Noetic Underwater Simulator (NUSim)

Akash Chaudhary

Compiled July 17, 2020

This document will highlight the tasks to be undertaken and steps to be followed to complete the first step towards building a complete Simulation for an Unmanned Underwater Vehicle (UUV) using Unity3D and Robot Operating System (ROS). In this version, we are going the build the environment in Unity3D and test it using keyboard controls without ROS integration.

1. INTRODUCTION

Simulations are a crucial part of the Design and Deployment process as it gives us an opportunity to test our software and algorithms before deploying them in the physical world. This not only ensures that the operators are well versed with the system that they are working with but also reduces the risk of deploying the product directly which can cost the user their hardware in case of failure. In this project, we are trying to build an Open Source Simulator to test the control system for Unmanned Underwater Vehicles (UUV). We will be using Unity3D, a physics engine to simulate the environment in which the UUV will be deployed, and Robot operating System (ROS) to build the control system for the same. This proposal deals with the first step towards that goal and aims to build the 3D environment in Unity3D with keyboard controls and no ROS Integration. This proposal highlights the steps and goals of the first version. These might change as we progress through the project, but this proposal can serve as a guiding base for the development process.

The Unity3D project will be uploaded to GitHub so that it can be accessed easily. The updated GitHub repository can be found here.

2. MISSION STATEMENT

The mission statement will highlight the tasks to be undertaken by the UUV after deployment from a remote Control Station.

1. Start from surface near the Control station.
2. Sink down to a predefined depth or predefined distance from the seabed.
3. Find and Follow the Pipeline.
4. Cross gate check points along the way.
5. Search for the leak in the pipeline.
6. Mark the coordinates of the leak and save its picture.
7. Automated return to the boat.

3. ASSETS AND ENVIRONMENTS

This section will list the major assets identified at this point of time to be used to build the simulation environment.

1. Boat control Station
2. Unmanned Underwater Vehicle
3. Water Container
4. Water
5. Gates/Checkpoints
6. Material, Textures and Effects

4. FORCES

The forces to be considered for the simulation are:

- Hydro static forces
- Lift
- Gravity
- Drag
- Thrust
- Buoyancy
- Collisions

Some of these forces will be taken care of by the Rigidbody plugin of Unity3D, and others will be defined by C# scripts.

5. SENSORS

The sensors to be deployed on the UUV are as follows:

- Two camera- Front facing and Downwards facing.
- Inertal Measurement Unit (IMU)
- Pressure Sensor
- Temperature Sensor
- Proximity Sensor
- Compass
- Navigation Lights

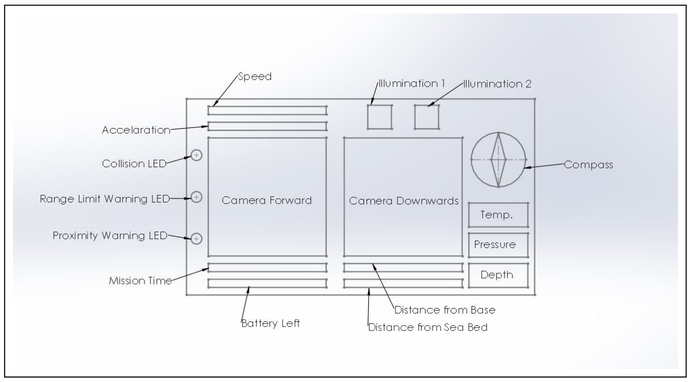

Fig. 1. User Interface for NUSim outlining the elements of the interface.

6. USER INTERFACE

The User Interface (UI) allows the operator to monitor the state of the vehicle and help them to control the UUV. A basic layout of the UI was prepared as shown in *Figure 1*.

7. SCRIPTS

Although the majority of the scipts will be made during the course of development as and when required, some of the scripts were identified as necessary keeping in mind the mission statement. Those scripts are listed below.

1. Control
2. Automate Sink
3. Automate Return to Station
4. Water and Underwater Effects
5. Triangle mesh data
6. Underwater Mesh Identifier
7. Calculations for Underwater Mesh
8. Water Controller
9. Teleoperation
10. Coordinates Calculator
11. Coordinates Marking
12. Surface
13. Video Capture
14. Collision
15. Buoyant Force
16. Sensor data
17. Relative Distance
18. Speed and Acceleration
19. Range Limit
20. Proximity Warning
21. Compass
22. Illumination
23. Mission Time and Battery left

and various other general and UI scripts which will be identified during the course of development.

8. AIM FOR VERSION 2

Some features that I don't intend to include in the first iteration of the simulation but want to include in the second iteration:

- Camera Movement and Zoom Controls
- Send Sensor data to ROS and save via rosbag
- Simple teleoperation commands using ROS

  This list will be updated when I will encounter problems during the development process and when new ideas might cross my mind.