**ACIT5930**

**MASTER'S THESIS**

**in**

**Applied Computer and Information Technology (ACIT)**
**May 2022**

**Robotics and Control**

# Verification and Validation of ROV Simulation Using Experimental Data

Jeen Ann Abraham

**Department of Computer Science**
**Faculty of Technology, Art and Design**

# Verification and Validation of ROV Simulation Using Experimental Data

Jeen Ann Abraham

**Verification and Validation of ROV Simulation Using Experimental Data**

http://www.oslomet.no/

Printed: Oslo Metropolitan University — OsloMet

# Abstract

Remotely operated vehicles (ROVs) are used widely in industries such as oil and gas, offshore, and renewable industries for various applications such as inspection, maintenance, and repair of their infrastructure. There is a great and underutilized opportunity for their scientific use across the world's oceans with depths varying from a few hundred meters to depths greater than 4,000 m. The ROV operations are normally controlled by a human operator, and their hydrodynamic performance is far from the ideal behavior. As a result, the outcome of the task primarily depends on the operator's experience. Model simulations that have close to real behavior can help in the training, development, testing, and verification of ROVs under varying test conditions and investigate critical situations cost-effectively without risking the safety of the user as well as equipment safety.

This research was focused on developing a high precision ROV simulation platform under Robot Operating System(ROS) and MATLAB/Simulink environment whose behavior stays as close to that of the hardware (IKM Subsea's Merlin) experimental data. The performance of the ROV simulator was tested and validated against the real-time data captured from the Merlin. In the early phase, the focus was more on understanding the problem. To get an insight into the components and working principle of an ROV, several related research works were reviewed, followed by analyzing the relevant works done under this research topic, and the simulation software environment was set up. Later, a great deal of attention was directed toward the ROV kinematics and behavior of the dynamic model from the existing simulator. Data collection was done for both simulators and hardware. Semi-automated scripts were made for data collection and graphical representation to perform a comparative study. Also, work has been done for ROV hydrodynamic modeling and simulation in MATLAB to test the simulator behavior across independent simulation environments. Simulation and real-time experiments along with the results were presented to demonstrate the reliability of the ROV simulator over the hardware and thereby signify the possibility of replacing physical hardware with virtual hardware for various applications.

# Preface

## The Audience

In the modern era, simulation is extensively being used as a tool to increase product safety as well as production capacity. The purpose of this thesis is to study how close are software simulation results of the ROV simulator against the real-time data captured from the hardware (Merlin UCV). This report is designed in a way that the reader can easily capture the research done during the work with basic knowledge of the ROS framework, MATLAB/Simulink framework, and control engineering. Each chapter begins with a chapter outline describing the topics the reader can expect to encounter.

## Acknowledgments

I would like to express my deep sense of gratitude towards Prof. Alex Alcocer [Professor, Faculty of Technology, Art and Design, Department of Mechanical, Electronics and Chemical Engineering, Teaching and Research, Oslo Metropolitan University, Oslo] who was my dissertation guide and Prof. Vahid Hassani [Vice-Dean, Faculty of Technology, Art and Design, Department of Mechanical, Electronics and Chemical Engineering, Teaching and Research, Oslo Metropolitan University, Oslo] who was my co-supervisor, for their guidance, interesting discussions and healthy criticism during the work, which has been a great motivation to me as a way forward. Furthermore, I would like to thank Ph.D. Candidate, Håkon Teigland for the support, knowledge exchange on the existing simulator of Merlin, and a good collaboration with the development of the simulator.

Also, I extend my sincere thanks to IKM Subsea AS for assisting with necessary support and granting an opportunity to work with the latest ROV technology as well as providing ROV data to use in my simulations. Lastly, I'm extremely grateful to all of the Department faculty members at the Oceanlab facility for their help and support.

Oslo, May 16, 2022

Jeen Ann Abraham

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## Chapter Outline

This chapter serves to introduce the subject of the research and details the purpose of the study to the reader. It justifies the research based on the identified problem, followed by exploring the objectives of the study. Once the objectives have been defined, the contribution of the study has been explored. Subsequently, the chapter discusses the followed methodology in this research and how the thesis has been structured.

## 1.1 Unmanned Underwater Vehicle - A Brief Discussion

Due to zero visibility, extreme environment, and high pressure, development in the field of deep-sea exploration was running at a slow pace during the early 80s. Additional factors that slag the exploration are lack of natural light because of scattering and attenuation, and high costs for development and testing. Recently the advancement in engineering and technology, accompany more modern and sophisticated tools that can yield high performance even in extreme environments. This plays a major role in the rapid development in the field of deep-sea exploration. An Unmanned Underwater Vehicle (UUV) is one of the best examples to explain the above statement. As the name suggests, an Unmanned Underwater Vehicle is a system that operates in a sub-aquatic environment without human inhabitants which are basically of two types, Remotely Operated Vehicles, and Autonomous Underwater Vehicles. The former one is called Remotely Operated Vehicles (ROVs) which are tethered and require an isolated human operator mostly from an onboard vessel. The first ROV ever built was named "Poodle" built-in 1953 by Dimitri Rebikoff [1]. The latter is called Autonomous Underwater Vehicles (AUVs) which are self-governing. In the absence of natural light in the deep sea, both the above vehicles utilize artificial light for illumination and advanced cameras to capture underwater images for gathering the required information. The captured images from cameras located at different positions of the UUVs are combined depending on the geometric relationship between the camera, light source, and the object to obtain perfect deep-sea images.

ROV simulator is a PC-based simulation program whose main purpose is pilot training and familiarization of ROV user controls. By utilizing ROV simulation and applying unique solutions to problems, organizations can significantly reduce engineering costs and risks, as well as safety issues encountered later in a project. The simulation can be recorded and replayed as a full 3D representation, which allows the user to visualize various views from any camera on the ROV, as well as the free-camera views of the work environment that can be seen from any angle, which will help the user to identify any issues in early stage. Deep-sea underwater image simulators which provide high precision pictures at interactive frame rates are still in the development stage due to various factors. Deep-sea exploration can be made achievable by integrating the simulator and UUV, thereby increasing the performance of the complete system and eliminating the need for sophisticated sensors which will reflect positively on the engineering cost.

## 1.2 Motivation and Need of Research Topic

Equipped with different tools and sensors, ROVs are highly maneuverable and are frequently used in a wide variety of industries such as oil and gas, offshore energy, military, shipping, search and rescue, aquaculture, and marine biology. The size and prices can vary depending on their

functionalities. They let the operators to capture photo and video footage for port inspection and monitoring, harbors and vessels, bring innovation to pipe inspections, locate underwater targets and ocean exploration in depths. The ROV missions are normally controlled by a human operator, and their hydrodynamic performance is far from the ideal behavior. As a result, the outcome of the task primarily depends on the operator's experience. Even though autonomy can bring a solution to mitigate the above risk, the solution itself produces another risk of safety for testing and verifying control algorithms.

An imitation of similar operations in a simulation environment can be a helpful tool from the user's perspective for both ROV and UAV operations. A simulation model can investigate outcomes for varying conditions, investigate critical situations without any risk and finally the method is cost-effective. The primary challenge for simulations are developing relevant scenarios and to make the behavior of the model close to reality. Several factors contribute to the above when the ROV is in water and the complexity increases over depth which needs to be investigated and taken into account while developing a simulation model. The environment itself can change in a short period of time (For example: sometimes the sea can be rough and in the next few hours the sea can be calm) which makes the task even more challenging. All the above challenges motivated me to do this research in a distinct way to verify and validate a high-fidelity ROV simulation that can behave close to reality using real-time data.

## 1.3 Problem Statement

The prime objective of this study is to implement a dynamic model of an ROV in two independent simulation environments followed by validating the model behavior from simulation against the experimental data captured from the hardware. For this purpose, a detailed study has to be done on the comparison of physical parameters such as mass and inertia for individual links between the simulator and hardware. Few developments have already been done on the simulator model from IKM Subsea AS which was built as a ROS package containing the implementation of Gazebo plugins and ROS nodes necessary for the simulation of Merlin as shown in Figure 1.1. The work stated in this thesis is an extension of the previous simulator and is focused on, how effectively real data captured from an ROV can be used for improving the quality of the simulator. The main goal also involves investigating various ways to perform the testing and verification of the complete system.

## 1.4 Software Framework

The software simulation platforms MATLAB/Simulink and ROS are used in the development of the ROV simulator. MATLAB is an application software with a sophisticated programming language and an interactive environment that can be served for understanding mathematical concepts,

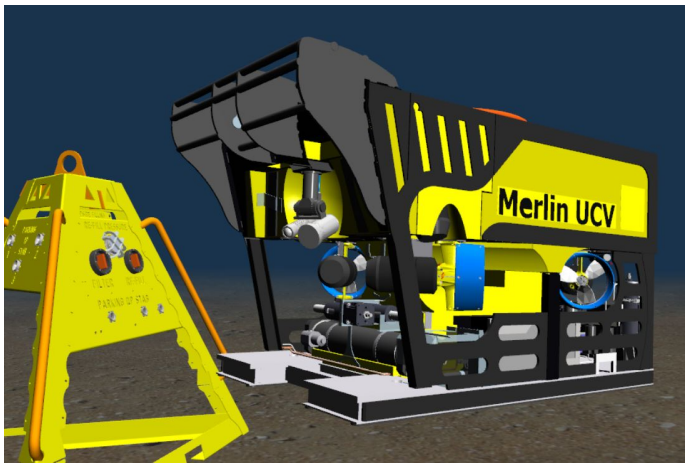



Figure 1.1: Merlin simulator

modeling, and testing complex systems interactively before encoding in a traditional programming language. On the other hand, the ROS package provides a vast collection of tools and packages allowing for rapid development of solutions without having to build systems from the ground up. This project employs the ROS Melodic branch, a ROS1 version 2 years old that is widely used and opted to avoid the newest branch Noetic Ninjemys as it is new, with fewer supported packages. The original UUV package [2] has been released for the ROS Melodic distributions which is the main reason behind choosing this distribution. Few additional reasons that contribute to selecting the ROS compared to other frameworks are flexibility, open-sourced, and a large community of developers and users with vast support. The message handling and package management in ROS are clean, straightforward, and easily understandable which is advantageous and improves the quality of the simulation.

### 1.4.1 MATLAB

MATLAB is a programming platform that has a high-level matrix-based language for numerical computation and application development. It allows to develop and test various algorithms, interface programs written in other languages, performs matrix manipulations, plotting and data analysis as well as creates complex system models and applications. It provides numerous libraries of mathematical functions for linear algebra, statistics, Fourier analysis, optimization, and solving ordinary differential equations. Inbuilt graphics helps the users to create custom plots for data visualization and data analysis.

### 1.4.2 ROS

ROS (Robot Operating System) is an open-source, meta-operating system commonly used for robotic development. It provides a vast collection of tools and packages allowing for rapid development of solutions without

having to build systems from the ground up. This project employs the ROS Melodic branch, a ROS1 version 2 years old that is widely used and opted to avoid the newest branch Noetic Ninjemys as it is new, with fewer supported packages.

- **ROS Master:** The ROS Master started through roscore is the name server for communication and node to node connections. This can run locally (which is the default) or run remotely on a separate station, this utility of communication is one of the strong suits of ROS, as it allows a strong station to run heavy loads remotely, instead of running them locally on the robotic unit, saving processing power.

- **Nodes, Subscribers and Publishers:** A node in ROS can be likened to an executable program. They are typically charged with handling a single task, such as sensors, movement commands, or control. ROS gives the advantage of having multiple small tasks solved in neatly arranged individual programs. Then passing the inputs, checks, or outputs between these different nodes through the ROS messaging system. These messages are sent from a Publisher, being the origin point, or received at a Subscriber through the topics. It is also possible to directly pass arguments to a topic from a terminal, commonly used for testing. The topics themselves are channels in the ROS messaging framework that runs in the background. These can be listened to (for instance by using subscribers) or passed information if the topic has been initialized.

- **Launch files:** Launch files are the premier way of running multiple ROS nodes with the necessary configuration easily. The launch files used in this project contain the required nodes, their arguments, file paths, initial configuration, and so on. Without launch files arguments would have to be hard-coded into the nodes themselves or added into the terminal window at execution. They also prevent the need for an individual terminal window for each node, as the launch file can allow multiple nodes to be executed in the same terminal.

### 1.4.3 Gazebo

For robotic systems, gazebo helps in performing dynamic simulation by utilizing the forces and torques that act on a body as inputs to the simulation environment. Gazebo is a robust physics engine with high-quality graphics and convenient programmatic and graphical interfaces which can simulate robots in complex environments, accurately and efficiently. It follows a user-friendly API for adding models and required interfaces for interaction with client programs. Figure 1.2 shows a general structure of components in Gazebo and how they interact each other to client programs. The purpose of Open Dynamic Engine (ODE) is to simulate the dynamics and kinematics associated with articulated rigid bodies [3] and it is employed in gazebo by a layer of abstraction between both gazebo and ODE. ROS provides necessary interfaces in the form of

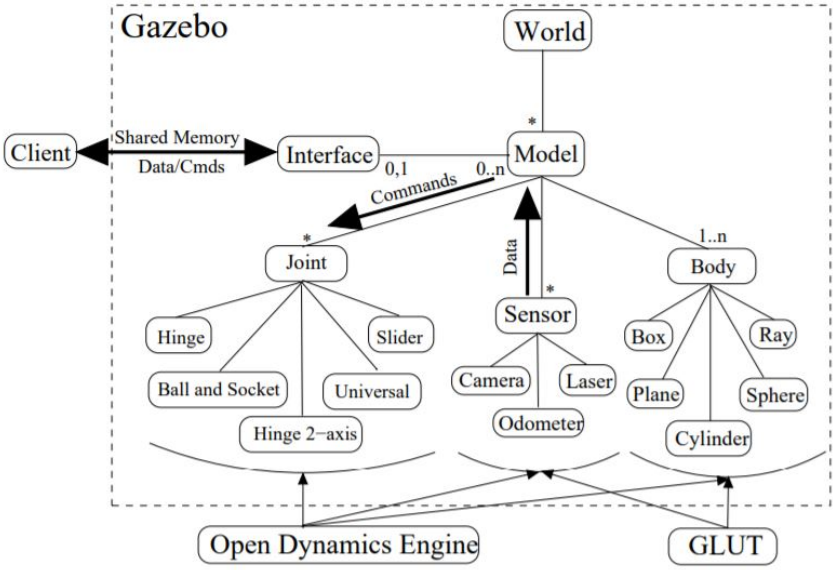



Figure 1.2: Overview of Gazebo components
[4]

packages that can simulate robots in Gazebo which can integrate with ROS using ROS messages, services and dynamic reconfigure. Gazebo uses physical engine for inertia, gravity, illumination, and so on. With the help of gazebo, the robots can be examined and evaluated in all the possible scenarios as for many applications it is necessary to test robot applications such as navigation, error handling, localization, grasping, and battery life. It is essential for avoiding any accidents or harm to the robot. It is much simpler and faster to test a robot using a simulator rather than testing the robot in the real world. The Gazebo consists of a client and server. The server evaluates all the physics and world, while the client works out the graphical interface for gazebo. There are five main components of Gazebo-world files, models, gzserver, gzclient, gzweb. World files contain all the elements such as lights, sensors, robots, and static objects required in the simulation, models represent the individual elements, gzserver, its main function is generating and populating a world by reading the world file, gzclient visualises the elements by connecting to gzserver, gzweb is the web version of gzclient.

## 1.5 Overview of Original Contributions

Indigenous work in this thesis focus on achieving the goals and objectives of this research work. Through this work, we conscientiously studied ROV kinetics and dynamics, specifically hydrodynamics modeling followed by deriving an ROV model for the six degrees of freedom in its operating range. During the initial phase, the studies were concentrated more on understanding the problem. Several related research works were

reviewed, followed by analyzing the relevant works done under the research topic which gave in-depth knowledge of theories behind ROV types and components, working principles, simulation environments, and modeling. After acquiring relevant knowledge and previous works done on the topic, the study moved on to the setting up of the simulation software environment which was one of the challenging tasks faced during the early phase. Lately, a great deal of attention was directed toward the ROV kinematics and behavior of the dynamic model from the existing simulator. Data collection was done for both the simulator and the hardware. Semi-automated scripts were made for data collection and graphical representation to perform a comparative study. Also, work has been done on ROV hydrodynamic modeling and simulation under MATLAB/Simulink environment. Even though the task was challenging, expected results were achieved in the end. Some of the noticeable contributions in this work are, the implementation of a model for Merlin ROV under MATLAB/Simulink environment and a feasible design for verification and validation with experimental data by comparing individual thrusters as input and achieved ROV velocities in six degrees of freedom, for the ROV simulator and the hardware.

## 1.6 Research Methodology

Quantitative approach has been exercised in this thesis as the research methodology and this section justifies, why the above method is best suited to achieve the research objective and how it helps this study to obtain valid and reliable results. The prime factor behind the selection of this method is that it utilizes measurable data to formulate facts and uncover patterns in research. At first, the studies were pointed in a direction towards understanding the problem and ROV concepts including kinetics and dynamics, especially hydrodynamics modeling followed by deriving an ROV model for the six degrees of freedom. The study later investigates previous works done on this topic and this formulates the drawbacks and limitations of the existing techniques and draws up the problem definition. Modeling the hardware based on mathematical equations and designing a method to validate the accuracy of the model simulations towards hardware was the next stage in this study. Data collections were made for both the simulator and the hardware for validating the model behavior from simulation. Semi-automated scripts were made to serve the above purpose. Finally, the method ends with a discussion and conclusion of achieved results that discuss the contribution and possibility for further works which can be done.

## 1.7 Thesis Outline

The Report is organized into 7 chapters. Chapter 1 presents a brief discussion on Unmanned Underwater Vehicle followed by motivation and need for a research topic, problem statement, software framework, an

overview of original contributions, and research methodology. The reader can acquire a background of the research work and the main goal that has to be achieved from this chapter. Chapter 2 concentrates on explaining the concepts and related work with a detailed discussion of ROV types and components, Merlin ROV. ROV modeling and approach toward achieving the research goal were discussed in Chapter 3. Chapter 4, covers the test plan and various experiments that have been conducted for studying the ROV simulator response under two independent simulation environments. it also details how the performance comparison study is done between simulator and hardware. Chapter 5 provides a concise summary of results for each experiment that has been done for studying the feasibility of the simulator. Chapter 6 interprets the results in detail and draws out their implications. Chapter 7 concludes the results with an overall answer to the main research question of this study along with future works that can be done.

# Chapter 2

# Background and Related Work

## Chapter Outline

This chapter outlines a review of the area being researched, current information surrounding the issue stated in the problem statement, previous studies on the same, and the relevant history on it. To easily capture the progress of research, many recent techniques in the research area are discussed and organized in a bottom to top hierarchical manner. The background section details the theory and concepts, which are part of this research while the literature review section details more on related work done in past and existing technology.

## 2.1 Background

### 2.1.1 ROV - A Brief Discussion

“ROV” stands for Remotely Operated Vehicle which is a highly maneuverable, unmanned underwater machine mainly intended to explore ocean depths by an operator at vessel on-board. The power and communication between the operator and the vehicle are done through a bunch of cables, or a tether that connects the ROV to the vessel, sending electrical signals back and forth. To achieve a high level of safety and efficiency for performing a sub-sea operation, ROVs are the best choice. These vehicles are vital in the offshore industry and some applications are as listed.

- **Drilling:** to monitor the Blow Out Preventer (BOP) and riser
- **Inspection maintenance and repair:** to perform inspection and tooling
- **Support during construction phase:** to perform surveys, touch-down monitoring and interfacing

### 2.1.2 Components of an ROV System

ROVs can vary in shape and size based on their applications, but they generally have some common elements. This section details the major components of a typical ROV system. The components can be classified into mechanical and electro/mechanical systems, primary subsystems, electrical systems, and control systems. The figure 2.1 details an overview of ROV components.

#### Mechanical and electro/mechanical systems

Frame, buoyancy, propulsion, and thrust cover this category. The frame is one of the critical components of an ROV and provides a firm platform for mounting the necessary mechanical, electrical, and propulsion components. The selection of materials for the frame depends on the criteria "maximum strength- minimum weight" and it can be made of materials ranging from plastic composites to aluminum tubing. The size of the frame depends on the weight of the complete ROV unit in the air, the volume of the onboard equipment, volume of the sensors and tooling, volume of the buoyancy, and load-bearing criteria of the frame. When it comes to underwater vehicle flotation systems, the main objective is to achieve a near-neutral buoyant state. The main requirements of these systems are that they should maintain their form and resistance to water pressure at the anticipated operating depth.

The propulsion system and thrust play a major role in the ROV design. In most cases, the type of thrusters, their configuration, and the power source to drive them takes priority in vehicle design over several other components. The propulsion systems in an ROV are mainly of three different types which are, electrical, hydraulic, and ducted jet

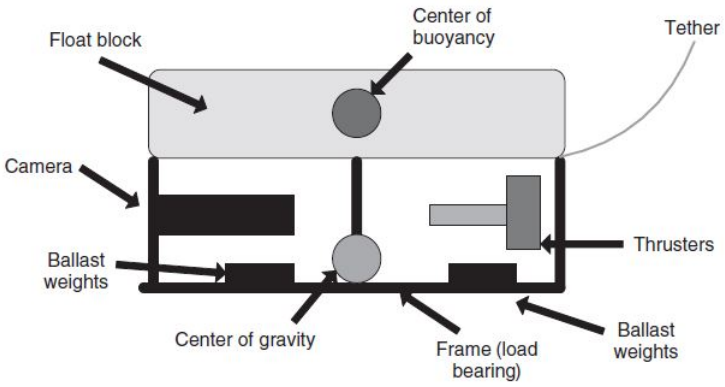



Figure 2.1: ROV submersible components
[5]

propulsion. Underwater electrical thrusters consist of a power source, an electric motor and its controller unit, thruster housing and attachment to the vehicle frame, gearing mechanism, driveshafts, seals and couplings, propeller, Kort nozzle, and stators.[5] Thrusters are solely responsible for the propulsion of ROVs which allows to control them in dynamic environments. Electric DC thrusters are widely used in Inspection-class ROVs using brushed DC motors, brushless DC motors, magnetically coupled motors, or rim-driven motors. The control of the motion of ROV depends on the configuration of the thrusters which includes size, available power, required thrust, and Degrees of Freedom (DOF) which describes every combination of a vehicle's movement, payload, and several other parameters.

**Primary subsystems**

The subsystems provide the ability to sense the environment either visually or through other means to the ROVs, for completing the task at a given location. Deep under the water, the operator can rely only on onboard cameras for the view of the ROV's work area. The lights provide illumination for the camera underwater since the availability of natural light decreases as the depth increases. Sensors play an important role in the proper functionality of ROVs as they sense desired physical phenomena without influencing the item being measured and are insensitive to other environmental or physical factors. ROV sensors are broadly divided into two categories namely survey sensors and vehicle sensors based on their responsibilities. These sensors include pressure sensors, depth sensors, rotation angle sensors, inclination sensors, proximity switches, sonars, cameras, pipe and cable tracking, motion sensors, and gyros which vary depending on electric connections, materials, mounting options, sizes, and water depth. applicability. The manipulator and tool pack sub-system consist of an electric motor running a worm gear to open and close valves or grabber arms for intervention duties.

**Electrical systems**

Communication linkage to the underwater vehicle depends upon the distance and the medium through which the communication takes place. The tether and the umbilical are vital components of an ROV and the main purpose is to serve as a medium to transfer the power to the vehicle as well as communication between the operator/control room and the vehicle. The cable linking the surface to the cage or tether management system (TMS) is termed as 'Umbilical', while the cable from the TMS to the submersible is called a 'Tether'. Underwater vehicles can be powered either by Alternating Current(AC) or Direct Current(DC) and can be sourced from one of the following ways:

- **Surface powered:** For the surface powered vehicles the power source is from the surface which means the vehicles must be practically tethered.
- **Vehicle powered:** Vehicle powered vehicles gets powered by itself in the form of battery, fuel cell, or some other means of power storage needed foe vehicle operation and propulsion.
- **Hybrid system:** It is a mixture of surface powered and vehicle power.

The connectors are vital for underwater vehicles because of the highly conductive nature of salt water, causing any exposed electrical component submerged in salt water to short to the ground. The purpose of an underwater connector is to conduct needed electrical currents through the connector and at the same time squeeze the water path and seal the connection to lower the risk of electrical leakage to the ground. [5]

**Control systems**

The control system provides a physical interface for the operator to control the vehicle and controls various ROV functions that can vary from switching the light(s) and video camera(s) to controlling the vehicle. These systems comprise control stations and various other subsystem control interfaces.

### 2.1.3 Degree of Autonomy

Autonomy has a direct relationship with human-robot interaction (HRI) and it influences the way in which humans and robots may interact with each other. There are several definitions related to the term "autonomy" in robotics literature. Few are "Autonomy refers to systems capable of operating in the real-world environment without any form of external control for extended periods" [6], "An Unmanned System's own ability of sensing, perceiving, analyzing, communicating, planning, decision making, and acting, to achieve goals as assigned by its human operator(s) through designed HRI; The condition or quality of being self-governing" [7]. According to the National Institute of Standards and Technology,

unmanned vehicles can be operated in the following four modes of operations [7]:

I **Fully autonomous:** A fully autonomous Underwater vehicle is the one which functions with no human intervention within a defined scope.

II **Semi-autonomous:** A semi-autonomous underwater vehicle requires a certain level of human-robot interaction.

III **Tele-operation:** In this mode of operation, the human operator either directly controls the motors/actuators or assigns incremental goals via a tethered or radio/acoustic/optic/other linked control device using video feedback or other sensory feedback.

IV **Remote control:** In this mode the human operator controls the actuators directly without any assistance of video or other sensory feedback on a continuous basis through a tethered or radio-linked control device within visual line-of-sight. No initiative is taken by the vehicle itself and sole operation is dependent on continuous input from the user.

### 2.1.4 Human Risk in handling UUVs

The demand for UUVs is increasing in various industries and applications such as military and intelligence, surveillance and reconnaissance (ISR), Anti-Submarine Warfare (ASW), and even time-critical strike operations. At the same time, little research has been done on human factors involved in the operation of UUVs. Some of the challenges for an operator when using UUVs are loss of sensor signals and spatial awareness, the control of the remote vehicle, problems with situation awareness (SA) and workload, problems with trust in automation, and challenges with robot communication. [8] The following human factors are mainly involved in the UUVs operation.

- **Operator's Perception in the Underwater Environment:** The two main root causes for restricting an operator's task such as the ability to navigate, control the vehicle, and object detection are poor vision and poor depth perception. These are mainly caused because of poor sensor signals or feedback from the vehicle in the operating environment which may result in accidents. It may also lead to high-cost impact and equipment damage for the owner due to which some of the ROVs are kept inside the cage to protect them from obstacles underwater. The operator's understanding of the underwater environment depends on the video camera images, sonar images, sensor indicators, and a physical model of the operating area. All of these have their limitations.

- **UUV Control and Displays:** The control of UUV is a very complex and hectic task that requires a lot of training and focus because the

operator must control the robot along 6 degrees of freedom which are surge (forward/backward), heave (up/down), sway (left/right), pitch, roll, and yaw at the same time. In addition to the above, the operator must also control the robot's velocity, altitude, and position in the water as well as the umbilical cord management. The operator controls the ROV from a control station which can vary in size, complexity, and location. As a result, unique human risks are associated with each type of control station. For instance, the operator who controls the vehicle outdoors, in a small boat, with waves splashing over the side will face different a challenge than the operator who controls an ROV from a comfortable office-like environment. [5]

- **Other factors:** Situation Awareness and Workload, Trust in robot and human-robot interaction and communication are the additional factors that are involved in handling of UUVs.

### 2.1.5 Classification of ROVs

The term "ROV" envelops wide range of equipment and no single vehicle can be described as 'typical'. But they are categorised depending on their functionalities, application and payload they can carry. [9]

- **Class I – Pure observation vehicles:** These vehicles are generally small in size fitted with camera, lights and thrusters and their functionality is limited to video surveillance. They will not be able to undertake any other task than observation without considerable modification.

- **Class II – Observation vehicles with payload option:** These vehicles are capable of carrying at least two additional sensors such as cathodic protection measurement systems and additional video cameras and sonar systems without any compromise in its original functionality.

- **Class III – Work class vehicles:** These vehicles are larger and more powerful compared to Classes I and II. They also have a multiplexing capability that allows additional sensors and tools to operate without being 'hard-wired' through the umbilical system. Based on their power they further classified as below:

  - Class III A: Workclass vehicles < 100 Hp
  - Class III B: Workclass vehicles 100 Hp to 150 Hp
  - Class III C: Workclass vehicles > 150 Hp

- **Class IV - Seabed working vehicles:** Class IV vehicles are designed for special purpose tasks such as cable and pipeline burial, dredging and other remotely operated seabed construction works, thereby, they are larger and heavier compared to Class III work class vehicles.

Due to their heavier nature, they manoeuvre on the seabed by a wheel or belt traction system, by thruster propellers or water jet power, or by combinations of any of these propulsion methods.

- **Class V – Prototype or development vehicles:** Autonomous Underwater Vehicles (AUVs) are part Class V. Vehicles under development, those regarded as prototypes and special-purpose vehicles that do not fit into one of the other classes are assigned to this class.

### 2.1.6 Inspection class ROVs

Inspection-class ROVs are also known as Observation-class ROVs which are smaller in size compared to Intervention-class ROVs. Inspection-class ROVs are of two types namely Micro or Handheld ROVs and Medium-sized ROVs. The weight of Medium-sized ROVs is between 30kg to 120kg which is usually powered by a DC supply with 600VDC voltage and 6kW power transmitted through copper cores or a combination of copper and fiber optic cores usually follow an open-frame model, whereas, Micro or handheld inspection ROVs weighs between 3 kg to 20 kg with smaller power requirements between 300W to 800W transmitted through copper cores in the umbilical which can be hand-fed from the surface vessel. Medium size inspection ROVs can be used to carry out underwater mapping and surveys with the help of accurate navigation systems and high-resolution imagining. [10] Inspection-class ROVs can be used in coastal monitoring, habitat monitoring, pollution assessments, dam wall inspections, blockage detection at pen-stock intake net inspection, seabed investigation, marine life studies, water and sediment sampling, and so on. [11]

### 2.1.7 Intervention-class ROVs

Intervention-class ROVs are of two types namely light work-class ROVs weighing up to 2000 kg and heavy work-class ROVs weighing up to 5000 kg. A strong hydraulic actuation is responsible for carrying out heavy-duty works. Due to the heavy mass, intervention-class ROVs require a Launch and Recovery System (LARS), along with a Tether Management System (TMS) which takes up a considerable volume of space on-board the surface vessel from which they are operated. [11]

### 2.1.8 Residential ROVs

The ROV sub-sea industry is demanding resident systems that can address various technology gaps and challenges along with integrated field support through more advanced and automated concepts opening up the door for unlimited potential in ROV and subsea operations for the future. This leads to the introduction of a different approach to ROV System, called Residential ROV (R-ROV). Residential ROVs (RROV) are unmanned underwater vehicles that are permanently installed in the sub-sea and are parked/caged on the seabed and have in-house designed Tether


Figure 2.2: Merlin UCV ROV
[12]

Management System (TMS) as well as Launch and Recovery Systems (LARS) rather than being deployed from a dedicated support vessel. The main advantages of this system are that it provides permanent installation on the seabed for long periods, free from the need to be accompanied by a support vessel, greater operational flexibility, increased safety, and greater quality of life for the pilots. The required offshore manpower is minimized due to the remote onshore control station via Tampnet.[10] Above the surface, 4G telecommunications networks have made remote control from onshore a reality whereas beneath the surface, "sub-sea GPS" and communications technologies are making navigation easier. RROV has revolutionized traditional sub-sea operations by reducing the waiting time on the weather and thereby saving cost and time. In addition to these benefits, they have also reduced Health Safety Environment (HSE) and personnel requirements. These vehicles are coupled with the onshore control centers. Having a permanently deployed RROV can lead to big savings as it's always available irrespective of the weather condition especially during marine riser disconnects and re-connections.

## 2.2 Merlin ROV

Merlin UCV (Ultra Compact Vehicle) is an electrical Work Class ROV [12] while Merlin UCV R-ROV (Residential- Remote Operated Vehicle) is an electrical work class residential ROV that is remotely operated from Onshore Control Centre (OCC) which can stay submerged for an extensive time and no activities are interrupted due to bad weather [13]. Both Merlin UCV and Merlin UCV R-ROV have the same dimensions with length 2.5-meter length, 1.5-meter width, 1.5-meter height and functions with 200 horsepower which is an extreme power to size ratio. Merlin UCV has a built-in tooling interface for electrical and hydraulic operated tooling while Merlin UCV R-ROV has sub-sea mating only for hydraulic tools. Merlin UCV weighs 2750 kilograms while Merlin UCV R-ROV weighs 3000 kg and both can handle up to 250 kg to 475 kg payload with an underwater

Figure 2.3: Merlin UCV Residential ROV
[13]

operational range up to a depth of 3000-meter sea water(msw). Various sensors are used in Merlin UCV such as depth sensor- Valeport Mini IPS, Sonar- Tritech Super Seeking, Heading sensor- Fiber Optical Gyro System - CDL Togs Nav with +/- 20 degrees pitch and roll. Merlin UCV R-ROV has a navigation sensor- SPRINT 500 with Syrinx DVL 603-0156-NS and a sonar - Tritech Super Seaking.

Both Merlin UCV and Merlin UCV R-ROV has one low light camera, one colour and zoom camera, but Merlin UCV has two colour cameras on front bar and the Merlin UCV R-ROV has only one camera at front down. On the rear side and at the center for TMS docking both the ROVs has a colour camera. Merlin UCV R-ROv has an additional wide angle colour camera at the front. Both are equipped with seven electrical thrusters (SA300), out of which 4 are horizontal and 3 are vertical. All the 7 thrusters are available in their full capacity 24*7. Both uses eight RS232 out of which, two can be configured to RS485 for subsea communication with 4 Ethernet connectivity in GBs and holds a spare fibre conductor.

## 2.3 ROV Modelling

Modeling and simulation is an effective technique to test software that plays an important role in different phases of development process. It can make the software testing process much more effective by observing the software's behavior in a virtual environment similar to the real environment whereas, testing in the real environment is not feasible which may involve a high risk of damage or even loss of vehicle if the algorithm fails to avoid on-board data collection, scenario construction and underwater recovery in the case of AUVs. Several pieces of researche have been done and are still ongoing in this field of modeling and simulation of ROVs. One such example is Underwater Rigid Body Library (URBL). [14] The URBL was developed as a framework for modeling underwater vehicles which has the base functional components to any ROV design.

Components and interfaces for modeling underwater vehicles, and an external interface to ROS are the two major sections of URBL. The modeling is done on Modelica which is a non-proprietary, object-oriented, equation-based language to conveniently model complex physical systems. The study considers the effect of water on submerged bodies as interactions with a "field" of water for encapsulating the effects of buoyancy and drag. Different methods of simulation include:

- **Offline simulation:** Offline simulators are used in the early stages of the development of the vehicle. The most used offline simulator is MATLAB/SIMULINK, which helps in modeling the vehicle and control systems but in this simulation, the time properties of the developed algorithms are not taken into consideration and cannot represent the external environment and the biggest drawback is that there is no consistency between simulation and reality.

- **Online simulation:** In online simulation one second of simulation is equal to one second of reality therefore we can say that there is consistency between simulation and reality and it also facilitates working concurrently on different features of the control system of a virtual robot. The commonly used online simulators in the field of robotics are ROS and MORSE.

- **Hardware in the loop (HIL) simulation:** In HIL simulation, a real-time computer is used as a virtual representation of the vehicle model and a real version of controller creates a test system that is scalable and ensures comprehensive test coverage. [15]

The deciding factors for selecting a robot simulation platform are physical correspondence for the simulation of rigid body dynamics and collisions, interface for ROS applications, low complexity for the setup of new world scenarios and robot models, extendable for integration of additional modules, adequate documentation, periodical check-ups, updated and maintenance, offering multiple-robot simulation, open-source application with a permissive license.

### 2.3.1 Mathematical modelling of an ROV

The initial step in simulating any real-world system is to replace the considered real-world system by a simplified mathematical model. To formulate the movement of the vehicle the degrees of freedom (DOF) is vital. For an ROV, there are a total of 6 (six) degrees of freedom (DOF), [16] where the vehicle can move as follows:

- Forward, surge sideways (sway) and float up (up) in the direction of the axis X, Y and Z.

- Roll, go up and turn (yaw) following the X, Y and Z axis.

The effects due to added mass and inertia terms as a result of the simultaneous acceleration of water particles and quadratic damping

caused by energy dissipation within the medium has to be considered while modelling a dynamic coupling of any system to the surrounding water. The work [17] details hydrodynamic model of a robot link as represented in equations 2.1 and 2.2 in which $p$ denotes the pose of a robot link expressed in the world frame, and $v$ denotes the velocity screw expressed in the local frame. The efforts related to hydrodynamic effects are considered as a negative impact on the vehicle.

$$M\dot{v} = G(v)\,v + d(v) + \tau_g + \tau_u + \tau_e \tag{2.1}$$

$$\dot{p} = Jv \tag{2.2}$$

where:

- $M$ represents the inertia matrix as detailed in the work [18] which is the sum of rigid body inertia matrix $M_d$ and inertia matrix $M_a$ on the body due to the fluid. $M_a$ is considered as zero since the error can be large in the model.
- $G(v)$ is the coriolis and centrifugal forces which is combination of dynamic matrix $G_d$ and hydrodynamic matrix $G_a$. The value of $G_a$ is considered as zero.
- $d(v)$ represents the hydro dynamic damping force.
- $\tau_g$ is the added forces due to gravity, buoyancy and induced torques
- $\tau_u$ is the added forces and torques by the user. For instance: force due to thrusters
- $\tau_e$ represents added forces and torques to the robot link.
- $J$ is the transformation matrix between the world frame and the link frame.

Let $k$ denotes the damping coefficients and $v_c = (v_c,\ 0)$ is the water velocity expressed in the world frame which has zero angular values. Then $d(v)$ can be expressed in the form of a quadratic function of velocity between the robot link and the water velocity as shown in equation 2.3.

$$d(v) = -k^T |v - J^{-1}v_c| \left(v - J^{-1}v_c\right) \tag{2.3}$$

## 2.4 Related works

Several studies regarding the modeling and simulation of ROVs have been carried out. The work [19] details the common tasks in ROV operations in its operational environment.

### 2.4.1 Implementation of Mathematical Model in ROS

Gazebo utilizes SDF (Simulation Description Format), which is an XML format for describing the objects and environments, that is capable of describing all properties of the robot model (links, joints, sensors, plugins), static and dynamic objects, lighting, terrain, and even physics. Links are defined by Inertial (mass, a moment of inertia), Collision and Visual (geometry) which are used by physics, collision and render engines, respectively. Joints serve as a connection between two links and define their movement, limiting the Degrees of Freedom (DOF) using various types such as fixed, revolute, prismatic, universal, ball, and screw based on their configuration. [20]

### 2.4.2 World Modelling

In the field of embedded robotics systems, tools like Gazebo and ROS are gaining higher importance considering the new concept of kinetic architecture or dynamic buildings. For ages architectural buildings were considered static structure incapable of any changes or modification. But, in recent years, the concept of kinetic architecture and dynamic buildings are currently being exploited by many architects around the globe and buildings are viewed as kinetic structures showing potential to changes and adaption. As of now, no tools are available in the field of architecture for the modeling, simulation, and operation of such complex kinetic architectures. [21] Therefore, for the implementation of such a concept, tools, and knowledge from the field of robotics is deployed for this purpose. The best applicable open-source system Robot operating system (ROS) permits the creation of a virtual model of the world along with the integration of different types of sensors as required. The concept of the dynamic building is utilized for making smart and sustainable buildings. A 3D model of the environment is required to simulate a dynamic, kinetic environment in ROS. Followed by the 3D model divided into smaller parts that relate to joints having different properties in mobility. ROS is equipped with tools that are capable of manual management of individual objects.

The water velocity $v_c$ and the buoyancy direction are two major factors that contribute to the world plugin for any UUVs. [17] The Gazebo simulator subscribes to a ROS topic that shows the intensity and direction of the water current. The direction of buoyancy is opposite to that of the gravity and decreases to zero as the vehicle move towards the water surface. A Gazebo world file typically has a .world extension and the Gazebo server (gzserver) reads this file to generate and populate the defined world.

### 2.4.3 Modelling of Sensors

Gazebo supports several plugin types which can be connected to ROS but only Model plugins such as differential drive controller, Sensor plugins such as cameras and visual plugins such as sonars can be accessed using

URDF file. When the robot model is loaded in the Gazebo, the plugin name in the code is given a reference to the sensor which provides access to its Application Programming Interface (API).

### 2.4.4 Camera Modelling

The sub-sea imaging simulators have received a lot of attention in recent researches since the captured images from real-time are not of very good quality due to numerous reasons such as low contrast, blurring details, colour deviations, non-uniform illumination. A prominent example is the deep sea robotic imaging simulator for UUV development. [1] The work focus on image restoration and enhancement which are vital for practical applications, are done using Jaffe-McGlamery model. According to this model, the underwater image is the linear superposition of three components (camera, light source, and image plane). An underwater image experiment consists of tracing the progression of light from a light source to a camera. The light received by the camera is composed by three components:

I The direct component (light reflected directly by the object that has not been scattered in the water) because the RID curve of the light is usually very smooth.

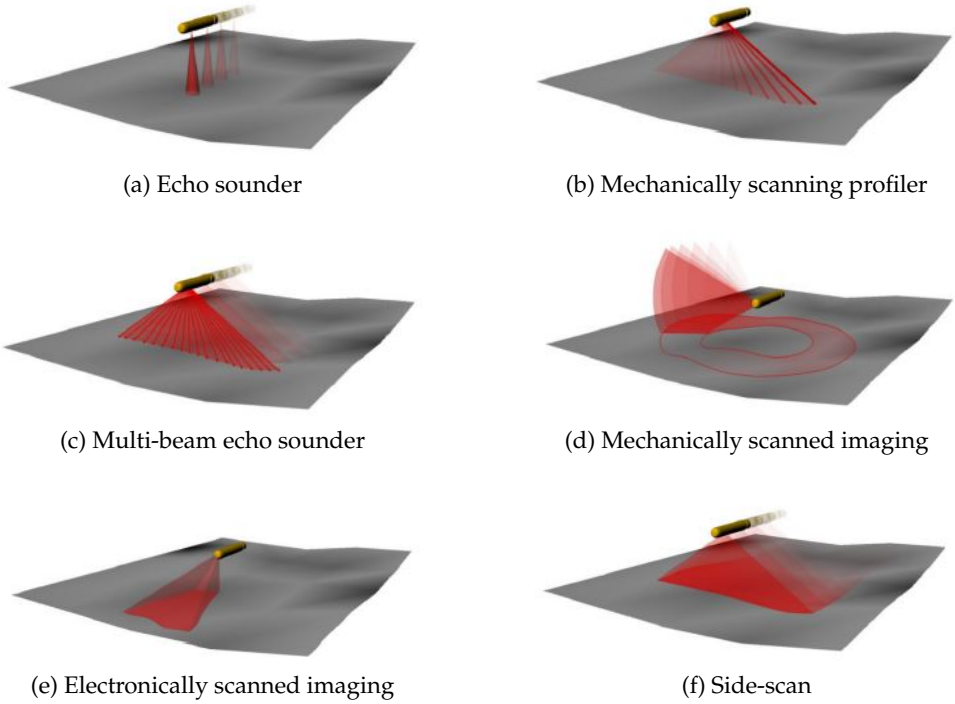
II The forward-scattered component (light reflected by the object that has been scattered at a small angle)

III The back scatter component (light reflected by objects not on the target scene but that enters the camera, for example, due to floating particles) which leads to a veiling light effect in the medium which happens along the whole light path.

The solid model of camera can be added to the robot model using visual tags in the URDF file. Once the necessary visual properties are added, the technical features can be added using a <gazebo> tag. There are many sub-tags for adjusting the properties of the camera such as update-rate, distortion and so on. Several examples are detailed by B.R.Japon in his work. [22] Gazebo also provides opportunity to create custom camera plugins for the robot model.

### 2.4.5 Sonar Modelling

Among various sonar technologies available active and passive are the most commonly used ones. For mapping related applications active sonar are more preferred since it can emit and receive acoustic symbols compared to its counterpart which can receive only echoes. Echo sounder, mechanically scanning profiler and multi-beam echo sounder are some of the types in active sonars. An echo sounder is a static type of sensor which emits a pulse from its transducer and calculates distance from the time of flight of reflected pulse by a surface to the sensor head. Most echo sounders compromise on precision, and doesn't compensate for water



(a) Echo sounder

(b) Mechanically scanning profiler

(c) Multi-beam echo sounder

(d) Mechanically scanned imaging

(e) Electronically scanned imaging

(f) Side-scan

Figure 2.4: Different variants of active sonar technology [23]

conditions such as temperature. Mechanically scanning profilers consists of mechanical actuators for orienting sensors at different angles varying from a few degrees to a complete 360-degree scan and produce a series of measurements to create an acoustic image of the environment in terms of intensity and position. Different variants in these types of sonars are mechanically scanned imaging sonar, electronically scanned imaging sonar and side-scan sonar. Multi-beam echo sounder consists of an array of hydrophones which emit fan shaped beams and thereby producing a complete strip of points in the direction of the pulses emitted. Figure 2.4 details how the scan is performed with various active sonar technology

For modeling the sonar, *Gazebo_ros_range* plugin can be utilized which publish messages according to *sensor_msgs/Range* Message format so that integration to ROS can be done easily. The work [23] details the modifications made on the SDF format to add the sonar sensor to a robot in gazebo. The sensor definition is done using the <*sensor*> XML element and the following sub-tags <*always_on*>, <*update_rate*> and <*plugin*> are used. The sonar sensor functions similar to an ultrasound range sensor by simulating the sound pulse emitted using a cone, and gets the closest collision between the cone and any surface in the environment followed by publishing as a message containing information about the 3D position of the collision. The <*sonar*> element, which is a child of the <*sensor*> element is used in the above work, with the following child elements

- <*min*> Minimum distance at which we detect collisions.
- <*max*> Maximum rage of the sonar.
- <*radius*> Radius of the cone at maximum range.

### 2.4.6 Thruster Modelling

A thruster, the combination of a motor and propeller, is an electro-mechanical device generating thrust to translate and rotate an underwater vehicle to control altitude, position, velocity, resist external disturbances and ensures maneuvering and station-keeping capabilities. Thrusters can be divided into two categories, hydraulic thrusters which are mainly used on work class hydraulic ROVs and electric thrusters which are mainly used battery operated underwater vehicles. The efficiency of a thruster decreases with speed so with thrusters we must choose between a thruster which produces an ample amount of thrust at a low speed or a thruster with high speed but less efficiency. For ROVs, we choose the latter, ROVs thruster needs to develop maximum thrust at zero forward speed as thrusters play a vital role in the functionalities of remotely operated vehicles (ROVs) and are the lowest control loop on the ROV system. A thruster is a good fit for an ROV if it possesses a high voltage range, works well in forward and reverse direction, generates at least 15 lbs of thrust from 25 to 50 volts, consist of a nozzle engineered for maximum thrust and an efficient brush-less motor with a decent gear ratio with a reliable thruster design requiring little maintenance.

Thrusters are composed of components like power source, thruster housing and attachment to vehicle frame, motor controller, electric motors, propeller, kort nozzle, stators, drive shafts, seals, couplings, gearing mechanism. While developing a thruster design following things must be kept in mind: cost, physical weight, thrust force and size. The steps involved in design and modelling of a thruster are problem specification, experimentation, system identification, testing/validation, analysis of performances, controller design and testing, implementation and testing and final testing.

### 2.4.7 Role of System Identification in Thruster Modelling

The modelling and control of underwater vehicle thruster systems using system identification has received a wide attention in the literature over the last years. [24] [25] The easiest and the fastest alternative for modelling a thruster is by system identification as compared to design by using a derivation of mathematical equations. Detection of the forces and moments generated by underwater vehicles plays an important role in the development of any physical models as well as for validating theoretical models. Hydrodynamic reaction torque must be considered while modelling underwater vehicles, which usually goes unnoticed. For characterising thrust performance, usually simple Bollard setups are implemented. All-axis translational and rotational flow sensing models are

complicated dynamic numerical model that have been used in the past. Additionally, underwater vehicles go through a complex full- vehicle test which can be harsh and time consuming. The process becomes even more tedious due to a large amount of information required to be collected and processed. Above mentioned difficulties are not present while using a 6 degree of freedom (lift, draw, side-force, yaw, roll, pitch) Gough-Stewart platform-based load cell. It is also very compact to fit in smaller test tanks.

Modelling of thrusters are a crucial part of underwater vehicle control and simulation. Motor modeling, propeller design, and hydrodynamic effects contributes to the thrust force from an ROV. The system identification can be done in modelling the thrusters of an underwater vehicle by using MATLAB system identification toolbox. The study [26] states that thruster modelling using system identification is an easier and faster alternative for modelling compared to design by using derivation of mathematical equations. The mathematical model of thruster is detailed in the above study and the final equations are shown in equation 2.4.

$$V = R_m \left(T/K_m\right) \; + \; K_E \left(\sqrt{T/K_T}\right) \tag{2.4}$$

Where,

- $V$ is the applied voltage to the motor.
- $R_m$ represents motor resistance.
- $T$ is the output thrust from the thruster
- $K_m$ is the modified motor constant.
- $K_E$ represents back EMF constant of the motor.
- $K_T$ equals lump parameter of various constants.

### 2.4.8 Modelling of Tether

Forces from the tether are one of the vital elements affecting the dynamics of an ROV. These forces can vary widely depending on sea current and how the ROV is deployed, thereby affecting the ROV motion. As a result, to achieve realistic simulation of real ROV motions, an estimation of these forces is important to include in the dynamics of ROV model. Using mathematical model, the study [27] details how much of these forces can contribute to dynamics of an ROV under water. Several pieces researches have been done in modelling and simulation of umbilical which can be categorized under two categories, force-based methods and position-based methods. Cable and chain models [28] fall under the first category while the study done by Jackobson [29] lies in the latter category. In a study by Ganoni et al. [30] found that, the simulation aspects of the tether attached to a ROV and to the launching platform is a major part of an underwater ROV simulation. The tether simulation is done by utilizing advantages from both the forced based and the position-based methods. The study

states that a close to real cable behavior can be achieved in simulation when a robot sees its own cable along with the aid from computer vision algorithms.

### 2.4.9 Manipulator and tool pack Modelling

Manipulator, also known as robot arm is a vital tool for executing sub-sea intervention operations. As a result, most of the ROVs that are under class III ROVs are equipped with one or more underwater manipulators. UUVs with manipulators are often called Underwater Vehicle Manipulator Systems (UVMS). These manipulators are comprised of links interconnected by means of revolute joints with a suitable angular displacement between them and grippers or other interchangeable tools attached at the end-effector. On the base underwater vehicle and/or on the manipulator itself, sensors such as spotlights and cameras are mounted, for observing their surroundings they are usually accompanied by additional equipment comprising of one or more cameras and spotlights. Generally, UVMS have two manipulators in which, one is anchor manipulator that holds UVMS still, as an underwater current countermeasure and another as the actual operation manipulator. The current and turbidity of water increase the complexity of controlling these systems. So, modeling and simulation can help UVMS operators to a greater extent in their tasks, by making tasks repeatable and reducing operation time.

Several works have been done explaining concepts of underwater manipulators out of which the prominent ones are, a study by Antonelli [31] provides a detailed theoretical background for underwater manipulators from the modeling and control point of view and another one by Yuh and West [32] giving a brief overview on underwater manipulators. The project [33] categorizes the semi-autonomous behavior of underwater manipulators into three main features, which are localization, UVMS pose control and manipulator pose control.

- **Localization:** The localization problem is common in most robotic applications. To estimate UVMS pose in the environment, necessary data from various sensors are used. Each sensor uses different algorithms to establish sensor's pose and consequently robot pose. For localization, aruco artificial markers are implemented in the above work for pose estimation using ROV camera.

- **UVMS Control:** UVMS can move to reach a pose by activating the thrusters. Velocity control through controlled thrusters is commonly used to move the ROV at a specific velocity in all 6 Degrees Of Freedom (DOF).

- **Manipulator Control:** A ROS tool called Moveit [34] was used in the above simulation to control the manipulator. It implements trajectory planning, joint controller, inverse and direct kinematics. The advantage is the possibility to define a pose as set-point to the manipulator's end effector.

# Chapter 3

# Approach

## Chapter Outline

The chapter describes plans and the procedures for the study to develop the dynamic model of ROV and explains basic concepts of ROV kinetics followed by mathematical modelling of the ROV. These mathematical derivations provide the basis for developing the dynamic models under two independent simulation environments and their implementations were discussed towards the end of this chapter.

## 3.1 UUV Kinematics

A rigid body in motion in space can be explained through two reference frames, a world-fixed reference frame (W) and a frame fixed to the body as reference (B). These two systems are represented in Figure 3.1. The body frame system was considered as placed with origin coinciding with the center of mass of the vehicle, where the degrees of freedom (DOF) of the body frame, respectively $X_B$ , $Y_B$ , $Z_B$, $\phi_B$, $\theta_B$, $\psi_B$ are named *Surge*, *Sway*, *Heave*, *Roll*, *Pitch*, *Yaw*. The position vectors for world frame and body frame are given by equations 3.1 and 3.2.

$$\begin{aligned} \chi &= \begin{bmatrix} surge & sway & heave & roll & pitch & yaw \end{bmatrix} \\ &= \begin{bmatrix} X_B & Y_B & Z_B & \phi_B & \theta_B & \psi_B \end{bmatrix} \end{aligned} \tag{3.1}$$

$$\eta = \begin{bmatrix} x & y & z & \phi & \theta & \psi \end{bmatrix} \tag{3.2}$$

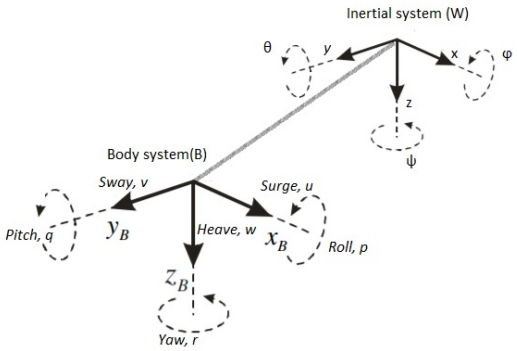



Figure 3.1: ROV reference frame and degrees of freedom

The velocity vectors in terms of vector of linear velocity $v_B$ and the vector of angular velocity $\omega_B$ for world frame and body frame can be written as shown in equations 3.3 and 3.4.

$$\begin{aligned} v &= \begin{bmatrix} \dot{X_B} & \dot{Y_B} & \dot{Z_B} & \dot{\phi_B} & \dot{\theta_B} & \dot{\psi_B} \end{bmatrix}^T \\ &= \begin{bmatrix} u & v & w & p & q & r \end{bmatrix}^T \\ &= \begin{bmatrix} v_B & \omega_B \end{bmatrix}^T \end{aligned} \tag{3.3}$$

$$\begin{aligned} \dot{\eta} &= \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} & \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T \\ &= \begin{bmatrix} v_W & \omega_W \end{bmatrix}^T \end{aligned} \tag{3.4}$$

The force/torque vector $\tau$ of the thruster input are given by the equation 3.5.

$$\tau = \begin{bmatrix} \tau_u & \tau_v & \tau_w & \tau_\phi & \tau_\theta & \tau_\psi \end{bmatrix} \tag{3.5}$$

Equation 3.6 shows the velocity vector transformation from the body frame

into the world frame.

$$\dot{\eta} = \underline{J}(\eta)\, v$$
$$\underline{J}(\eta) = \begin{bmatrix} \underline{J_1}(v_W) & 0 \\ 0 & \underline{J_2}(\omega_W) \end{bmatrix} \tag{3.6}$$

where, $\underline{J}(\eta)$ is the coordinate transform matrix, which brings the world frame into alignment with the body frame.

Equation 3.7 represents the Euler angle rotation matrix $\underline{J_1}(v_W)$ defined in terms of principal rotations as shown in equation 3.8

$$\begin{aligned} \underline{J_1}(v_W) &= \underline{R}^{BW}(v_W) \\ &= \begin{bmatrix} c(\theta)c(\psi) & -c(\phi)s(\psi) + s(\theta)s(\phi)c(\psi) & s(\phi)s(\psi) + s(\theta)c(\phi)c(\psi) \\ c(\theta)s(\psi) & c(\phi)c(\psi) + s(\theta)s(\phi)s(\psi) & -s(\phi)c(\psi) + s(\theta)c(\phi)s(\psi) \\ -s(\theta) & c(\theta)c(\psi) & c(\phi)c(\theta) \end{bmatrix} \end{aligned} \tag{3.7}$$

$$R_{x,\phi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(\phi) & -s(\phi) \\ 0 & s(\phi) & c(\phi) \end{bmatrix} R_{y,\theta} = \begin{bmatrix} c(\theta) & 0 & s(\theta) \\ 0 & 1 & 0 \\ -s(\theta) & 0 & c(\theta) \end{bmatrix} R_{z,\psi} = \begin{bmatrix} c(\psi) & -s(\psi) & 0 \\ s(\psi) & c(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

where, $s$ represents *sin*, $c$ represents *cosine* and $t$ represents *tangent*.

On the other hand, the Euler angle attitude transformation matrix is shown in equation 3.9.

$$\underline{J_2}(\omega_W) = \begin{bmatrix} 1 & s(\phi)t(\theta) & c(\phi)t(\theta) \\ 0 & c(\phi) & -s(\phi) \\ 0 & s(\phi)/c(\theta) & c(\phi)/c(\theta) \end{bmatrix} \tag{3.9}$$

## 3.2 Dynamic modelling

Dynamic model is vital for performing simulations as well as to formulate control algorithms. In order to generate control forces for thruster-actuated vehicles, it is necessary to compute the Thruster Allocation Matrix, which will translate the output of the controller into the output thruster forces for individual thrusters. ROV simulator expects each thruster unit to have its unique frame, which enables the use of tf package in ROS to lookup the transformation matrix between the vehicle's body frame and each thruster during runtime.

Equation 3.10 shows the dynamic model which has been derived from the Newton-Euler equation of a rigid body in fluid and does not take account of environmental disturbances, such as underwater currents.

$$\underline{M}\dot{v} + \underline{C}(v)\, v + \underline{D}(v)\, v + \underline{g}(\eta) = \tau \tag{3.10}$$

Where,

- $\underline{M}$ is the inertia matrix.

- $\underline{C}$ represents the Coriolis and centripetal matrix.
- $\underline{D}$ is the drag matrix

### 3.2.1 System Inertia Matrix

The system inertia matrix is the sum of of rigid body mass $\underline{M}_{RB}$ and added mass $\underline{M}_A$. The rigid body mass can be written in terms of mass of the ROV $m$ and the inertia $I_{()}$of the ROV with respect to the Body frame as shown in equation 3.11.

$$\underline{M}_{RB} = \begin{bmatrix} m & 0 & 0 & 0 & mz_G & -my_G \\ 0 & m & 0 & -mz_G & 0 & mx_G \\ 0 & 0 & m & my_G & -mx_G & 0 \\ 0 & -mz_G & my_G & I_{xx} & -I_{xy} & -I_{xz} \\ mz_G & 0 & -mx_G & -I_{yx} & I_{yy} & -I_{yz} \\ -my_G & mx_G & 0 & -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix} \tag{3.11}$$

The calculated values for rigid body mass and the effect of the hydrodynamic added mass for the Merlin UCV is detailed in equation 3.12. The values are used in the

$$\underline{M}_{RB} = \begin{bmatrix} 2936.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 2936.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 2936.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1224.179892 & -28.489418 & -82.409928 \\ 0.0 & 0.0 & 0.0 & 0.0 & 2234.271553 & 7.821611 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 2125.224693 \end{bmatrix}$$

$$\underline{M}_A = \begin{bmatrix} 1468.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1468.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1468.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 122.4179892 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 223.4271553 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 212.5224693 \end{bmatrix} \tag{3.12}$$

### 3.2.2 Drag Matrix

The hydrodynamic damping in underwater vehicles are caused mainly by the drag and lift forces. The lift forces are negligible compared to the drag forces since ROV is operated at very low speeds. The drag forces is the sum of linear $\underline{D}_l(v)$ and quadratic term $\underline{D}_q(v)$. The calculated values for the above terms are given by the equation 3.13.

$$\underline{D}_l(v) = \begin{bmatrix} -150.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & -200.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -200.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & -100.0 & 0 & 0 \\ 0.0 & 0.0 & 0.0 & 0.0 & -100.0 & 0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -50 \end{bmatrix}$$

$$\underline{D}_q(v) = \begin{bmatrix} -1500.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & -2000.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -2000.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & -500.0 & 0 & 0 \\ 0.0 & 0.0 & 0.0 & 0.0 & -500.0 & 0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -500 \end{bmatrix} \tag{3.13}$$

### 3.2.3 Gravitational and Buoyancy Matrix

The gravitational and buoyancy vector, $\underline{g}(\eta)$ , can be denoted in matrix form as shown in equation 3.14.

$$\underline{g}(\eta) = g \begin{bmatrix} (m - \rho\nabla)\, s(\theta) \\ -(m - \rho\nabla)\, c(\theta)\, s(\phi) \\ -(m - \rho\nabla)\, c(\theta)\, c(\phi) \\ (mz_g - z_b\rho\nabla)\, c(\theta)\, s(\phi) - (my_g - y_b\rho\nabla)\, c(\theta)\, c(\phi) \\ (mz_g - z_b\rho\nabla)\, s(\theta) + (mx_g - x_b\rho\nabla)\, c(\theta)\, c(\phi) \\ -(mx_g - x_b\rho\nabla)\, c(\theta)\, s(\phi) - (my_g - y_b\rho\nabla)\, s(\theta) \end{bmatrix} \tag{3.14}$$

Where,

- $g$ is the acceleration due to gravity.
- $\rho$ represents the density of fluid and $\nabla$ is the ROV volume.
- $x_g$, $y_g$, $z_g$ is the ROV center of gravity.
- $x_b$, $y_b$, $z_b$ is the ROV center of buoyancy.

### 3.2.4 Force and Torque Vector

Seven motors works together to deliver the required thrust force and torque for the ROV to move in *surge*, *sway*, *heave* and *yaw* directions. The force and torque vector is defined in equation 3.15

$$\tau = MU \tag{3.15}$$

Where,

- $M$ is the thruster mapping matrix.
- $U$ represents the thrust vector.

## 3.3 Model Implementation

### 3.3.1 Implementation overview in ROS

Most of the development of IKM Subsea's Merlin Simulator under the ROS/Gazebo environment were already been done and in this work, the studies were more towards validating the existing model performance. The simulator was built on ROS package and the simulation engine was gazebo. The common file extensions used are listed below:

- **.urdf:** known as Unified Robot Description Format. These are robot description files used by ROS.
- **.xacro:** also known as XML macros. These are urdf files with macro features and are converted to .urdf before being used.
- **.launch:** These files contains launch instructions for the ROS package. For example, using these files it can be determined, which models to load and which scripts to run.

- **.sdf:** also known as Simulation Description Format. These are Simulation/model description files for Gazebo and it is also used in .urdf files where ever there is a <gazebo> tag.

The folder organisation of ROS simulator is shown in Figure 3.2 and the files under each folder are not included in the figure due to its large structure. The files under the folder *bagfiles* are part of the original contributions of this research with which validation of the existing simulator has been done whereas the *srov − master* is part of the existing simulator where the works has already been done. The procedure for setting-up and running the ROS simulation environment are detailed in Appendix A.

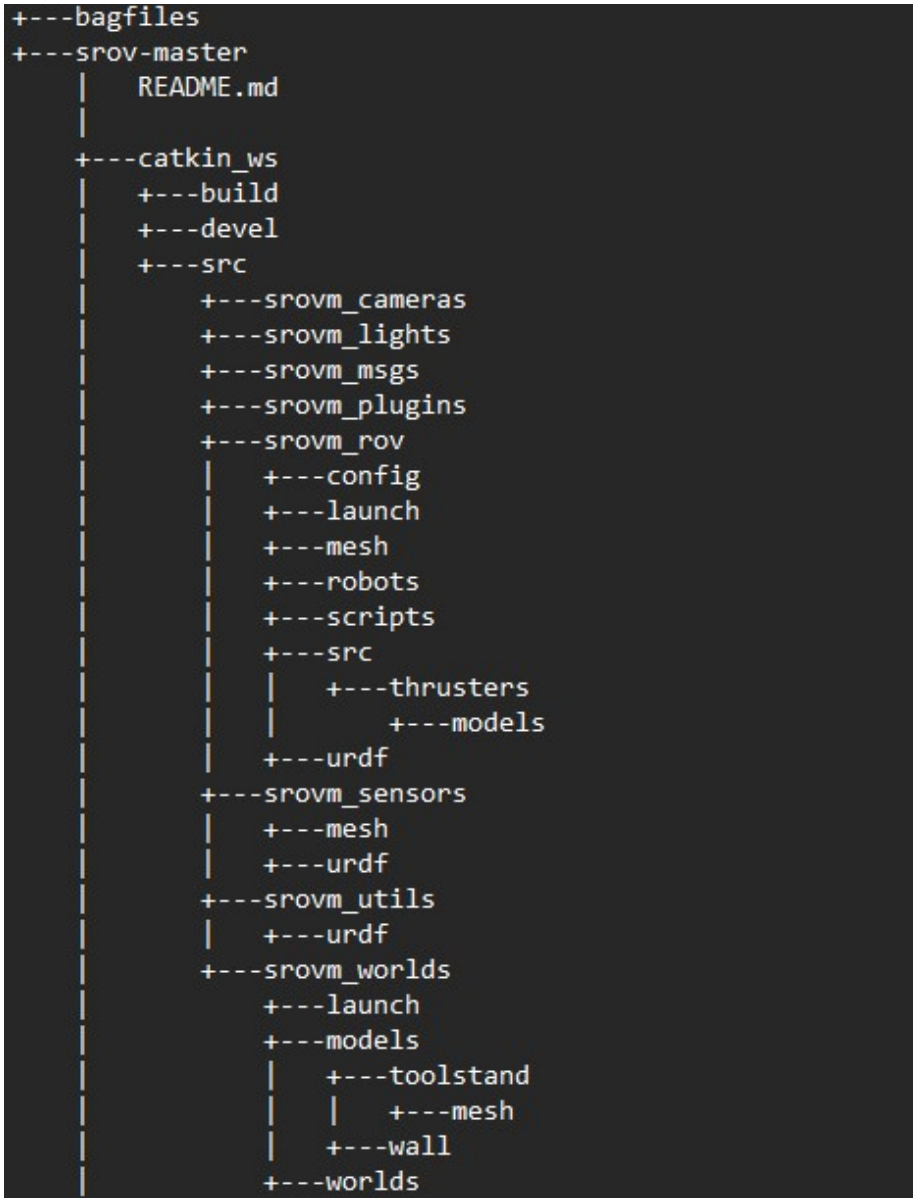



Figure 3.2: ROS package folder structure

### 3.3.2 Implementation overview in MATLAB/Simulink

The mathematical modelling of the ROV in MATLAB/Simulink environment follows the equations discussed in section 3.1 and section 3.2. The project package has been adapted to that of Merlin UCV hardware by some modifications on top of UUV dynamics project [27]. The ROV dynamics and thruster modelling are implemented in C language using the C MEX API followed by calling them in Simulink using C s-functions. The

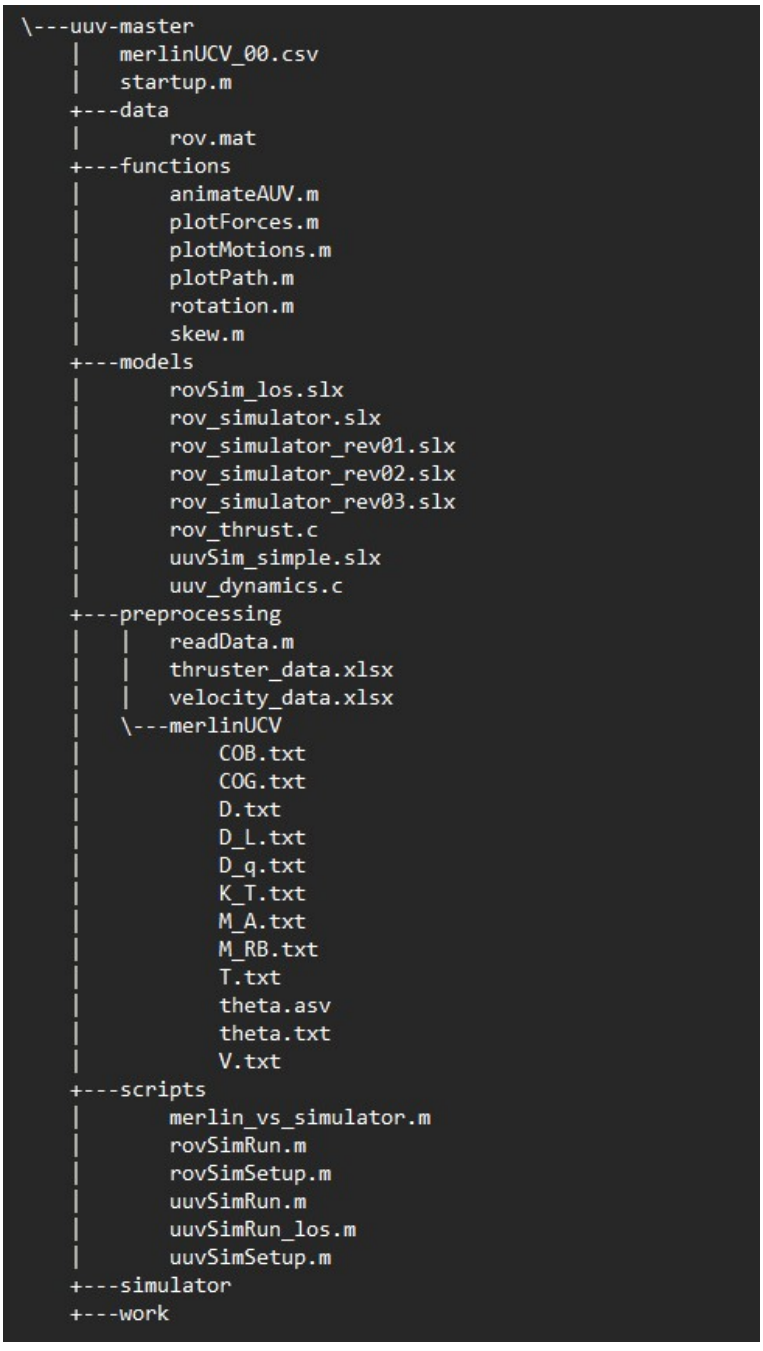



Figure 3.3: MATLAB package folder structure

"rov_thrust.c" details the thruster behaviour for seven thrusters by taking propeller revolutions as an array of input and returns the output as a thrust vector with six degrees of freedom. On the other hand the dynamics of ROV in six degrees of freedom is detailed in "uuv_dynamics.c" taking account of system inertial mass which includes rigid body matrix and added mass, drag matrix consisting of linear and quadratic damping, and other external effects from water current. A basic PID controller to control depth, speed and steering is designed to verify the model in Simulink environment. Functions for plotting and animation of ROV in 3D space are also implemented to capture the results.

The ROV is defined under the directory $uuv - master/preprocessing/merlinUCV$ as shown in Figure 3.3. All the physical parameters related to ROV, thruster and the operational environment are stored under this directory. The MATLAB scripts (with *.m* extensions) are responsible for calling various parameters and individual models as required for simulation. All the main scripts for running the simulations are kept under the directory *scripts*. The Simulink models and the *C* codes that control the ROV behaviour and dynamics are stored under the directory *models*. The procedure for setting-up and running the MATLAB simulation environment are detailed in Appendix B.

# Chapter 4

# Test Plan and Experiments

## Chapter Outline

This chapter discusses the test plan and various experiments that have been conducted for studying the ROV simulator response under two independent simulation environments. it also provides an overview of additional tools and libraries utilised to perform the data capture and analysis. Lastly, how to perform a performance comparison study between simulator and hardware is detailed.

## 4.1 Introduction

Experiments have a vital role in the testing and verification of mathematical models. It can provide practical ground for the designed model is correct, or it can prove the the model is wrong by a deviation from the expected behaviour which opens a new area of attention for the study. They also facilitate evidence for the existence of the entities involved in our theories and mathematical equations. In this work, MATLAB/Simulink and ROS/Gazebo environments are utilized to perform experimentation on the ROV simulator. Few additional tools such as *PlotJuggler*, *RQT_GUI*, and a few custom-made scripts in *python* are utilized for data capture to perform data analysis and comparative study.

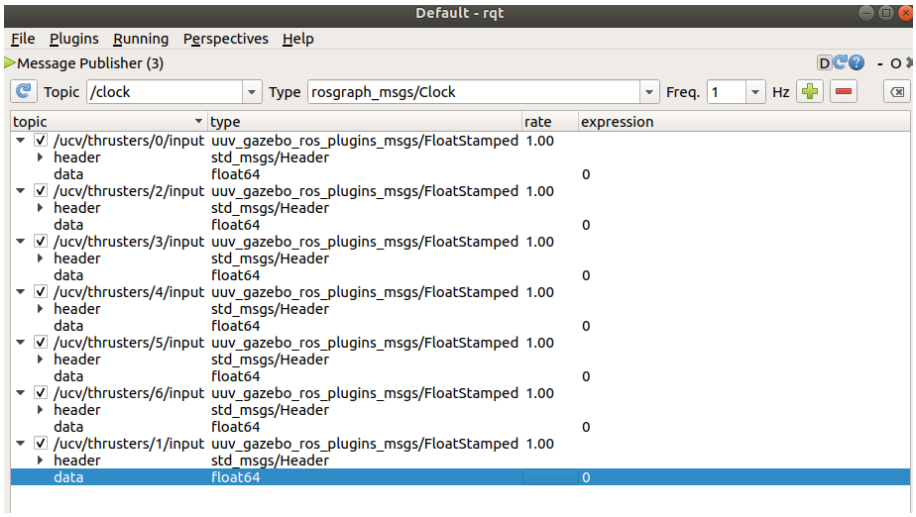



Figure 4.1: Input Commands to Individual Thruster using *RQT_GUI*

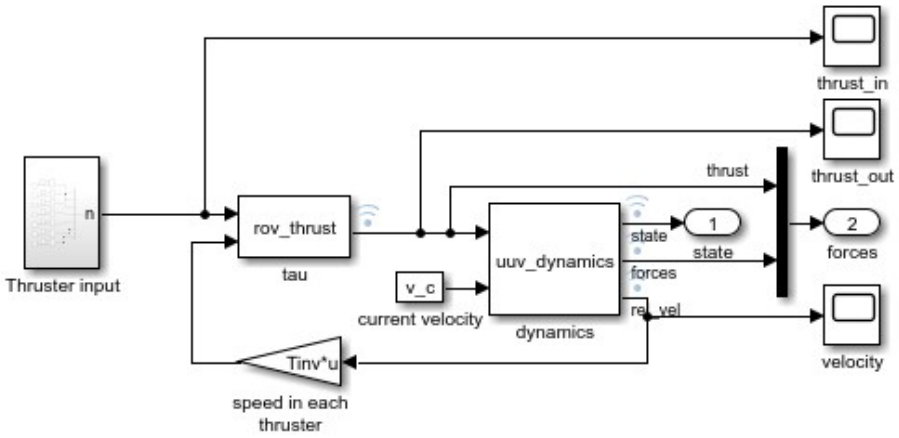



Figure 4.2: Simulink model for stability test

## 4.2 ROV model stability

Initial experiments were focused on studying the stability of the ROV model in MATLAB and ROS environment under ideal conditions. The test plan for the above is detailed in tables 4.1 and 4.2.

| **To study ROV stability under ideal conditions in MATLAB** | |
|---|---|
| **S No.** | **Procedure** |
| **Test preconditions** | Make sure all necessary directories are included in the path and the C-coded S-functions are compiled. |
| 1 | Run the startup script named *startup.m* |
| 2 | Run the Simulink model named *rov_simulator.slx* as shown in Figure 4.2, under the directory *models* |
| 3 | Observe and record thrust and velocity of ROV in all six degrees of freedom. |
| **Expected result** | Both the thrust and velocity in all six degrees of freedom must be zero under ideal conditions (zero RPM to thruster input). |

Table 4.1: Test scenario 1 - Model stability under MATLAB environment

To serve the purpose of this study, all inputs to individual thrusters were set to zero followed by comparing the performance between models in ROS and MATLAB.

## 4.3 Thruster response for varying inputs

The main goal of this study is to compare the individual thruster response from the ROV model in MATLAB/Simulink and ROS environment. The test plan is detailed in tables 4.3 and 4.4. The inputs to individual thrusters are changed with respect to time but they remain the same across the simulation environments.

## 4.4 Thruster response for varying movement commands

Few additional experiments have been done with simulator in ROS environment to test casual relationship by manipulating ROV input variables related to movements and measure their effect on thruster inputs and outputs. The initial experiments were done to study the list of available feed-backs from the simulator and ROS message types followed by storing the complete data in time synchronised form. Scripts were made to extract time series data for individual message types from the complete data file to study the co-relation between inputs and outputs to thrusters at varying environmental conditions. The test plan for the above under ROS platform is detailed in Table 4.6. On the other hand, ROV simulator under MATLAB

| To study ROV stability under ideal conditions in ROS | |
|---|---|
| **S No.** | **Procedure** |
| **Test preconditions** | Make sure all the UCV ROS packages are build and sourced. |
| 1 | Run a terminal and launch the world in gazebo using the command *snorre* |
| 2 | Run another terminal and spawn the robot to gazebo environment using the command *snorre* |
| 3 | Run another terminal and launch the ROS tool *rqt_gui* by running the command *rosrun rqt_gui rqt_gui*. |
| 4 | Run another terminal and launch the *PlotJuggler* which is a tool to visualize time series by running the command *rosrun plotjuggler plotjuggler*. |
| 5 | Load the topics for individual thruster input with data value equal to zero in *rqt_gui* as shown in Figure 4.1 and publish the topics at 10 Hz. |
| 6 | Load the subscribed topics to *PlotJuggler* data analysis window and record thrust and velocity of ROV in all six degrees of freedom. |
| **Expected result** | Both the thrust and velocity in all six degrees of freedom must be zero under ideal conditions(zero RPM to thruster input). |

Table 4.2: Test scenario 1 - Model stability under ROS environment

Environment, controls the movement directions using 3-dimensional way points in a Cartesian coordinate system. The test plan is detailed in Table 4.5.

## 4.5 Data Collection

Data collection plays a vital role in any field of research to evaluate their hypothesis or validate a model with hardware in real-time. Through data collection, accurate insights of research can be measured and analysed using standard validation techniques. Quantitative approach is used in this study to find a relationship between the data from simulator and that from the Merlin UCV. The data from the simulator is gathered using *PlotJuggler* tool and *rosbag* commands in ROS and extracting the required data through custom-made scripts. To serve the above purpose *bagpy* library - (A Python package to facilitate the reading of a *rosbag* file based on semantic datatypes) has been used. On the other hand, MATLAB simulation captures the simulator data using standard Simulink library functions.

The hardware (Merlin UCV) data has been collected in real-time from various sensor feed-backs. The data sampling rate was 10Hz. A total

| To study individual thruster response in MATLAB | |
|---|---|
| **S No.** | **Procedure** |
| **Test preconditions** | Make sure all necessary directories are included in the path and the C-coded S-functions are compiled. |
| 1 | Follow the steps 1 and 2 listed in table 4.1 |
| 2 | Vary the inputs to individual thrusters by running the script *.m* |
| 3 | Observe the scope and record the thruster output as well as the velocity of ROV in all six degrees of freedom. |
| **Expected result** | The thrust output varies with the varying rotational speed of individual thrusters and ROV starts the movement accordingly |

Table 4.3: Test scenario 2 - Thruster response under MATLAB environment

of seven thrusters are available in the Merlin UCV which are named $VFP$, $VFS$, $VAC$, $HFP$, $HFS$, $HAP$, $HAS$ where the naming convention is as follows: $V$ - vertical, $H$ - horizontal, $F$ - forward, $A$ - aft, $P$ - port, $S$ - starboard. For each thrusters, both the feedback value/ process value and the command/ set-point were collected. The *Latitude* and *Longitude* are computed by Inertial Navigation System on the ROV using measurements from the Inertial Measurement Unit, Doppler Velocity Log and transponders through a Kalman filter

## 4.6 Performance Comparison - Simulator v/s Merlin UCV

Real-time simulation plays a vital role in the design and test of complex systems, especially when interfacing the subsystems. The main goal of this experiments is to compare the corresponding experimental data from Merlin UCV and verify to represent the physical behavior of the ROV simulator. A dynamics simulation model was developed considering the characteristics of individual seven thrusters, the ROV dynamics, and the operating conditions of the hardware. The kinetic behavior of both the Hardware (Merlin UCV) and the MATLAB ROV simulator is compared when they are in underwater conditions. A close proximity to environmental conditions was maintained during the experiment by tuning the ROV parameters as well as sea bed conditions. The test plan is detailed in Table 4.7.

| To study individual thruster response in ROS | |
|---|---|
| **S No.** | **Procedure** |
| **Test preconditions** | Make sure all the UCV ROS packages are build and sourced . |
| 1 | Follow the steps from 1 to 4 listed in table 4.2 |
| 2 | Publish data to the topics for individual thruster input with a value varying from 0 RPM to 70 RPM and back to 0 RPM manually at a rate of 10 Hz. |
| 3 | Load the subscribed topics to *PlotJuggler* data analysis window and record thrust and velocity of ROV in all six degrees of freedom. |
| **Expected result** | The thrust output varies with the varying rotational speed of individual thrusters and ROV starts the movement accordingly. |

Table 4.4: Test scenario 2 - Thruster response under ROS environment

| To study thruster response for movement commands in MATLAB | |
|---|---|
| **S No.** | **Procedure** |
| **Test preconditions** | Make sure all necessary directories are included in the path and the C-coded S-functions are compiled. |
| 1 | Follow the steps 1 and 2 listed in table 4.1 |
| 2 | Load the directory *scripts* and run the script *rovSimRun.m* |
| 4 | Record the plots for forces, thrust and velocity of ROV in six degrees of freedom. |
| **Expected result** | Not applicable |

Table 4.5: Test scenario 3 - Thruster response for varying movement commands under MATLAB environment

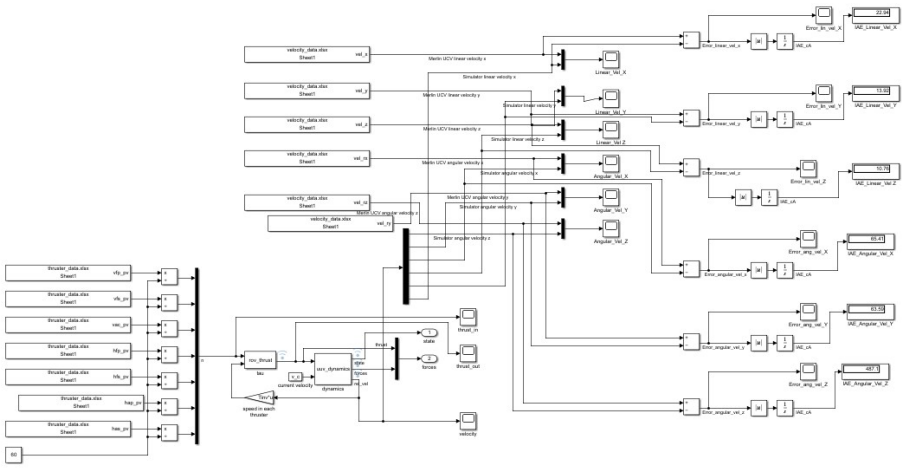


Figure 4.3: Simulink Model for Comparison Study

| To study thruster response for movement commands in ROS | |
|---|---|
| **S No.** | **Procedure** |
| **Test preconditions** | Make sure all the UCV ROS packages are build and sourced . |
| 1 | Follow the steps from 1 and 2 listed in table 4.2 |
| 2 | Run another terminal and launch the *PlotJuggler* tool by running the command *rosrun plotjuggler plotjuggler* |
| 3 | Load the subscribed topics to *PlotJuggler* data analysis window and record individual thruster's output, thrust and velocity of ROV in all six degrees of freedom. |
| 4 | To study the effect of following movements *surge*, *sway*, *heave* and *yaw* in positive and negative directions, on individual thrusters, perform modifications on the script *teleop.py* so that each type of movement is performed for 30 seconds with a waiting period of another 30 seconds. The waiting time is for retaining the ROV stability after each movement finishes. |
| **Expected result** | Not applicable |

Table 4.6: Test scenario 3 - Thruster response for varying movement commands under ROS environment

| To compare the performance between the hardware and simulator | |
|---|---|
| **S No.** | **Procedure** |
| **Test preconditions** | Make sure all necessary directories are included in the path and the C-coded S-functions are compiled. |
| 1 | Load the directory *models* and open the Simulink model *rov_simulator_rev*03.*slx* as shown in Figure 4.3 |
| 2 | Once the Simulink model is opened, Load the directory *scripts* and open the m-script *merlin_vs_simulator.m* |
| 3 | Run the script and record individual thruster's output, thrust, velocity of ROV in all six degrees of freedom and the error plots. |
| **Expected result** | Not applicable |

Table 4.7: Test scenario 4 - Performance Comparison - ROV Simulator Versus Merlin UCV

# Chapter 5

# Results

## Chapter Outline

This chapter provides a clear idea of exactly what is found and keeps the data itself separate from the interpretation i.e., discussion and analysis. A concise summary of results for each experiment listed in the previous chapter is discussed. The chapter objectively report the findings and only brief observations were presented in relation to each question.

The validation of results from the experiments play a vital role to measure the extent to which the results reflect the truth. While performing data validation, several things have to be considered such as the accuracy of sensors in data measurements, time synchronisation when working with real-time data, and so on. The question raised is, whether the value of the dependent variable has been changed due to the manipulated variable from the experimental study or if confounding factors have been the cause. As a result, time synchronised data has been collected across the data to keep track of what is measured and when is it measured. This chapter details the data capture.

## 5.1 Test scenario 1 - ROV Model Stability

For test scenario 1, inputs to individual thrusters were set to zero. In MATLAB the results were captured using blocks from the standard Simulink library functions while in ROS, the same was captured by using the tool *PlotJuggler*. The Figure 5.1 shows the ROV velocity in six degrees of freedom. On the other hand, Figure 5.2 was to study the inputs to individual thruster and corresponding response. The test results from MATLAB environment are shown from Figure 5.3 to Figure 5.5. Inputs to individual thrusters were plotted along with ROV's thrust and velocity in six degrees of freedom. The main goal behind this experiment is to compare the ROV stability in ROS and MATLAB environments which will be closely examined in latter chapter. For better understanding, all the plots are plotted as time series with simulation time on X-axis.


Figure 5.1: ROV Velocity Versus Time in ROS - Test Scenario 1

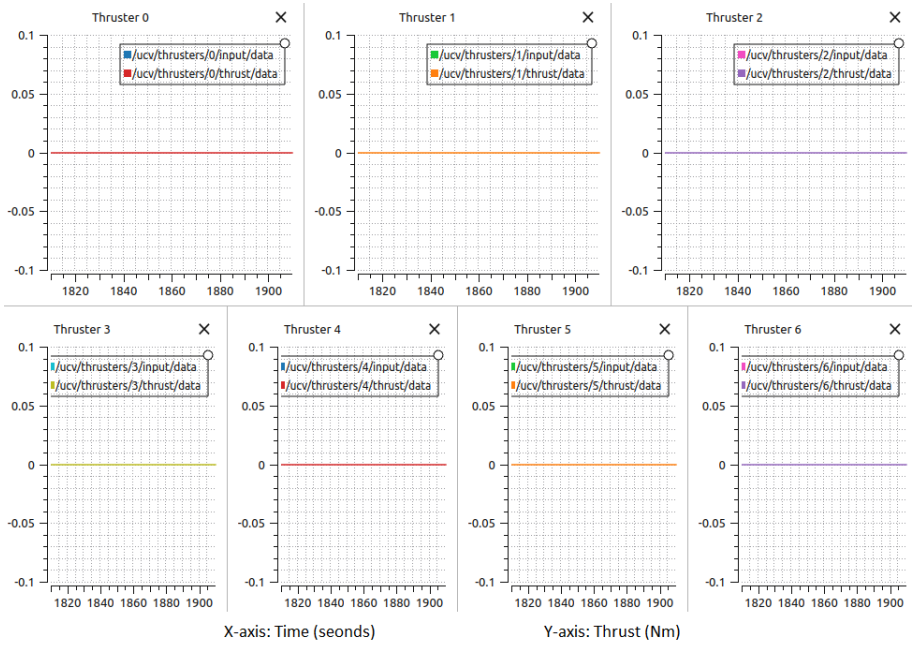



Figure 5.2: ROV Individual Thruster Response in ROS - Test Scenario 1

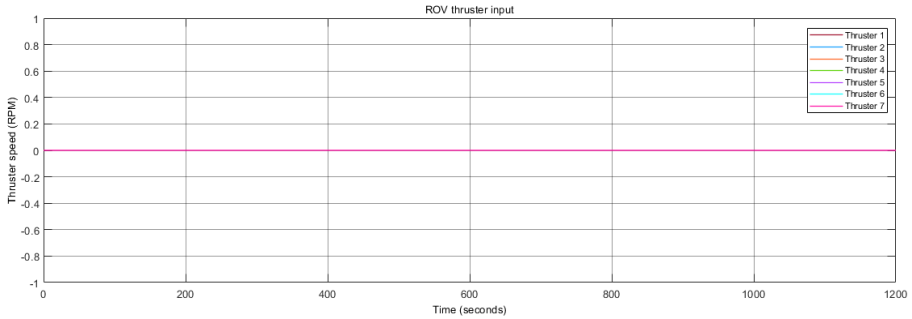



Figure 5.3: Input to individual thruster in MATLAB - Test Scenario 1

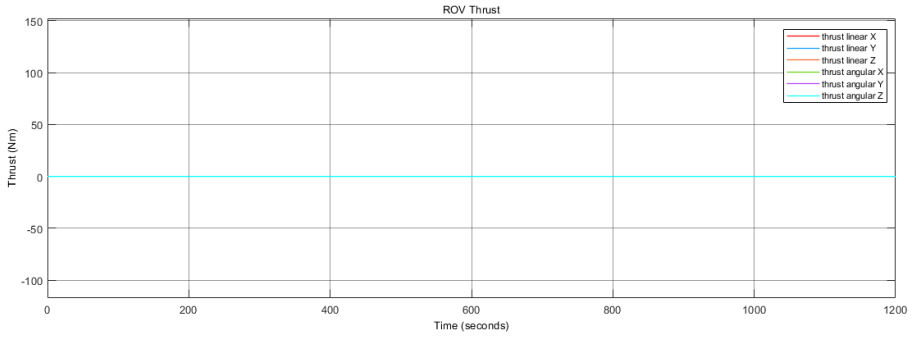



Figure 5.4: ROV Thrust in MATLAB - Test Scenario 1

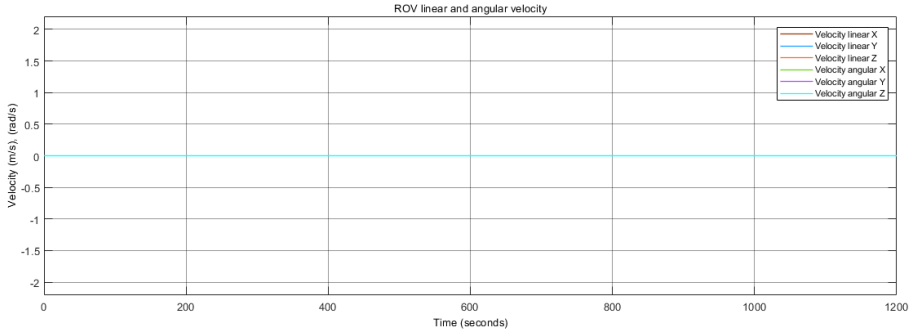



Figure 5.5: ROV Velocity in MATLAB - Test Scenario 1

## 5.2 Test scenario 2 - Thruster Response for Varying Inputs

Thrusters are one of the vital components responsible for ROV's dynamics. As a result, it is important to test and verify individual thruster responses. Test scenario 2 focuses the test coverage for individual thruster response for varying inputs. In ROS simulation, the rotation of individual thruster are varied at regular intervals and published with a constant frequency rate using the *RQT_GUI* tool manually and results were captured using the tool *PlotJuggler*. On the otherhand, MATLAB simulation runs a test script that performs the parameter change in input, automatically and capture the results using standard Simulink library functions. The motive behind this experiment is to study the behaviour of individual thrusters of ROV under independent simulation environments, in this study, the ROS and the MATLAB environments. All the plots are done in time series with simulation time on X-axis.

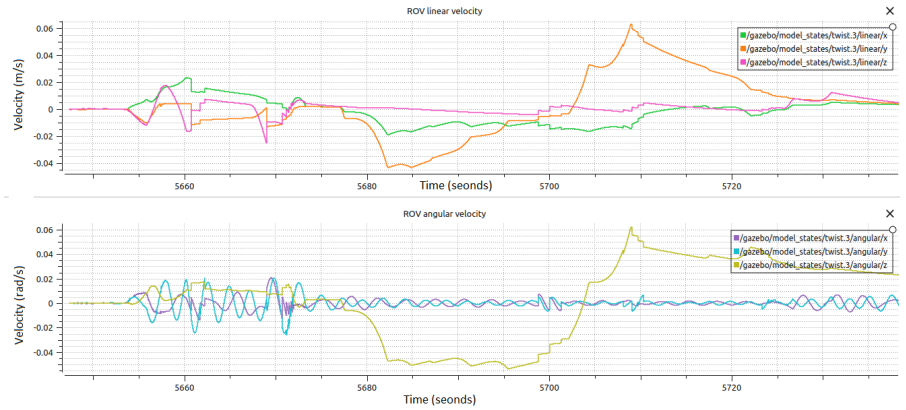



Figure 5.6: ROV Velocity Versus Time in ROS - Test Scenario 2

The results from ROS are shown from Figure 5.6 to Figure 5.9 in which Figure 5.6 shows the ROV velocity in six degrees of freedom, 5.7 shows the thrust from individual thruster for corresponding inputs i.e., thruster rotational speed. The Figure 5.8 and Figure 5.9 shows the individual

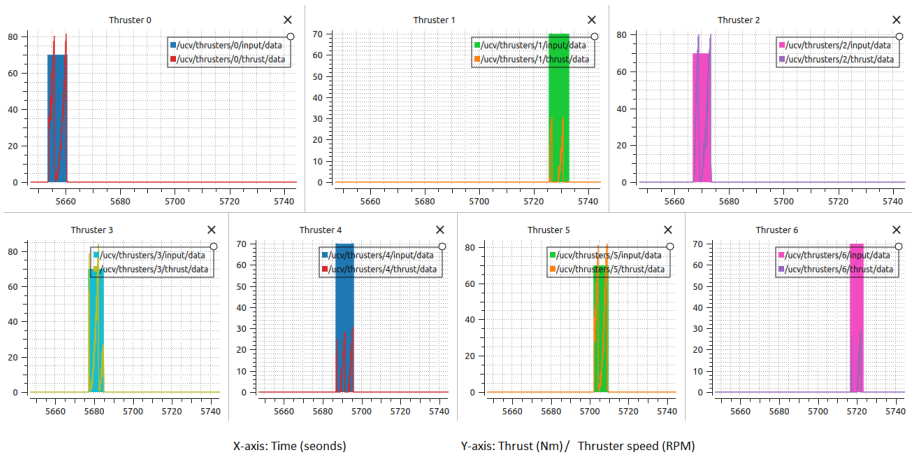



Figure 5.7: ROV Individual Thruster Response in ROS - Test Scenario 2

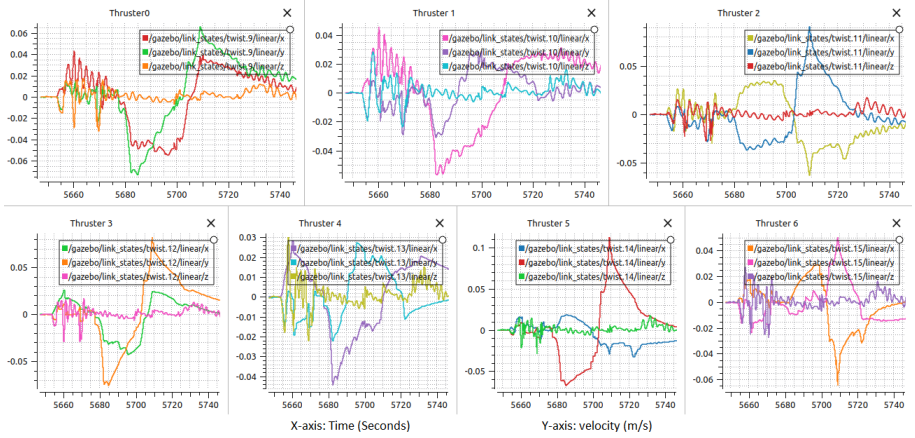



Figure 5.8: Individual Thruster Velocity (linear) in ROS - Test Scenario 2

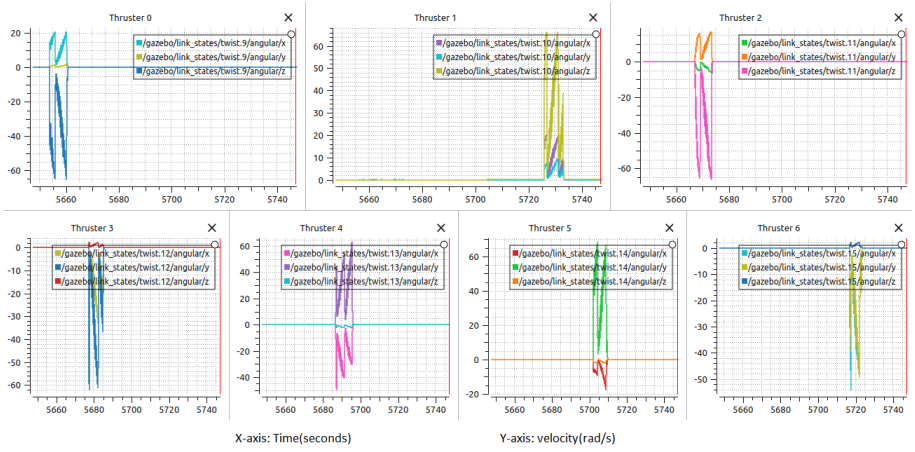



Figure 5.9: Individual Thruster Velocity (angular) in ROS - Test Scenario 2

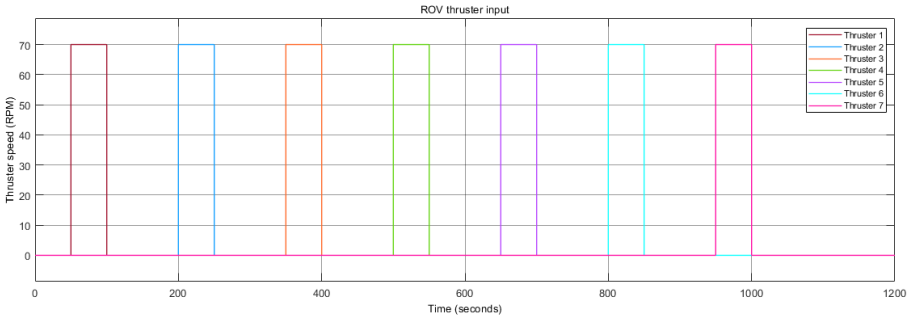



Figure 5.10: Input to individual thruster in MATLAB - Test Scenario 2

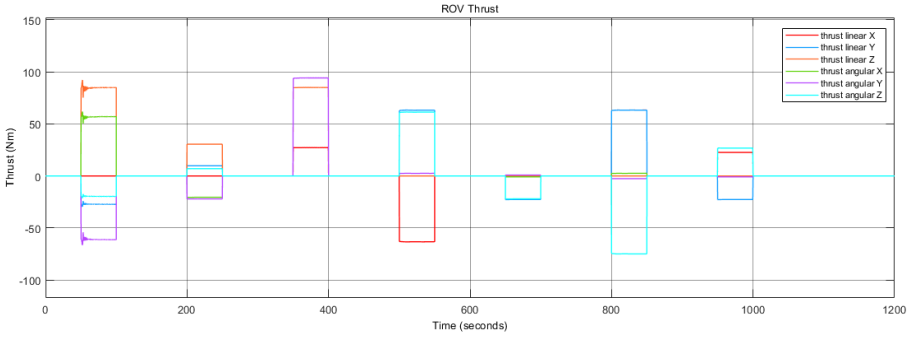



Figure 5.11: ROV Thrust in MATLAB - Test Scenario 2

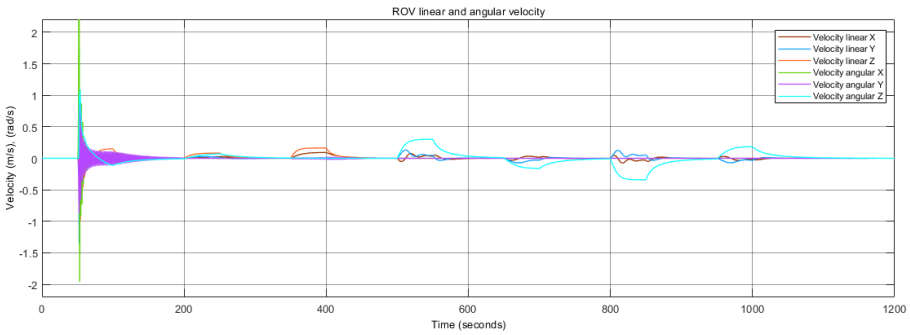



Figure 5.12: ROV Velocity in MATLAB - Test Scenario 2

thrusters linear and angular velocity feedback in six degrees of freedom. The test results from MATLAB environment are shown from Figure 5.10 to Figure 5.12. Figure 5.10 shows the inputs i.e., thruster rotational speed commanded to individual thruster. The feedback i.e., ROV's thrust and velocity in six degrees of freedom are shown in Figure 5.11 and Figure 5.12. The analysis on results will be detailed in later chapter.

## 5.3 Test scenario 3 - Thruster Response for Varying Movement Commands

The test scope for this section covers the response of individual thruster in ROS simulation for the movements - *surge*, *sway*, *heave* and *yaw* both in positive and negative directions. The rotation of individual thruster are controlled depending on the movement commands received from the python script *teleop.py* and results were captured using the tool *PlotJuggler*. Individual thruster response at ideal conditions for corresponding inputs were captured and are shown from Figure 5.16 to Figure 5.13. On the other hand, MATLAB simulations control the

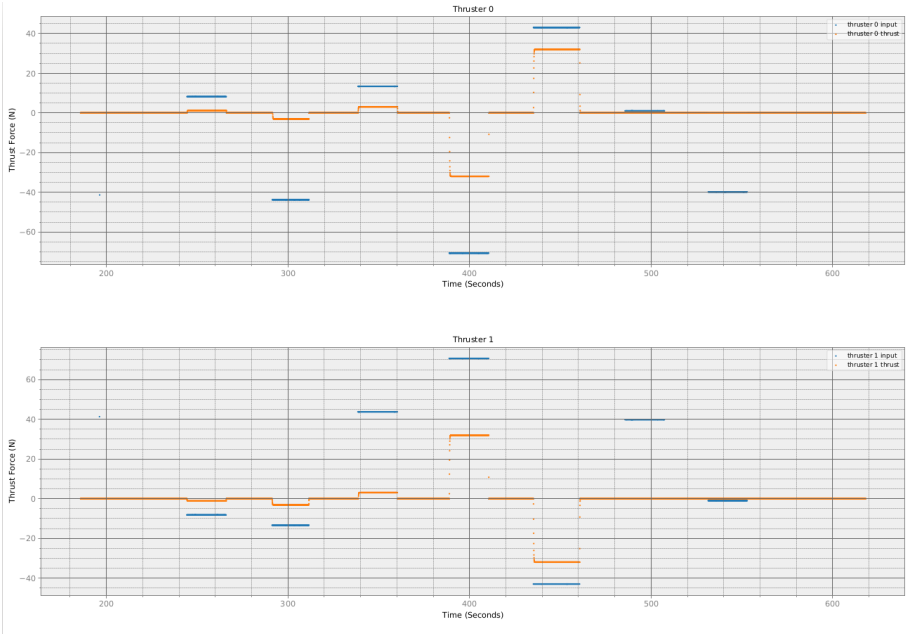



Figure 5.13: Thruster 1 and 0 Input/ Output Versus Time - Test Scenario 3

movement directions using way points in Cartesian coordinate system. The test results from MATLAB environment are shown from Figure 5.17 to Figure 5.18. Figure 5.17 shows the linear and angular displacement, while the linear and angular velocities are plotted in Figure 5.19. Figure 5.18 details the ROV simulator's movement as a three dimensional plot.

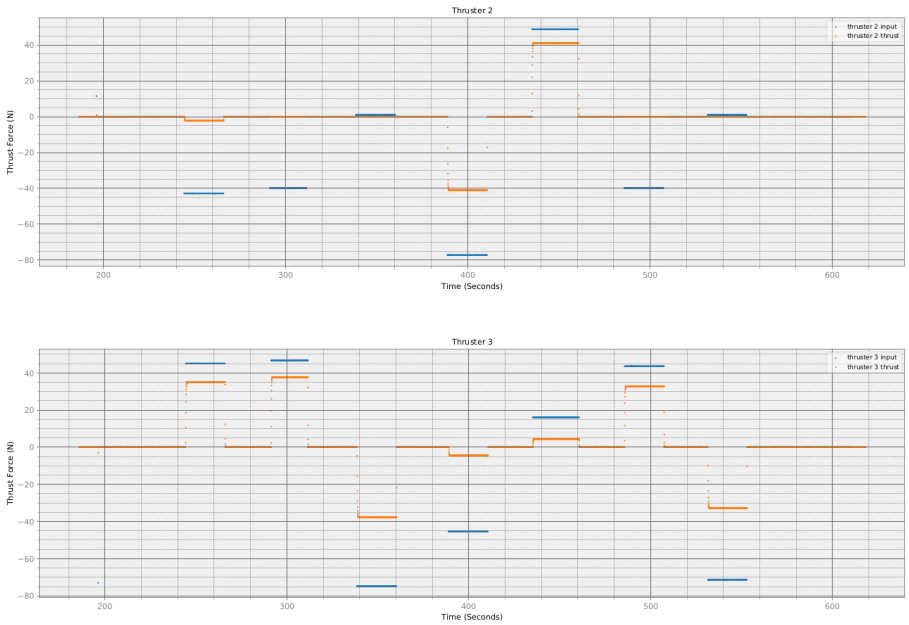



Figure 5.14: Thruster 3 and 2 Input/ Output Versus Time - Test Scenario 3

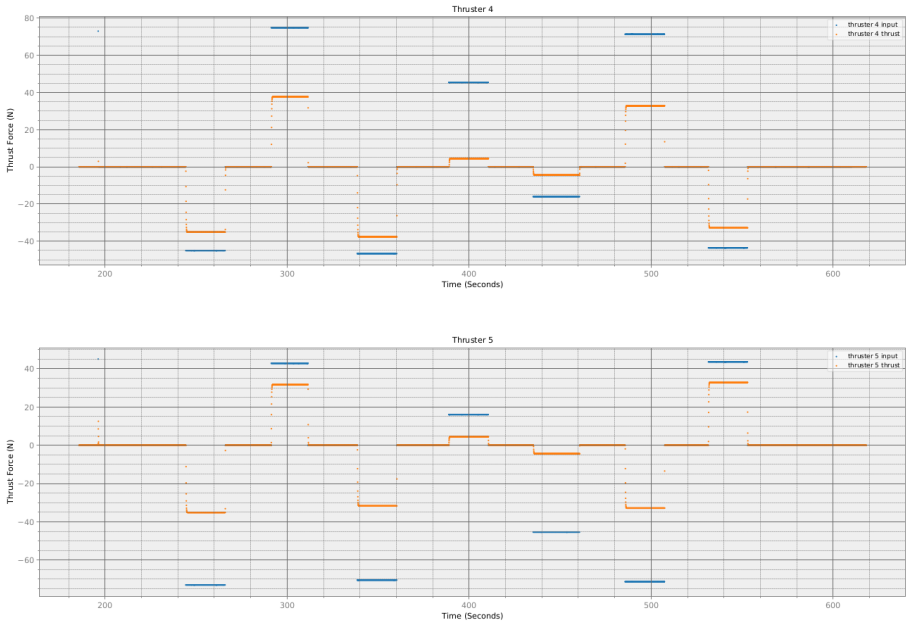



Figure 5.15: Thruster 5 and 4 Input/ Output Versus Time - Test Scenario 3

## 5.4 Test scenario 4 - Data Samples from Hardware

The available seven thrusters in the Merlin UCV are named as $VFP$, $VFS$, $VAC$, $HFP$, $HFS$, $HAP$, $HAS$ where the naming convention is as follows: $V$ - vertical, $H$ - horizontal, $F$ - forward, $A$ - aft, $P$ - port, $S$ - starboard. For each thrusters, both the feedback value/ process value and the command/

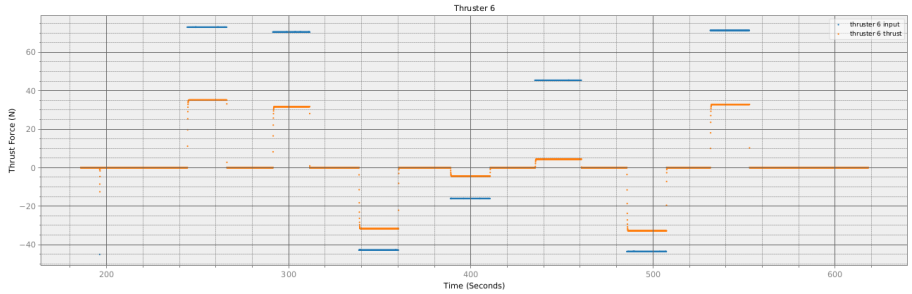



Figure 5.16: Thruster 6 Input/ Output Versus Time - Test Scenario 3

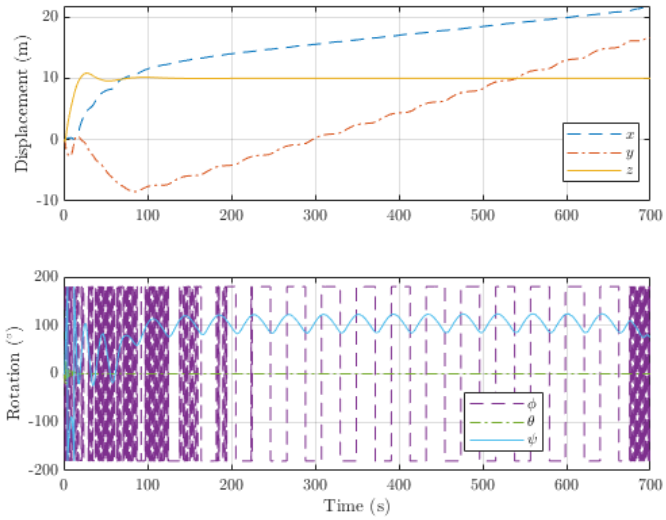



Figure 5.17: ROV Change in Position in MATLAB - Test Scenario 3

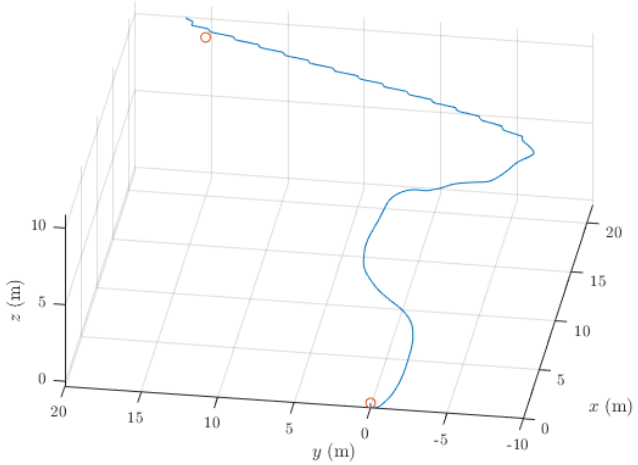



Figure 5.18: ROV motion 3-D plot in MATLAB - Test Scenario 3

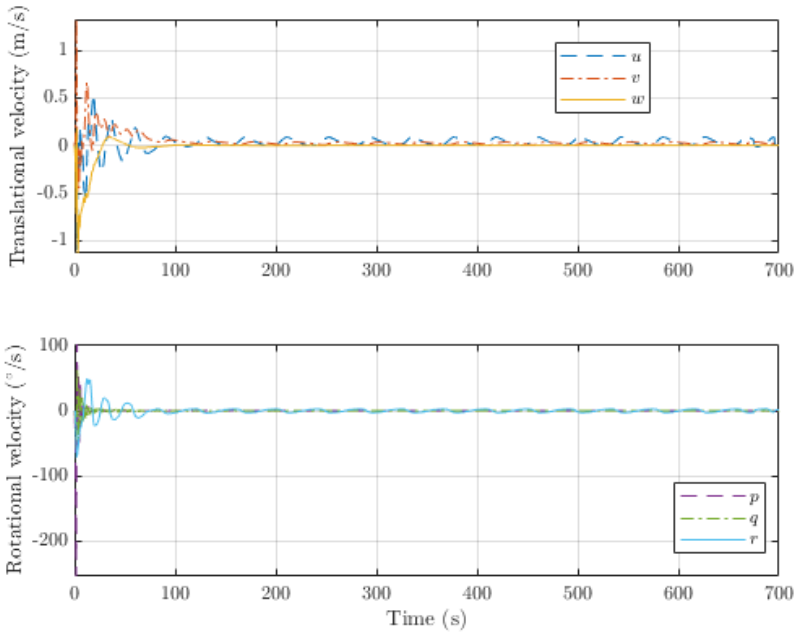
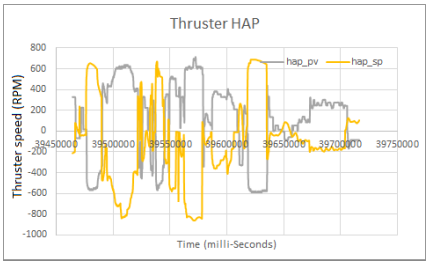



Figure 5.19: ROV Velocity in MATLAB - Test Scenario 3

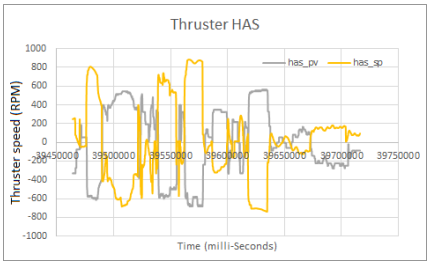
set-point were plotted. The data also consists of *Timestamps* in units *hour*, *minutes*, *seconds* and *milli − seconds*; *Latitude* and *Longitude* in *decimal degrees*; *depth* in *meters*; *attitude* in *degrees*; *linear velocity* in *meters/second*; *angular velocity* in *degrees/second*; *linear acceleration* in $meter^2/second$ and *rotation* of thrusters in *rpm*. Figure 5.20 details the thruster rotation from the hardware for varying movement commands from the control station.

## 5.5 Test scenario 5 - Performance Comparison

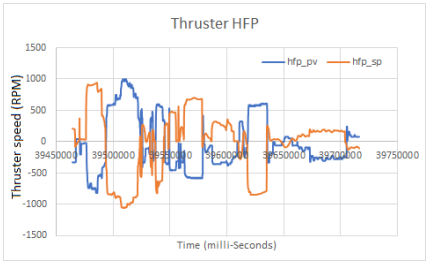
The test coverage under this section focuses on comparing performance between hardware data and ROV simulator under matlab. The achieved ROV velocity for same thruster input were taken as the parameter to compare the performance. Figure 5.21 shows the comparison on achieved thruster velocity for linear and angular motions from the hardware and simulator.



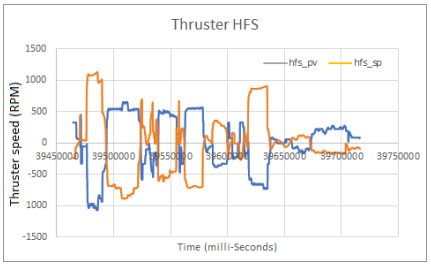

(a) HAP



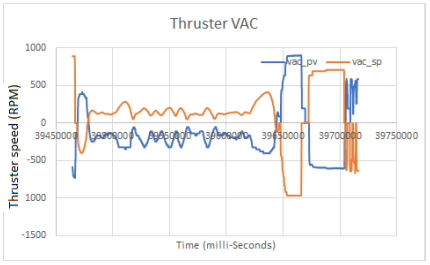

(b) HAS



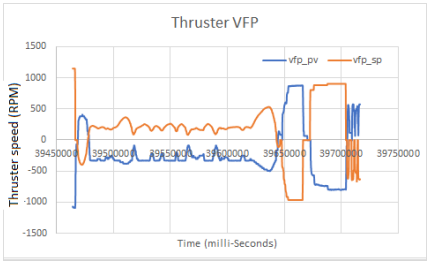

(c) HFP



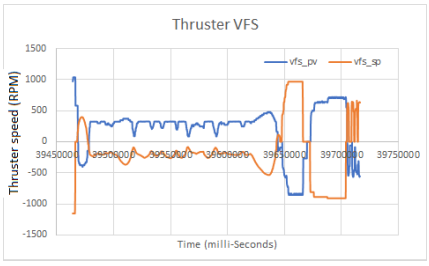

(d) HFS




(e) VAC

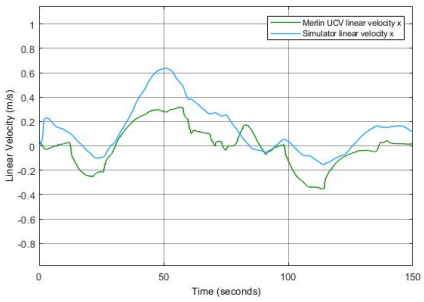



(f) VFP

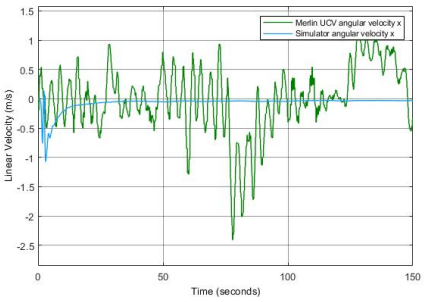



(g) VFS

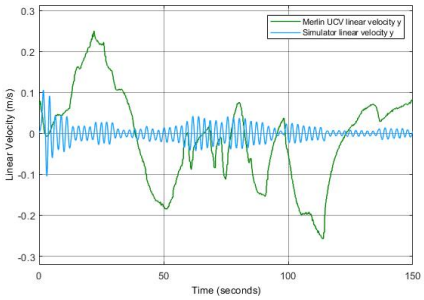
Figure 5.20: Response From Various Thrusters in Merlin UCV



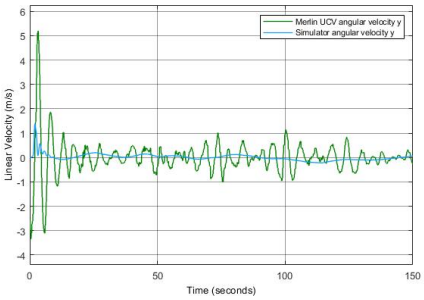

(a) Linear velocity X-axis



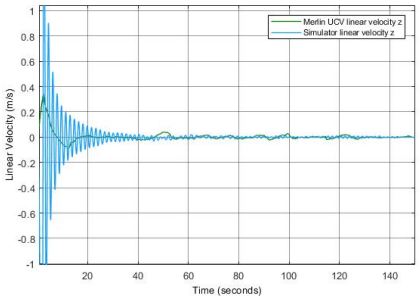

(b) Angular velocity X-axis



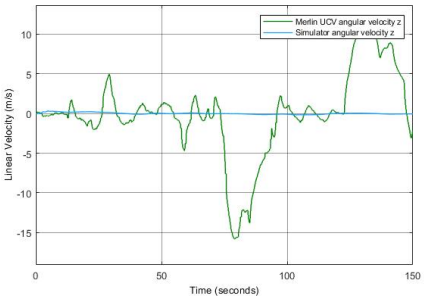

(c) Linear velocity Y-axis




(d) Angular velocity Y-axis




(e) Linear velocity Z-axis




(f) Angular velocity Z-axis

Figure 5.21: Performance comparison Merlin Hardware Versus Simulator

# Chapter 6

# Discussion

## Chapter Outline

This chapter interprets the results listed in the previous chapter in detail and draws out their implications. The research findings from the simulations were elaborated on and evaluated in a thoroughly and coherently manner throughout the chapter.

## 6.1 Preliminary Analysis on ROV Dynamics

The ROV motion in 3-D space was investigated in two entirely independent simulation platforms: ROS and MATLAB/ Simulink based on the mathematical model. The initial simulation phase was focused on comparing the model uncertainties among the different simulation environments and the results were promising with a 10% variation in ROV dynamics across the platforms. The above behaviour can be seen from the individual thruster response as seen in Figure 5.9 and Figure 5.10. It is also noticeable that the two systems have similar sensitivity towards underwater dynamic uncertainties where the variations in rotational velocities are slightly greater than that in linear velocities (translational motions). The simulation results indicates that light variations in thruster parameters causes a considerable impact on simulator performance. As a result, the accuracy of the model was vital for studying the performance comparison between the simulator and the hardware which is discussed under Section 6.5. To prove the model is accurate, the analysis has to start with the ROV model stability with all the thrusters running at zero speed and is discussed under Section 6.2.

## 6.2 ROV Simulator Stability

When discussing the stability of an ROV, the two main factors taken into account are: the vehicle must be buoyant enough so that it can be maneuvered easily up or down without using too much energy and it must also be stable and doesn't tip over. Results from the test scenario 1 shows that the stability of an ROV is affected by the distance between the center of gravity and the center of buoyancy. Figure 5.5 and Figure 5.1 shows both the values for linear and angular velocities of the ROV in all the 3 axes are almost zero when there is no thrust generated from the thrusters. It has also been observed that the result stays consistent in both MATLAB/Simulink as well as ROS simulation environment.

During the initial phase of modelling, the main challenges to achieve stability was from the physical parameters of ROV such as system inertia matrix comprising the rigid body mass($\underline{M}_{RB}$) and added mass($\underline{M}_A$), drag matrix which consists of linear damping($\underline{D}_l$) and quadratic damping($\underline{D}_q$), volume($V$), center of buoyancy ($x_b$, $y_b$, $z_b$) and center of gravity($x_g$, $y_g$, $z_g$). When the above parameters were tuned as close to the hardware, the simulator got its stability. Another observation from both simulation platforms are that the ROV sinks to the seabed gradually over time with out tip over when the thrusters is running at zero speed. The noticeable difference across the simulation platforms are on the noise in velocity measurements. In ROS simulation platform the noise in velocities is with in the magnitude of 1 x $10^{-3}$ $m/s$ while in MATLAB the noises has to be manually fed into the ROV dynamics based on the real-time data from hardware.

## 6.3 Thruster Inputs Versus Response

The one-to-one relationship of thrusters among simulators is vital for a precise ROV simulation and can be studied from the recorded results from the test scenario 2 detailed under Section 5.2. From the Figure 5.6 and Figure 5.12 shows that the ROV simulator in the MATLAB/ Simulink environment as well as in the ROS environment has similar and consistent results for velocities in six degrees of freedom for the corresponding thruster speed. It can also be perceived from the simulation results that the thrust output from individual thrusters has almost identical results across the two independent simulation environments and is detailed in the Table 6.1.

| **Comparing thruster response across simulation platforms** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Platform** | **Thruster/ input(RPM)** | **Output thrust (Nm)** | | | | | | |
| | | T0 | T1 | T2 | T3 | T4 | T5 | T6 |
| MATLAB | T0 - 70.0 | 82.7 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| | T1 - 70.0 | 0.0 | 31.3 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| | T2 - 70.0 | 0.0 | 0.0 | 83.8 | 0.0 | 0.0 | 0.0 | 0.0 |
| | T3 - 70.0 | 0.0 | 0.0 | 0.0 | 87.0 | 0.0 | 0.0 | 0.0 |
| | T4 - 70.0 | 0.0 | 0.0 | 0.0 | 0.0 | 35.2 | 0.0 | 0.0 |
| | T5 - 70.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 86.7 | 0.0 |
| | T6 - 70.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 31.4 |
| ROS | T0 - 70.0 | 80.5 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| | T1 - 70.0 | 0.0 | 30.4 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| | T2 - 70.0 | 0.0 | 0.0 | 78.7 | 0.0 | 0.0 | 0.0 | 0.0 |
| | T3 - 70.0 | 0.0 | 0.0 | 0.0 | 82.1 | 0.0 | 0.0 | 0.0 |
| | T4 - 70.0 | 0.0 | 0.0 | 0.0 | 0.0 | 31.7 | 0.0 | 0.0 |
| | T5 - 70.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 80.3 | 0.0 |
| | T6 - 70.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 29.5 |

Table 6.1: Thruster response comparison

## 6.4 Movement Command Versus Thruster response

The control algorithm and the PID tuning parameters for the ROV simulator in MATLAB environment is premature and the movement directions were fed using waypoints in the Cartesian coordinate system which increases the complexity to perform this study. The above unstability can be visualised from the plots shown from Figure 5.17 to Figure 5.18. On the other hand, the simulator under ROS environment can be commanded to move in positive and negative directions for *surge*, *sway*, *heave*, and *yaw* movements under the ROS environment. For each commanded movement, the activated thrusters and their corresponding thrust output vary depending on thruster characteristics which consist of thruster allocation matrix, thruster diameter, thruster coefficient, and thrust

loss factors for individual thruster. Detailed analysis of results from Figure 5.16 to Figure 5.13 following observations were made for various ROV motions.

- **Backward (surge negative):** For a *backward* movement, thrusters namely 3, 4, 5 and 6 where in active state on which, thrusters 3 and 6 where rotating opposite to 4 and 5. The thrust force was at negative side for thrusters 3 and 6 while it was at positive side for thrusters 4 and 5.
- **Forward (surge positive):** In case of $forward$ movement, the difference in thrusters behaviour compared to *backward* movement is that the thrust force was at positive side for thrusters 3 and 6, while it was at negative side for thrusters 4 and 5.
- **Right (sway positive):** Thrusters 3, 4, 5 and 6 were active and all of their individual thrust were at positive side, for a movement towards *right* direction.
- **Left (sway negative):** The only difference for a $left$ movement as compared to the *right* movement, is that the thrust force was at negative side.
- **Upward (heave positive):** For an $upward$ movement, the active thrusters were 0, 1 and 2 in which, thruster 1 rotates in the opposite direction to the thruster 0 and thruster 3. For thrusters 0 and 2, the thrust force was at negative side while, it was opposite for thruster 1.
- **Downward (heave neagtive):** In comparison to $upward$ movement, the difference for *downward* movement is that thrusters 0 and 2, the thrust force was at positive side while, it was at negative side for thruster 1.
- **Rotate right (roll positive):** For a movement command to *rotate right*, thrusters namely 3, 4, 5 and 6 where in active state on which, thrusters 3 and 4 where rotating opposite to 5 and 6. The thrust force was at positive side for thrusters 3 and 4 while it was at negative side for thrusters 5 and 6.
- **Rotate left (roll negative):** In case of *rotate* $left$ movement, the difference in thrusters behaviour compared to *rotate right* movement is that the thrust force was at negative side for thrusters 3 and 4, while it was at positive side for thrusters 5 and 6.

## 6.5 Simulator Versus Merlin UCV

The stability of the ROV was analyzed and an open-loop simulation was performed under MATLAB/Simulink platform with a motive to compare the performance between the hardware and the simulator. Preliminary analysis draws the inference that there are few variations in the magnitude

of thrust force from individual thrusters compared to that of the real hardware, but the simulation results were realistic. In real life scenario, it is merely impossible to achieve the perfect reality, but by taking account of ROV's physical parameters and disturbance in operational environment the results indicate that the model is correct. The error calculations from the comparison study are detailed in the table 6.2. The initial problems faced was to achieving the stability of the simulator. The root-cause was realised from consecutive tests and simulation which contribute to instability are the ROV physical parameters such as weight of ROV , sea water density, system inertia, and drag matrix. Once the values for the above parameters correspond to that of the hardware, the stability was achieved. Next challenge was to achieve the performance and it has been observed that, the major factors that contribute towards it are thruster allocation matrix, thruster diameter, thruster coefficient and thrust loss factors for individual thruster. In the above thruster diameter and thruster coefficient are constants but thruster allocation matrix and thrust loss factors needed tuning to match simulator performance with that of hardware.

| **Comparing thruster response across simulation platforms** | | | |
|---|---|---|---|
| **Velocity** | **Axis** | **Error** | **Integral Absolute Error** |
| Linear | X | -0.096 | 22.94 |
| | Y | 0.078 | 13.92 |
| | Z | -0.006 | 10.76 |
| Angular | X | -0.299 | 65.41 |
| | Y | -0.32 | 63.59 |
| | Z | -2.01 | 487.1 |

Table 6.2: Thruster response comparison

# Chapter 7

# Conclusion and Future Work

## Chapter Outline

This chapter interprets the results to an overall answer to the main research question of this study. The main research question was that, how far is the performance between the hardware (Merlin UCV) and the software simulations. The chapter provides a summary of findings and an answer to the research question and concludes with future work and improvements.

## 7.1 Conclusion

The main goal of this work was to compare the model performance between simulator and hardware at its operational conditions. To get an insight into the components and working principle of an ROV, several related research works were reviewed, followed by analysing of the relevant works done under the research topic. Installation and setup of simulation environments, specifically ROS/Gazebo was a challenging task. A few challenges worth mentioning are the simulation of custom components in an ROV, fixing Operating System(OS) compatibility related issues while settingup the simulation environment, unsupported custom ROS messages in MATLAB. Data collection was done on both simulator and hardware. Initially, Semi-automated scripts were made for data collection and graphical representation to perform a comparative study under ROS but, lately more sophisticated and open source data visualisation tools that are compatible with ROS were found and utilized.

### 7.1.1 Simulation Environment

The two simulation platforms ROS/Gazebo and MATLAB/Simulink are used in the development of the ROV simulator. ROS provides software management and real-world physics in a high fidelity simulation that can be done using Gazebo, while MATLAB provides dynamics simulation where algorithms can be easily implemented and tested, easy debugging capability and help to perform extensive data analysis and visualization. MATLAB provides vast built-in library of functions as part of various tool kits which makes task more comfortable and saves development time. It offers plotting and data visualisation tools as well as tools to develop GUI based applications. Both the simulation platform requires a configuration file of the ROV model before simulations can be launched. This file includes the ROV physical parameters, thruster characteristics as well as parameters that define operational environments. In ROS simulation, the above file cannot be done after the simulation is loaded, while in MATLAB, the ROV parameters (ex: thruster characteristics) can be tuned even when the simulation is loaded which provides a several advantages during simulation. One example is, to study the effect of change in the value of thruster allocation matrix against the stability of ROV without compiling the code each time.

### 7.1.2 Dynamic Model of ROV Against Hardware

A dynamic model of an ROV is implemented in both ROS and MATLAB, and an analysis is done to test the stability and flexibility to compare the performance which resulted in choosing MATLAB/Simulink environment to perform a comparison study between the simulator and the hardware. The results under ROS environment have dynamic noise response while in MATLAB the results were more or less stable which is in contradiction to real-time response from the hardware. During the initial phase, out

of several modelling problems encountered with the ROV dynamics, the prime one to be mentioned is that, In MATLAB the simulation does not manage to simulate for a long period. It was discovered later that the root cause for the above issue was that the ROV lose its stability and solved later by tuning the ROV parameters close to the hardware. The validation of results across two different simulation platforms was done and observed realistic simulation results. The simulation results were compared against the experimental data from the hardware and the results were promising with slight variations in linear and angular velocities for both simulator and the hardware. It has been noticed during simulations that the thruster allocation matrix, thruster coefficient and thruster loss factor were the main factors contributing to the tuning of ROV velocities in six degrees of freedom.

## 7.2 Further Work

This research work introduces a way of simulating an ROV model which can be a replica of the hardware and compare the performance between both. Few proposals to improvise the simulator as a part of further work to continue the development of the platform, including some specific proposals of how the problems can be solved are detailed below.

- **Expansion of the Simulation Platform:** Addition of new modules (for example: an Inertial Navigation System) and control algorithms in the existing simulator can be contained as future work. Graphical User Interface is another improvement that can be looked into, that makes the simulator user-friendly. As of today, it requires certain knowledge at developer level to load the simulator.

- **Improvements under MATLAB simulation:** For simulating the disturbance caused by wave loads under the desired weather conditions, noise blocks that are available in MATLAB tool kits can be modified and fed into the simulator input. It will help the simulator to produce more close results to the hardware at its operating conditions. In addition, It would be better if the simulation can take account of how the current varies with depth, and also how it varies with other environmental conditions such as wind speed.

- **Improvements under ROS simulation:** The ROS simulator provides only a few tool-kits for verification of simulation results with output from the hardware and it is an area of improvement. Also, simulator under ROS Environment can be improvised with an option for feeding the same input as that of hardware instead of controlling the simulator with a separate input from a keyboard/ joystick.

# Bibliography


[1] Yifan Song, David Nakath, Mengkun She, Furkan Elibol, and Kevin Köser. Deep sea robotic imaging simulator for uuv development. *arXiv preprint arXiv:2006.15398*, 2020.

[2] Musa Morena Marcusso Manhães, Sebastian A. Scherer, Martin Voss, Luiz Ricardo Douat, and Thomas Rauschenbach. UUV simulator: A gazebo-based package for underwater intervention and multi-robot simulation. In *OCEANS 2016 MTS/IEEE Monterey*. IEEE, sep 2016.

[3] Russell Smith. Open dynamics engine v0. 5 user guide. *http://ode. org/*, 2005.

[4] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2149 – 2154 vol.3, 04 2004.

[5] Robert D Christ and Robert L Wernli Sr. *The ROV manual: a user guide for remotely operated vehicles*. Butterworth-Heinemann, 2013.

[6] George A Bekey. *Autonomous robots: from biological inspiration to implementation and control*. MIT press, 2005.

[7] Kerry Pavek, James Albus, and Elena Messina. Autonomy levels for unmanned systems (alfus) framework: An update. *Proceedings of SPIE - The International Society for Optical Engineering*, 05 2005.

[8] Geoffrey Ho, Nada Pavlovic, and Robert Arrabito. Human factors issues with operating unmanned underwater vehicles. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, pages 429–433. SAGE Publications Sage CA: Los Angeles, CA, 2011.

[9] NORSOK Standard. Remotely operated vehicle (rov) services. *U-102, Rev*, 1, 2003.

[10] Yulian Mahendra, Muhammad Iqbal, and Husnul Furqon. Introduction to residential ROV: New concept of ROV for oil and gas activities. In *Proc of Indonesian Petroleum Association 42nd Annual Convention*. Indonesian Petroleum Association, May 2018.

[11] Romano Capocci, Gerard Dooly, Edin Omerdić, Joseph Coleman, Thomas Newe, and Daniel Toal. Inspection-class remotely operated vehicles—a review. *Journal of Marine Science and Engineering*, 5(1):13, 2017.

[12] IKM Subsea AS. Merlin UCV product datasheet, 2018.

[13] IKM Subsea AS. Merlin UCV RROV product datasheet, 2019.

[14] Shashank Swaminathan and Srikanth Saripalli. A framework for modeling underwater vehicles in modelica. In *2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV)*, pages 1–6, 2018.

[15] O Matsebe, CM Kumile, and NS Tlale. A review of virtual simulators for autonomous underwater vehicles (auvs). *IFAC Proceedings Volumes*, 41(1):31–37, 2008.

[16] Iis Hamsir Ayub Wahab, Rintania Elliyati Nuryaningsih, and Achmad Pradjudin Sardju. Proposed mathematical modeling of small remotely operated vehicle (rov) movement. *Journal of Physics: Conference Series*, 1569:042002, Jul 2020.

[17] Olivier Kermorgant. A dynamic simulator for underwater vehicle-manipulators. In Davide Brugali, Jan F. Broenink, Torsten Kroeger, and Bruce A. MacDonald, editors, *Simulation, Modeling, and Programming for Autonomous Robots*, pages 25–36, Cham, 2014. Springer International Publishing.

[18] L. Bevilacqua, W. Kleczka, and E. Kreuzer. On the mathematical modeling of rov's. *IFAC Proceedings Volumes*, 24(9):51–54, Sep 1991.

[19] Håkon Teigland, Vahid Hassani, and Ments Tore Møller. Operator focused automation of rov operations. In *2020 IEEE/OES Autonomous Underwater Vehicles Symposium (AUV)(50043)*, pages 1–7. IEEE, 2020.

[20] Zandra B. Rivera, Marco C. De Simone, and Domenico Guida. Unmanned ground vehicle modelling in gazebo/ros-based environments. *Machines*, 7(2):42, Jun 2019.

[21] Thomas Linner, Alaguraj Shrikathiresan, Maxim Vetrenko, Bernhard Ellmann, and Thomas Bock. Modeling and operating robotic environments using gazebo/ros. In *proceedings of the 28th isarc*, Jun 2011.

[22] Bernardo Ronquillo Japon. *Hands-On ROS for Robotics Programming Program highly autonomous and AI-capable mobile robots powered by ROS*. Packt Publishing Ltd., UK, 2020. OCLC: 1241687096.

[23] Aridane Jesús Sarrionandia de León. Underwater mapping using sonar. B.S. thesis, Universidad de Las Palmas de Gran Canaria, 2015.

[24] Jordan Boehm, Eric Berkenpas, Bradley Henning, Michelle Rodriguez, Charles Shepard, and Alan Turchik. Characterization, modeling, and

simulation of an rov thruster using a six degree-of-freedom load cell. In *OCEANS 2018 MTS/IEEE Charleston*, page 1–7. IEEE, Oct 2018.

[25] Kristian Fotland. Analysis of rov thrusters and small marine propellers at specific rotational speeds. Master's thesis, University of Stavanger, Norway, 2018.

[26] Mohd Shahrieel Mohd Aras, Shahrum Shah Abdullah, Azhan Ab. Rahman, and Muhammad Azhar Abd Aziz. Thruster modelling for underwater vehicle using system identification method. *International Journal of Advanced Robotic Systems*, 10(5):252, May 2013.

[27] Sigrid Marie Mo. Development of a simulation platform for rov systems. Master's thesis, NTNU, 2015.

[28] R.N. Marshall, R.K. Jensen, and G.A. Wood. A general newtonian simulation of an n-segment open chain model. *Journal of Biomechanics*, 18(5):359–367, Jan 1985.

[29] Thomas Jakobsen. Advanced character physics. In *Game developers conference*, volume 3, pages 383–401. IO Interactive, Copenhagen Denmark, 2001.

[30] Ori Ganoni, Ramakrishnan Mukundan, and Richard Green. A generalized simulation framework for tethered remotely operated vehicles in realistic underwater environments. *Drones*, 3(1):1, Dec 2018.

[31] Gianluca Antonelli and G Antonelli. *Underwater robots*, volume 3. Springer, 2014.

[32] Junku Yuh and Michael West. Underwater robotics. *Advanced Robotics*, 15(5):609–639, 2001.

[33] Yuri M. A. Oliveira, Diogo A. Martins, Leizer Schnitman, and Marco A. Reis. Semi-autonomous uvms simulation: vehicle and manipulator pose control using artificial markers for localization. In *Proceedings of the III Brazilian Humanoid Robot Workshop (BRAHUR) and IV Brazilian Workshop on Service Robotics (BRASERO)*. Even3, 2020.

[34] Sachin Chitta, Ioan Sucan, and Steve Cousins. Moveit! [ros topics]. *IEEE Robotics and Automation Magazine*, 19(1):18–19, Mar 2012.

# Appendix A

# ROS Environment Set-up

## Prerequisites

I Install Ubuntu 18.04, ROS melodic.

II Open a new terminal and install uuv-simulator using command $sudoaptinstallros - melodic - uuv - simulator$

## To Build srov-master

I Load a new terminal and go to the source directory using the command $cdsrov - master/catkin\_ws$.

II Build the package using command $catkin_make$

III Source the setup file using command $source../devel/setup.bash$. This tells ROS where to look for packages. This step must be performed for each new terminal. To avoid having to do this, the bash script file /.bashrc can be edited so that this is performed in each new window. In addition, some Gazebo paths needs to point to the repository so that Gazebo is able to find models and plugins. To do this open the file with command $gedit\ /.bashrc$ and add the following lines to the end of the file

- $source\ /srov - master/catkin\_ws/devel/setup.bash$
- $export\ GAZEBO\_PLUGIN\_PATH = GAZEBO\_PLUGIN\_PATH$ : $/srov - master/catkin\_ws/src/srovm\_plugins/build/devel/lib$
- $export\ GAZEBO\_MODEL\_PATH = GAZEBO\_MODEL\_PATH$ : $/srov - master/catkin\_ws/src/srovm\_worlds/models$
- $export\ GAZEBO\_RESOURCE\_PATH = GAZEBO\_RESOURCE\_PATH$ : $/srov - master/catkin\_ws/src/srovm\_worlds$

## To Run the Simulator

I Launch a world in Gazebo using command *roslaunchsrovm_worldssnorre.launch*.

II Launch the ROV using command *roslaunchsrovm_rovucv.launch*

III To control the ROV with a keyboard run the command *roslaunch srovm_rov keyboard_control.launch*. Default setup is: $w/s$ : *surge*, $a/d$ : *sway*, $r/f$ : *heave*, $q/e$ : *yaw*.

IV To view output from camera use command *roslaunch srovm_cameras viewer.launchcamera* := *color_zoom*

V Launch camera processor with the command *roslaunch srovm_cameras viewer.launch camera* := *color_zoom*. This looks for circles and draws them on a new image that is subscribed to by the viewer

# Appendix B

# MATLAB Environment Set-up

## Prerequisites

I Install MATLAB R2021b.

II Install a C/C++ compiler supported by/compatible with Matlab which are listed below.

- In case of Windows, install the *_MinGW − w64_* compiler from the Add-Ons app.
- In case of Linux, install *_gcc_*.
- In case of OSX, install *_Xcode_*.

## To Build UUV-master

I Compile the C-coded S-functions located in the *uuv − master/models* directory for the rov thrust and dynamics as listed below.

- *mex rov_thrust.c*
- *mex uuv_dynamics.c*

## To Run the Simulator

After the C files has been compiled, the desired simulations can be run as stated below.

I First run the script *startup.m* to include the necessary files and directories to Matlab.

II Run the desired simulation file.

III Once finished with the simulation, run the script *cleanup.m* to clean up the temporary files and the Matlab path.