UNMANNED UNDERWATER VEHICLE SIMULATION FOR OBSTACLE AVOIDANCE AND MAPPING USING ROS & GAZEBO

A REPORT

submitted in partial fulfillment of the requirements

for the award of the dual degree of

Bachelor of Science-Master of Science

in

ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

by

MEHUL PAITHANE

(18142)

DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
INDIAN INSTITUTE OF SCIENCE EDUCATION AND RESEARCH BHOPAL
BHOPAL - 462066

April 2023

भारतीय विज्ञान शिक्षा एवं अनुसंधान संस्थान भोपाल
Indian Institute of Science Education and Research Bhopal
Department of Electrical Engineering and Computer Science
Bhopal By-pass Road, Bhauri, Bhopal 462066

CERTIFICATE

This is to certify that **Mehul Paithane**, BS-MS (Electrical Engineering and Computer Science), has worked on the project entitled **'Unmanned Underwater Vehicle Simulation for obstacle avoidance and mapping using ROS & Gazebo '** under my supervision and guidance. The content of this report is original and has not been submitted elsewhere for the award of any academic or professional degree.

April 2023 **Prof. Sujit PB**
IISER Bhopal

Committee Member	Signature	Date
Dr. Ankur Raina	_____	_____
Dr. Arijit Sen	_____	_____
Dr. Santanu Talukder	_____	_____

ACADEMIC INTEGRITY AND COPYRIGHT DISCLAIMER

I hereby declare that this project is my own work and, to the best of my knowledge, it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at IISER Bhopal or any other educational institution, except where due acknowledgement is made in the document.

I certify that all copyrighted material incorporated into this document is in compliance with the Indian Copyright Act (1957) and that I have received written permission from the copyright owners for my use of their work, which is beyond the scope of the law. I agree to indemnify and save harmless IISER Bhopal from any and all claims that may be asserted or that may arise from any copyright violation.

April 2023 **Mehul Paithane**
IISER Bhopal

ACKNOWLEDGEMENT

As I sit down to write this acknowledgement, I am filled with a sense of gratitude, humility and emotions. My journey has been long and challenging, but it would not have been possible without the help and support of so many people. To all those who have been a part of my life and supported me. I want to express my heartfelt gratitude.

First and foremost, I would like to thank my parents, **Mr. Makarand Paithane** and **Mrs. Jayshree Paithane** for their constant support. their constant love, guidance, and support have been the pillars of my success.

I would like to thank my supervisor, **Dr. P.B. Sujit** for his invaluable guidance, mentor-ship, input and encouragement throughout my final year. His expertise and patience to my research project has been instrumental in shaping my thesis.

To my dear friends, "Oops Proxy" and "The Boys" - **Gango, Antara, Anuja, Dhawal, Hitaishi, Parvathy, Sonam, Aditya, Agastya, Ayyappan, Sachin and Sutti**. I want to thank you for your support and companionship throughout these years. You have stood by me through thick and thin, and I feel blessed to have you all in my life. From impromptu trips at nikki and Pawan to planned picnics, from celebrations of small things to consolations of having a bad day at Amul, we have shared it all together. Thank you for all the memories and laughs. You will all be very missed.

There are a few more people who I would like to acknowledge as they have grown closer to me and have become great friends - **Kasi Viswanath, Deependra Singh Sinsinwar, Aman Kumar, Dishant Digdarshi**. from gym buddies to work colleagues. Thanks for all the support.

In closing, I would like to express my deepest gratitude to all those who have helped me along the way. Your kindness, support, and encouragement have meant the world to me, and I will be forever grateful.

Special thanks to **S Gangothri, Virat Kohli and Cricket.**

Mehul Paithane

ABSTRACT

This thesis focuses mainly on the implementation and analysis of the control and navigation algorithms used for an unmanned underwater vehicle (UUV) named Rexrov2 in windows sub-system linux 2 (WSL2) using robot operating system (ROS - melodic) and gazebo. The UUV is equipped with different kinds of sensors and cameras for mapping the topography of the ocean floor along with obstacle avoidance. This vehicle can be controlled using a joystick for manual operation or autonomously navigate along predefined way-points that have been generated through linear, cubic and Helical curve trajectories. The performance of the UUV is evaluated based on the accuracy of following way-points seamlessly, the effectiveness of obstacle avoidance algorithms and the precision of trajectory tracking during manual and autonomous control modes. Our result indicate that the UUV is capable of effectively mapping the underwater terrain. Also, it demonstrated high levels of accuracy in trajectory tracking. Overall, this thesis presents a comprehensive study of the capabilities of Rexrov2, which can be applied to a range of underwater exploration and surveillance applications.

LIST OF SYMBOLS OR ABBREVIATIONS

ROS	Robot Operating System
UUV	Unmanned Underwater Vehicle
SWARMs	Smart Networking Underwater Robots in Cooperation Meshes
AUV	Autonomous Underwater Vehicle
API	Application Programming Interface
URDF	Unified Robotics Description Format
SDF	Spatial Data File
DOF	Degrees of Freedom
STL	Standard Tesselation Language

LIST OF FIGURES

Contents

Chapter 1

Introduction

1.1 Problem Statement

Development of an UUV named RexROV2 using gazebo and ROS. The simulator must be able to perform the following tasks:

- Manual joystick control of the vehicle.
- Pre-defined way-points following of different kinds: 1). Helical curves 2). Cubic interpolator 3). Linear interpolator with polynimial blends.
- Bathymetric mapping of the ocean floor.
- Implementing obstacle avoidance and path planning algorithms.

1.2 Basic Introduction

UUVs have become increasingly popular in the recent years as they provide us with a safe and efficient way of exploring and studying the vast ocean depths. One of the most important aspects of designing a real life UUV is to ensure that it is able to withstand the harsh conditions of the underwater environment. [5] The development of said (UUVs) for development and testing different algorithms along with various configurations of sensors requires a stable and reliable simulation environment. This environment should be

able to give us reproducible results and a somewhat accurate depiction of how it would be to field test the vehicle in real life.

The hardware for such a vehicle is expensive and needs to be tested properly. Creating a simulation for UUVs and AUVs is more difficult than that of their aerial and ground counterparts. This is due to the fact that realistically modelling the hydrodynamic forces along with the complexity of the environment in which the vehicle will be deployed is much more difficult. A number of assumption are made before the simulation work starts, namely:

- The vehicle itself behaves as a rigid body.
- As compared to the various acceleration components of the vehicle, earth's rotation is negligible.
- The primary forces acting on the vehicle are inertial and gravitational like **hydrostatic** and **hydrodynamic**. [6]

As stated in the problem statement, to perform these simulations, we are gonna use Robot Operating System (ROS) - Melodic along with Gazebo.

While taking a closer look at the basics of modelling a UUV, we shall use Rexrov2 as an example and perform our simulations like different types of way-point generation. Rexrov2 is specifically designed for deep-sea exploration and research. We can also use other UUVs for the given task.

1.3 Fossen's Equations of Motion & Frames of References

The dynamics of an underwater vehicle is similar to that of an ariel vehicle, the main forces involved are as follows [7]:

- Thrust: Thrust is the force generated by the vehicle's propulsion system - Thrusters and fins, which helps to propels the vehicle through the water.
- Lift: This is basically a hydrodynamic force, that acts perpendicularly to the motion of the vehicle.

- Drag: This is a hydrodynamic force, that acts parallely to the motion of the vehicle - like friction.

- Buoyant force: Buoyancy is the upward force exerted on an object immersed in fluid, which in this case is water. The buoyant force acting on the vehicle is equal to the weight of the water displaced by the vehicle.

The dynamics of AUVs and UUVs is guided by Fossen's equations of motion. The equations are based on the principle of classical mechanics and they describe the motion of any underwater vehicle in 6 degrees of freedom (DOF). The list of these DOFs along with their corrosponding equation of motion are as follows:

- Surge: Linear movement in the backward or forward position (x-axis):

$$m\ddot{x} - m\dot{y}\dot{\psi} + m\dot{z}\dot{\theta} = X$$

- Sway: Linear movement in the side-to-side direction (y-axis):

$$m\ddot{y} + m\dot{x}\dot{\psi} - \dot{z}\dot{\phi} = Y$$

- Heave: Linear movement in the up-down direction (z-axis):

$$m\ddot{z} + mg = Z$$

- Roll: Rotation of vessel around it's longitudinal axis (x-axis):

$$I_{xx}\ddot{\phi} + (I_{zz} - I_{yy})\dot{\theta}\dot{\psi} + I_{xz}(\ddot{\theta}\cos\psi + \dot{\theta}\dot{\psi}\sin\psi) = \tau_{\phi}$$

- Pitch: Rotation of vessel around it's transverse axis (y-axis):

$$I_{yy}\ddot{\theta} + (I_{xx} - I_{zz})\dot{\phi}\dot{\psi} + I_{xz}(\ddot{\phi}\cos\psi - dot\phi\dot{\psi}\sin\psi) = \tau_{\theta}$$

- Yaw: Rotation of vessel around the vertical axis (z-axis):

$$I_{zz}\ddot{\psi} + (I_{yy} - I_{xx})\dot{\theta}\dot{\phi} + I_{xz}(\ddot{\theta}\sin\psi + \dot{\theta}\dot{\psi}\cos\psi) = \tau_{\psi}$$

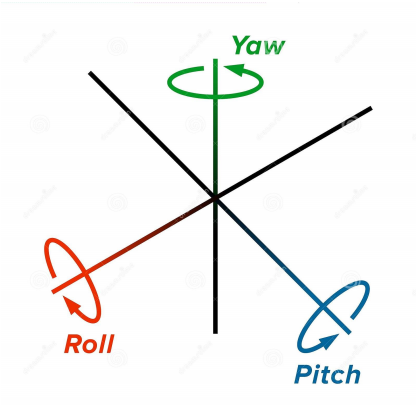


Figure 1.1: Representation of the 6 DOF motions [1]

In case of AUVs and UUVs, there are several different frames of references that we can use to define it's motion in 3D space. Some of the most commonly used frames of references are:

- Earth-fixed frame or NED frame: This is a fixed frame of reference and does not move as the vehicle moves. It is typically known as the North-East-Down (NED) frame. The x-axis pointing north, y-axis pointing east and the z-axis pointing down. The figure below depicts NED frame of reference:

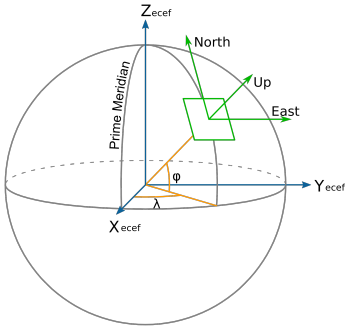


Figure 1.2: NED frame of reference [8]

- Body-fixed frame: This frame of reference is fixed to the vehicle and moves as the vehicle moves. The x-axis of this frame aligns with the longitudinal axis of the vehicle, y-axis with the transverse axis and the z-axis with the vertical one.

- Local-level frame: This frame is similar to that of the earth-fixed frame but it is rotated to align with the vehicles' orientation. This frame is particularly used for navigation applications.

1.4 Modelling of UUV

1.4.1 About Rexrov2

Rexrov2 is an Unmanned Underwater Vehicle (UUV) designed for underwater exploration and research. The Rexrov2 package is a part of UUV simulator and is a versatile platform that can be used for a wide range of applications, such as oceanographic research, marine biology, underwater inspection, and underwater surveys. It is capable of operating at depths of up to three thousand meters and has a maximum speed of four knots. The UUV is equipped with a range of sensors, cameras, and other instruments that allow it to collect data and images of the underwater environment. It can be operated remotely from a surface vessel or from a shore-based control center, and it can be programmed to operate autonomously for extended periods of time.

The Rexrov2 UUV is powered by a high-capacity lithium-ion battery pack, which provides it with a long operating time. It is equipped with a sophisticated propulsion system that enables it to maneuver in all directions, including vertically. The UUV also has a modular design, which allows it to be easily customized for specific applications.

The different types of sensors present in rexrov2 are:

1. Cameras: Cameras are used to provide with video feed to the operator and observe the environment it is in.

2. Sonar: This sensor is used to detect underwater objects and obstacles, this sensor is used in cases where the water is translucent and murky. A Lidar can also be used instead of sonar.

3. Depth sensor: Depth sensors are used to calculate the depth of the water.

4. Temperature and Pressure sensors: These sensors are used to monitor the surroundings of the UUV.

5. Compass: The UUV is also equipped with compass that gives the operator the heading of the vehicle.

We shall use the already developed rexrov2 model to run the simulations.

1.4.2 Actuators

Actuators are devices that convert different kinds of energy from electrical signals or a physical force to mechanical motion. Some common examples of actuators are electric motor, hydraulic cylinders. In case of UUVs, we have namely 2 actuators:

Thrusters

This is a propulsion system present in the rexrov2, that helps the UUV move underwater. The rexrov2 is equipped with a total number of 4 thruster units - 2 located in the vertical and 2 in horizontal plane. The structure of a thruster looks like:

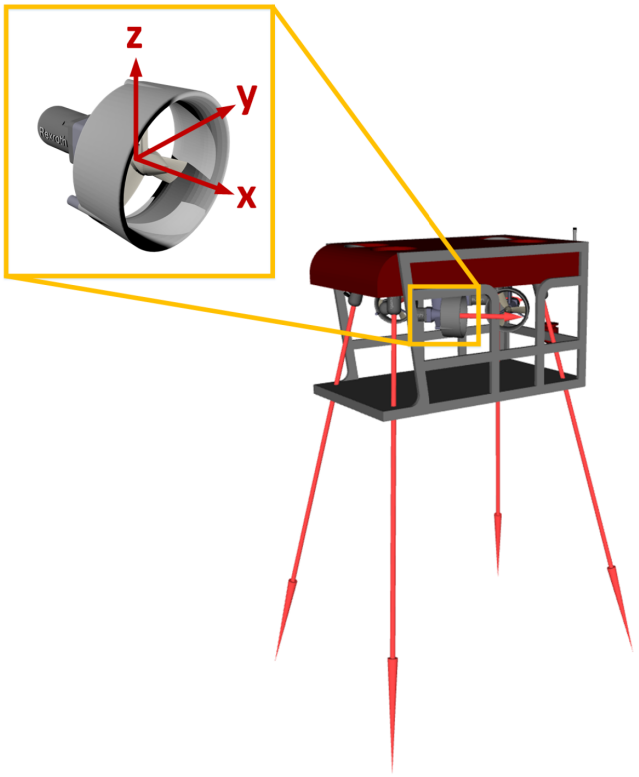


Figure 1.3: Rexrov2 - thrusters [6]

The thruster units are mainly compromised of two modules: a dynamic model that describes the dynamics of the the thruster's rotors and the conversion function, which describes the steady-state relationship between the rotor's angular velocity and the thruster force output.

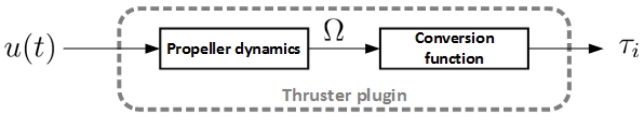


Figure 1.4: Thruster plugin model [6]

The different kinds of **dynamic models** are as follows:

1. Zero - order dynamic model: The zero-order dynamic model is a simple mathematical model that describes the relationship between the input and the resulting output. The basic form of the zero-order dynamic model is:

$$Output = K * Input$$

where k is some constant that represents the gain of the actuators.

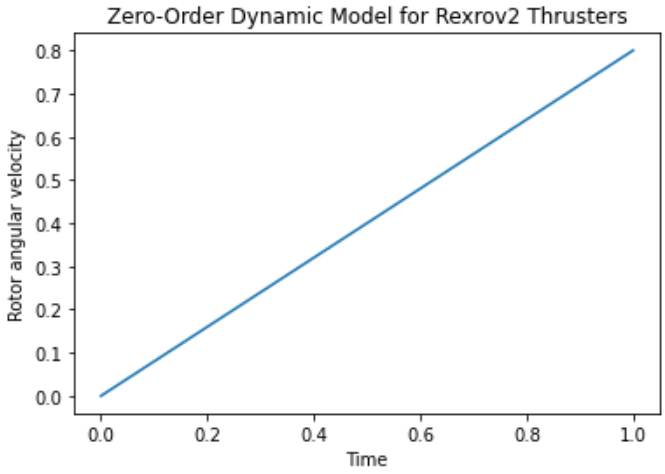


Figure 1.5: zero order dynamic model for thrusters

2. First - order dynamic model: The first-order model is a more complex mathematical model that takes into account the internal dynamics of an actuator, as well as any delays in the actuator's response to an input signal. The basic form of the first-order dynamic model is:

$$output = K * (1 - e^{-t/\tau}) * Input$$

K = Gain of the actuators
τ = Time constant.

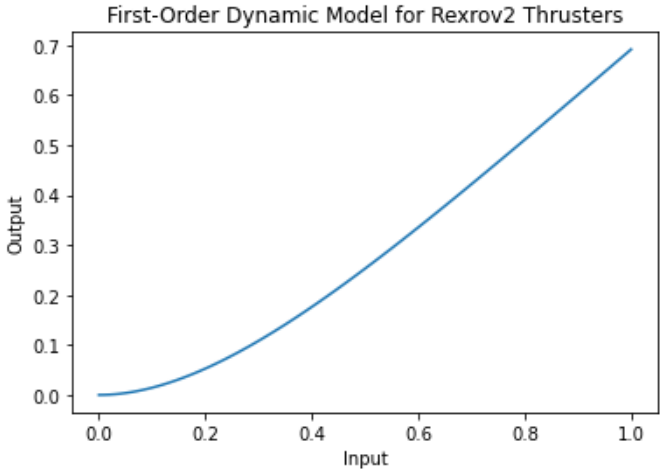


Figure 1.6: First order dynamic model for thrusters

3. Yoerger dynamic model: This is a more advanced mathematical model that takes into account the internal dynamics of the actuator alongside the dynamics of the vehicle itself. The basic form of the yoerger dynamic model is:

$$\frac{d^2x(t)}{dt^2} + 2\zeta w_n \frac{dx(t)}{dt} + w_n^2 x(t) = K_p u(t)$$

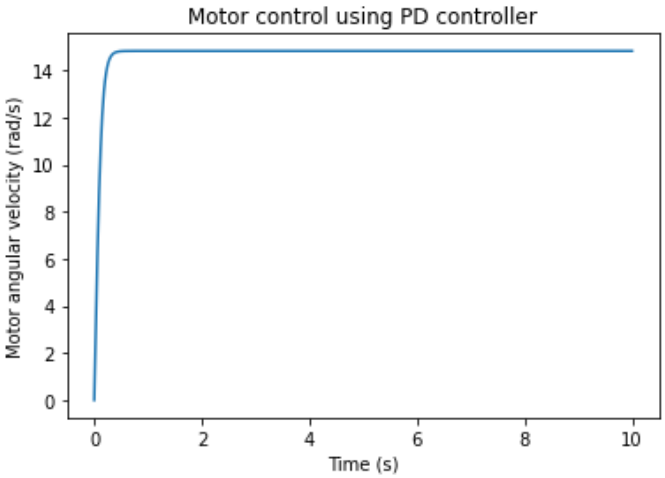


Figure 1.7: Yoerger dynamic model for thrusters

4. Bessa dynamic model: This is a second-order linear time-variant differential equation that takes into account the effects of damping and inertia on the motion of the vehicle. The basic form of this model is:

$$m\frac{d^2x(t)}{dt^2} + c\frac{dx(t)}{dt} + kx(t) = K_p u(t)$$

m = Mass of the vehicle
c = Damping coefficient
k = Stiffness of the vehicle
$u(t)$ = Input signal to the thruster
$x(t)$ = Position of the vehicle at time t.

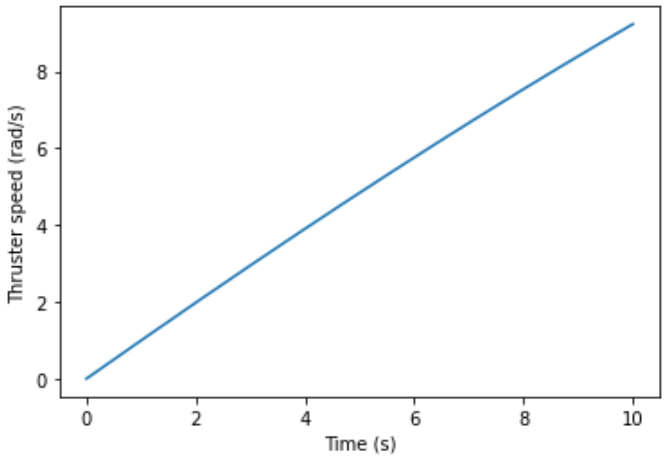


Figure 1.8: Bessa dynamic model for thrusters

The different kinds of **conversion function** are as follows:

1. Basic conversion function: The basic conversion function is a simple linear function that maps the input signal to the thruster to the output signal of the thruster. It is used to convert the input signal, which is typically a voltage or current signal, into the appropriate output signal that will drive the thruster. The conversion function is often modeled as a linear equation:

$$y(t) = K * u(t) + b$$

$u(t)$ = Input signal to the thruster
$y(t)$ = Output signal of the thruster
K = Conversion constant that determines the slope of the linear function.
b = Conversion constant that determines the intercept of the linear function.

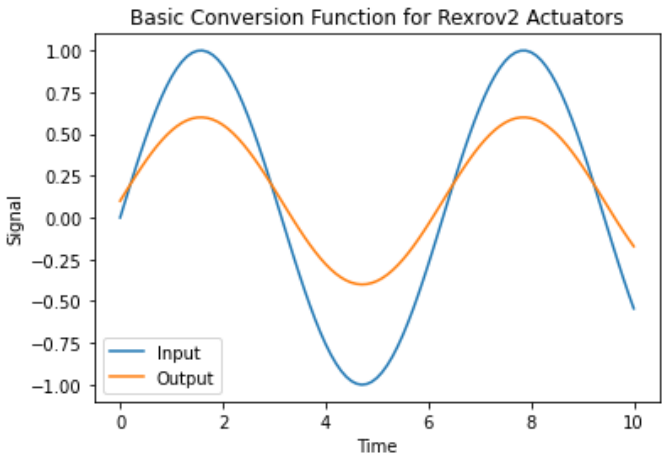


Figure 1.9: 'Basic' conversion function for thrusters

2. Dead_zone conversion function: The dead zone conversion function introduces a dead zone around the zero input signal. The dead zone is a range of input signals where the thruster does not respond, effectively creating a dead zone in the output signal. This function is often used to prevent small fluctuations in the input signal from causing unnecessary thruster movements. The dead zone conversion function can be modeled as follows:

$$y(t) = K * (u(t) - d) * (|u(t)| > d) + b$$

$u(t)$ = Input signal to the thruster
$y(t)$ = Output signal of the thruster
K = b are the conversion constants that determine the slope and intercept of the linear function. d is the dead zone parameter, which

determines the range of input signals where the thruster does not respond.

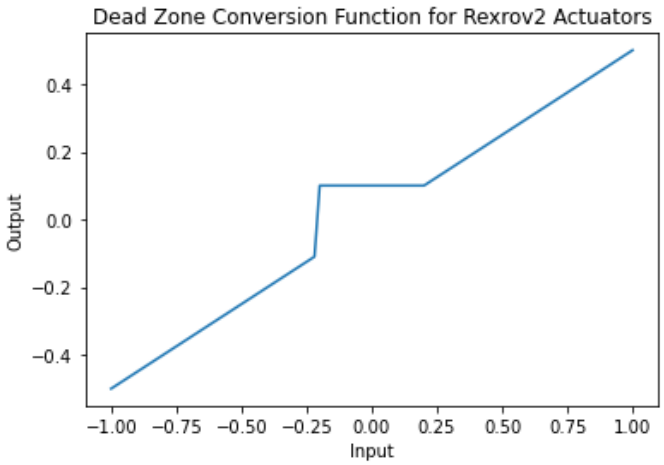


Figure 1.10: 'dead zone' conversion function for thrusters

3. Linear_interpolator conversion function: The linear interpolation conversion function is a function that maps the input signal to the output signal using a linear interpolation between two points. This function is often used when the output signal of the thruster needs to change smoothly with the input signal. The linear interpolation conversion function can be modeled as follows:

$$y(t) = K * (u(t) - x1) * (x2 - u(t))/(x2 - x1) + b$$

$u(t)$ = Input signal to the thruster
$y(t)$ = Output signal of the thruster
K = Conversion constant that determines the slope of the linear function.
b = Conversion constant that determines the intercept of the linear function.
$x1$ & $x2$ = Two points that define the range of the input signal, and the output signal is computed by interpolating between the two points.

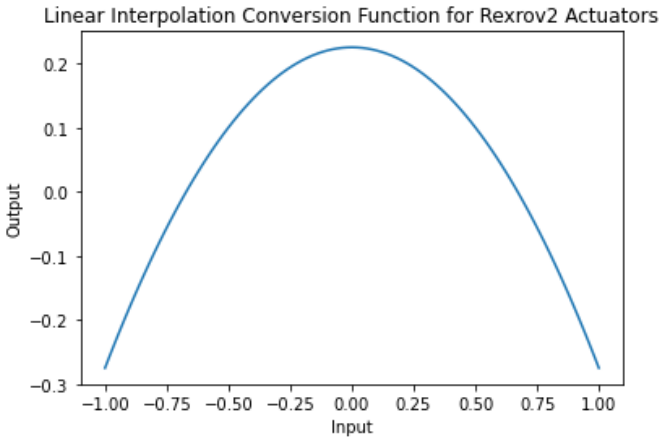


Figure 1.11: 'Linear_inter' conversion function for thrusters

Fins

Fins are built very similar to that of thrusters, a dynamic model to describe the fin's revolute joint's state and a lift and drag model that models the lift and drag forces generated at the fin's surface taking into account the fin's relative velocity with respect to the current velocity. There are basically 2 types of lift and drag models:

1. Quadratic: The quadratic lift model is a mathematical model used to calculate the lift and drag forces generated by a hydrofoil, such as a fin on a UUV like the Rexrov2. This model takes into account the angle of attack, the speed of the UUV, and the properties of the fluid (water) surrounding the hydrofoil. The drag force (D) generated by the hydrofoil can be calculated using a similar equation:

$$D = 1/2 * \rho * V^2 * A * Cd$$

ρ = The density of the fluid
V = Speed of the vehicle relative to the fluid.
Cd = the lift coefficient, which depends on the angle of attack of the hydrofoil.
A = the area of the hydrofoil.

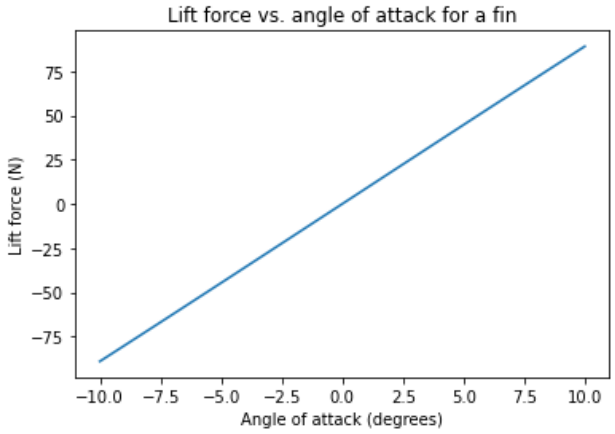


Figure 1.12: 'Quadratic' lift model for fins

2. Two_lines: In this model, the lift and drag coefficients are represented by two separate lines: a linear part at small angles of attack and a nonlinear part at larger angles of attack. The linear part is based on the concept of thin airfoil theory and assumes that the lift and drag coefficients are directly proportional to the angle of attack. The nonlinear part accounts for the stall effect, where the lift coefficient reaches a maximum value and then drops sharply as the angle of attack increases further.
   The lift force (L) generated by a hydrofoil using the Two-Lines Model can be calculated using the following equation:

$$L = 1/2 * \rho * V^2 * A * (Cl1 + (Cl2 - Cl1) * (\alpha - \alpha1)/(\alpha2 - \alpha1)) * F$$

   The drag force (D) generated by the hydrofoil using the Two-Lines Model can be calculated using a similar equation:

$$D = 1/2 * \rho * V^2 * A * (Cd1 + (Cd2 - Cd1) * \alpha - \alpha1)/(\alpha2 - \alpha1)) * F$$

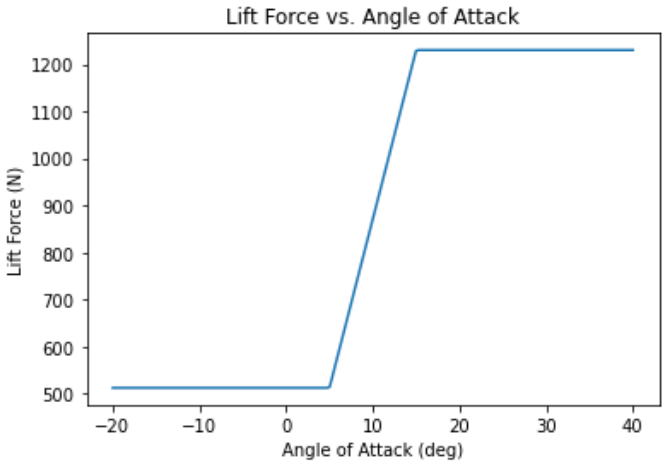


Figure 1.13: 'Two_line' lift and drag model for fins

1.4.3 Gazebo worlds

These are pre-built underwater world scenerios with in-built ODE physics engines that enable us to spawn multiple AUVs. The different kinds of gazebo worlds are:

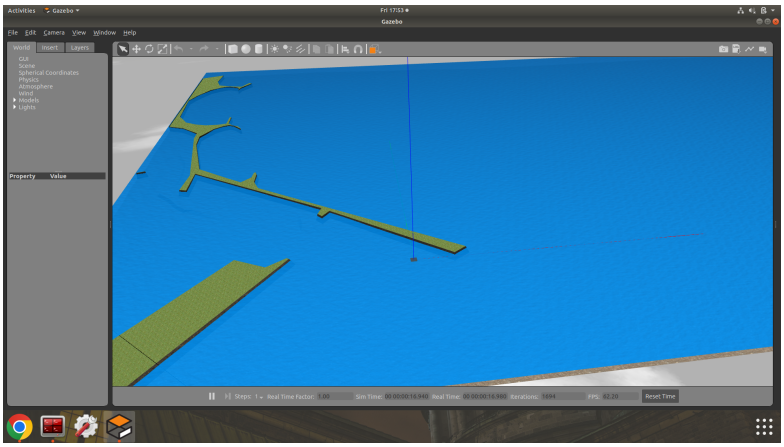


Figure 1.14: Gazebo world: coast of Magnalia, Romania

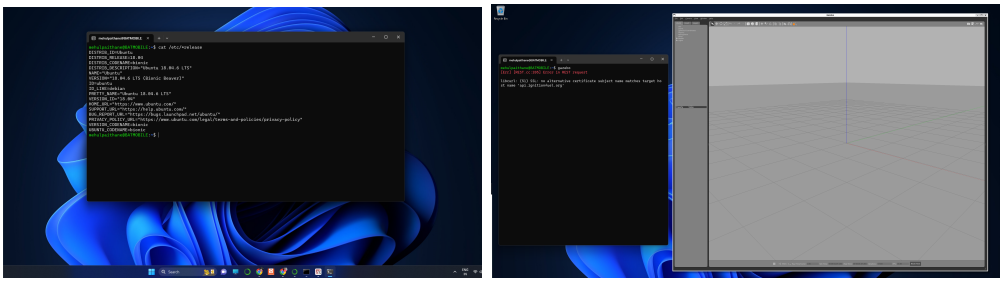
Chapter 2

Software Description

2.1 Tools used

2.1.1 Windows Sub-system Linux (WSL2)

Windoes subsystem Linux (WSL2) is a feature of windows 10 & 11, it allows users to run a linux environment (Ubuntu 18.04.5 LTS) directly on any windows machine.[9] The simulation softwares like ROS and gazebo are widely compatible with the linux interface, therefore we shall use WSL2 for any simulation wotk on these softwares alongside some third-part applications used for GUI.



(a) WSL2 (b) Gazebo on WSL2

Figure 2.1: Windows Subsystem Linux 2

2.1.2 Robot Operating System (ROS) & Gazebo

Due to the problems of simulating AUVs and UUVs, there aren't many open-source simulation software that can accurately simulate the physics and complex environments accurately. **MATLAB** and **Gazebo** are mostly used for simulation work. Gazebo is more often prefered on the account of it being open-source.

ROS [2] is an open-source framework used for building robotic software. For the simulation work of UUVs, we shall use ROS-melodic as it provides a range of libraries, tools and applications to help developers create robust and scalable robotic systems that can later be used in real-life. The other reason behind using ROS-melodic was bacause of the uuv-simulator package already present in it. This version of ROS is most compatible with Ubuntu 18.04.5 LTS.

Some advantages of using ROS over other open-source simulation frameworks are:

1. The modular architecture of ROS, allows users to add and remove components easily.

2. It supports multiple programming languages - python and C++.

3. for the simulation of UUVs, there is a pre-developed package named UUV simulator, alongside different uuvs for further development.

Gazebo [3] is a 3-D simulation environment that allows developers to create and test robotic systems in a virtual environment. It provides a range of tools and libraries for simulating robotic systems, including sensors, actuators, and environmental models.

The combination of ROS and gazebo together provides a range of tools for creating and testing robotic systems. This integration allows developers to easily create and test complex robotic systems in a virtual environment. Overall, ROS Melodic and Gazebo are powerful and flexible tools for developing and testing robotic systems, and by using them together, developers can take advantage of the strengths of both systems.

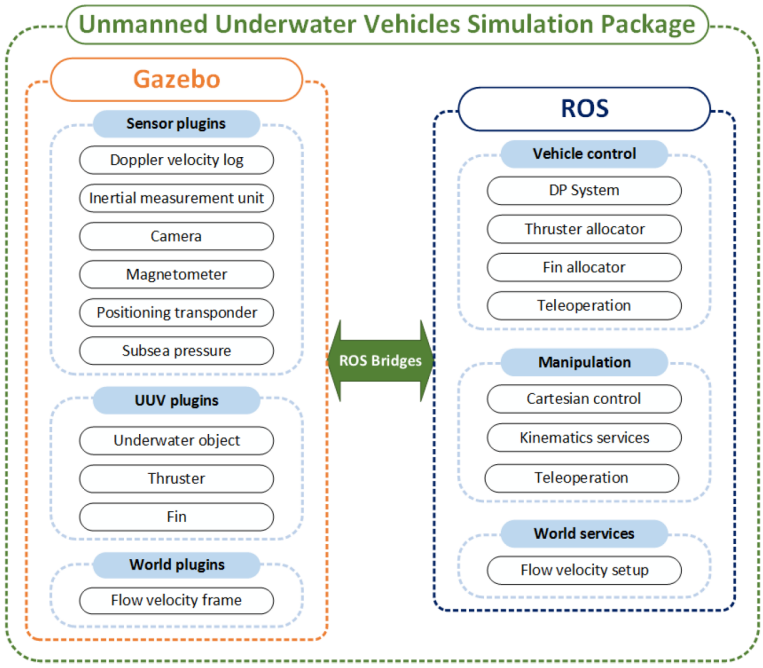


Figure 2.2: Software structure for UUV simulator [5]

2.1.3 Rviz

Rviz is a 3-D visualization tool designed for use with ROS. It allows users to visualize and interact with various aspects of a robotic system, including sensor data, robot models, and environment maps. RViz is an essential tool for robot developers and engineers, as it allows them to visualize and test their systems in a virtual environment before deploying them in the real world. One of the main uses of RViz is to visualize sensor data from various sensors, including cameras, lidars, and range finders. RViz allows users to visualize this data in real-time, which can help them understand how their sensors are working and identify any issues or areas for improvement. RViz also allows users to visualize robot models and animations, which can help them understand how their robots are moving and interacting with their environment.

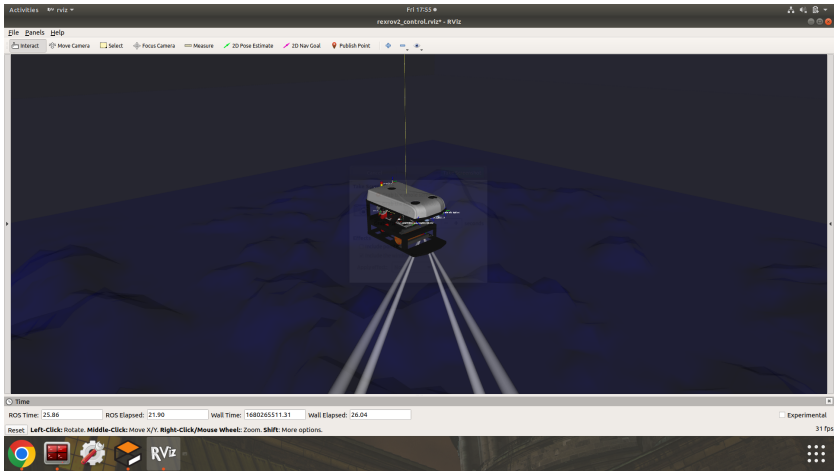


Figure 2.3: Visualizing rexrov2 in rviz

2.1.4 Blender

Blender is a 3-D creation software that has a wide variety of applications, we use this software to create and edit STL files of ocean floors that will later be used for bathymetric mapping.

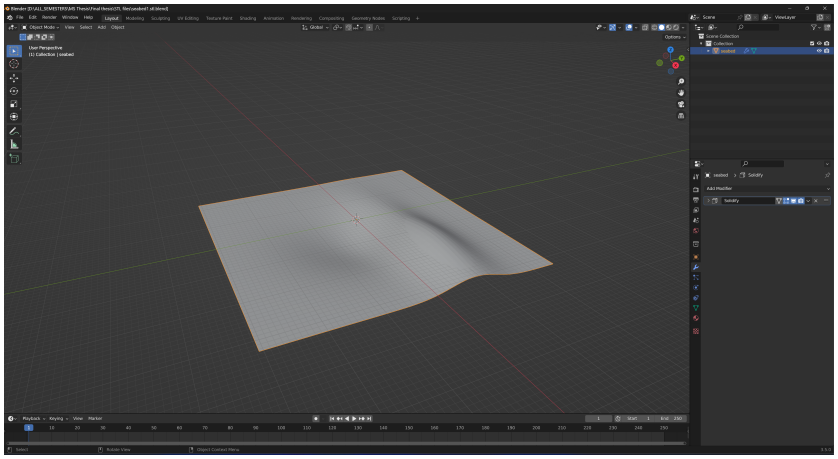


Figure 2.4: Ocean floor - STL file in blender

Chapter 3

Results

3.1 Manual control using joystick

Before starting autonomous driving of the UUV, we should be able to control the same vehicle manually using a joystick or keyboard.

3.1.1 Controlling a turtlesim simulation

Turtlesim is a popular open-source simulation environment that allows users to simulate the movement and behavior of a turtle. The turtle can be controlled using a joystick, and the movements of the turtle are displayed in real-time on the screen.

The algorithm behind Turtlesim joystick control is relatively simple. The user can move the joystick in different directions to control the movement of the turtle. The joystick sends a signal to the computer, which then translates the signal into a specific movement command for the turtle. here is what the outcome looks like:

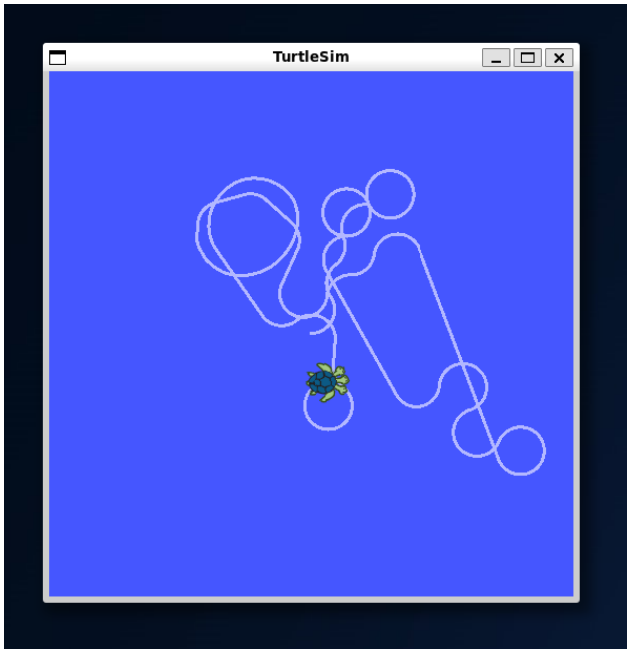


Figure 3.1: Controlling turtlesim using joystick

3.1.2 Controlling Rexrov2 simulation

The algorithm behind controlling rexrov2 manually is same as that of the turtlesim node. The result is as follows:

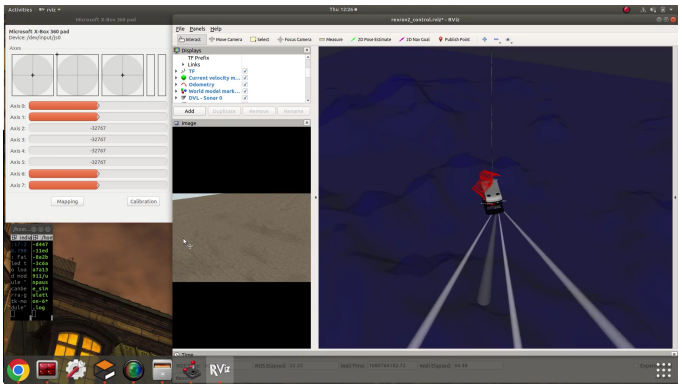


Figure 3.2: Controlling rexrov2 using joystick

The red arrows in the visualization shows the odometry of the rexrov2 vehicle.

3.2 Path and trajectory generators

3.2.1 Helical curves

A helical waypoint path used for a UUV, is a pre-defined trajectory composed of a series of waypoints forming a helix shape in 3-D space.

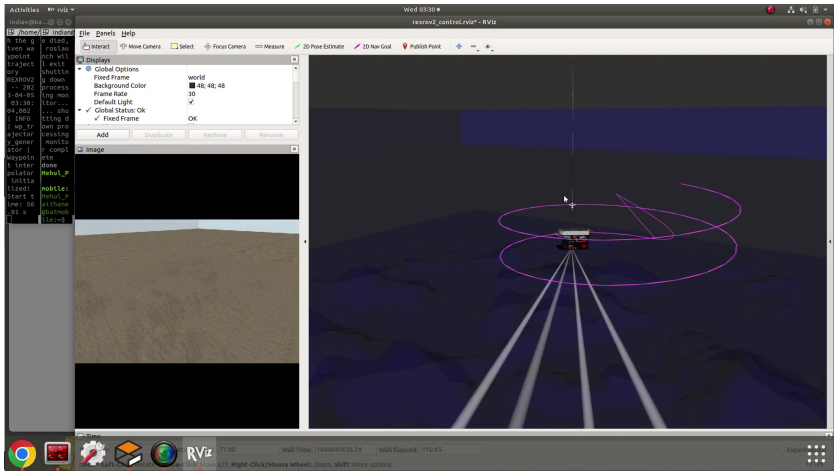


Figure 3.3: Helical curve for rexrov2

3.2.2 Cubic interpolator

Cubic interpolators are another way to generate smooth paths or trajectories for a UUV like the Rexrov2. A cubic interpolator generates a smooth curve that passes through a set of control points, which are defined by the user. The algorithm for generating a cubic interpolator involves a few key steps:

1. Defining the control points
2. Calculating the coefficients
3. Generate points on the curve

The following figure shows the actual rexrov2 simulation for cubic interpolator:

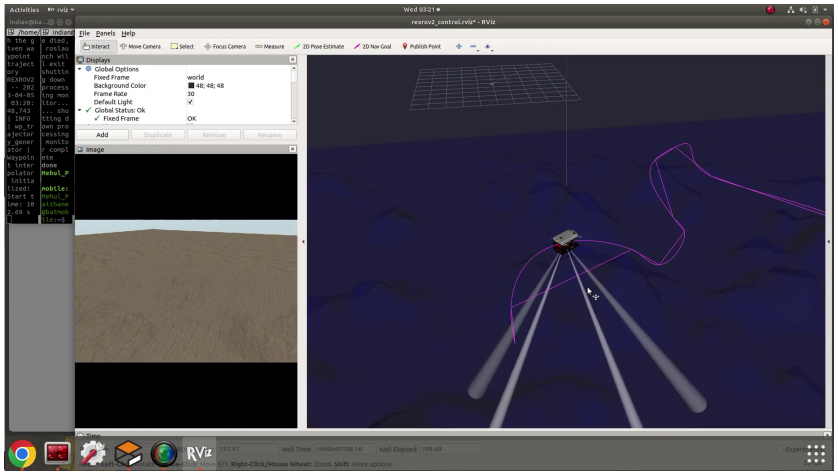


Figure 3.4: Cubic interpolator waypoint path for rexrov2

3.2.3 Linear interpolator with polynomial blends

Linear interpolator with polynomial blends is a path generation technique. This technique divides the path between the start and end points into series of smaller segments. Each of these segments is connected smoothly using polynomial blends. A polynomial blend is a mathematical function that smoothly connects two adjacent polynomial curves or segments.

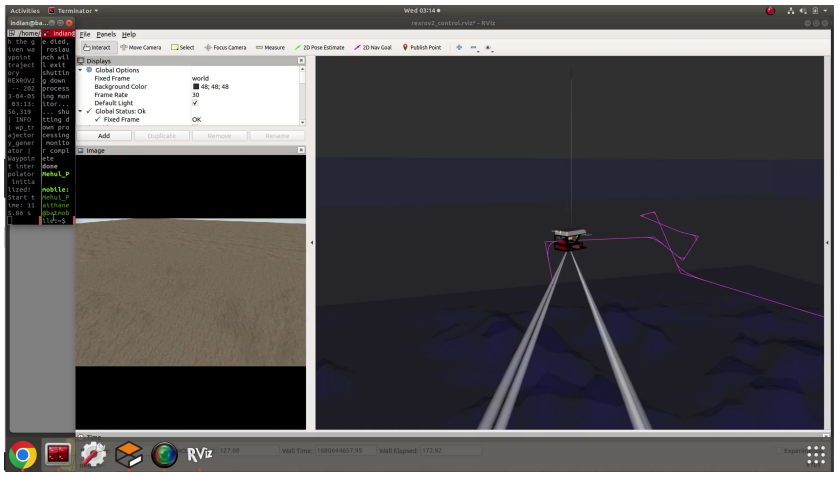
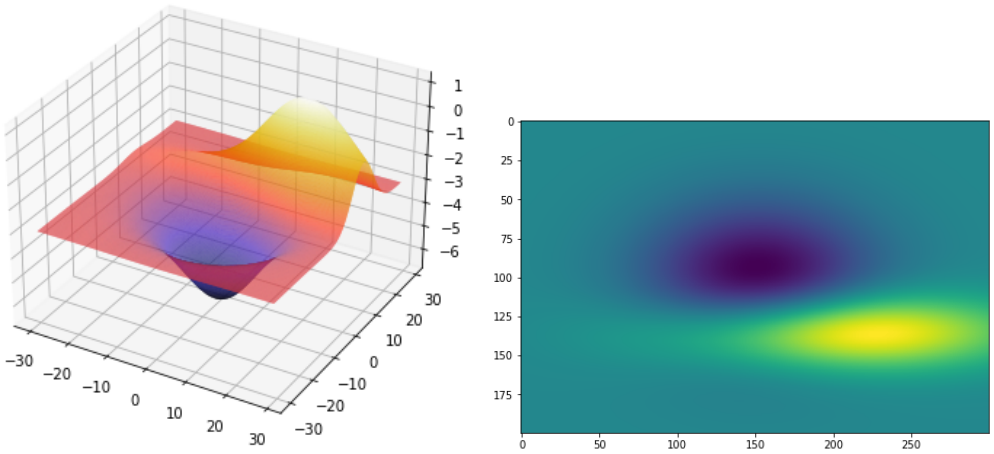


Figure 3.5: linear interpolator with polynmial blend in rexrov2

3.3 Bathymetric mapping of ocean floor

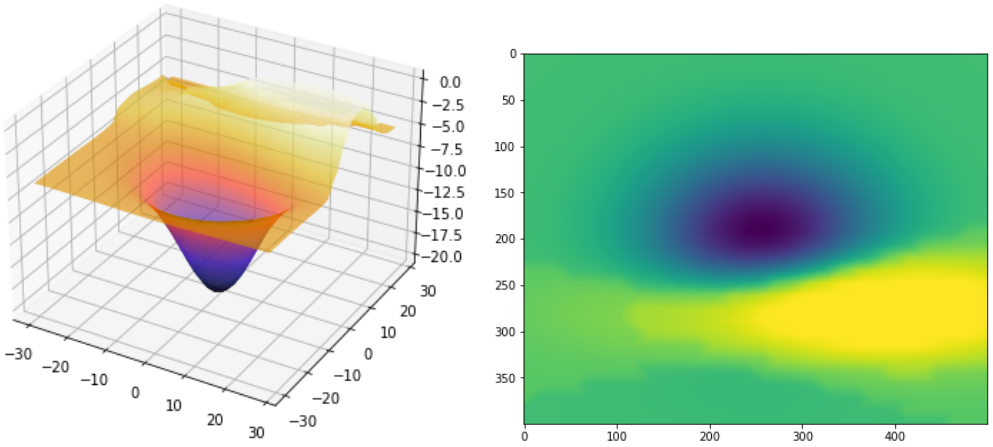
Bathymetric mapping refers to the process of creating a map of the ocean floor's topography, including its depth and terrain. The Rexrov2 is equipped with various sensors, including sonars and cameras, that are used to collect data about the ocean floor. The sonars emit sound waves that bounce off the ocean floor and return to the sensor, allowing the Rexrov2 to measure the depth of the ocean at various points. The camera provides visual data that can be used to create detailed maps of the terrain.
To create a bathymetric map of the ocean floor using the Rexrov2 in Gazebo and Rviz, the following steps can be followed:

1. Creating a new world with a custom seabed: we have created 2 different custom seabed for this process, the images below show the heightmap and the 3-D plot of the ocean floor



(a) 3-D plot of ocean floor 1 (b) Heightmap of ocean floor 1

Figure 3.6: Ocean floor 1



(a) 3-D plot of ocean floor 2 (b) Heightmap of ocean floor 2

Figure 3.7: Ocean floor 2

2. Configure the Rexrov2 robot in Gazebo: This involves setting up the robot's physical properties such as its mass, dimensions, and sensors. The robot can be placed in a virtual ocean environment to simulate real-world conditions.

3. Develop a control algorithm for the Rexrov2: This involves programming the robot to move in a predefined pattern, such as a zigzag or circular path, while collecting data from its sensors. The algorithm should also account for the robot's movement in 3D space, including its pitch, roll, and yaw.

4. Visualize the robot and its sensors in Rviz: This step involves setting up Rviz to display the robot and its sensor data in real-time. The sonar data can be visualized using a depth map or point cloud, while the camera data can be displayed as a 3D mesh or textured surface.

5. Collect data and generate a bathymetric map: Once the Rexrov2 is deployed in the virtual ocean environment, it can begin collecting data from its sensors as it moves along its predefined path. The collected data can be used to generate a bathymetric map of the ocean floor using software tools such as python.

3.4 Obstacle avoidance with path planning algorithms

This task is possible using all the sensors present in rexrov2 working in tandem.

To make this simulation possible we need to follow the following steps:

1. Make a gazebo world with different types of objects present at different poses. These objects will act as the obstacles that the rexrov2 has to maneuver through.

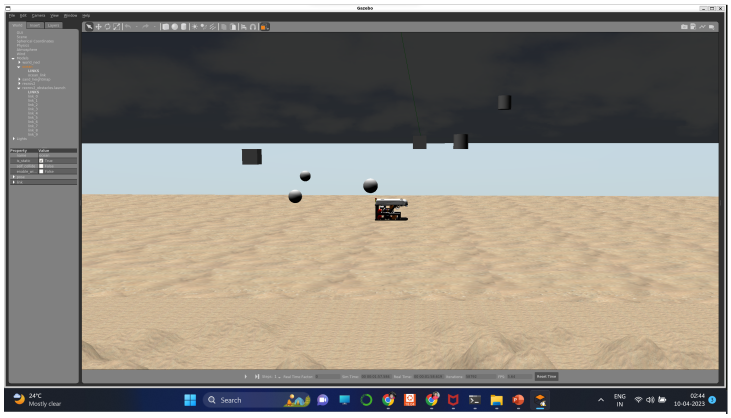


Figure 3.8: rexrov2 world with obstacles

2. Collection of sensor data: The rexrov2 is equpped with different types of sensors. Collec the data from either SONAR or LIDAR. Ghis data will be usefull later.

3. Creating a map of the surroundings. The data collected earlier will be used to create a environment of the rexrov2.

4. Path planning: As we now know the environment, we can use algorithms like A - star and Dijkstra.

5. Once the path has been planned, we will use object detection in case of dynamic obstacles. This is achieved using model predictive control.

Let us discuss more about the algorithms used in path planning:

3.4.1 Dijkstra algorithm

Dijkstra is a popular iterative algorithm that is used to calculate the shortest distance between any two nodes in a given graph. In our case, these two nodes are the starting point and the goal for rexrov2.
The basic idea behind Dijkstra is that it iteratively explores the edges of one node and choose the edge that has the least weight and continue this until goal is reached. Also, no 2 edges can be repeated.

3.4.2 A - star algorithm

This algorithm is an extension of Dijkstra, it uses a heuristic function that helps to find the shortest distance between 2 nodes. The end goal is to minimize the f-score, which is the sum of h-score and g-score. g-score is the cost of travelling from the start node to the current node and h-score is the estimated cost of the cheapest path from the current node to goal node. [4]

Chapter 4

Conclusions

4.1 Applications

Due to the hazardous nature of the underwater world, there are multiple uses of an unmanned underwater vehicle such as:

1. Oceanography: UUVs are used in the field of oceanography to collect data about the water temperature, salinity, and currents, as well as to map the seafloor.

2. Military: UUVs can be used to detect mines and other objects

3. Monitoring of Underwater environment: comparing water quality and tracking the corresponding marine life in that water.

4. Oil exploration and checks: UUVs can be used for checking the oil pipes being used for leaks and repairs. They can also be used to find potential drilling sites.

5. Underwater archaeology: These vehivles can be used to explore centuries old ship-wrecks and take images without disturbing the site.

4.2 Future work

The applications of UUVs are vast, but due to it being in the developmental stages, the future work is basic and has a lot of scope:

1. Increased range: Current UUVs have limited range, this restricts them to perform tasks that take large amount of time.

2. SWARM: These vehicles can be deployed in large numbers, they can work in tandem to perform laborious tasks like: mapping of a large segment of ocean's floor.

Appendices

I Basic Definitions

1. Thrust: Thrust is the force generated by the vehicle's propulsion system - Thrusters and fins, which helps to propels the vehicle through the water.

2. Lift: This is basically a hydrodynamic force, that acts perpendicularly to the motion of the vehicle.

3. Drag: This is a hydrodynamic force, that acts parallely to the motion of the vehicle - like friction.

4. Buoyant force: Buoyancy is the upward force exerted on an object immersed in fluid, which in this case is water. The buoyant force acting on the vehicle is equal to the weight of the water displaced by the vehicle.

5. Surge: Linear movement in the backward or forward position (x-axis).

6. Sway: Linear movement in the side-to-side direction (y-axis).

7. Heave: Linear movement in the up-down direction (z-axis).

8. Roll: Rotation of vessel around it's longitudinal axis (x-axis).

9. Pitch: Rotation of vessel around it's transverse axis (y-axis).

10. Yaw: Rotation of vessel around the vertical axis (z-axis).

Bibliography

[1] Denis Barbulat. Roll, pitch, yaw three rotation angles corresponding to euler angles royalty free svg, cliparts, vectors, and stock illustration. image 135478783., 2019.

[2] Amanda Dattalo. Wiki, 2018.

[3] Willow Garage. Gazebosim, 2004.

[4] Peter E Hart. Stanford artificial intelligence laboratory, 1966.

[5] 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.

[6] Musa Manhães, Sebastian Scherer, Martin Voss, Luiz Douat, and Thomas Rauschenbach. Uuv simulator: A gazebo-based package for underwater intervention and multi-robot simulation. 09 2016.

[7] J.W. Nicholson and A.J. Healey. The present state of autonomous underwater vehicle (auv) applications and technologies. *Marine Technology Society Journal*, 42:44–51, 03 2008.

[8] Torge Torge and Wolfgang Wolfgang. Local tangent plane coordinates, Feb 2023.

[9] Ubuntu Ubuntu. Ubuntu dual booting with windows 10, 2018.