as follows in equation 5.7, where it consists of the gravitational constant and the mass of the ROV.

$$F_g = m_{rov} g \tag{5.7}$$

The placement of these forces can be seen on figure 5.2, where they are shown aligned meaning that in an ideal steady state the ROV will have a pitch angle q equal to 0. If there is misalignment between the center of buoyancy and center of gravity, it will create instability in terms of non-zero pitch, roll and yaw angles.

$$\rho V g = m_{rov} g \tag{5.8}$$

By having the buoyancy equal to the gravitational force, as shown in equation 5.8, the ROV is neutrally buoyant. Neutral buoyancy allows the ROV to stay at certain depths without continuously actuating the thrusters.

5.4 Surge and Heave Motion

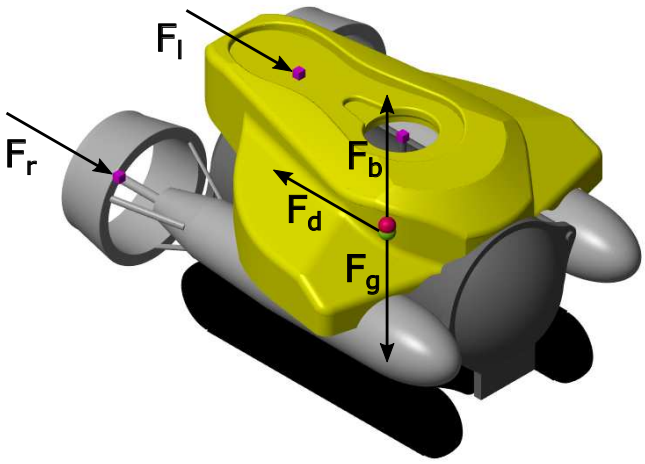


Figure 5.3: Free body diagram for the surge motion [8].

Due to the placement of the thrusters, the ROV is unable to directly create sway movement. Using the available thrusters it is possible to obtain an expression for the surge and heave motion.

$$\boldsymbol{X} = F_r + F_l - F_d^u + f(r, q, p) \tag{5.9}$$

Equation 5.9 shows the expression for the surge motion, where the two rear thrusters are utilized in order to create backwards and forwards motion. The term F_d^u is the velocity dependant drag force mentioned previously in equation 5.4 and 5.5, where the superscript u denotes that its is dependant on the surge directional velocity. The added term $f(r, q, p)$ captures the induced angular changes in the body frame due to surge movement. By assuming that the ROV is neutrally buoyant, the effects of the gravitational force and buoyancy force can be neglected.

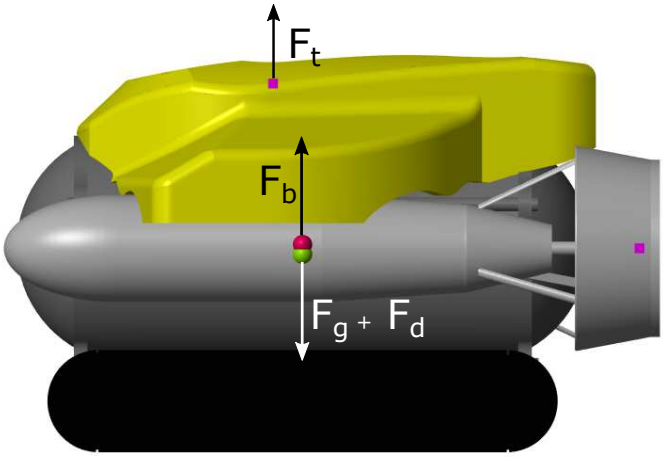


Figure 5.4: Free body diagram for the heave motion [8].

Similar to the surge motion, the expression for the heave motion uses instead the top thruster and is influenced by a velocity dependant drag in the heave direction, this is shown in equation 5.10. This equation for the heave motion assumes neutral buoyancy and therefore the gravitation force and buoyancy forces are neglected. Positive buoyancy may be advantageous for some controller implementations and can be modelled by included the resulting force from equation 5.8 when it is not equal to 0.

$$\boldsymbol{Z} = F_t - F_d^w + f(r, q, p) \tag{5.10}$$

The forces and moments described in table 1.1 can be denoted in terms of their respective double derivatives and either the mass moment of inertia or the mass, based on Newtons second law of motion.

$$\boldsymbol{X} = m_{rov}\dot{u} \tag{5.11a}$$
$$\boldsymbol{Z} = m_{rov}\dot{w} \tag{5.11b}$$

Then by substituting the expressions from equation 5.11 into equations 5.9 and 5.10 respectively, the following is obtained in equations 5.12 and 5.13.

$$m_{rov}\dot{u} = F_r + F_l - F_d^u + f(r, q, p) \tag{5.12}$$

$$m_{rov}\dot{w} = F_t - F_d^w + f(r, q, p) \tag{5.13}$$

Finally by inserting the known expression for the drag with respect to the two linear velocities and setting the induced angle changes equal to 0, two differential equations are obtained for surge and heave motion respectively.

$$m_{rov}\dot{w} = F_t - (a_d w|w| + b_d w) \tag{5.14}$$

$$m_{rov}\dot{u} = F_r + F_l - (a_d u|u| + b_d u) \tag{5.15}$$

The coupling between the linear movements and the angular changes in the body frame are neglected due to being insignificant at lower velocities; this will be shown

experimentally later in the report. Substituting in the mass of the ROV, which is 6.6kg, and isolating the acceleration term, the equations for heave and surge become the following in equations 5.16 and 5.17.

$$\dot{u} = 0.152(F_r + F_l) - 0.152(a_d u|u| + b_d u) \tag{5.16}$$

$$\dot{w} = 0.152F_t - 0.152(a_d w|w| + b_d w) \tag{5.17}$$

5.5 Yaw Motion

Besides the surge and heave motion, the ROV is able to directly control its yaw movement. The torque generated by the two rear thrusters, F_r and F_l, is dependant on their placement with respect to the centre of mass.

$$\boldsymbol{N} = \tau_{thrusters} - \tau_{drag} + f(q, p) \tag{5.18}$$

Equation 5.18 shows the generated moment from the yaw movement, which depends on the torque generation from the rear thrusters but also the rotational drag. The included term $f(q, p)$ is a change in roll or pitch induced from the yaw movement. Similar to the linear movement the notation for the moment can be substituted with its respective double derivative and the inertia of the ROV.

$$\boldsymbol{N} = I_r \dot{r} \tag{5.19}$$

Then by substituting the new expression for the moment in equation 5.19 into equation 5.20, an expression which depends on the moment of inertia and angular acceleration is formed.

$$I_r \dot{r} = \tau_{thrusters} - \tau_{drag} + f(q, p) \tag{5.20}$$

The moment of inertia can be approximated as a solid cuboid, which is based on a known equation shown in 5.21. W and L are the width and length of the ROV,and by using the dimensions specified in the VideoRay's datasheet the moment of inertia is calculated [17].

$$I_r = \frac{m_{rov}}{12}(L^2 + W^2) = 0.105 \tag{5.21}$$

Similar to the drag force for the translational motion shown in equation 5.5, the rotational drag motion is approximated using the same format. By substituting the expression for drag into the equation and the torque generated by the thruster, the following in equation 5.22 is obtained. As mentioned previously, the angular changes induced due to coupling are neglected.

$$\dot{r} = \frac{\tau_{thrusters}}{I_r} - \frac{(a_d r|r| + b_d r)}{I_r} \tag{5.22}$$

The torque generated by the thrusters depends on the distance from and their angle with respect to the center mass, since this is required in order to convert a force to a torque. In order to calculate the torque, equation 5.23 is used, where R is the

radius from the center of rotation, F_l and F_r are the applied forces and γ is the angle between the force and the line towards the center of rotation.

$$\tau_{thrusters} = F_l R sin(\gamma) + F_r R sin(-\gamma) \tag{5.23}$$

In order to create rotation using both thrusters they must be powered in separate directions, and in order to avoid the torques cancelling out from giving the one thruster a negative input and the other a positive, the one angle must be negative.

$$\dot{r} = \frac{F_l R sin(\gamma) + F_r R sin(-\gamma)}{I_r} - \frac{(a_d r|r| + b_d r)}{I_r} \tag{5.24}$$

Substituting the force to torque translation from equation 5.22 into equation 5.23, the final description for the yaw movement can be found in equation 5.24. Using the center of rotation specified in [8] based on a cad model, the γ is calculated to be 0.423rad and R to be 0.22m.

$$\dot{r} = \frac{1}{0.105}(0.0903(F_l - F_r) - (a_d r|r| + b_d r)) \tag{5.25}$$

5.6 Further Considerations

The desired outputs of the mathematical model depend on the current state or purpose of the developed controller. When controlling the depth, the measured depth from the pressure sensor will always be with respect to the world frame, due to the insignificant measurement errors introduced if the ROV is slightly pitched or rolled. Therefore the measured depth will be equal to the world frame coordinate z. By assuming that the ROV has no significant pitching or rolling when navigating in the plane, the angle measured by the compass is the correct heading angle with respect to the euler angle ψ in the world. When navigating with respect to a detected object, the distance towards the object in the surge direction and the angle between the current heading and the object, will be specified in terms of the integration of the body frame linear and angular velocities as follows in equation 5.26.

$$\int u = d_u, \quad \int w = d_w, \quad \int r = a_r \tag{5.26}$$

The distance d_u is the distance towards the object in the surge direction, d_w the depth compared to the depth of the object and a_r the angle between the detected object and the ROV. The aforementioned distances and angles can seen in figures 5.5, 5.6 and 5.7 where they are explained using the notation from figure 5.1.

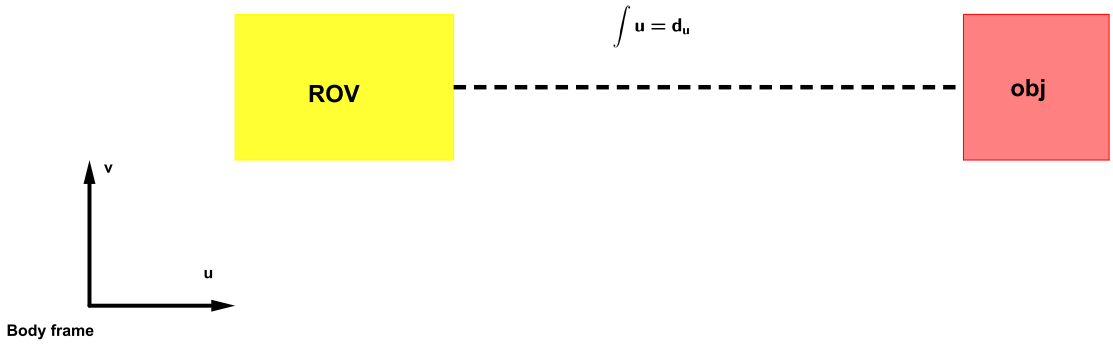


Figure 5.5: Distance from the ROV to an object.

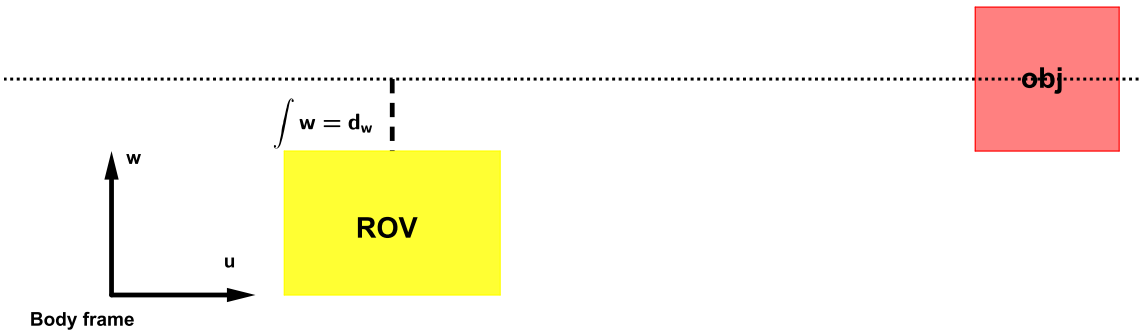


Figure 5.6: Depth difference between the ROV and an object.

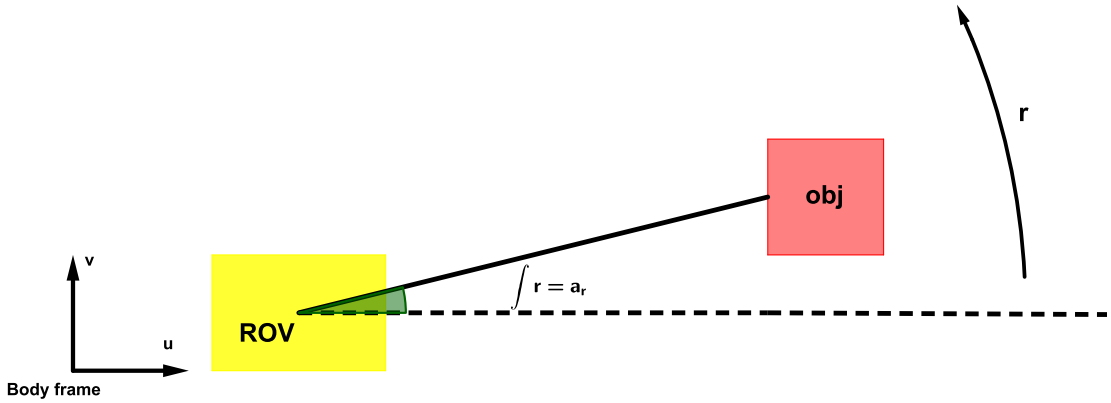


Figure 5.7: Angle difference between the ROV and an object.

Due to the placement of the thrusters on the ROV, it is only fully controllable for three movements, these being surge, heave and yaw. Compared to a system such as a quadcopter/drone the ROV is under-actuated, due to having all 6DOF available but limited due to having no actuators for direct movement for the remaining degrees of freedom. Therefore the modelling will focus on describing motions and the associated dynamics in the before mentioned available degrees of freedom. The current mathematical model represents the dynamics as a set of uncoupled differential equations, meaning that the coupling effects between the 6 degrees of freedom have been neglected. The coupling between the different movements is significant and are amplified at sudden high bursts of force or at high velocities. Coupling is also amplified if the center of gravity is not the center of the vehicle. If the center of gravity is in the geometrical center there will be zero coupling. The current model can be expanded to include the coupling effects later on, and this actual amount of coupling will be revisited later and experiments showcasing the amount of coupling between the different movements will be examined. By attaching additional equipment to the ROV, the additional added mass will alter the center of mass and therefore the dynamics of the system. Therefore additional measures must be taken into consideration, either by adding mass to counter-act the mass from the equipment or alter the thruster positions. Due to the construction of the given VideoRay, alterations to the thrusters are not possible, but it is equipped with a balancing system with weights.

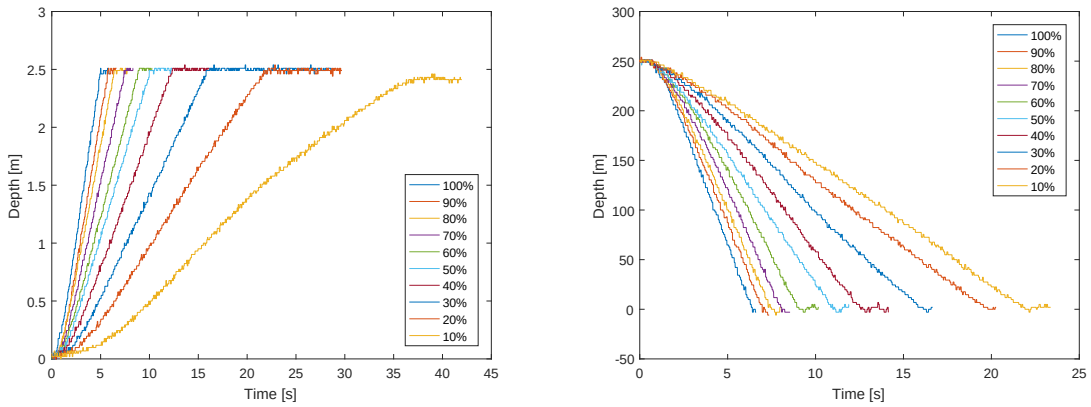
Chapter 6

Parameter Estimation

6.1 Linear and Rotational Velocities

6.1.1 Heave

In order to determine the downward and upward velocities, the depth position is differentiated. The depth data is captured using the pressure sensor, where the pressure value is converted to meters. Due to the derivative being noise sensitive, the positional data is filtered using a low pass filter before being differentiated to obtain the velocities. The filtering is performed offline on the recorded depth data, therefore the plots comparing the filtered and non-filtered data have neglected the possible induced phase-shift.

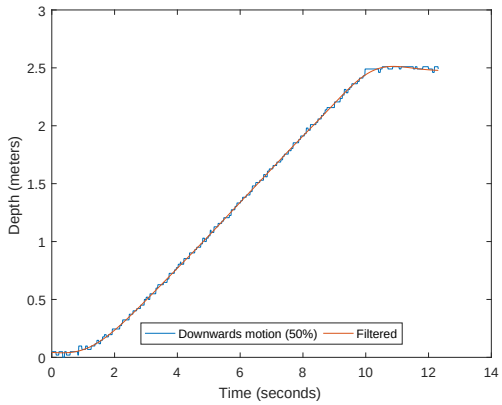


(a) Downward. **(b)** Upward.

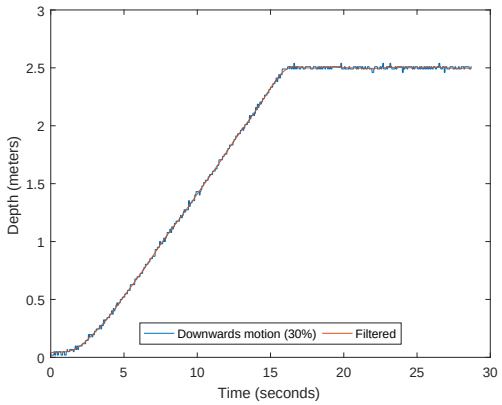
Figure 6.1: Depth over time at different thrust percentages in both directions.

Figures 6.1a and 6.1b compare the depth changes over time at different thrust percentages. Through comparison of the values for both directions, it is observed that the middle percentages share similar velocities, and this will be confirmed using the

derivatives. All of the experiments related to the depth were performed with the ROV as close to neutrally buoyant as possible.



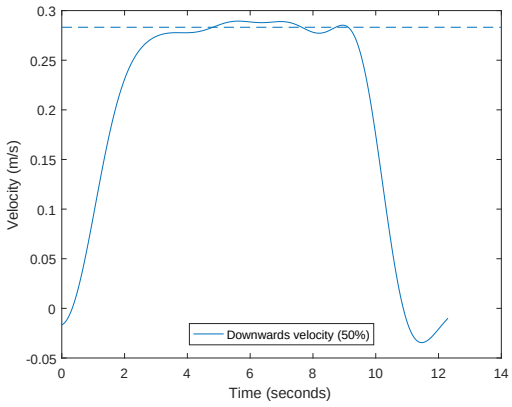
(a) Downward depth over time at 50% thrust.



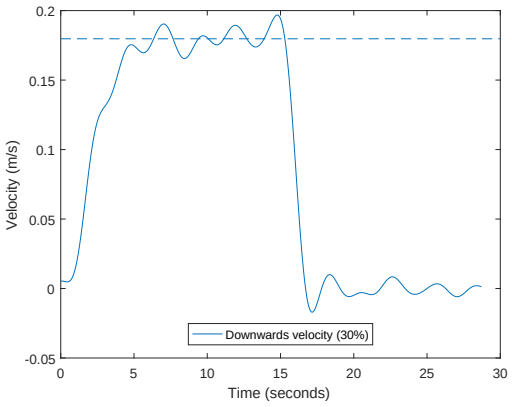
(b) Downward depth over time at 30% thrust.

Figure 6.2: Comparison of the full setup.

The filtered positional data can be seen in figure 6.2, where the remaining filtered data sets can be found in appendix 12.1.1 and 12.1.2, together with the upward depth data.



(a) Downward velocity at 50% thrust.



(b) Downward velocity at 30% thrust.

Figure 6.3: Comparison of the full setup.

The range of which the mean value of the velocity is calculated is based on whenever it becomes constant. This is important since the steady value will be used for calculating the drag coefficient later on the in the chapter. The two figures in 6.3 shows two different thrust percentages and their mean velocities plotted, where in this case it is clear when the ROV reaches a constant velocity. The violent drop-off towards the right-hand side is when the ROV reaches the bottom of the test environment. The remaining velocities and depth measurements for the downwards experiments from 0 to 100% thrust and similar data for the upwards experiments

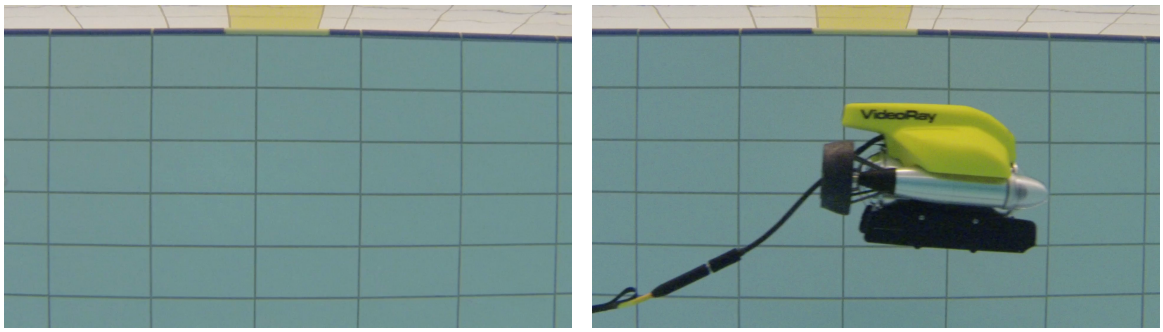
can be found in the appendix as mentioned previously. Based on the data in the appendix, the following velocities in table 6.1 are calculated.

Thruster %	Upward Velocity	Downward Velocity
10%	-0.1212m/s	0.0760m/s
20%	-0.1376m/s	0.1309m/s
30%	-0.1716m/s	0.1797m/s
40%	-0.2305m/s	0.2334m/s
50%	-0.2570m/s	0.2832m/s
60%	-0.3390m/s	0.3231m/s
70%	-0.3762m/s	0.3825m/s
80%	-0.3973m/s	0.4526m/s
90%	-0.4220m/s	0.5217m/s
100%	-0.4638m/s	0.5802m/s

Table 6.1: Heave velocities at 10% thrust increments.

6.1.2 Surge

When finding the velocity of the heave motion it is possible to use reliable positional data in terms of the pressure to differentiate to velocity. In order to determine the surge velocity there were two options available, integrating the accelerometer data or using recordings of the ROV at different velocities. Integrating the accelerometer data yields data that drifts heavily over time, due to the integration amplifying the errors that appear when accelerating and decelerating, the integration also amplifies any measurement noise. Therefore it was determined not to be a viable option for determining the velocity of the ROV.



(a) Test environment. **(b)** Showing the ROV compared to the tiles.

Figure 6.4: Setup for determining the surge velocities.

Using an underwater camera, specifically a GoPro, the ROV was recorded travelling a set distance at different thruster percentages. Figure 6.4 shows the view from the camera and also the scale of the ROV compared to the tiled background. Each white tile has a width of 25cm, and 1m will be used for the velocity calculations. Before entering the frame it assumed that the ROV has reached a constant velocity.

At higher thruster percentages (70%+) the coupling between the pitching and surge motion became too significant, and this resulted in changes to the ROVs path from travelling in a straight line to a curve. Therefore any forward surge data above 70% has been disregarded. When attempting to perform the same experiments for the reverse surge motion, the tether interfered with the ROVs path at almost all thruster values and therefore yielded a lower amount of data. Besides the tether, the pool also had an outlet along the top-side edge used for water filtering, that in turn caused a constant pulling force towards the wall. At times this interfered with the straight movement of the ROV, but was deemed to be insignificant, compared to its interference with the reverse motion at the lower velocities.

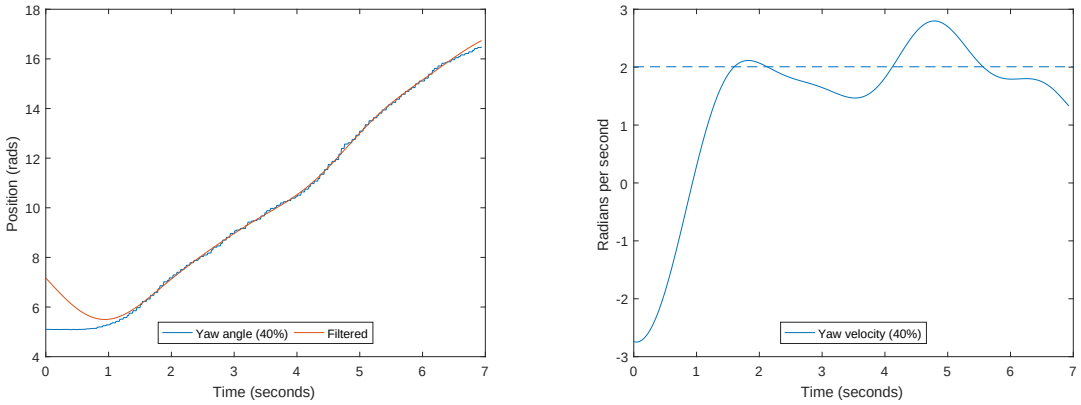
Thruster %	Backward Velocity	Forward Velocity
15%	0.19m/s	0.32m/s
20%	0.37m/s	0.31m/s
30%	0.33m/s	0.47m/s
40%	Tether interference	0.69m/s
50%	-	0.86m/s
60%	-	1.14m/s
70%	-	1.47m/s
80%	-	Too much pitching

Table 6.2: Surge velocities at 10% thrust increments.

The video files of the surge velocity experiments are available as a digital appendix and the calculated forward and backwards velocities can be found in table 6.2. Due to the small amount of force generated at the lower thrust percentages not being enough to create movement, they were left out of the experiment.

6.1.3 Yaw

In order to determine the angular velocity of the ROV for the yaw movement, the thrusters were set to the same thruster percentages but with opposite signs, and this in turn created a counter-clockwise or clockwise rotation. Due to the symmetry of the ROV it is assumed that the velocity obtained in one rotational direction is the same for the opposite. By differentiating the compass data, the angular velocity is obtained. Similar to the depth experiments, the derivative is sensitive to noise and the data is therefore passed through a low pass filter before performing the differentiation.



(a) Angular position over time at 40% thrust. **(b)** Yaw velocity at 40% thrust.

Figure 6.5: Yaw experimental data.

Figure 6.5 shows the obtained data from powering the thrusters at 40%, and the remaining yaw data can be found in appendix 12.2. The interval used to calculate the mean velocity is whenever the ROV seems to have reached a constant angular velocity, similar to the heave and surge motion, but for most of the collected data there seems to be some interference from the surrounding water dynamics induced by the rotational motion. This resulted in oscillating values in the recorded velocity data, making it hard to determine when the steady sate is reached. When rotating the ROV the tether also becomes twisted and starts to tangle up, and this also had an influence on the final values, due to the added strain from the twisted tether.

Thruster %	**Yaw velocity**	**Thruster %**	**Yaw velocity**
20%	1.3453rad/s	60%	3.7655rad/s
30%	1.5182rad/s	70%	4.0677rad/s
40%	2.0068rad/s	80%	5.0068rad/s
50%	2.6190rad/s		

Table 6.3: Yaw velocities.

Table 6.3 shows the calculated mean values when the yaw movement has reached a constant angular velocity. Due to the torque created from lower thruster percentages being too small to induce rotation they were left out. The higher thruster values caused the tether to twist at a high rate, resulting in interference and possible damage, therefore the higher values were left out. The duration of the experiments were also reduced to protect the ROV and tether from becoming damaged from the twisting.

6.2 Thruster Parameters

Figure 6.6: PASCO PS-3202 [46].

In order to gather the forces generated from the thrusters, the PASCO PS-3202 force sensor was used for data acquisition, and the force sensor can be seen in figure 6.6. In order to connect the sensor to the ROV, nylon string was tied to the ROV and to a set of metal beams that were configured on an experiment to experiment basis. The specific setups will be mentioned later during this this chapter when relevant. The sensor has an operating range of $\pm 50N$ and the data is obtained using PASCOs provided software [47]. All the thruster forces were recorded with a sampling frequency of 1KHz and the control inputs were given from the top-side computer using our custom made GUI mentioned in section 3.3.

6.2.1 Rear Thrusters

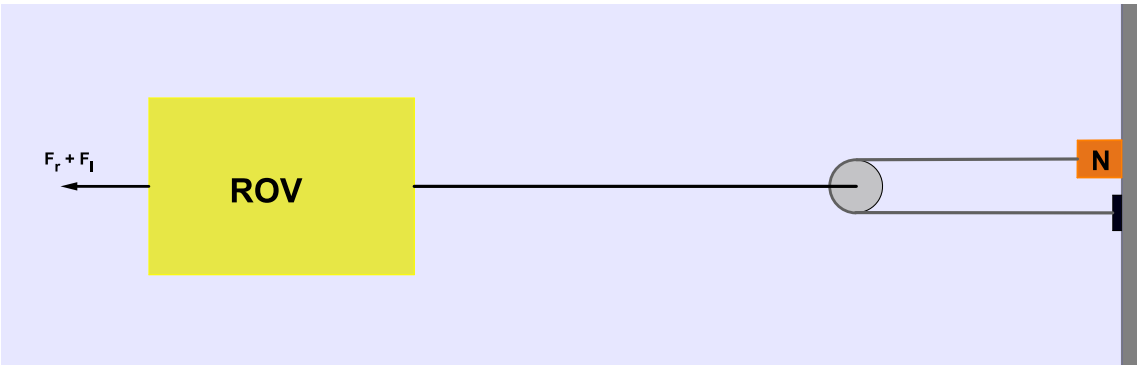
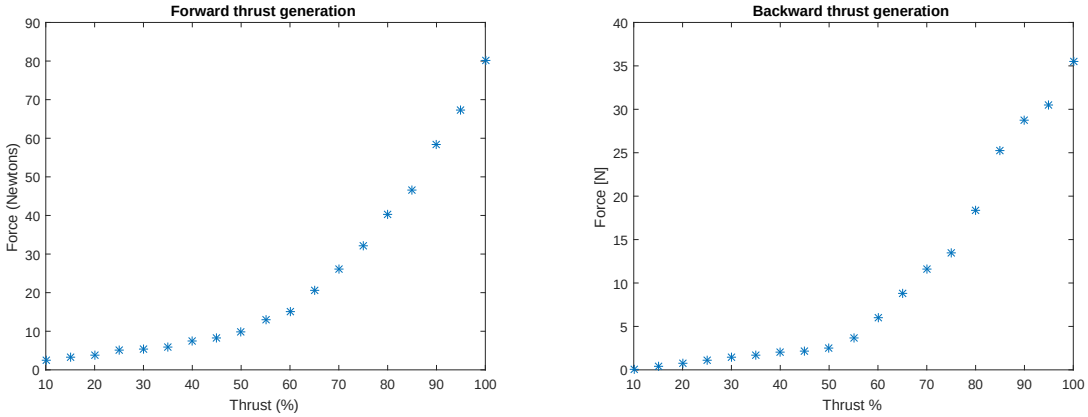


Figure 6.7: The physical setup for measuring the rear thruster forces.

As mentioned previously, the available force sensor has a maximum range of $\pm 50N$. The VideoRay datasheets for the thrusters specify a combined force generation of roughly $\pm 100N$ when operating at maximum input power. Due to this being outside the operating range of the sensor, a pulley system is utilized to halve the force. The setup shown in figure 6.7 will, in the ideal case, equally divide the force produced by the ROV into to two equal amounts, effectively halving the force. By halving the force, the measurements are within the operating range of the digital newton meter.



(a) Forward thruster characteristics. **(b)** Backwards thruster characteristics.

Figure 6.8: Rear thruster characteristics.

Figure 6.8 shows the forward and backwards thrusters characteristics after taking the halving-effect into consideration, where the force is the combined force from the rear left and right thruster $(F_r + F_l)$. The force produced when reversing with the ROV is significantly lower compared to the forward thrust. Due to the objective being navigating towards an object, the dynamics for the forward motion is prioritised. This means that the modelling of the combined rear thruster dynamics will consist mostly of characteristics related to the forward motion, since this will be the primary direction of movement.

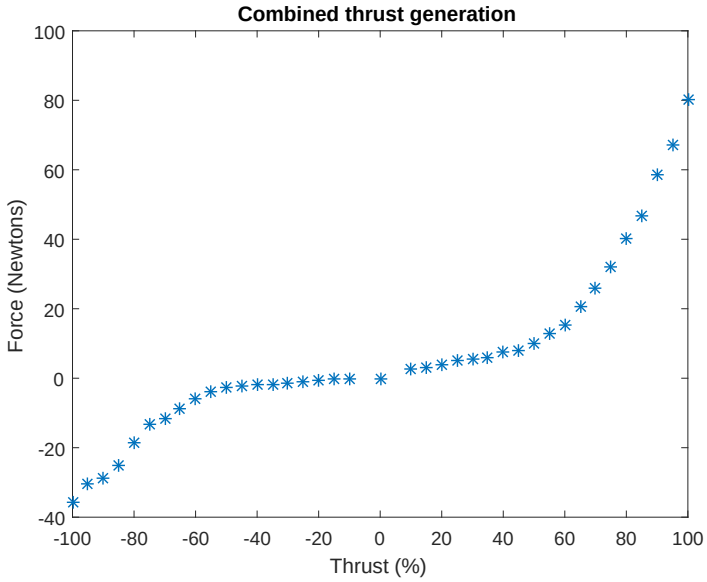


Figure 6.9: Combined thrust characteristics for the surge motion.

Figure 6.9 shows the entire input range and its characteristics, where the forward and backward characteristics are similar at lower input percentages, but with the backwards thrust being slightly less powerful, and then deviating at the higher thrust percentages.

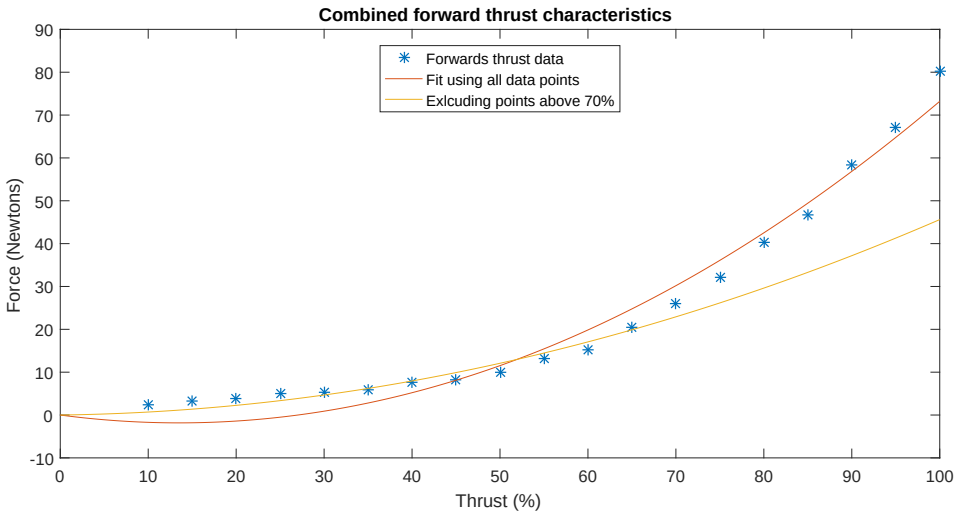


Figure 6.10: Curve fits for the forward thrust.

Using the gathered data points from the experiment and the quadratic equation for the thruster formulated during the mathematical modelling (equation 5.2), it possible to find a suitable curve fit for the data. Figure 6.10 shows two curve fits using the previously mentioned quadratic expression, where the first uses all the data points and the second all the points up to and including 70%.

$$(F_l + F_r)_{alldata} = 0.01003|u_r|u_r - 0.2709u_r \tag{6.1}$$

$$(F_l + F_r)_{selectdata} = 0.00429|u_r|u_r + 0.02711u_r \tag{6.2}$$

Due to the large forces at the higher thruster percentages, it is unlikely to be part of the operating range for the controller implementation. Therefore the secondary curve fit using less data points is favoured, since it provides a tighter fit at the lower thruster percentages. The expressions for the two curve fits are found in equation 6.1 and 6.2, where u_r is the control input given to the two rear thrusters simultaneously.

6.2.2 Top Thruster

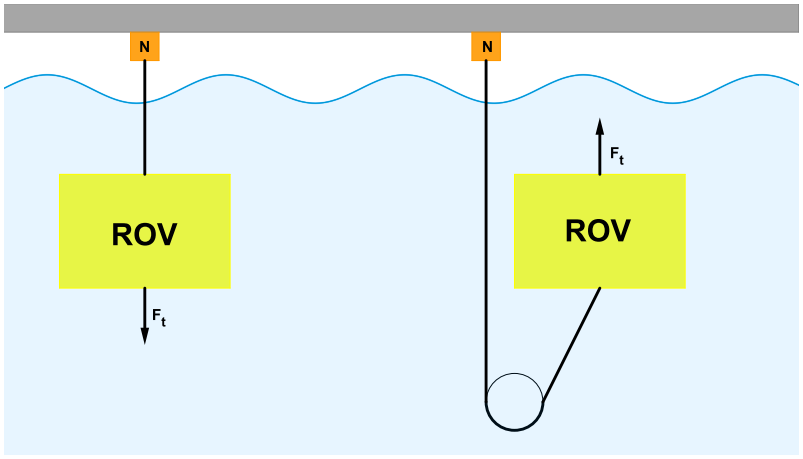
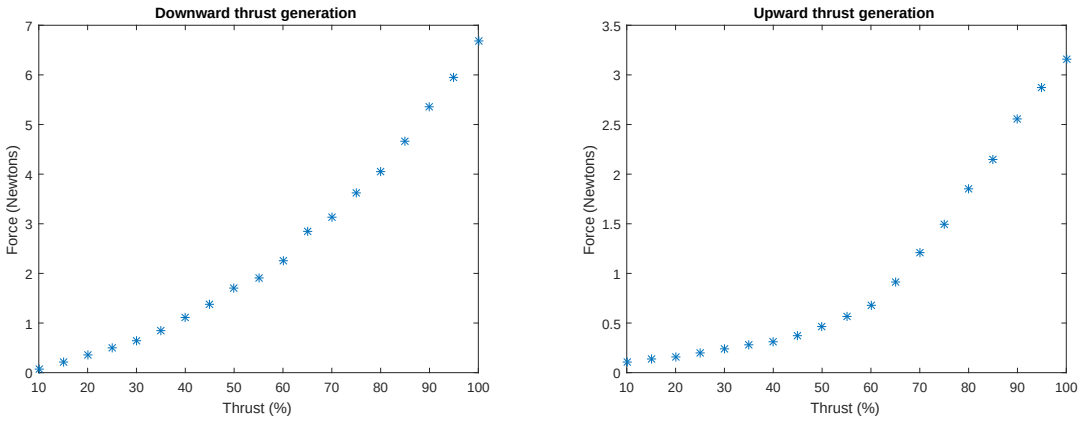


Figure 6.11: The physical setup for measuring the top thruster forces.

Finding the force generated in the downward direction was significantly easier than in the upward direction. Figure 6.11 shows the experimentation setup for the top thruster, where the upward motion required a pulley system at the bottom of the pool. Due to the pulley system for the upward direction, the force was influenced by a significant amount of static friction. This is evident when looking at the gathered data, where there is a lack of thrust characteristics for the upward direction. The static friction was due to the wet string making contact with the metal pipe used as a pulley placed at the bottom of the pool. The angle on the right-hand experiment setup on figure 6.11 is exaggerated; during the actual testing the ROV moved further towards the vertical string from the newton meter, making contact, resulting in more friction between the ROV and string.



(a) Downward thruster characteristics. **(b)** Upward thruster characteristics.

Figure 6.12: Downward and upward thruster characteristics.

Figure 6.12 shows the thruster characteristics for the up and downward motions, where the downward characteristics are well-captured compared to the upward. Figure 6.13 shows the comparison between the gathered data sets for the top thruster.

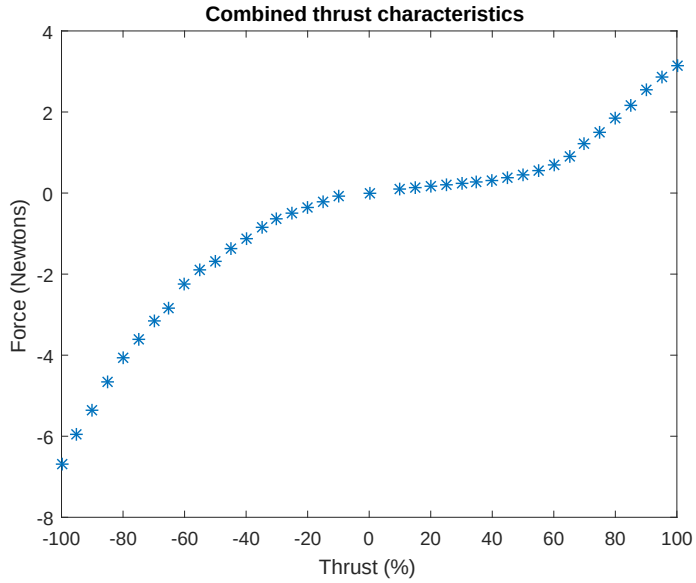


Figure 6.13: Combined thrust characteristics for the heave motion.

During the depth experiments discussed previously in this chapter, in section 6.1.1, the gathered velocities for the heave motion (table 6.1) show that the velocities are similar for both the up and down motion at the middle thruster percentages. Due to this fact, the data for the downward motion will be used for both up and down direction. Since it is clear that some of the characteristics for the upward motion has been lost due to additional friction.

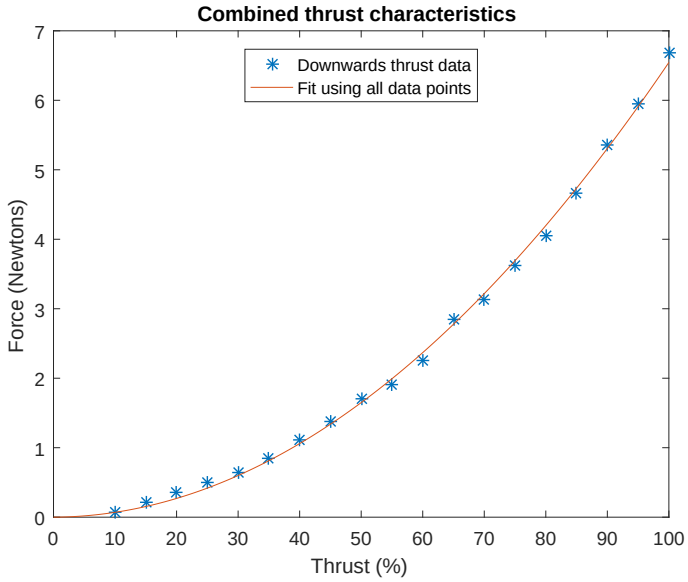


Figure 6.14: Curve fit for the downward thrust.

Similar to the surge motion, the quadratic formulation for the thruster is used when performing the curve fits on the downward force data. Equation 6.3 shows the resulting curve fit for the top-side thruster, where u_t is the given input thruster percentage.

$$F_t = 0.0006494|u_t|u_t + 0.000513u_t \tag{6.3}$$

6.3 Drag Coefficients

To determine the drag coefficients for the three movements, the equations of motion for surge, heave and yaw from mathematical modelling are utilised. At zero acceleration the equations for the surge and heave motion found in equations 5.15 and 5.14 become the following.

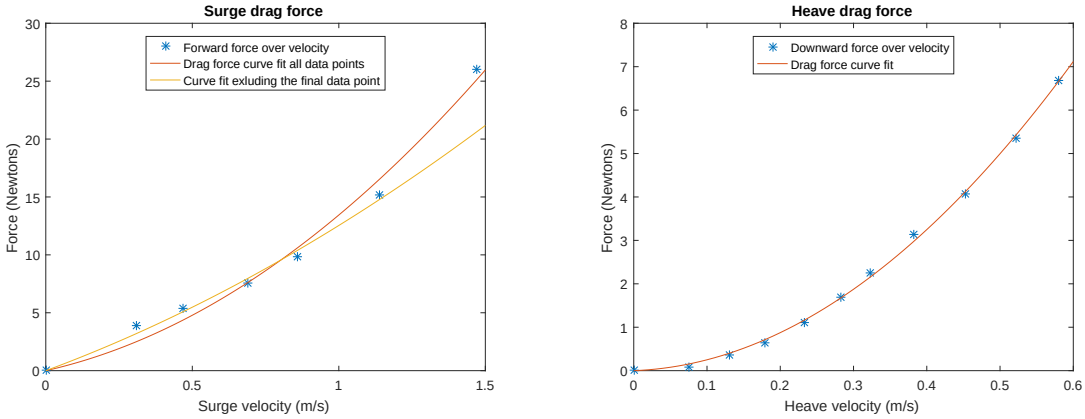
$$F_t = F_d^w = a_d w|w| + b_d w \tag{6.4}$$

$$F_r + F_l = F_d^u = a_d u|u| + b_d u \tag{6.5}$$

Then, by plugging in the velocities measured at the given thruster forces it is possible to obtain the different velocity dependant drag coefficients. Similar to surge and heave, the drag force from the yaw movement will be calculated using the torque produced by the thrusters and the measured angular velocities. At zero angular acceleration equation 6.6 becomes the following:

$$\tau_{thrusters} = \tau_{drag} = a_d r|r| + b_d r \tag{6.6}$$

In order to obtain the torque generated by the thrusters, the coefficient calculated during mathematical model of 0.0903 is multiplied with the forward force data from figure 6.8.



(a) Surge drag coefficient fit. **(b)** Heave drag coefficient fit.

Figure 6.15: Curve fits for the drag coefficients.

Figure 6.15 shows the curve fits for the drag force related to heave and surge. Two fits have been made for the surge drag force as shown in figure 6.15a, since the initial curve fit using all data points captures the characteristics of the middle velocities worse than the curve fit using all the data except the highest velocity, the equation describing the surge drag force is then as follows in equation 6.7, where the highest velocity data point is excluded.

$$F_d^u = 3.159u|u| + 9.379u \tag{6.7}$$

The curve fit for the heave drag force fits the data points well for the given velocity interval and the equation for the heave drag force can be found in equation 6.8.

$$F_d^w = 18.72w|w| + 0.6324w \tag{6.8}$$

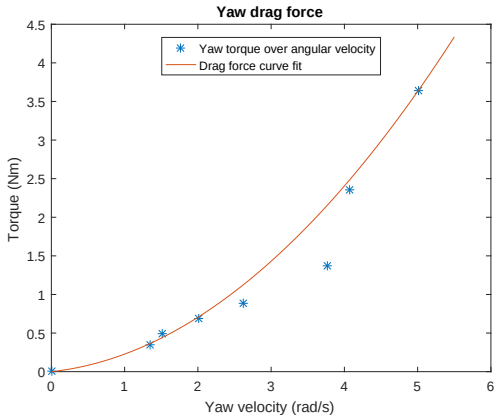


Figure 6.16: Yaw drag coefficient fit.

Figure 6.16 shows the curve fit for the yaw drag torque. Due to a combination of the simplicity of the drag modelling and the inaccurate velocity data due to the extra dynamics introduced when rotating. The current curve fit does not accurately track

the data, but was chosen due to its fit around the desired operating point of 30%, which is equal to 1.5182rad/s. Equation 6.9 is curve fit for the yaw drag, where r is yaw velocity.

$$\tau_{drag} = 0.1246|r|r + 0.103r \tag{6.9}$$

6.4 Coupling

In order to identify the amount of coupling present between the surge motion and the available angular movements some angular data has been recorded when surging at different thrust percentages. The pitch angle is the most important, since unwanted pitching dynamics changes the course of the ROV dramatically. When surging at the higher thruster percentages (70%+) the ROV pitches a significant amount causing the ROV to dive.

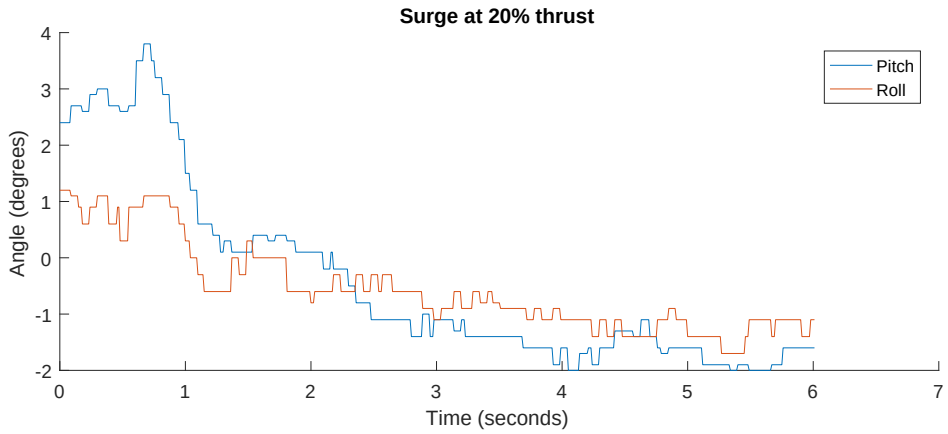


Figure 6.17: Pitch and Roll data when surging at 20% thrust.

Figure 6.17 shows the pitch and roll angles when surging at 20% thrust. During the initial second the ROV has a positive pitch angle caused from suddenly powering on the thrusters. This angle is reduced to a slightly negative pitch angle at around -2 degrees.

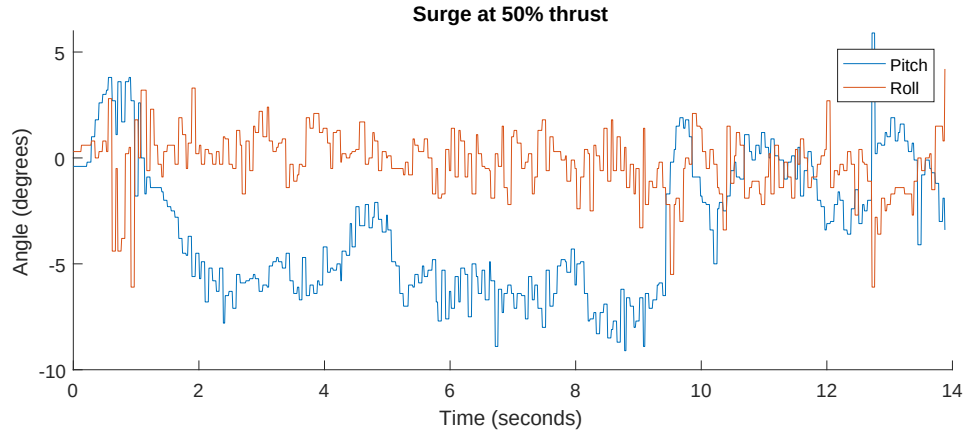


Figure 6.18: Pitch and Roll data when surging at 50% thrust.

Similar to the previous experiment, figure 6.18 shows the pitch and roll data when surging with 50%. The pitch angle also becomes positive during the initial second due to the initial power on and then reduces to a slightly more negative angle at -5 degrees. The pitch angle becomes 0 at 10 seconds due to powering off the ROV, this is to indicate the pitch angle when standing still.

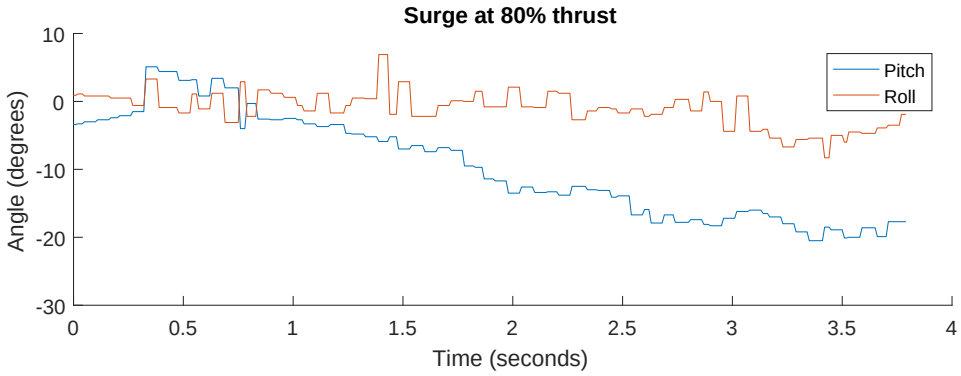


Figure 6.19: Pitch and Roll data when surging at 80% thrust.

Figure 6.19 shows the angular data from surging at 80%. Here there is a significant negative itching angle, at approximately -15 degrees. This causes the ROV to change its depth greatly over time whilst moving forward. The above data for the three different surge motions shows that working within the range of 20%-50% the pitching angle is insignificant compared to the higher thrust value.

Chapter 7

State Space

7.1 Introduction

$\boldsymbol{A}$	System matrix	$p \times p$
$\boldsymbol{B}$	Input matrix or control matrix	$p \times q$
$\boldsymbol{C}$	Output matrix	$r \times p$
$\boldsymbol{D}$	Feed-forward matrix	$r \times q$
$\dot{\boldsymbol{x}}$	Future State	$p \times 1$
$\boldsymbol{x}$	Current State	$p \times 1$
u	Input (SI)	1×1
$\boldsymbol{u}$	Input vector (MI)	$q \times 1$
$\boldsymbol{K}$	General Feedback matrix (SI)	
$\boldsymbol{K_s}$	Feedback matrix (SI)	$1 \times p$
$\boldsymbol{K_m}$	Feedback matrix (MI)	$q \times p$
$\boldsymbol{Q}$	State Weighting Matrix	$p \times p$
$\boldsymbol{R}$	Control Weighting Matrix	$q \times q$
$\boldsymbol{P}$	ARE solution	$p \times p$
$\boldsymbol{L}$	Observer Gain Matrix	$r \times p$

Table 7.1: Matrix and vector dimensions.

The state space or state variable representation is a mathematical model of a system, where the equations are represented as a set of first-order differential equations relating to input, output and state vectors. State-space control design is advantageous when dealing with MIMO systems (Multiple Input Multiple Output), due to the concept of states, matrix and vector notation. This makes it easier when creating larger scale systems, in terms of keeping track of the system inputs and outputs.

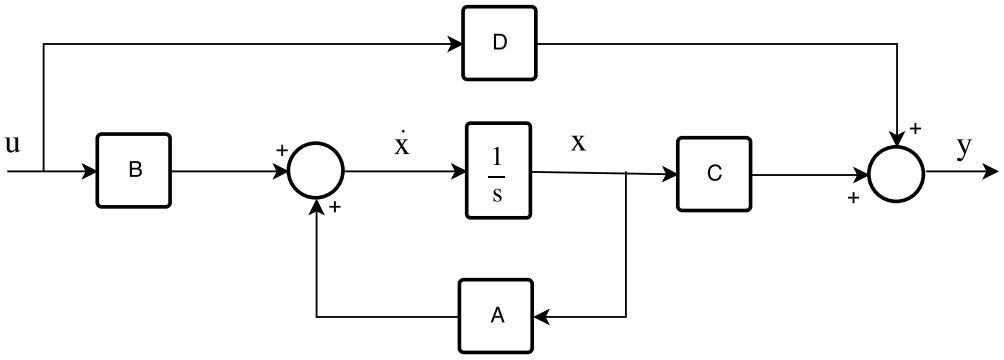


Figure 7.1: State-Space block-diagram.

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{7.1a}$$
$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u} \tag{7.1b}$$

The state space representation is as shown in equation 7.1, where equation 7.1a is the state equation and 7.1b the output equation. The state equation shows the relationship between the current state ($\boldsymbol{x}$) and the given input($\boldsymbol{u}$), where $\dot{\boldsymbol{x}}$ is the calculation of the future state. Matrix $\boldsymbol{A}$ contains all the information regarding the dynamics of the system, while the $\boldsymbol{B}$ matrix is the actuator dynamics. The output equation displays the current output value based on the entries in the $\boldsymbol{C}$ matrix, while the final matrix $\boldsymbol{D}$ is the feedforward matrix [48]. Due to dealing with a MIMO system, the system input is a vector rather than a scaler for the SISO case. Similarly, the system output $\boldsymbol{y}$ is a vector due to it being a multiple output system.

7.2 Full-State Feedback

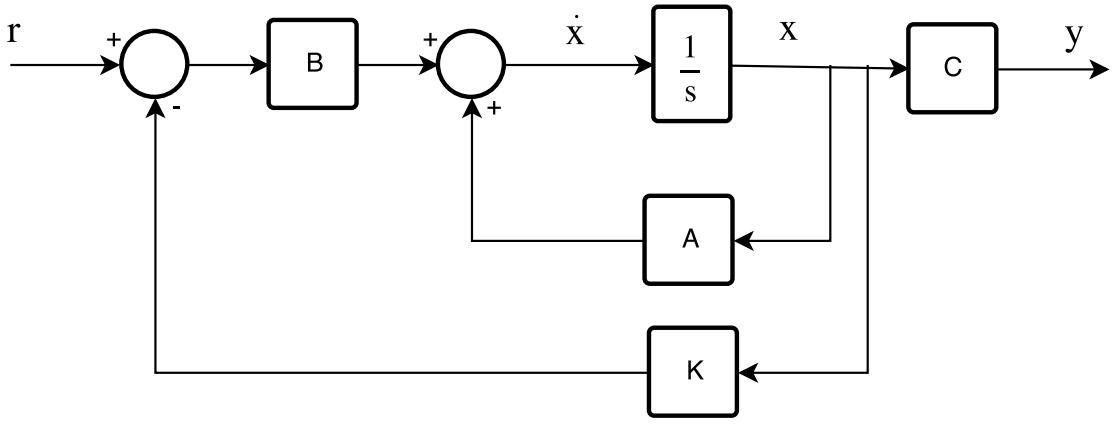


Figure 7.2: Full-state feedback.

When dealing with full-state feedback it is assumed that all the information is available from the state vector $\boldsymbol{x}$. It is then possible to introduce the negative feedback

matrix $\boldsymbol{K}$, which can be seen on figure 7.2. The entries in $\boldsymbol{K}$ are calculated based on the desired pole placements and due to the properties of full-state feedback if the system fulfils the controllability requirement, the poles can be placed freely by the designer.

$$Co = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{AB} & \boldsymbol{A^2B} & ... & \boldsymbol{A^{p-1}B} \end{bmatrix} \tag{7.2}$$

The controllability of the system depends on the rank of the controllability matrix $\boldsymbol{Co}$. In order for the system to be completely controllable the matrix must have full rank p. Controllability depends purely on the system dynamics defined in the $\boldsymbol{A}$ matrix and the actuator dynamics formulated in the $\boldsymbol{B}$ matrix. If the system is not controllable, it means that the effect of the input is not sufficient to influence the system. The amount of control authority must then be increased, by either revisiting the mathematical model of the system dynamics or by modifying the actuators [48] [49]. Through the analysis of the block diagram shown in figure 7.2 it is possible to formulate the control law found in equation 7.3, where for any p number of states there will be p number of entries in $\boldsymbol{K_s}$.

$$u = -\boldsymbol{K_s x} = -\begin{bmatrix} k_1 & k_2 & ... & k_p \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} \tag{7.3}$$

The notation shown in equation 7.3 is for the single input case, where it is possible to extend it based on the dimension of the input matrix $\boldsymbol{B}$. The multiple input extension is shown in equation 7.4.

$$\boldsymbol{u} = -\boldsymbol{K_m x} = -\begin{bmatrix} k_{11} & k_{12} & ... & k_{1p} \\ \vdots & \vdots & \vdots & \vdots \\ k_{q1} & k_{q2} & ... & k_{qp} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} \tag{7.4}$$

Depending on the number of system inputs, the control law from equation 7.3 or 7.4 can then be substituted into the previously found state equation, equation 7.1a, which re-formulates the expression creating a new $\boldsymbol{A}$ for the system as shown in equation 7.5.

$$\dot{\boldsymbol{x}} = \boldsymbol{Ax} + \boldsymbol{Bu} = \boldsymbol{Ax} - \boldsymbol{BKx} = (\boldsymbol{A} - \boldsymbol{BK})\boldsymbol{x} \tag{7.5}$$

As mentioned previously, by using full-state feedback it is assumed that there is complete knowledge of the current state of the the system, which enables the freedom to place the poles freely. Using pole-placement methods it is possible to place the system poles as desired by calculating the equivalent feedback gains based on desired time-domain specification or through an iterative process. The stability of the system depends on the pole locations, which are the eigenvalues of the $\boldsymbol{A}$ matrix, where in order for the system to be stable their location must be in the left-half plane. By introducing the feedback gains from the matrix defined in equation

7.4, the control input to the system is amplified according to the entries in $\boldsymbol{K}$. Too-large feedback gains may cause the system so saturate, meaning that the controller is sending an input signal greater than what the given actuator is capable of. This introduces a non-linearity to the system and reduces the precision of the developed controller.

7.2.1 SISO System

When dealing with a SISO system, typically a second order system, it is possible to place the poles according to desired time-domain specifications by calculating the desired poles and placing them as the dominant second order poles [48] [50]. The time-domain specification used for the performance calculations are the following:

- Settling time t_s
- Overshoot M_p
- Rise time t_r

The rise time is the time it takes the system to transition from the input value to within close range of the new set point. This property is based on the natural frequency of the second order system and calculated as follows in equation 7.6.

$$t_r \tilde{=} \frac{1.8}{\omega_n} \tag{7.6}$$

Overshoot is the maximum amount that the system overshoots and is specified in terms of percentage due to calculating it based on the maximum overshoot divided by the final value. The calculation for the desired overshoot relies on the damping ratio ζ, this is shown in equation 7.7.

$$M_p = e^{-\pi\zeta/\sqrt{1-\zeta^2}} \quad 0 \leq \zeta < 1 \tag{7.7}$$

Settling time is the the time it takes the transients to decay, and this can be viewed as the total time elapsed until the system reaches the final value within some error threshold, in this case $\pm 1\%$.

$$e^{-\zeta\omega_n t_s} = 0.01 \tag{7.8a}$$

$$ts = \frac{4.6}{\zeta\omega_n} \tag{7.8b}$$

Equation 7.8a is set equal to 0.01 because the desired error at settling time is $\pm 1\%$ as specified above. The settling time t_s can then be expressed in terms of damping ratio ζ and natural frequency ω_n.

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = 0 \tag{7.9}$$

By choosing the desired time-domain specifications in terms of settling time, rise time and overshoot, it is possible to calculated the associated damping ratio and

natural frequency. These calculated values are then substituted into equation 7.9 and set equal to 0, which then can be solved for the pole location based on the desired specifications.

$$(s - p1)(s - p2) = det[s\boldsymbol{I} - (\boldsymbol{A} - \boldsymbol{BK})] \tag{7.10}$$

Once the poles are calculated in equation 7.9, they can be used to calculate the feedback gain matrix $\boldsymbol{K}$ using equation 7.10.

$$(s^2 + 2\zeta\omega_n s + \omega_n^2)(s^2 + \alpha s + \beta) = 0 \tag{7.11}$$

As mentioned previously, the time domain specifications can be enforced by calculating second order dominant poles. By extending equation 7.9 to equation 7.11, it is possible for the calculated poles based on the specifications to dominate the system's response. This is only true if the remaining pair α and β are placed further into the left-half plane, they can be scaled by a separation factor that is varied in size during the controller design process. [49]

7.3 Linear Quadratic Regulation

Linear Quadratic Regulation (LQR) is a method used for choosing feedback gains for optimal control. This method uses the control law on the form as shown in equation 7.3 mentioned previously in the chapter. Regulator implies that the desired state values is zero, that the system starts from an initial state and that the state vector is then driven to the zero vector [48]. System eigenvalues designed for regulation have the same performance when introducing a non zero reference.

The goal is to minimize the cost function J, where the cost function is defined as follows with the objective of driving $\boldsymbol{x} \to 0, t \to \infty$.

$$J = \int_0^\infty (\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u})dt \tag{7.12}$$

Equation 7.12 shows the cost function, where $\boldsymbol{x}$ is the state vector and $\boldsymbol{u}$ the input vector. By modifying the values of the matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ the different weights for the states and inputs are modified.

$$\boldsymbol{Q} = \begin{bmatrix} q_{11} & q_{12} & \cdots & q_{1p} \\ q_{12} & q_{22} & \cdots & q_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ q_{1p} & q_{2p} & \cdots & q_{pp} \end{bmatrix} \quad \boldsymbol{R} = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1q} \\ r_{12} & r_{22} & \cdots & r_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ r_{1q} & r_{2q} & \cdots & r_{qq} \end{bmatrix} \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} \quad \boldsymbol{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_q \end{bmatrix}$$

The symmetric $\boldsymbol{Q}$ matrix is positive definite, which results in the scalar $\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x}$ being positive for any value in the $\boldsymbol{x}$ vector. The property of positive definite ensures that the resulting cost is a positive value. This is similarly true for the $\boldsymbol{R}$ matrix and vector $\boldsymbol{u}$.

$$\boldsymbol{Q} = \begin{bmatrix} q_1 & q_3 \\ q_3 & q_2 \end{bmatrix} \quad \boldsymbol{R} = \begin{bmatrix} r_1 & r_3 \\ r_3 & r_2 \end{bmatrix} \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{7.13}$$

$$\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} = q_1x_1^2 + q_2x_2^2 + 2q_3x_1x_2 \tag{7.14}$$

Equation 7.14 shows the influence of the weightings in the $\boldsymbol{Q}$ matrix, where the diagonal terms influence each individual state from the $\boldsymbol{x}$ vector and the off-diagonal entries influence the state combinations. A similar relationship exists between the values in the $\boldsymbol{R}$ matrix, where the diagonal entries influence each individual input and the off-diagonal entries the input combinations. By increasing the value of the entries in $\boldsymbol{Q}$ and $\boldsymbol{R}$ the respective states and inputs are slowed down.

$$\boldsymbol{u} = -\boldsymbol{K}\boldsymbol{x}, \quad \boldsymbol{K} = \boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P} \tag{7.15}$$

Compared to the pole placement technique discussed in the previous section, gain matrix $\boldsymbol{K}$ is calculated based on the $\boldsymbol{R}$,$\boldsymbol{B}$ and $\boldsymbol{P}$ matrices, as shown in equation 7.15, where P is found by solving the Algebraic Ricatti Equation (ARE) shown in equation 7.16.

$$\boldsymbol{A}^T\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} - \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P} + \boldsymbol{Q} = 0 \tag{7.16}$$

According to Bryson's rule it is possible to calculate possible initial values for the diagonal entries in the $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrices based on the maximum value of the given state or input. Finding the optimal values for the given matrices is an iterative process based on simulation results. [48] [51]

$$q_{pp} = (maximum\,value\,of\,[x_p^2])^{-1} \tag{7.17a}$$

$$r_{qq} = (maximum\,value\,of\,[u_q^2])^{-1} \tag{7.17b}$$

During the iterative tuning process the values for the $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrix values are adjusted, where the desire is to find an acceptable performance and control effort.

7.4 Observer

One of the requirements of using full-state feedback is knowing all the current values of the states, but this is rarely the case and therefore an alternative is needed in order to obtain the remaining unknown states. Using an observer, it is possible to estimate the remaining states based on the measured outputs of the given system. In order to use an observer it is a requirement that the given system is oberservable, meaning that it is required that sufficient information is obtained by reading the available outputs [48].

$$O = \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{C}\boldsymbol{A} \\ \vdots \\ \boldsymbol{C}\boldsymbol{A}^{n-1} \end{bmatrix} \tag{7.18}$$

Equations 7.18 shows the the calculation of the observability matrix, where the system is completely observable if the matrix has full rank.

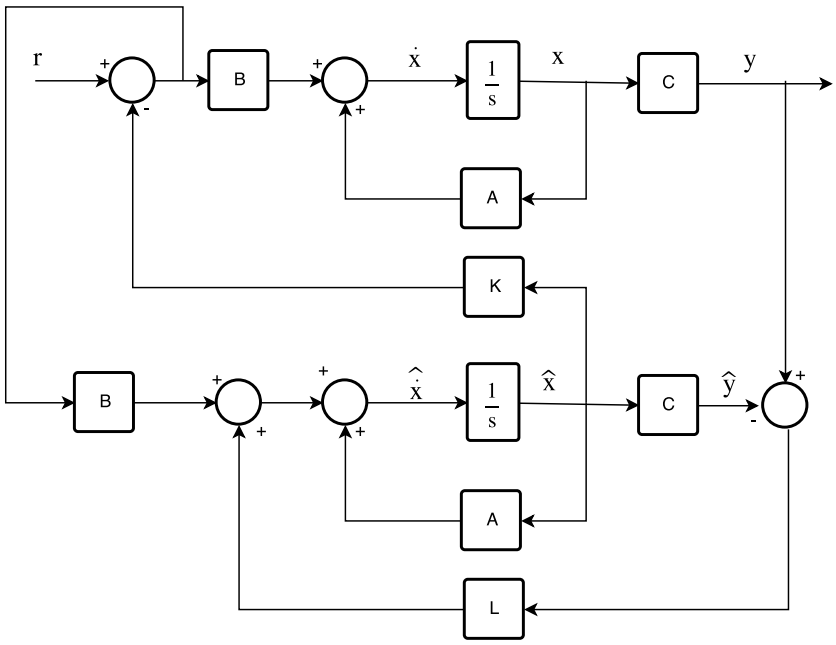


Figure 7.3: State Estimator for the feedback matrix.

The observer is a copy of the original system, taking in the same input as the regular system. The goal of the observer is for the estimated states to converge with the actual states. This is done using the feedback matrix $\boldsymbol{L}$, which takes in the difference between the system and observer system outputs. The feedback gains determine the rate of convergence of the estimated states $\hat{\boldsymbol{x}}$, and since the observer is a software construct the gains are not limited by the physical limitations of the system, though too-large gains will significantly amplify the given noise. Based on the separation principle, it is then possible to the determine the feedback gain $\boldsymbol{K}$ using LQR or a similar pole-placement method while assuming that there is full-state feedback available, and during the implementation the input to signal to the feedback matrix $\boldsymbol{K}$ will then be replaced by the state estimation $\hat{\boldsymbol{x}}$ from the observer, using the configuration shown in figure 7.3. The observer gains will then be placed accordingly using a similar pole-placement technique as mentioned previously.

$$\dot{\hat{\boldsymbol{x}}} = \boldsymbol{A}\hat{\boldsymbol{x}} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{L}(\boldsymbol{y} - \boldsymbol{C}\hat{\boldsymbol{x}}) \tag{7.19}$$

$$\hat{\boldsymbol{y}} = \boldsymbol{C}\hat{\boldsymbol{x}} \tag{7.20}$$

Equations 7.19 and 7.20 show the state equation and output equation for the observer. These will be used during the controller development and controller implementation.

7.5 Reference Tracking

When introducing the feedback, there will be a steady-state error without the use of input scaling or integral control. Using a reference scaling method is based on calculating a gain to scale the desired reference value accordingly. This gain is calculated based on the system matrices and is therefore sensitive to any errors in the model, compared to the real system dynamics. [48]

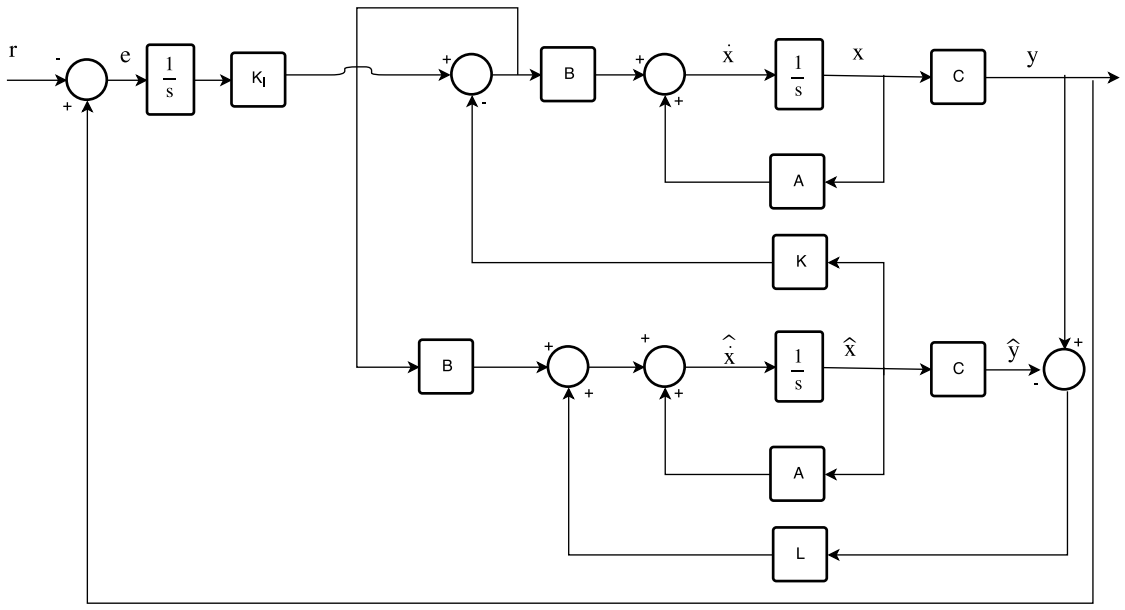


Figure 7.4: State Estimator with integral control.

By introducing the integral control as shown in figure 7.4, over time the system will remove the steady-state error due to the accumulative properties of the integrator. Besides the integrator there is an integral gain, which is tuned based on the closed loop poles of the system. When using integrators there is the possibility of anti-windup. Whenever the system saturates due to the limitations of the physical actuators, if the error signal is still applied to the integrator it will grow in size until the system becomes unsaturated. This may result in undesired effects on the system performance, including a very large overshoot due to the amount of accumulated saturation. In in order to circumvent integral windup an anti-windup method is applied to disable the integrator whenever the system reaches saturation. [48]

Chapter 8

Model combination and validation

8.1 Linearisation and State Space Representation

In order to use the current mathematical model for the creation of a controller using State-Space, it is a requirement that the system is linear. The linearisation will be done using the Taylor Series, where a suitable operating point is defined. Equation 8.1 shows the first order approximate, where a is the operating point.

$$f(x) = f(a) + \dot{f}(a)(x - a) \tag{8.1}$$

There are several non-linearities in the model, these being the thruster dynamics and drag forces. During the parameter estimation the curve fits for the acquired data were focused around the operating point of 30% thrust input, as this is outside the deadzones of the respective thrusters and still at a low enough percentage to neglect the pitching dynamics. The equivalent velocities at the 30% thrust values are read from tables 6.1, 6.2 and 6.3 respectively.

$$x_0 = [u_0 \quad w_0 \quad r_0]^T = [0.47 \quad 0.1797 \quad 1.5182]^T \tag{8.2}$$

$$u_0 = [u_{r0} \quad u_{t0} \quad u_{y0}]^T = [30 \quad 30 \quad 30]^T \tag{8.3}$$

x_0 and u_0 indicate the operating points for the velocity states and for the thrusters. The system of non-linear equations depends on the states $x = [u \quad w \quad r]^T$ and input $u = [ur \quad ut \quad uy]^T$, and the three non-linear equations for the motion can be found in 8.4.

$$f(x,u) = \begin{bmatrix} 0.152(0.00429u_r|u_r| + 0.02711u_r) - 0.152(3.159u|u| + 9.379u) \\ 0.152(0.0006494u_t|u_t| + 0.000513u_t) - 0.152(18.72w|w| + 0.6324w) \\ \frac{1}{0.105}(0.0903(0.00429u_y|u_y| + 0.02711u_y) - (0.1246r|r| + 0.103r)) \end{bmatrix} \tag{8.4}$$

It is possible to obtain the A and B matrices by taking the partial derivative of the given function in equation 8.4 and evaluating at the given operating points as shown in equation 8.5.

$$\boldsymbol{A} = \frac{\delta f}{\delta x}\Big|_{(x_0)}, \quad \boldsymbol{B} = \frac{\delta f}{\delta u}\Big|_{(u_0)} \tag{8.5}$$

After evaluating the partial derivatives, the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are obtained and can be found in equation 8.6. The resulting offsets from the linearisation will be added during the simulations and implementation in order to shift the operating point of the system accordingly, these offsets being $x_c = [0.106069 \quad 0.09188 \quad 2.7351]^T$ for the linearised state matrix and $u_c = [-0.586872 \quad -0.0887 \quad -3.3204]^T$.

$$\begin{bmatrix} \dot{u} \\ \dot{w} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} -1.877 & 0 & 0 \\ 0 & -1.119 & 0 \\ 0 & 0 & -4.584 \end{bmatrix} \begin{bmatrix} u \\ w \\ r \end{bmatrix} + \begin{bmatrix} 0.04325 & 0 & 0 \\ 0 & 0.006 & 0 \\ 0 & 0 & 0.2446 \end{bmatrix} \begin{bmatrix} u_r \\ u_t \\ u_y \end{bmatrix} \tag{8.6}$$

In order to introduce a positional reference point, additional states are added to obtain the positional output of the system. The additional states d_u,d_w and a_r are the distance and depth with respect to the detected object and also the heading angle, and this is introduced during the mathematical modelling in section 5.6. The addition of these states can be seen in equation 8.7.

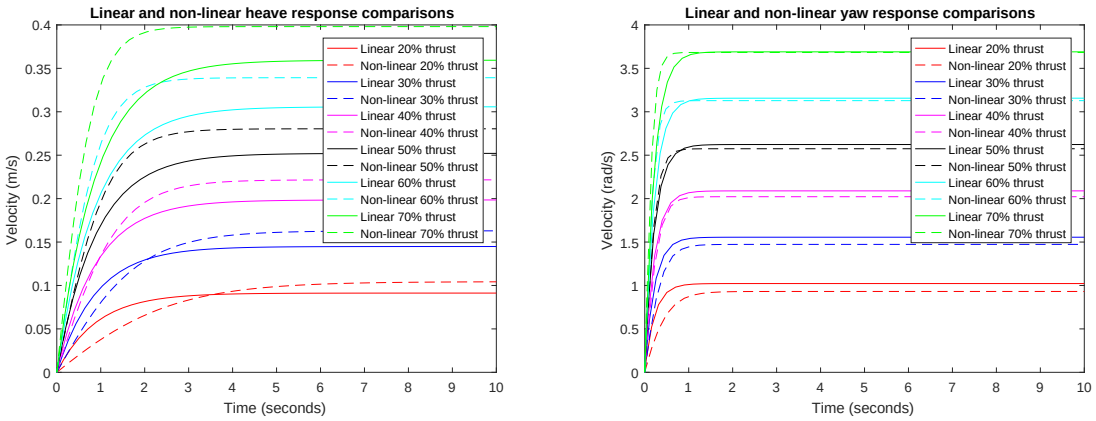
$$\begin{bmatrix} u \\ w \\ r \\ \dot{u} \\ \dot{w} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1.877 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1.119 & 0 \\ 0 & 0 & 0 & 0 & 0 & -4.584 \end{bmatrix} \begin{bmatrix} d_u \\ d_w \\ a_r \\ u \\ w \\ r \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0.04325 & 0 & 0 \\ 0 & 0.006 & 0 \\ 0 & 0 & 0.2446 \end{bmatrix} \begin{bmatrix} u_r \\ u_t \\ u_y \end{bmatrix} \tag{8.7}$$

When creating the depth and heading controller which positions are with respect to the world frame, the body frame velocities are changed to derivatives of the world frame positions, and since velocity information can be obtained by taking the derivative of the heading measured from the compass and from the pressure sensor which returns the depth. Substitution of states can be seen in equation 8.8, where the body frame notation is replaced by world frame notation.

$$\begin{bmatrix} \dot{z} \\ \dot{\psi} \\ \ddot{z} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -1.119 & 0 \\ 0 & 0 & -4.584 \end{bmatrix} \begin{bmatrix} z \\ \psi \\ \dot{z} \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0.006 & 0 \\ 0 & 0.2446 \end{bmatrix} \begin{bmatrix} u_t \\ u_y \end{bmatrix} \tag{8.8}$$

8.2 Model Validation

In order to validate the model, the linear and non-linear step responses are compared. As expected when stepping outside the operating point, the non-linear model performed closer to actual performance from the recorded experimental data. Figure 8.1 shows the comparisons between the linear and non-linear models for the heave and yaw motion. The linearised model has lower steady state value and a slower response due to the quadratic features being removed in the linearisation process. The offset between the linear and non-linear models at the operating points are small, but the heave model deviates further at higher percentages. The linearised yaw model also deviates from the non-linear model, though significantly less than the heave motion.



(a) Heave step response. **(b)** Yaw step response.

Figure 8.1: Comparison between linear and non-linear step responses.

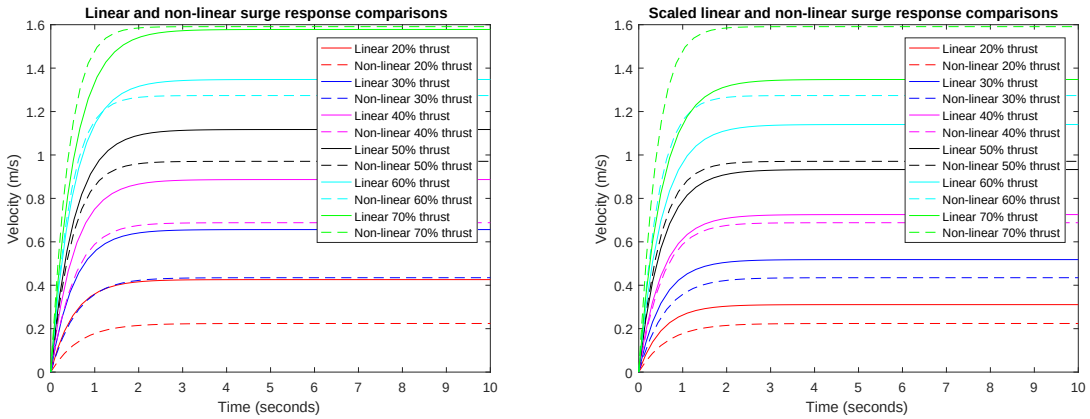
The overall deviations may also be due to the choice of operating point for linearisation, in this case 30%, and also to general uncertainties in the obtained data and curve fits from the parameter estimation. Tables 8.1 and 8.2 show the steady-state velocity comparisons between the linear model, the non-linear model and the experimental data for the heave and yaw motion.

Thruster %	Linear	Non-Linear	Experimental
20%	0.09104m/s	0.1041m/s	0.1309m/s
30%	0.1448m/s	0.1628m/s	0.1797m/s
50%	0.2516m/s	0.2804m/s	0.2832m/s
70%	0.3581m/s	0.3981m/s	0.3825m/s

Table 8.1: Heave velocity comparisons.

Thruster %	Linear	Non-Linear	Experimental
20%	1.022rad/s	0.9311rad/s	1.3453rad/s
30%	1.556rad/s	1.473rad/s	1.5182rad/s
50%	2.623rad/s	2.574rad/s	2.6190rad/s
70%	3.69rad/s	3.683rad/s	4.0677rad/s

Table 8.2: Yaw velocity comparisons.



(a) Surge step responses. **(b)** Surge step responses scaled by 0.9.

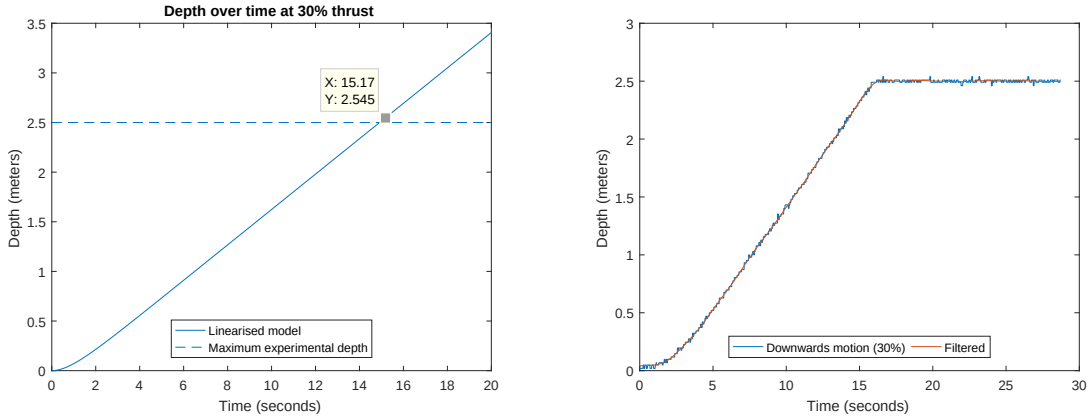
Figure 8.2: Comparison between linear model and non-linear model.

Figure 8.2 shows the responses for the surge motion at the different thruster percentages. Figure 8.2a shows the initial behaviour of the surge velocity where at the lower thrust percentages, the linearised model is significantly faster than the non-linear model, and the non-linear model is also overall slower than the experimental data. Since the controller development will be performed on the linearised model, the input for the linear surge motion is down-scaled in order to obtain a better performance surrounding the operating point of 30%. The input is scaled by 0.9 and the new step responses can be seen in figure 8.2b.

Thruster %	Linear	Non-Linear	Experimental
20%	0.3106m/s	0.2239m/s	0.31m/s
30%	0.518m/s	0.4347m/s	0.47m/s
50%	0.9327m/s	0.9707m/s	0.86m/s
70%	1.347m/s	1.591m/s	1.47m/s

Table 8.3: Surge velocity comparisons.

Table 8.3 shows the comparison between the scaled linear model, the non-linear model and the experimental data for the surge velocity. The now-scaled linear model tracks the experimental data more precisely around the operating point and also at the higher percentages.



(a) Linear model depth over time. **(b)** Experimental depth data.

Figure 8.3: Comparison between the depth experimental data and linear model.

By comparing the positional depth output d_w of the linear model with recorded experimental data, it is possible to further validate the model. The comparison in figure 8.3 shows that the depth of 2.5m is reached at approximately 15 seconds for the both the linearised model and the experimental data. Due to not having experimental positional data for the remaining displacement outputs d_u and a_r, a similar comparison cannot be made.

$$\begin{bmatrix} u \\ w \\ r \\ \dot{u} \\ \dot{w} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1.877 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1.119 & 0 \\ 0 & 0 & 0 & 0 & 0 & -4.584 \end{bmatrix} \begin{bmatrix} d_u \\ d_w \\ a_r \\ u \\ w \\ r \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0.0389 & 0 & 0 \\ 0 & 0.006 & 0 \\ 0 & 0 & 0.2446 \end{bmatrix} \begin{bmatrix} u_r \\ u_t \\ u_y \end{bmatrix} \tag{8.9}$$

Equation 8.9 shows the final system to be used for the controller development, and in terms of creating the controller to navigate towards an object with respect to itself. The $\boldsymbol{B}$ matrix has had the surge thruster input scaled by 0.9 in order to obtain the desired performance at the operating point.

$$\begin{bmatrix} \dot{z} \\ \dot{\psi} \\ \ddot{z} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1.119 & 0 \\ 0 & 0 & 0 & -4.584 \end{bmatrix} \begin{bmatrix} z \\ \psi \\ \dot{z} \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0.006 & 0 \\ 0 & 0.2446 \end{bmatrix} \begin{bmatrix} u_t \\ u_y \end{bmatrix} \tag{8.10}$$

Equation 8.10 shows the final system for the positioning with respect the to the world frame, due to not having the surge as part of the system no additional scaling is needed.

Chapter 9

Controller Development

9.1 Controller development

In order to avoid using very large or small numbers for the control gains, the thruster polynomials are normalized from the input range of -100 to 100 to the range of -1 to 1.

$$u_r = (F_l + F_r)_{selectedata} = 42.9|u_r|u_r + 2.711u_r \tag{9.1}$$

By normalizing the expression for the rear thrusters from equation 6.2, the small coefficients in the polynomial are enlarged to the following in equation 9.1. Similar to the rear thrusters, the input values for the top thruster (Equation 6.3) are normalized to the interval -1 to 1, where equation 9.2 shows the new polynomial.

$$u_t = F_t = 6.494|u_t|u_t + 0.0513u_t \tag{9.2}$$

Since the two polynomials have been normalized they must be linearised using the new operating point, which in this case is 0.3 instead of 30%. The new $\boldsymbol{B}$ matrix is then calculated. The scaling previously mentioned during the model validation is also applied to the surge motion after normalizing to smaller range.

$$B_{obj} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 3.89 & 0 & 0 \\ 0 & 0.6 & 0 \\ 0 & 0 & 24.46 \end{bmatrix}, \qquad B_{worldframe} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0.6 & 0 \\ 0 & 24.46 \end{bmatrix} \tag{9.3}$$

Equation 9.3 shows the two matrices, where the one is for the navigational controller with respect to a detected object and the second with respect to the world frame. From the linearisation at operating point $u_o = [0.3 \quad 0.3 \quad 0.3]^T$, the following constants $u_c = [-0.586872 \quad -0.0887 \quad -3.3204]^T$ were obtained and will be added during the controller developed to correctly shift the coordinate system to the new operating point.

The controllability and observability matrices are calculated for the body frame system in order to determine if the system is controllable and observable. Both the controllability and observability matrices have full rank and therefore meet both requirements. Since the world frame system is a reduced version of the body frame system, but with the same coefficients, the controllability and observability requirements are also met. The control gains will be designed using LQR, where the feedback matrix is calculated based on the linear system but simulated on the non-linear system, and this is done in order to mimic the real life behaviour of the system. As initial values for the $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrices Bryson's rule is used, where the different inputs are weighted accordingly thereafter through an iterative process.

9.1.1 Body frame controller

The reference input values for the body frame controller will be the desired position of the ROV with respect to the given object. The desired distance for the surge direction will be 1 meter (d_u) and 0 difference in depth and heading. The initial position of the ROV with respect to the object will be as follows: $d_u = 3m$, $d_w = 2m$, $a_r = 1rad$. These particular reference values and initial values are chosen based on a typical scenario that the ROV will encounter and also the given testing facilities available.

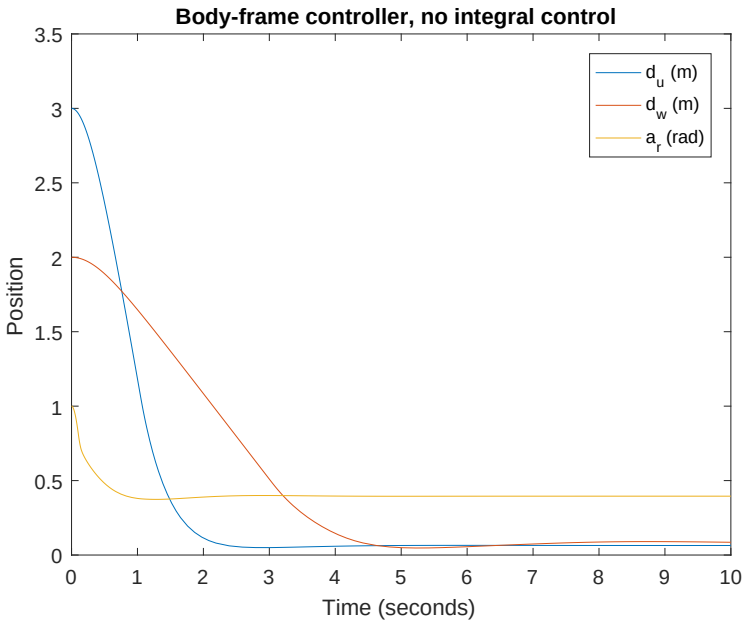
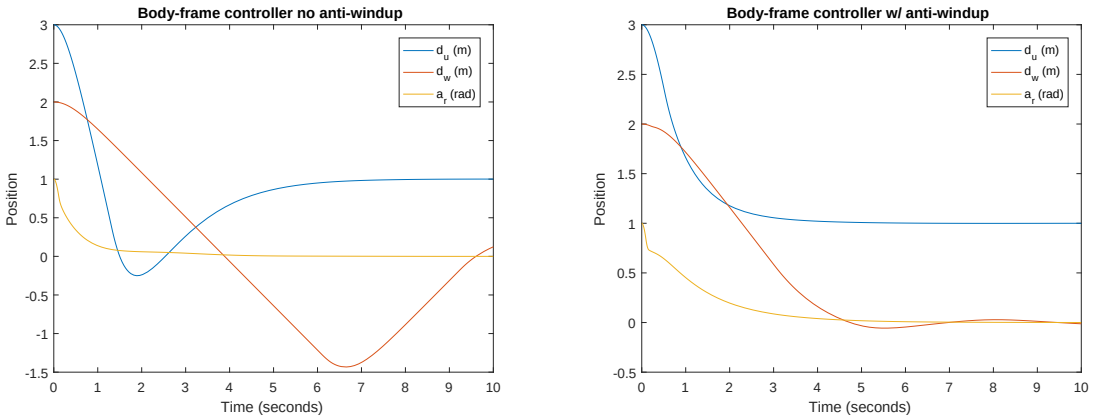


Figure 9.1: Reference input without integral control.

Figure 9.1 shows the performance of the calculated feedback gains when simulated using the non-linear model. The controller gains were calculated iteratively, changing the values for the $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrices accordingly until the performance matched the one in the figure. This performance is with reference input values as previously

mentioned, where in this case there is a steady-state error due to lack of proper reference tracking.



(a) No anti-windup. **(b)** With anti-windup.

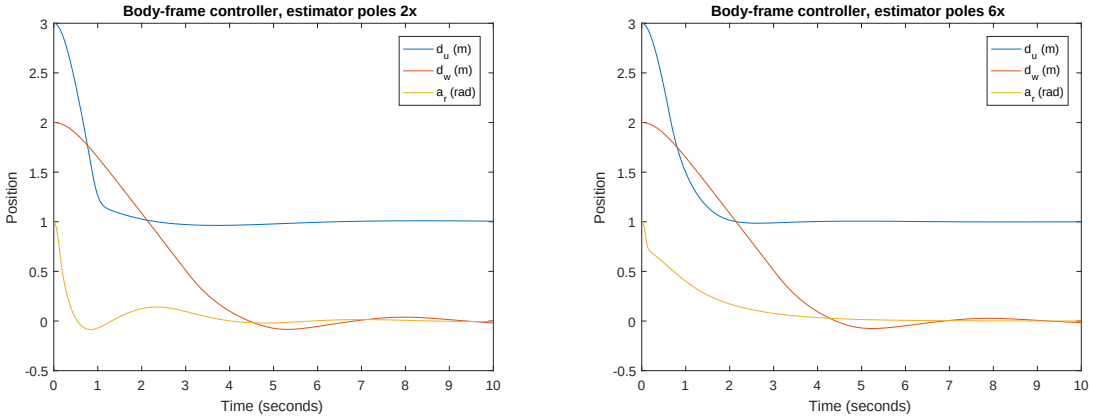
Figure 9.2: Controller simulation with integral control.

Integral control was then added in order to track the desired reference values. The controller over-saturates and this causes the integrator to accumulate when it is not desired and this in turn results in large overshoots. The overshoot caused by the integrator is removed using an anti-windup technique called back-calculation. Back-calculation relies heavily on the chosen coefficient for the back-calculation. Both the integral gain and back-calculation gain is chosen through trial and error, where the integral gain is picked first and then a back-calculation gain is selected based on the given integral gain value.

The estimator gains are calculated based on the closed loop system poles using the control gains calculated using LQR. Fast estimator poles increase the sensitivity to noise, but slower poles decrease the convergence rate of the estimated states to the actual states. The initial poles for the estimator are placed at 6 times faster than the closed loop system.

$$L_{poles} = 6(eig(\boldsymbol{A} - \boldsymbol{BK})) \tag{9.4}$$

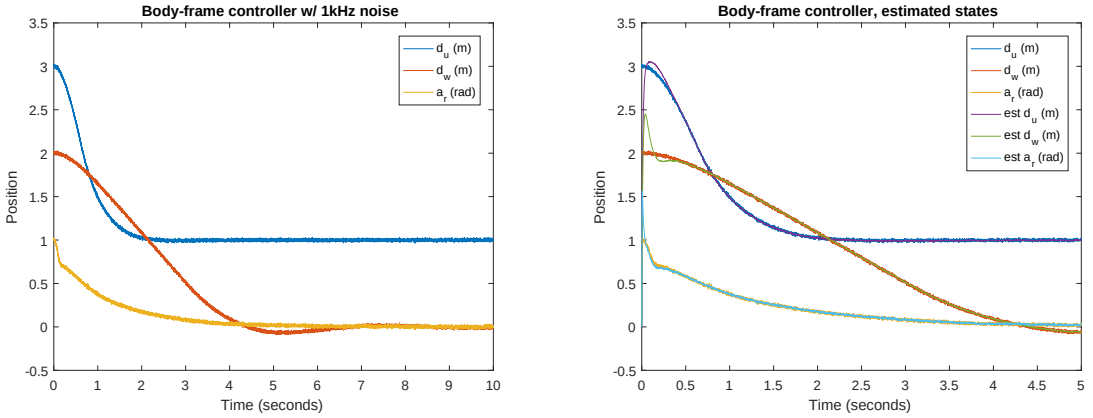
In equation 9.4 the closed loop system eigenvalues are multiplied by 6, in order to move the poles further to the left away from the imaginary axis. The matrix $\boldsymbol{L}$ is calculated using a pole-placement method with the desired poles from equation 9.4.



(a) 2 times CL poles. **(b)** 6 times CL poles.

Figure 9.3: Comparison between estimator pole locations.

Figure 9.3 shows the difference between placing the estimator poles 2 and 6 times faster than the closed loop system poles. The slower estimator poles cause some changes in performance due the slower convergence with the estimated states, as mentioned previously.



(a) Added output noise. **(b)** Estimated states w/ output noise.

Figure 9.4: System output and estimator states with added noise.

Measurement noise is added to the simulation in order to emulate the real-life noise that will be present. The noise is high frequency noise at 1kHz with a random seed using the built-in block in Simulink. Figure 9.4 shows the performance of the system using an estimator with the output noise present and also a comparison between the actual states and estimated states with the added noise.

9.1.2 World frame controller

Since the dynamics used for the world frame controller are a reduced version of those for the body-frame controller the same control gains still yield a suitable performance.

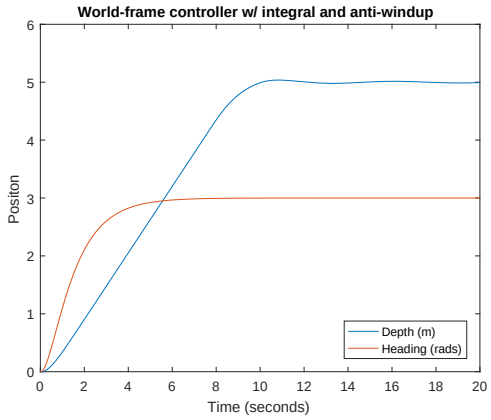
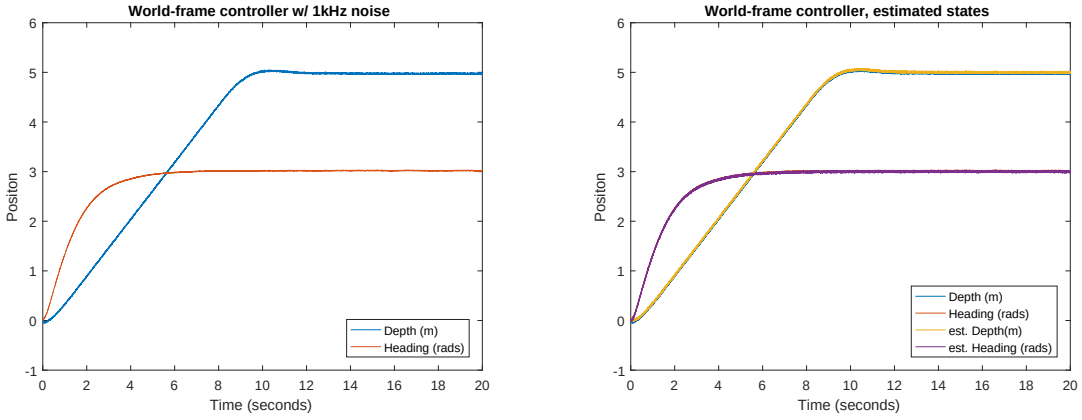


Figure 9.5: World frame controller.

The initial conditions for the world frame controller is a heading of 0 radians and a depth of 0 meters. The performance shown in figure 9.5 is using the same gains, both for the feedback control and integral control, as calculated previously. Due to the limited depth of indoor pools, the controller was designed around being capable of reaching 5 meters of depth and heading error of 3 rads, which is roughly equal to half a rotation.



(a) Added output noise. **(b)** Estimated states with added noise.

Figure 9.6: World frame controller simulation with output noise and estimator.

Figure 9.6 shows the performance of the estimator with the added output noise and also the convergence of the estimated states with the actual states.

9.2 Implementation

9.2.1 Controller Implementation

In order to implement the developed controllers it is necessary to create a discretization of the different continuous time components. Based on the closed loop system bandwidth it is possible to establish the required sampling frequency for the controller. The controller can be implemented directly on the top-side computer, where the desired control signals are calculated directly and sent to the ROV.

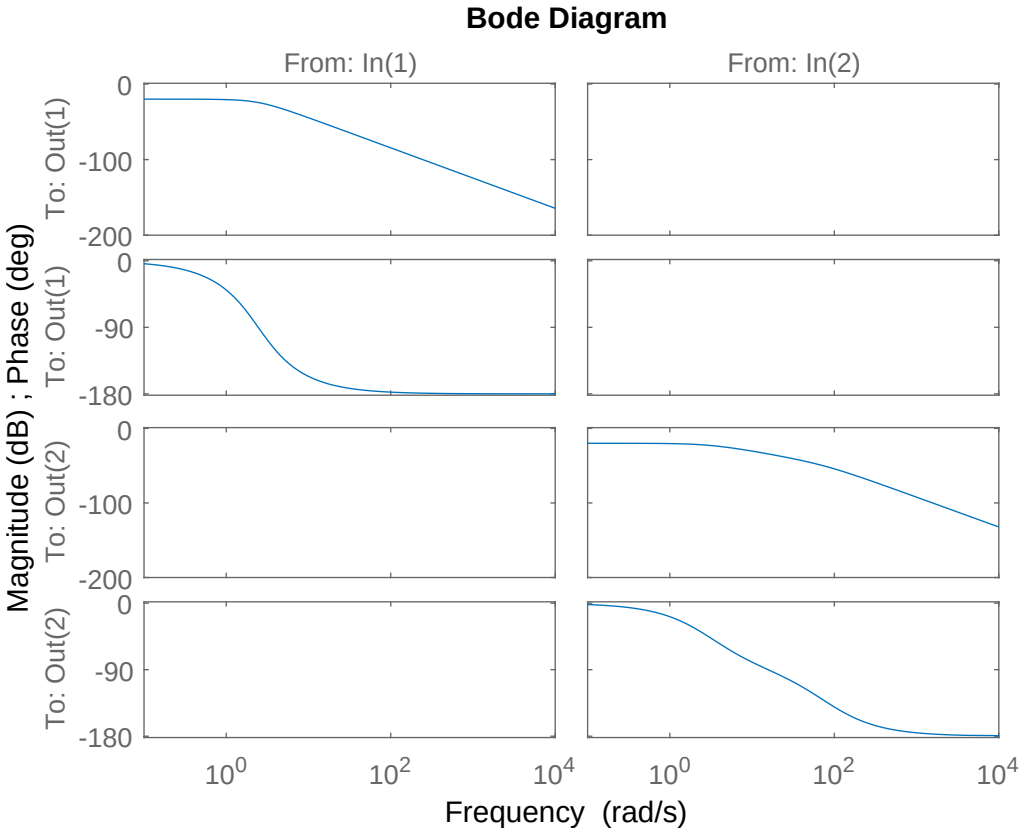


Figure 9.7: Bode plot for the world frame closed loop.

Figure 9.7 shows the bode plot for the world frame closed loop system, where the -3dB mark indicates the closed loop system bandwidth. The sampling frequency of the implementation is required to be at least 20 times greater than the system bandwidth in order to ensure good performance.

$$f_z = 2rad/s = 0.32Hz \tag{9.5a}$$

$$f_\psi = 3.17rad/s = 0.5Hz \tag{9.5b}$$

Equation 9.5 shows the respective bandwidth frequencies for the two dynamics, where the sampling frequency will be 20 times faster than the fastest dynamic. This

requires the calculations to at least be performed at 10Hz which is 10 times per second.

The equations to be implemented are obtained through the analysis of the system block diagram in figure 7.4. The control input vector $\boldsymbol{u}$ which is used to perform the actuation of the three available thrusters is calculated based on the integrated error signal subtracted by the feedback signal.

$$u = e\frac{K_i}{s} - \boldsymbol{K}\hat{\boldsymbol{x}} = \frac{K_i}{s}(r - y) - \boldsymbol{K}\hat{\boldsymbol{x}} \tag{9.6}$$

The integrator needs to be discretized in order to be implemented since the block diagram and simulations assume continuous time, where a common method for numerical integration is Eurler's method.

$$y(t_{k+1}) = y(t_k) + hf(t_k) \tag{9.7}$$

Equation 9.7 shows the numerical integration method, where h is the step size, $f(t_k)$ the current output values, $y(t_k)$ the current accumulation and $y(tk + 1)$ the next accumulated value. The back-calculation anti-windup scheme also needs to be implemented in code, since the saturation of the integrator must also be taken care of during the implementation. As mentioned previously in equation 7.19, the state equation for the estimator is as follows in equation 9.8.

$$\dot{\hat{\boldsymbol{x}}} = \boldsymbol{A}\hat{\boldsymbol{x}} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{L}(\boldsymbol{y} - \boldsymbol{C}\hat{\boldsymbol{x}}) \tag{9.8}$$

The equation must be calculated once per sampling period in order to update the control input $\boldsymbol{u}$ since it is calculated based on the estimated states. In order to calculate the state estimate, it is necessary to discretize the derivative $\dot{\hat{\boldsymbol{x}}}$. The derivative will be approximated using a finite-difference approximation, which divides the difference between the next state and current state over the step size.

$$\frac{\hat{\boldsymbol{x}}(t_{k+1}) - \hat{\boldsymbol{x}}(t_k)}{h} = \boldsymbol{A}\hat{\boldsymbol{x}}(t_k) + \boldsymbol{B}\boldsymbol{u}(t_k) + \boldsymbol{L}(\boldsymbol{y}(t_k) - \boldsymbol{C}\hat{\boldsymbol{x}}(t_k)) \tag{9.9}$$

Equation 9.9 is the state equation with the derivative substituted by the finite difference approximation. It is then reformulated so that the calculation of $\hat{\boldsymbol{x}}(t_{k+1})$ is isolated, so that it can be used to calculate $\boldsymbol{u}$, which expressed in equation 9.6. The reformulated expression for the state estimation is shown in equation 9.10.

$$\hat{\boldsymbol{x}}(t_{k+1}) = (\boldsymbol{I} + h\boldsymbol{A} - h\boldsymbol{L}\boldsymbol{C})\hat{\boldsymbol{x}}(t_k) + h\boldsymbol{B}\boldsymbol{u}(t_k) + h\boldsymbol{L}\boldsymbol{y}(t_k) \tag{9.10}$$

Using Math.NET Numerics [52] it is possible to directly implement equation 9.10 using regular notation for matrix addition and multiplication. Math.NET numerics is part of the open-source Math.NET initiative, providing different algorithms for numerical computations and different tool sets. The implementation will use the vector and matrix operations available to implement the given equation for the estimated state vector. The step size h used for both equation 9.10 and 9.7 has to be 0.1 or less, and this is based on the 10Hz calculated from the closed loop system bandwidth mentioned in equation 9.5.

9.2.2 Computer Vision Implementation

The testing of the computer vision code was divided into two test setups, the first having the pitch angle of the ROV at 0 and the second having the ROV pitched. The pitched setup of the ROV is shown in figure 9.8. The difference between the two viewing angles can been seen in figure 9.9, where, due to the shallowness of the pool, having a non-zero pitching angle resulted in a visible reflection of the buoy. This reflection is an issue since the reflection looks like the buoy and will therefore also be detected by the computer vision programme. Therefore, during the stationary tests the ROV was pitched in order to confirm the calculated position of the buoy with respect to the ROV. The setup shown in 9.9a was used in order to confirm that the calculated d_w was correct and the setup in figure 9.9b used to confirm the distance and heading towards the buoy.

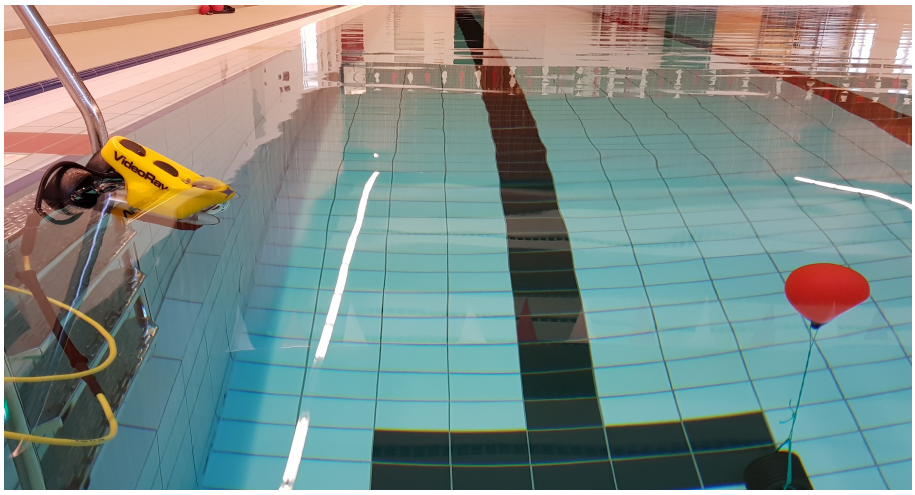
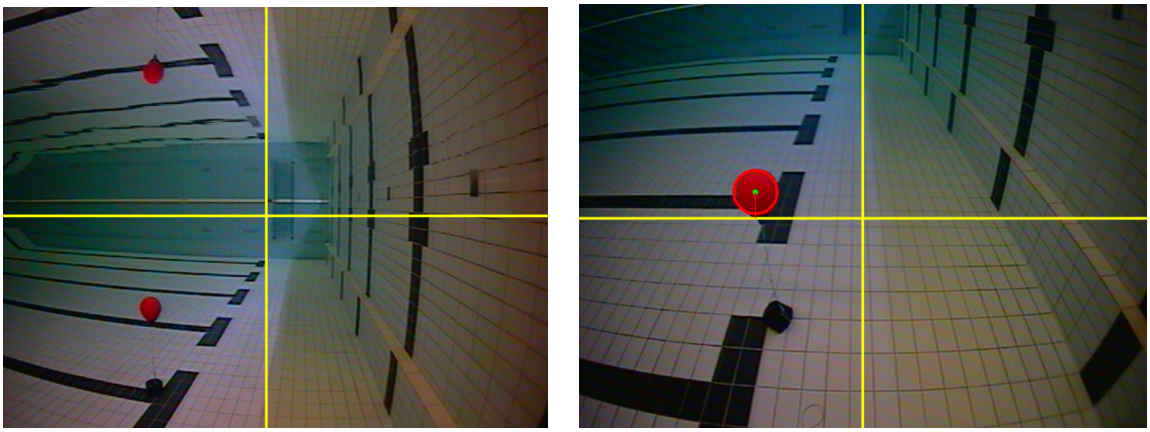


Figure 9.8: The setup of the computer vision test.

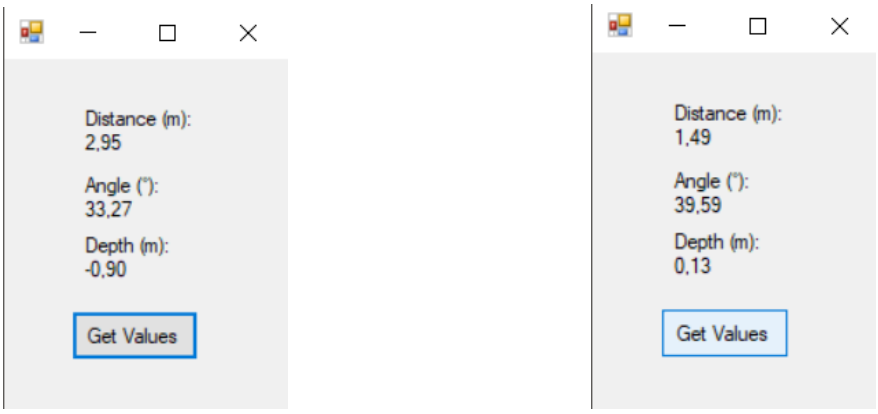


(a) Buoy being mirrored by the reflection.

(b) Buoy in more level view.

Figure 9.9: Looking at the buoy with two different angles.

The measured values from the setup with a pitch angle of 0 can be seen in figure 9.10a. The depth of the buoy with respect to the ROV was measured to be -0.9m, meaning that the ROV was situated above the buoy. When confirming the actual buoy position it was measured to be 0.8m from the surface of the pool, which fits with the measured value by the camera since the camera looks at the centre of the buoy and the radius of the buoy is 0.1m. The second set of data shown in figure 9.10b shows that the buoy is located 1.49m away and at a 40 degree angle. The buoy was measured being 1.6 m away. The 0.1 m of differences comes from inaccuracies of the curve fit. The measured and actual values are still within a reasonable range, but at greater distances the error may be amplified due to the nature of the given curve fit. The angle calculated using the camera was 40 degrees between the ROV and buoy. After calculating the actual angle using trigonometry the value was 36 degrees.



(a) Non-pitched test. **(b)** Pitched test.

Figure 9.10: Data from computer vision test.

In general the data generated by the computer vision code are well within an error margin and could therefore be sent to the controller and used as reference values for trajectory control.

Chapter 10

Discussion

In chapter 2, specific success criteria were created in order to limit the scope and also to allow the project to be evaluated. This discussion will focus on what was accomplished in terms of the success criteria and on the possible future steps to improve or advance the current project. The list of success criteria determined during the Problem Delimitation is shown below.

- Create a 3DOF mathematical model for the VideoRay
- Identify and estimate the model parameters
- Design controllers for world frame and body navigation
- Detect and identify a predefined object in water
- Implement a navigation system with minimizing the distance towards the detected object

In order to perform model based control, it was crucial to have a mathematical model with its associated parameters identified. In chapter 5, the general descriptions of the respective dynamics were determined for all of the fully controllable degrees of freedom. Due to the choice of operating point, this being at 30% thrust, the coupling dynamics between the translational and angular motion could be neglected. This was confirmed through experimentation during the parameter estimation, since the pitching induced from the surge motion was shown to be insignificant. In order to further expand the mathematical model, inclusion of the coupling dynamics would enable more functionality. By modelling the dynamics so that the there is a coupling between the full 6 degrees of freedom it may be possible to enable the ROV to roll or pitch, without having direct control over those movements.
The definitions for the drag and thrust forces from the mathematical modelling were required whilst creating the curve fits for the parameter estimation. While performing the curve fitting, the data at the higher velocities and forces were weighted less than the lower-mid range, since the higher velocities introduced unwanted dynamics and velocities greater than suitable for the given testing environment. When determining the yaw velocity, it became apparent that the modelling for the thrust torque and drag were lacking some of the dynamics induced from the rotation of the ROV,

and when performing the model validation for the yaw movement it was determined that the estimated data and model were acceptable. During the experiments for the parameter estimation, the computer vision software had not yet been completed. The finished software would have been beneficial and could aid in determining the surge velocity, along side using the video recordings, and this could help improve the quality of the coefficients.
The accuracy of the determined thrust characteristics were limited by the equipment and data available. The official datasheet from VideoRay only provided a maximum thrust value. The experiments for the thrust in the surge direction were all performed with the described pulley system to halve the force, due to the limitations of the given newton meter. In order to better capture the characteristics at the lower thruster percentages the pulley system should have been removed when operating within the range of the meter. The heave force was, as mentioned, impacted severely by friction whilst determining the force in the upward direction. This turned out not to be an issue due to the similarity between the upward and downward thrust. Other equipment such as a waterproof force sensor or an advanced rig with force cells or strain gauges may improve the overall quality of the thruster polynomials if redoing the parameter estimation.

The created mathematical model and the estimated parameters were validated and were particularly precise around the choice of operating point. Therefore, the model was accepted as possible representation of the VideoRay and used during the controller design phase. The controllers were designed with the indoor testing environments in mind, since the controller should be focused around the size limitations of the given testing environments available to us. The controller for the body frame was designed with as little overshoot as possible in the surge direction (d_u) in order to avoid hitting the object that is being tracked. Similarly, the depth of the ROV with respect to object (d_w) also had little to no overshoot so as to avoid hitting and damaging the vehicle whilst positioning it.

When attempting to detect the given object, one of the limitations of the given sonar setup was the lack of access to the SDK from BlueView, and also the use of a single forward facing camera. By analysing the images captured by the sonar and performing real-life measurements to confirm those readings, it was determined that the sonar is capable of detecting an object at a distance and return an image that can be processed accordingly. However, the processing required is unknown due to not having the SDK for the sonar available.
The use of a single forward facing camera has its limitations since the camera lacks depth perception. The creation of the curve fit for pixel to distance ratio, for the given camera and object does help. This approach has the possibility to be really inaccurate at certain distances and requires great accuracy while performing the curve fit. A possible solution to increase the feasibility of the single camera would be to add a laser to either side of the camera, where the angle between the laser and center of the camera can be determined using trigonometry. A disadvantage to the laser setup would be that the camera needs to look directly at the object since both the laser and the centre of the camera need to be pointed at the object. A second, and much better, solution would be the addition of stereo vision. Stereo vision is

the use of two cameras that are on the same platform with a set distance between them, and the calculations of the depth in the given scenario is done by comparing the two pictures. Due to lack of experience with stereo vision, it is uncertain how well it would perform underwater and at what operating range. A third solution would be to use a sonar that has a more suitable operating range or to use two sonars, where one of them has an operating range at the longer distances and the other at the shorter distances.

In general, some of the limitations of using computer vision for object detection underwater is determining the different threshold values needed for detecting a particular object. When operating closer to the surface of the water there is an increased possibility of encountering false positives due to the reflection of the object. The computer vision software can be extended in order to combat these false positive by adding a weighting algorithm that uses a confidence level to determine which of the detected objects is the actual object. The confidence takes into an account the size of the object detected and also the previously stored position of the object. By comparing where the previous location of the object was, it possible to rule out newly detected objects that are a great distance away.
Besides using the computer vision as sensory data for the body frame controller, the computer vision could act as aid for the operator of the ROV. It could display information about the detected objects and also warn the operator if there is an object to be aware of whilst manoeuvring the vehicle. The current implementation of the computer vision code has the capability of detecting anything circular and can be extended to detect any circular object with a given radius. As mentioned, due to the use of a single camera, the implementation relies heavily on the accuracy of the curve fit for the distance to pixel ratio of the detected object. Based on the data generated through the experiments and the following validation of the measurements, it is confirmed that the current state of vision code is suitable as a proof of concept for detecting and locating an object with respect to the ROV's body.

Due to the lack of time during the final stages of the project, it was not possible to perform the actual implementation of the created controllers alongside the computer vision system. The current state of the body and world frame controllers is that the software implementation has to be finalised. This includes finalising the implementation of the integrator, anti-windup scheme and the actual estimator. The world frame controller uses the on-board sensory information as feedback signals and is independent of the computer vision. The navigational controller also lacks the implementation of the state-space related components as mentioned previously, but the sensory information from the computer vision is available as a feedback signal. The next steps would be to finalise the state-space implementations and perform tests accordingly, in order to determine the accuracy of the theoretically developed controllers based on the validated model. The choice of operating point and if the thruster deadzones are significant cannot be validated until the implementations of the controller have taken place, therefore after some initial implementation it may be required to create a new model with a different operating point or the addition of deadzone compensation for the thrusters.

Chapter 11

Conclusion

Based on the experiments and results discussed in the previous chapter, we can conclude the majority of the success criteria were met. In order to create the model-based controller, extensive parameter estimation was performed, and the acquired parameters were then validated during model validation. Based on the populated mathematical model, the gains for the two controllers were determined and simulated on the non-linear model, in order to mimic the real-life performance. The computer vision software supplied adequate sensory information needed for the navigation controller, given that the object is pre-defined. Due to a lack of time, the proposed implementation did not take place, but based on the controller simulations and computer vision experiments, the two separate proof of concepts can be combined into a single functioning system. Using additional or improved equipment for the computer vision, as proposed during the discussion, the accuracy and flexibility of the object detection system can be improved and expanded to other objects.

The respective results from the computer vision and the navigational controller simulations seem to indicate the possibility of implementing semi-autonomous capabilities on an existing ROV platform.

References

[1] International Chamber of Shipping, "Shipping and World Trade." http://www.ics-shipping.org/shipping-facts/shipping-and-world-trade, (Accessed March 2017).

[2] International Maritime Organization, "IMO profile." https://business.un.org/en/entities/13, (Accessed March 2017).

[3] National Oceanic and Atmospheric Administraion (NOAA), "National Ocean Service." http://oceanservice.noaa.gov/facts/exploration.html, (Accessed March 2017).

[4] ROV - Marine Technology Society, "ROV Categories." http://www.rov.org/rov_categories.cfm, 2017.

[5] ROV - Marine Technology Society, "What is an ROV?." http://www.rov.org/rov_overview.cfm, 2017.

[6] Nautilus Live. http://www.nautiluslive.org/, 2017.

[7] R. Capocci and G. Dooly, "Inspection-class remotely operated vehicles - a review." http://www.mdpi.com/jmse/jmse-05-00013/article_deploy/html/images/jmse-05-00013-g001-550.jpg, 2017.

[8] C. Mai, S. Pedersen, L. Hansen, K. L. Jepsen, and Z. Yang, "Modeling and Control of Industrial ROV's for Semi-Autonomous Subsea Maintenance Services," *IFAC Congress*, 2017.

[9] The society of Naval Architects and Marine Engineers (SNAME), "Nomenclature for treating motion of a submerged body through a fluid," *Technical and Research Bulletin No.1-5*, 1950.

[10] W. Hackmann, *SEEK and STRIKE. Sonar, Anti-Submarine Warfare and the Royal Navy 1914-54*. HMSO Science Museum. 1984 1st, 1984.

[11] R. E. Hansen, "Introduction to sonar, Course materiel to INF-GEO4310, University of Oslo." https://www.uio.no/studier/emner/matnat/ifi/INF-GEO4310/h12/undervisningsmateriale/sonar_introduction_2012_compressed.pdf, Autumn 2012.

[12] HyperPhysics, "Speed of Sound in Various Bulk Media." http://hyperphysics.phy-astr.gsu.edu/hbase/Tables/Soundv.html#c1, (Accessed April 2017).

[13] Project Studio Handbook, "Why do sound waves lose energy and diminish." http://www.projectstudiohandbook.com/videos/playlists/sound-wave-theory/why-do-sound-waves-lose-energy-and-diminish/why-do-sound-waves-lose-energy-and-diminish-script.html, (Accessed April 2017).

[14] BlueView, "BlueView P450 specification." http://www.blueview.com/products/2d-imaging-sonar/pseries-archives/p450-series/, (Accessed April 2017).

[15] BlueView, "BlueView P-450-45-I SONAR." http://download.videoray.com/documentation/v1_3_1/pro4/html/acc_blueview.html, (Accessed April 2017).

[16] VideoRay, "Videoray 4 with blueview sonar." https://www.inspectahire.com/media/products/545cb1387b746pro4.jpg, (Accessed April 2017).

[17] VideoRay, "PRO 4 PLUS BASE." http://www.videoray.com/images/specsheets/2014/2014_PRO4PLUSBASE_FINAL.pdf, 2014.

[18] VideoRay, "VideoRay 4 Plus Base." http://www.videoray.com/homepage/new/professional-rovs/videoray-pro-4/pro-4-plus-base.htm, (Accessed April 2017).

[19] VideoRay, "Videoray tether." http://www.videoray.com/homepage/new/tether.html, (Accessed April 2017).

[20] Subconn, "Male SubConn Micro Low Profile 9 tether connector." https://www.macartney.com/what-we-offer/systems-and-products/connectivity/subconn, 2017.

[21] VideoRay, "Videoray tether pinout." http://www.videoray.com/support/manuals.html, 2017.

[22] VideoRay, "Videoray thrusters." http://www.videoray.com/images/specsheets/2013/Brushless_Thrusters.pdf, (Accessed April 2017).

[23] D. Bryant for Maritime Musings, "Kort nozzel." http://images.maritimeprofessional.com/images/storage/kort-a.png, February 14, 2012.

[24] N. Waytowich, K. Stilson, and S. Kennett, "BLUE RAY ROV." http://projects.ccec.unf.edu/oilspill/docs/UNF_ROV_Technical_Report.pdf, 2010.

[25] VideoRay, "VideoRay Thruster Shaft." http://download.videoray.com/documentation/pro_4_mnt_a/html/_r_VR-PRO4-01-0001_r_MAR-004.html, (Accessed April 2017).

[26] VideoRay, "vrLib SDK." http://download.videoray.com/developer/, (Accessed May 2017).

[27] VideoRay, "vrLib SDK Documentation." http://download.videoray.com/developer/docs/vrLib/Index.html, (Accessed May 2017).

[28] SlimDX. https://slimdx.org/, (Accessed May 2017).

[29] SlimDX, "SlimDX Documentation - Software Development Kit ." https://slimdx.org/docs/#SlimDX_Software_Development_Kit, (Accessed May 2017.

[30] SlimDX, "SlimDX Github Repository." https://github.com/SlimDX/slimdx, (Accessed May 2017).

[31] R. E. Hansen, "Introduction to imaging, Course materiel to INF-GEO4310, University of Oslo." http://www.uio.no/studier/emner/matnat/ifi/INF-GEO4310/h12/undervisningsmateriale/imaging-kap1.pdf, Autumn 2012.

[32] Terratec, "Terratec G1 Capture Card." http://terratec.com/details.php?artnr=10680#.WRt09Wh96Mo, (Accessed May 2017).

[33] OpenCV, "OpenCV Library." http://opencv.org/about.html, (Accessed May 2017).

[34] Emgu, "Emgu vision library." http://www.emgu.com/wiki/index.php/Main_Page, (Accessed May 2017).

[35] http://www.astronomersgroup.org/, "EM spectrum." http://www.astronomersgroup.org/images/EMspectrum.jpg, 2008, November 11(Accessed May 2017).

[36] S. Mallick, "Why does OpenCV use BGR color format?." https://www.learnopencv.com/why-does-opencv-use-bgr-color-format/, Septempet 27, 2015 (Accessed May 2017).

[37] Kirupa, "RGB spectrum." https://www.kirupa.com/images/rgb_image.png, (Accessed May 2017).

[38] R. Sporea, "Color Models." http://www.photozone.de/colorimetric-systems-and-color-models, (Accessed May 2017).

[39] A. W. R. Fisher, S. Perkins and E. Wolfart., "Mathematical Morphology." https://homepages.inf.ed.ac.uk/rbf/HIPR2/matmorph.htm, 2003 (Accessed May 2017).

[40] A. W. R. Fisher, S. Perkins and E. Wolfart., "Mathematical Morphology, Erosion." https://homepages.inf.ed.ac.uk/rbf/HIPR2/erode.htm, 2003 (Accessed May 2017).

[41] A. W. R. Fisher, S. Perkins and E. Wolfart., "Mathematical Morphology, Dilation." https://homepages.inf.ed.ac.uk/rbf/HIPR2/dilate.htm, 2003 (Accessed May 2017).

[42] U. Sinha, "Circle Hough Transform." http://www.aishack.in/tutorials/circle-hough-transform/, (Accessed May 2017).

[43] Microsoft, "What Is Windows Communication Foundation." https://msdn.microsoft.com/en-us/library/ms731082(v=vs.110).aspx, (Accessed May 2017).

[44] W. Wang and C. Clark, "Modeling and Simulation of the VideoRay Pro III Underwater Vehicle," *IEEE OCEANS 2006*, 2006.

[45] J. R. Taylor, *Classical Mechanics*, ch. 2. University Science Books, 2005.

[46] PASCO, "Wireless Force Acceleration Sensor." https://www.pasco.com/prodCatalog/PS/PS-3202_wireless-force-acceleration-sensor/index.cfm, (Accessed May 2017).

[47] PASCO, "Wireless Force Acceleration Sensor Manual(PS-3202)." https://www.pasco.com/file_downloads/Downloads_Manuals/Wireless-Force-Acceleration-Sensor-Manual-PS-3202.pdf, (Accessed May 2017).

[48] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*, ch. 7. Pearson, 7. ed., 2015.

[49] R. C. Dorf and R. H. Bishop, *Modern Control Systems*, ch. 11. Prentice Hall, 12. ed., 2011.

[50] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*, ch. 3. Pearson, 7. ed., 2015.

[51] B. Friedland, *Control System Design - An introduction to State-Space Methods*, ch. 9. McGraw-Hill, 1. ed., 1986.

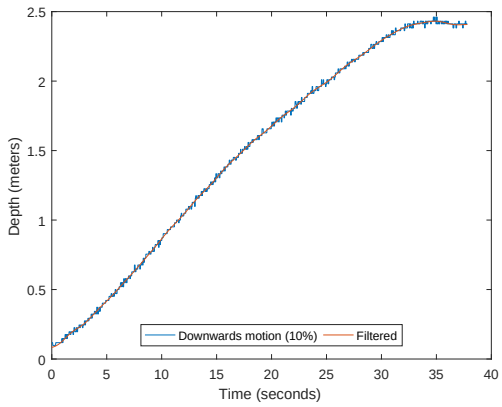
[52] Math.NET, "Math.NET Numerics." https://numerics.mathdotnet.com/, (Accessed May 2017).

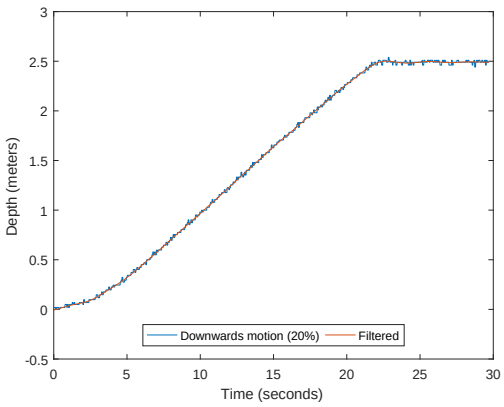
Chapter 12

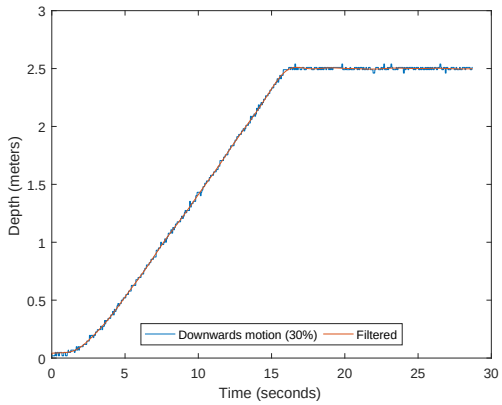
Appendix

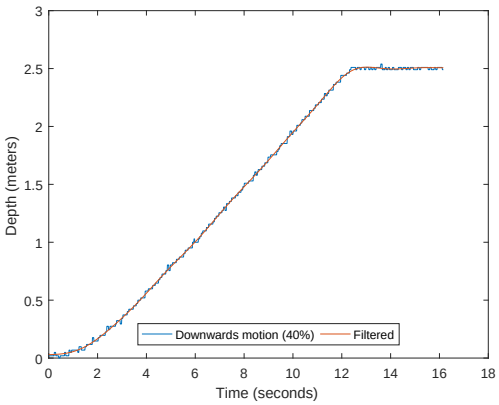
12.1 Depth Measurements

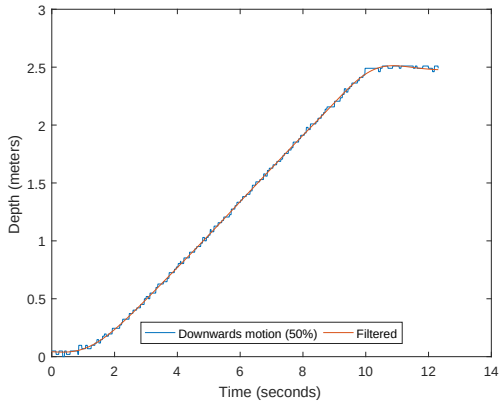
12.1.1 Downward

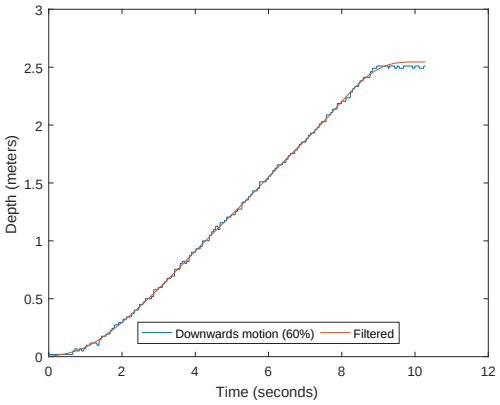


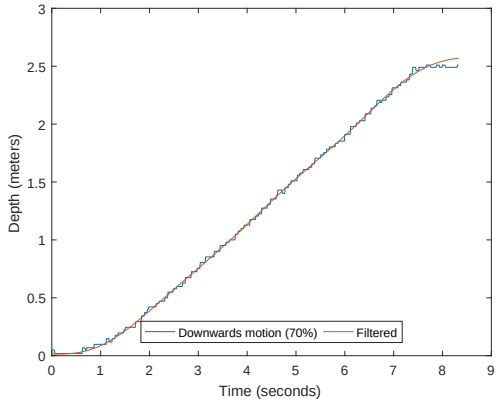


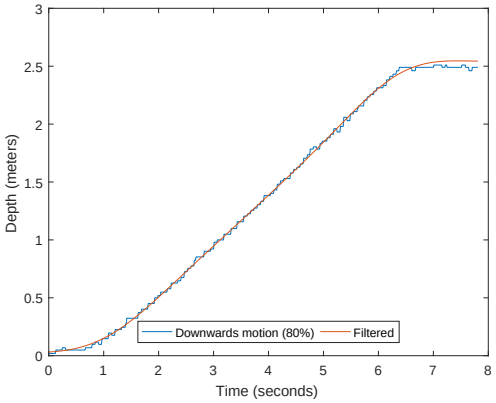


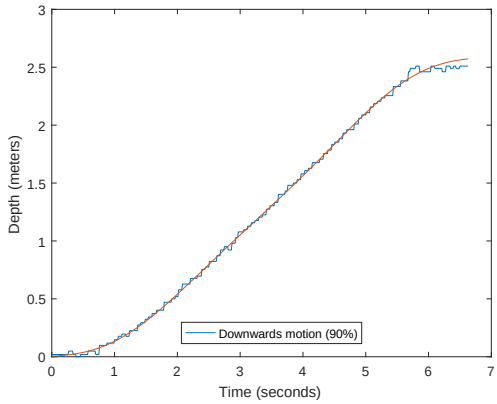


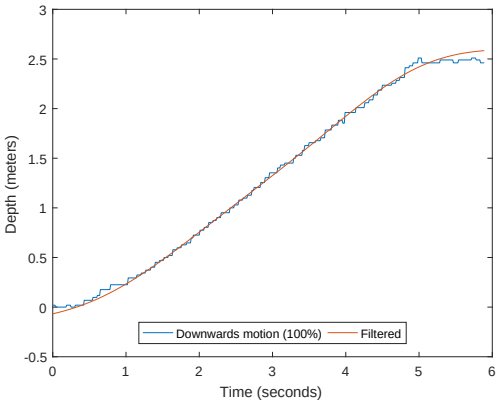

Depth (meters)
Time (seconds)
Downwards motion (50%)
Filtered

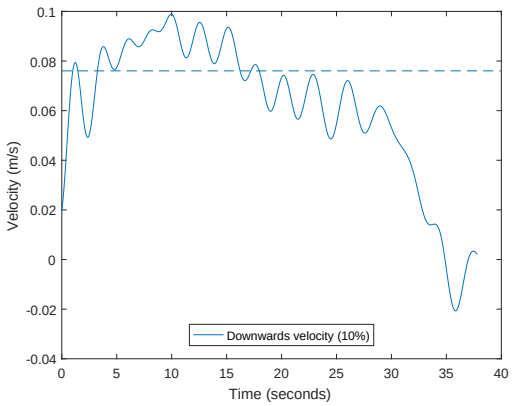

Depth (meters)
Time (seconds)
Downwards motion (60%)
Filtered

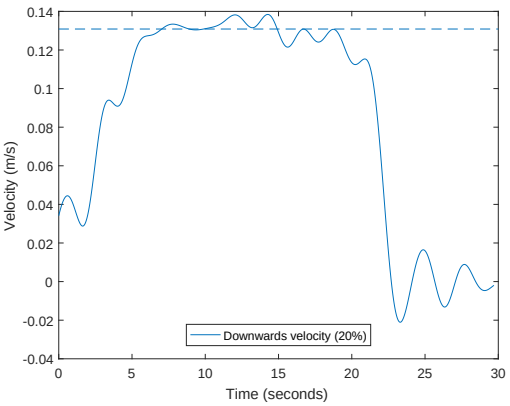

Depth (meters)
Time (seconds)
Downwards motion (70%)
Filtered

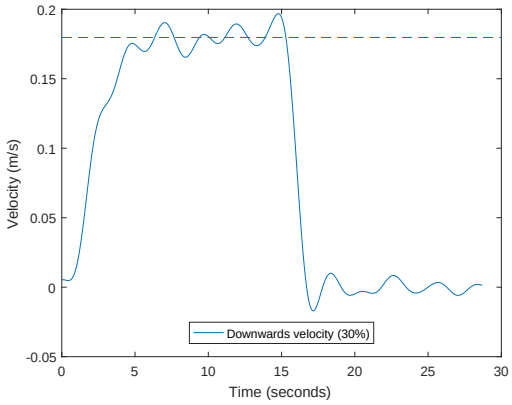

Depth (meters)
Time (seconds)
Downwards motion (80%)
Filtered

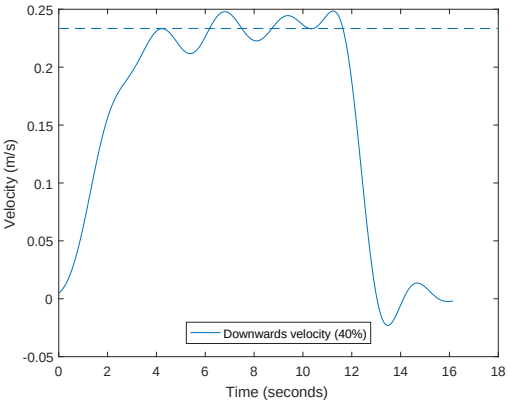

Depth (meters)
Time (seconds)
Downwards motion (90%)

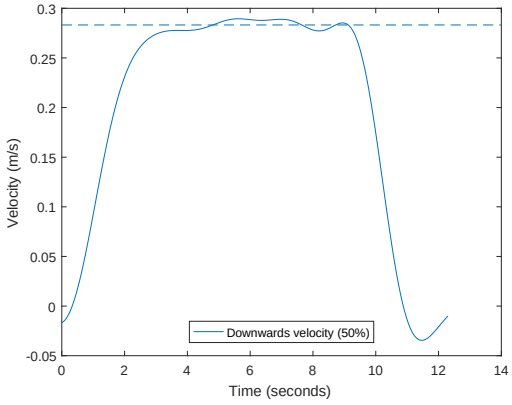

Depth (meters)
Time (seconds)
Downwards motion (100%)
Filtered

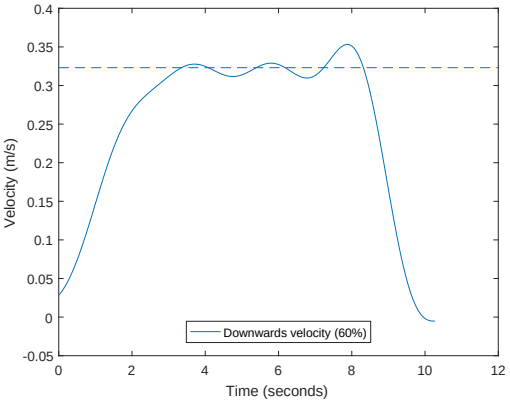

Velocity (m/s)
Time (seconds)
Downwards velocity (10%)

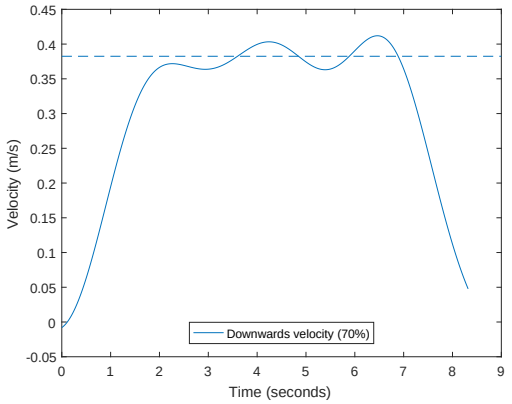

Velocity (m/s)
Time (seconds)
Downwards velocity (20%)

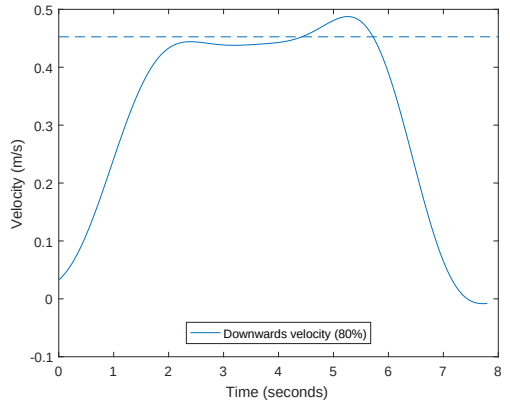

Velocity (m/s)
Time (seconds)
Downwards velocity (30%)

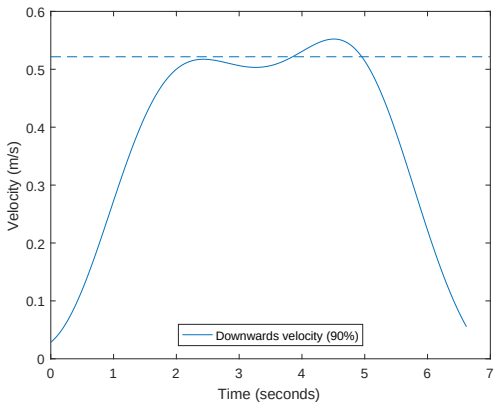

Velocity (m/s)
Time (seconds)
Downwards velocity (40%)

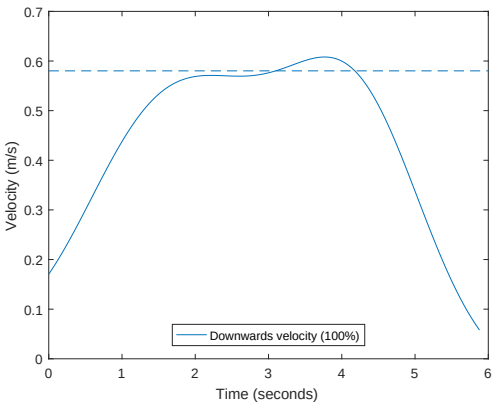

Velocity (m/s)
Time (seconds)
Downwards velocity (50%)


Velocity (m/s)
Time (seconds)
Downwards velocity (60%)

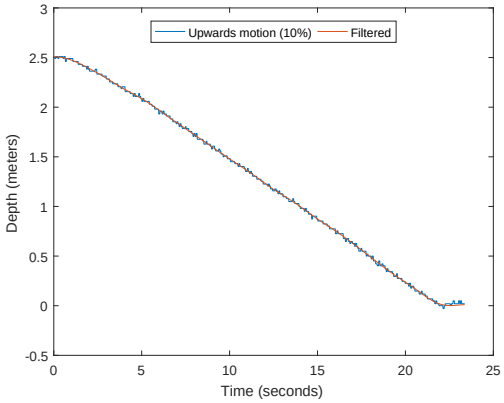

Velocity (m/s)
Time (seconds)
Downwards velocity (70%)

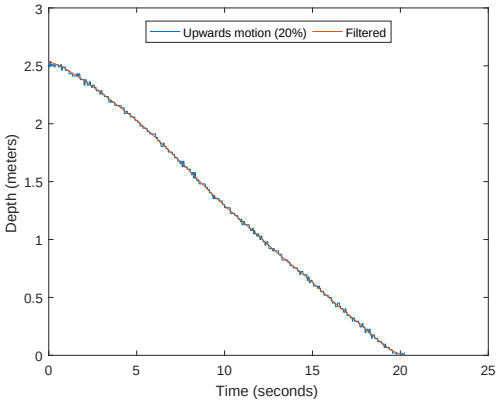

Velocity (m/s)
Time (seconds)
Downwards velocity (80%)

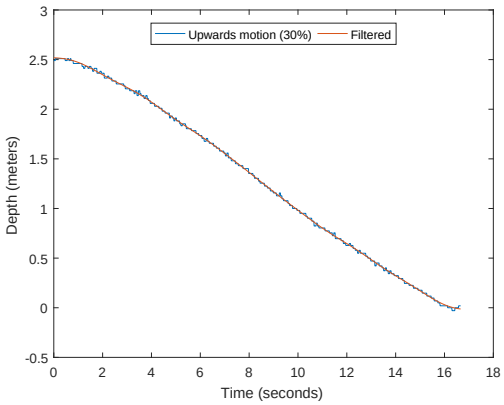

Velocity (m/s)
Time (seconds)
Downwards velocity (90%)

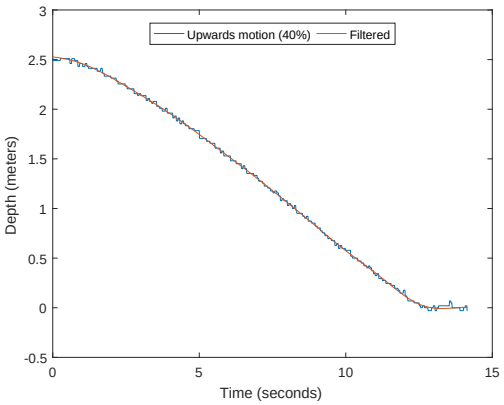

Velocity (m/s)
Time (seconds)
Downwards velocity (100%)

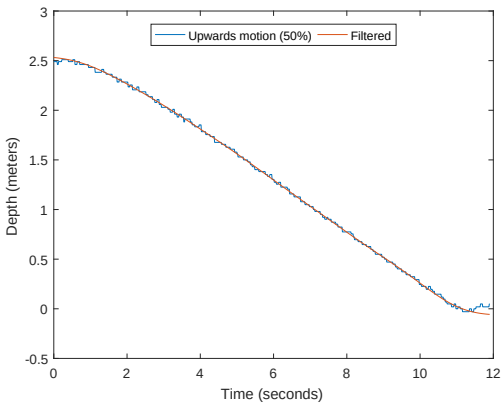
12.1.2 Upward

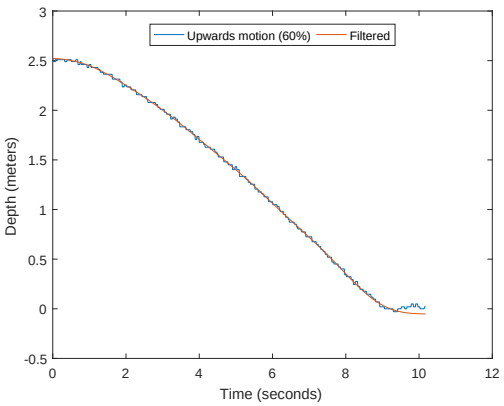

Upwards motion (10%)
Filtered
Depth (meters)
Time (seconds)

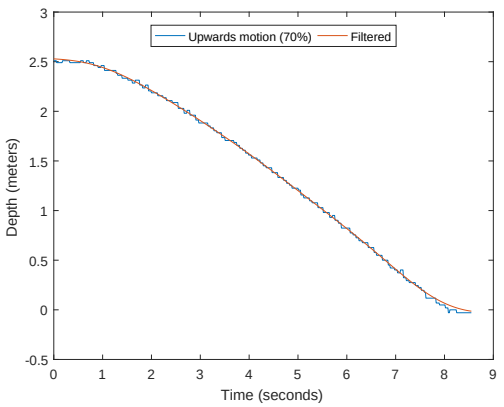

Upwards motion (20%)
Filtered
Depth (meters)
Time (seconds)

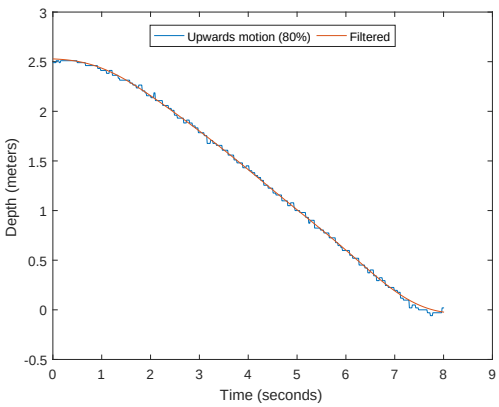

Upwards motion (30%)
Filtered
Depth (meters)
Time (seconds)

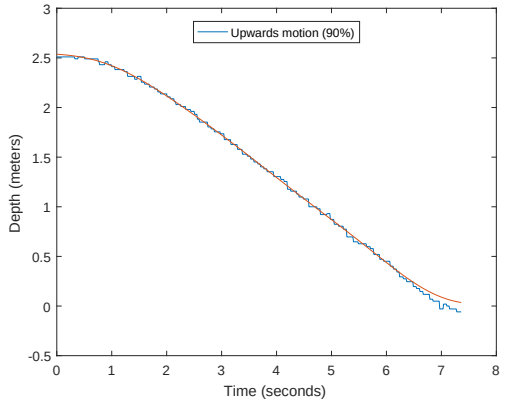

Upwards motion (40%)
Filtered
Depth (meters)
Time (seconds)

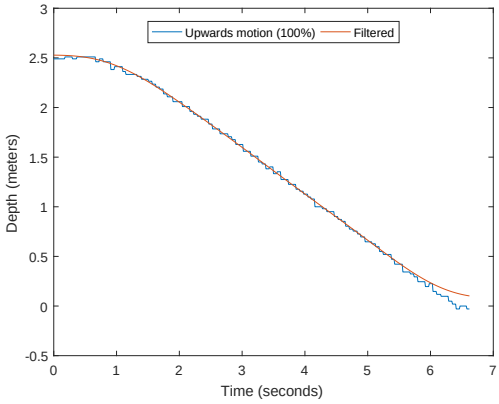

Upwards motion (50%)
Filtered
Depth (meters)
Time (seconds)

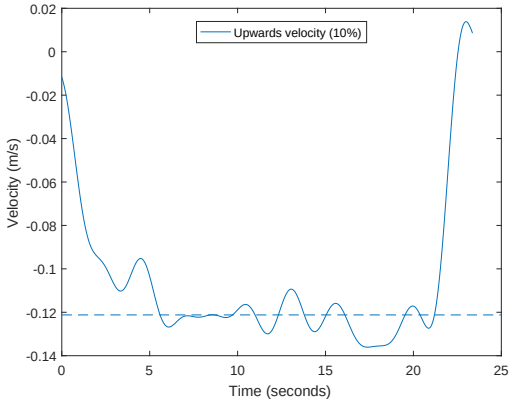

Upwards motion (60%)
Filtered
Depth (meters)
Time (seconds)

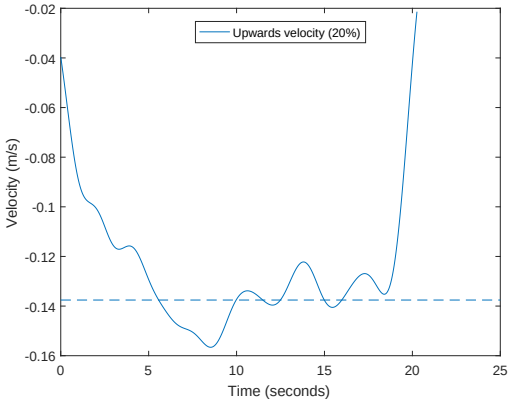

Upwards motion (70%)
Filtered
Depth (meters)
Time (seconds)

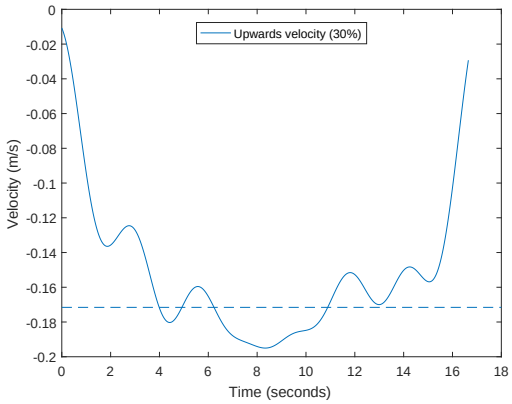

Upwards motion (80%)
Filtered
Depth (meters)
Time (seconds)

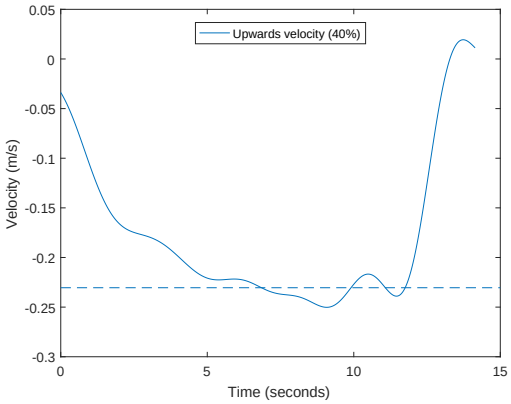

Upwards motion (90%)
Depth (meters)
Time (seconds)

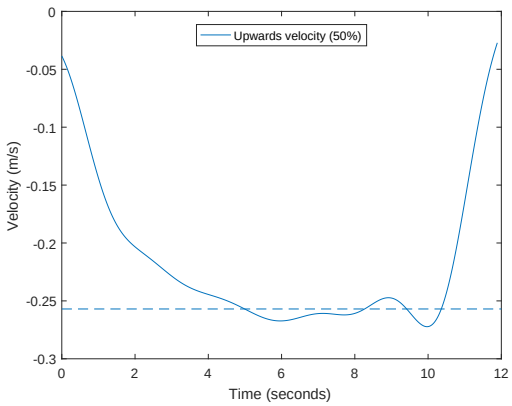

Upwards motion (100%)
Filtered
Depth (meters)
Time (seconds)

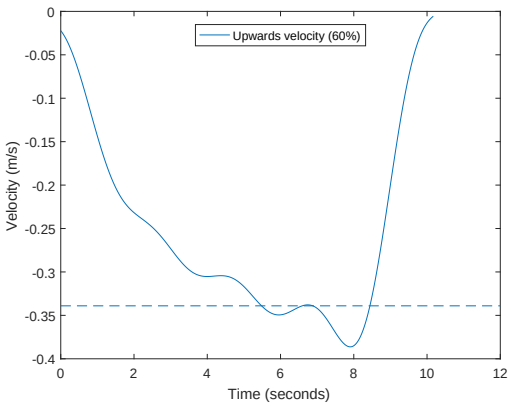

Upwards velocity (10%)
Velocity (m/s)
Time (seconds)

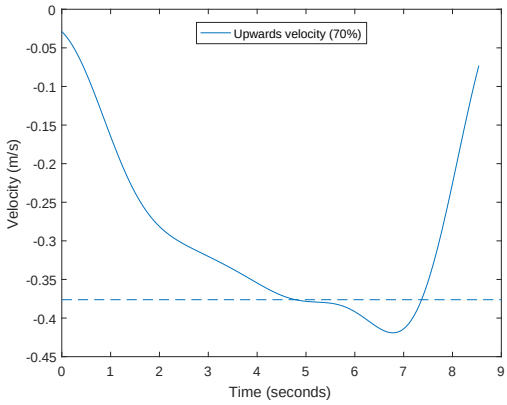

Upwards velocity (20%)
Velocity (m/s)
Time (seconds)

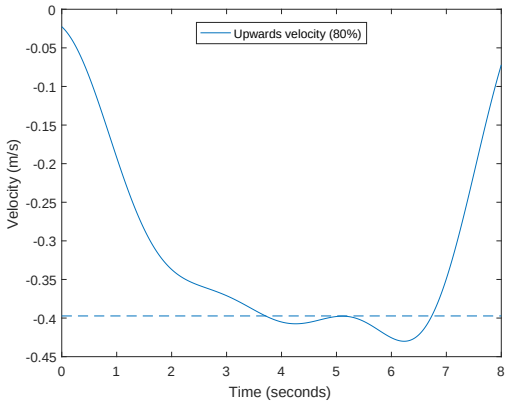

Upwards velocity (30%)
Velocity (m/s)
Time (seconds)

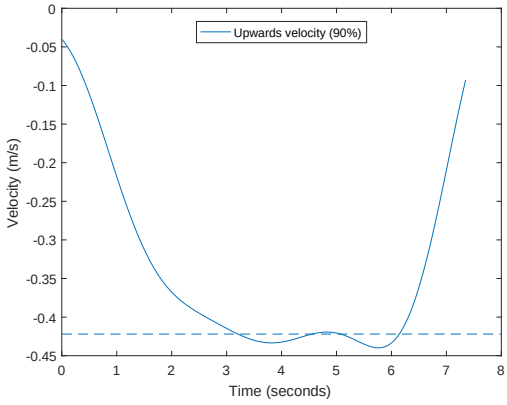

Upwards velocity (40%)
Velocity (m/s)
Time (seconds)

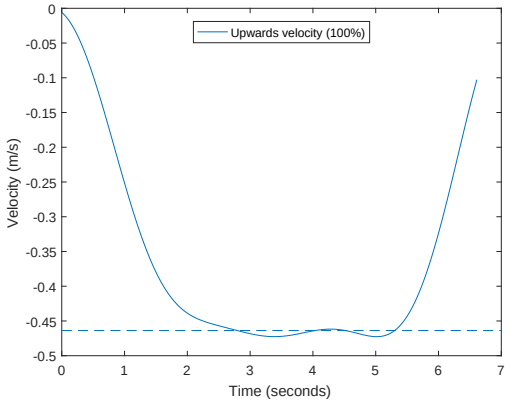

Upwards velocity (50%)
Velocity (m/s)
Time (seconds)


Upwards velocity (60%)
Velocity (m/s)
Time (seconds)

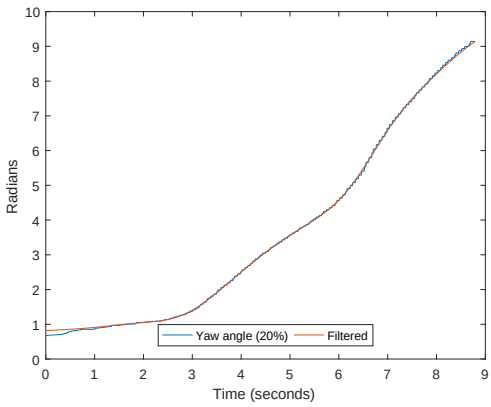


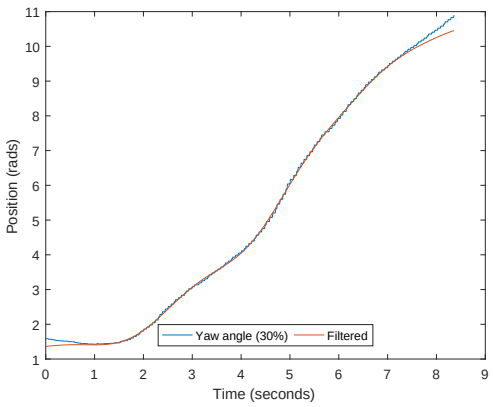


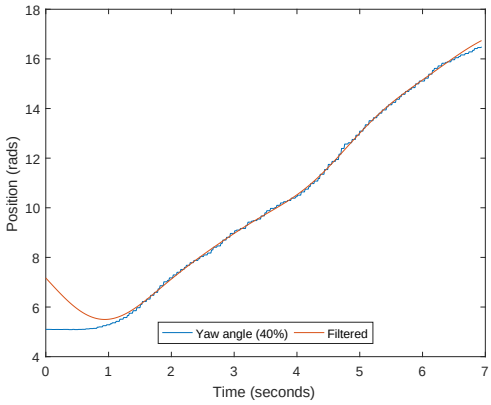


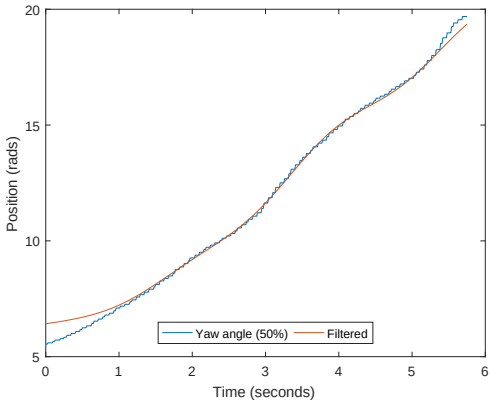


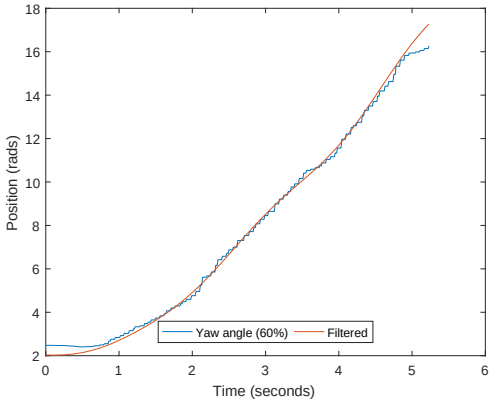
12.2 Yaw Measurements

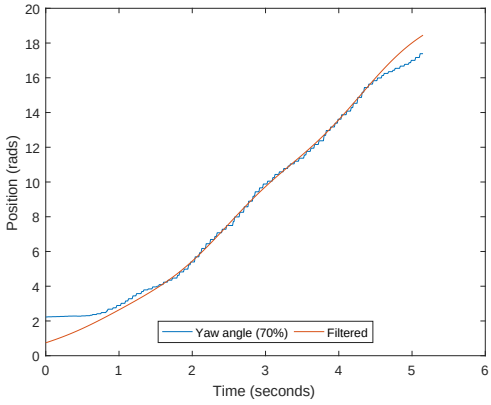


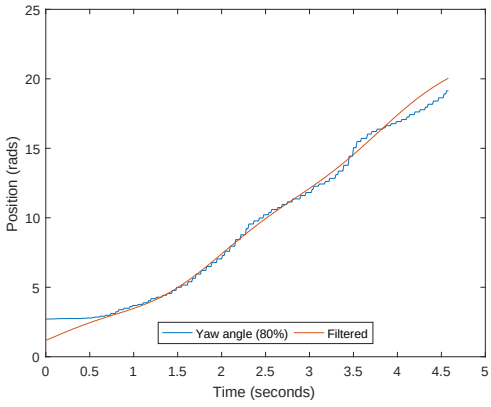


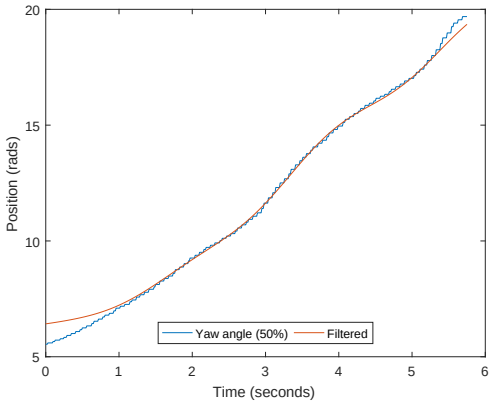

Position (rads)
Time (seconds)
Yaw angle (40%)
Filtered

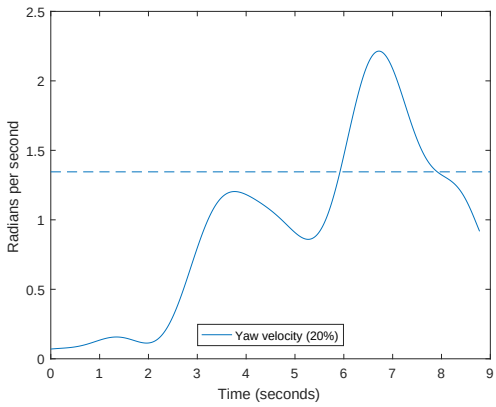

Position (rads)
Time (seconds)
Yaw angle (50%)
Filtered

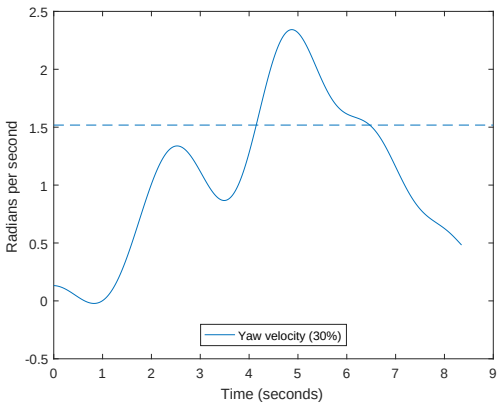

Position (rads)
Time (seconds)
Yaw angle (60%)
Filtered

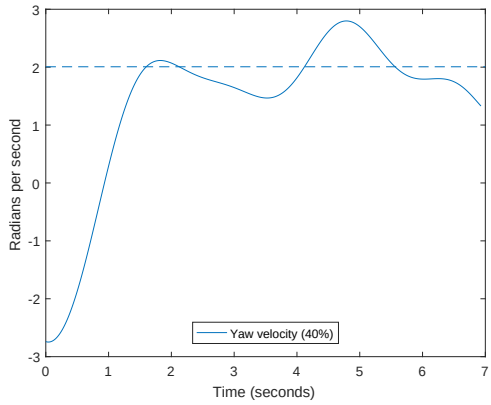

Position (rads)
Time (seconds)
Yaw angle (70%)
Filtered

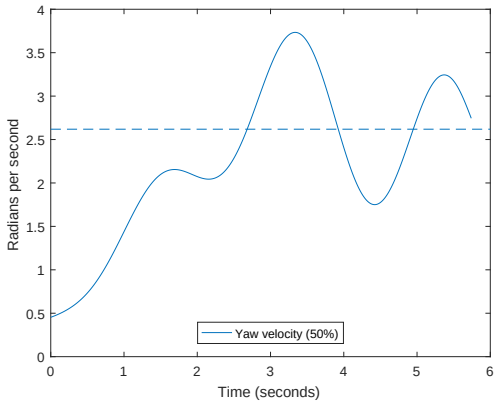

Position (rads)
Time (seconds)
Yaw angle (80%)
Filtered

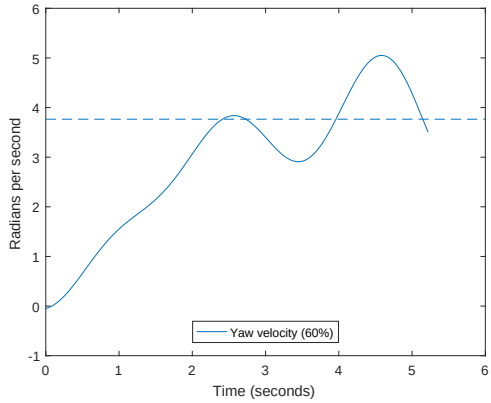

Position (rads)
Time (seconds)
Yaw angle (50%)
Filtered

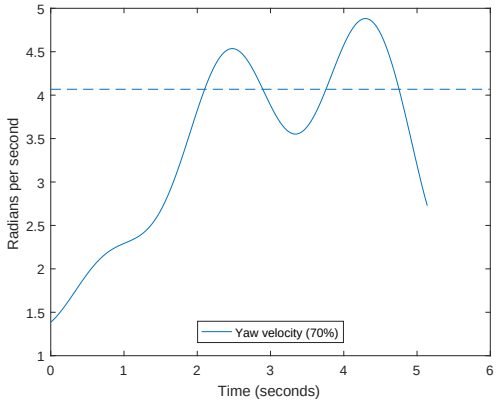

Radians per second
Time (seconds)
Yaw velocity (20%)

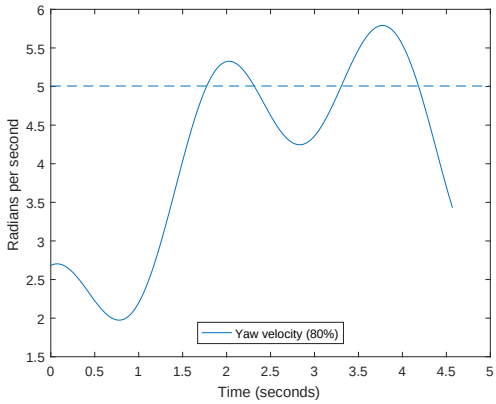

Radians per second
Time (seconds)
Yaw velocity (30%)


Radians per second
Time (seconds)
Yaw velocity (40%)


Radians per second
Time (seconds)
Yaw velocity (50%)


Radians per second
Time (seconds)
Yaw velocity (60%)


Radians per second
Time (seconds)
Yaw velocity (70%)


Radians per second
Time (seconds)
Yaw velocity (80%)