Autonomous Underwater Vehicle Simulation

Haroon Muhammed

May 1, 2025

Abstract

Autonomous Underwater Vehicle Simulation is a research project aiming to develop a comprehensive modern underwater drone simulator using ROS 2 (Robot Operating System) and ignition gazebo. The simulator provides a realistic environment for testing and evaluating the performance of underwater drones and their associated algorithms. The simulation platform is designed to replicate real-world conditions and provides a suite of tools to design, test, and validate various control strategies for underwater vehicles. The simulator also supports the integration of various sensors and actuators that are commonly used in underwater robotics.

1 Introduction

Simulators play a crucial role in robotics by allowing developers to test algorithms and system integrations without constant reliance on physical prototypes. In underwater settings—where each AUV trial can require a support vessel, crew, and days or weeks at sea, with risks of equipment damage or weather delays—software simulation testing becomes even more valuable. But many existing underwater-robotics tools are rigid: they bundle tightly coupled components, offer limited sensor and thruster models, and often fail to run on recent Linux or ROS releases. This project tries to address these practical gaps in a modest, open-source framework: it accepts custom sensor or control inputs and produces matching outputs, lets you import your own chassis and thruster models, supports a range of virtual sensors and environment layouts, and integrates cleanly with current and future ROS versions without requiring extensive rewrites..

2 Software Architecture

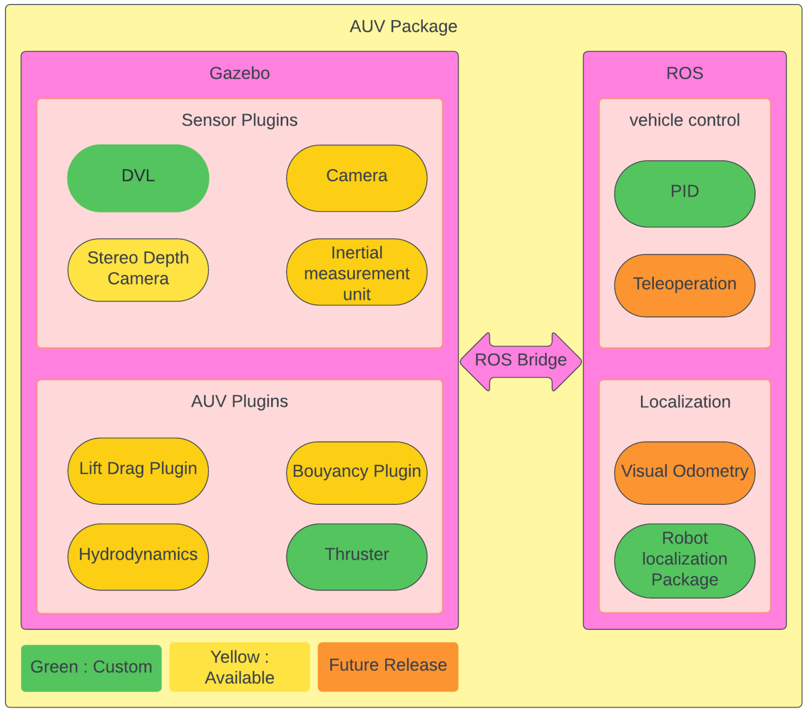


Figure 1: A figure highlighting the plugins,sensors and nodes used in the simulation colored according to their availability.

2.1 Software Plugins

The simulator is equipped with a comprehensive array of essential sensors vital for facilitating autonomous underwater traversal. These sensors include:

1. **Stereo Depth Camera:** This camera provides depth information in a stereo format, enabling the autonomous Underwater Vehicle (AUV) to perceive its surroundings in three dimensions, thus facilitating precise navigation and obstacle avoidance.

2. **2D Camera:** Primarily capturing 2D images or video streams, the 2D camera serves various functions such as object detection and visual navigation, contributing to situational awareness and decision-making processes of the autonomous underwater vehicle.

3. **Inertial Measurement Unit (IMU)**: Responsible for measuring vehicle acceleration and angular velocity, the IMU plays a pivotal role in navigation and orientation estimation, ensuring stable and accurate motion control underwater.

4. **Doppler Velocity Logger (Doppler velocity logger):** Among the most widely usedrs for underwater navigation, the Doppler velocity logger plugin accurately measures linear velocity along the x, y and yaxes. This sensor significantly enhances pose estimation by providing a precise description of linear movement, complementing the data obtained from the IMU.

2.2 Gazebo Plugins

1. **Thrusters :** As Gazebo provides no native thruster support, a custom plugin was implemented to simulate the Blue Robotics T200 thruster by driving rotational joints via ROS 2 control velocity controllers. In addition, a lift-drag plugin was configured using mesh files and thrust parameters supplied by Blue Robotics to accurately model thrust.

2. **Lift Drag Plugin :**The Ignition Gazebo Lift–Drag plugin computes hydrodynamic forces on thruster meshes using parameters such as angle of attack, radius of the fin, area of the fin surface, fluid density and relative velocity to generate realistic lift and drag.

3. **Bouyancy :** The Ignition Gazebo Buoyancy plugin applies upward forces based on the volume of the displaced fluid, the density of the fluid and the center of buoyancy of the model, allowing AUVs to simulate neutral, positive or negative buoyancy by configuring parameters such as the buoyancy coefficient, the density of the fluid and the reference link.

2.3 ROS Nodes

1. **Robot Locaization Package :** This ROS 2 package provides configurable nodes like the EKF node for state estimation via Kalman filters, supporting multiple sensor inputs. It was set up in EKF mode to fuse linear velocity data from the Doppler Velocity Log with angular and orientation measurements from the IMU, producing a drift-reduced, accurate pose estimate for the AUV.

2. **Visual Odometry :** Visual Odometry will process RGB-D camera streams—using visual feature tracking and depth data for scale—to estimate frame-to-frame motion; these VO pose estimates will be fused with the Robot Localization EKF alongside DVL velocities and IMU rotations, with full integration into the EKF framework slated for a future development phase.

2.4 Vehicle Control

1. **PID Controller :** The PID module serves as the primary navigation controller: it takes in state estimates from the Robot Localization EKF and computes RPM commands to drive the AUV toward user-specified coordinates. Tunable proportional, integral, and derivative gains ensure responsive yet stable closed-loop performance.

2. **Tele-operation :** The tele-operation node employs ROS Serial to establish connections with external gaming controllers. This node enables remote operators to send control commands and values directly to the simulator through ROS topics. This feature is valuable for testing and manual control of the autonomous underwater vehicle during development and experimentation.

3 Thruster Configuration

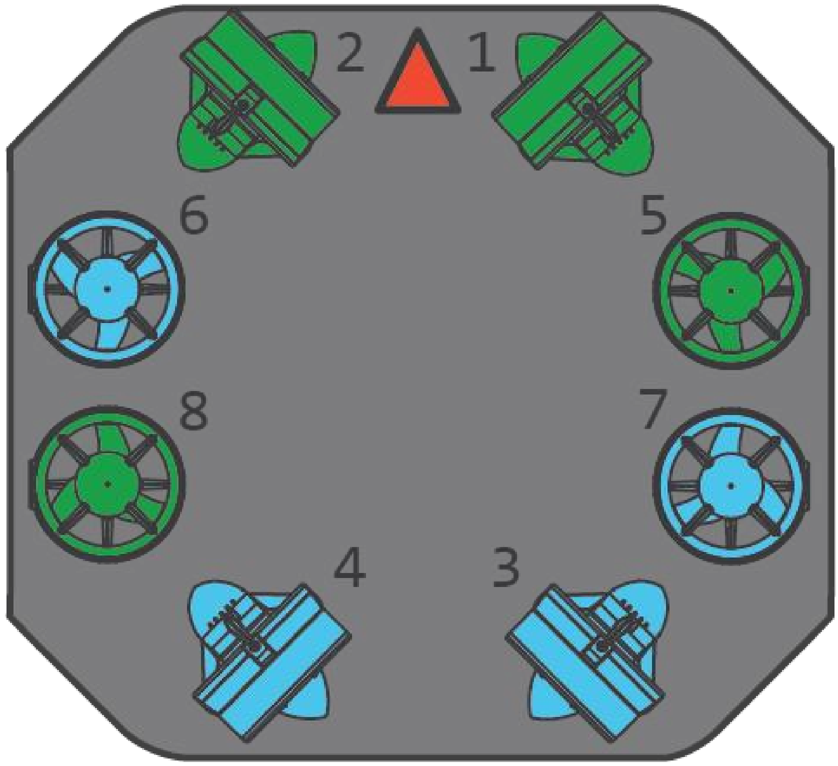
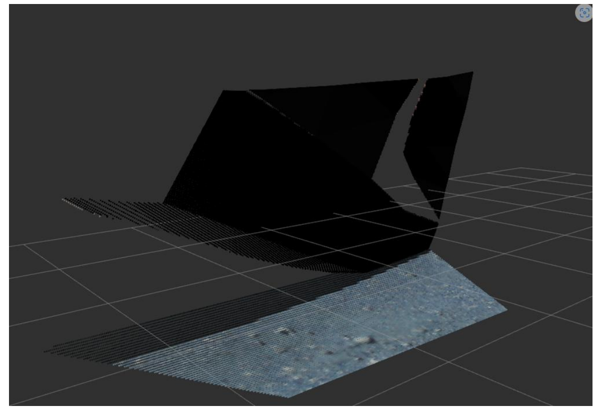


Figure 2: 8 - Thruster configuration from blue robotics

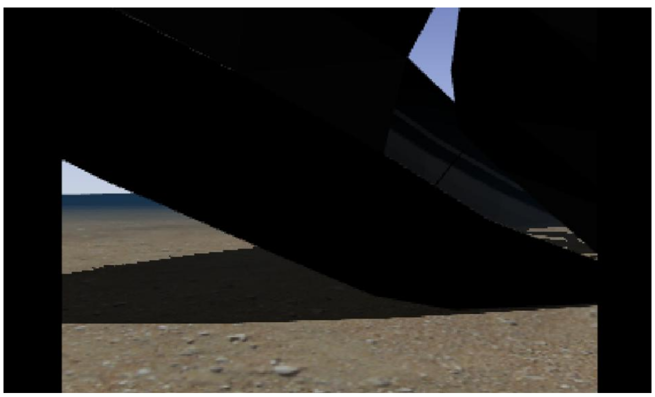
The AUV in the simulation adopts a conventional 8-thruster configuration fig 2., featuring four thruster's oriented along the xy plane and four thruster s aligned with the z plane. This configuration grants the autonomous underwater vehicle the capability to execute roll, pitch, and yaw movements within 3D space, thereby facilitating smoother and more stable control. To counteract the rotational momentum induced by the thruster s and prevent unintended autonomous underwater vehicle rotation, each thruster pair is equipped with both clockwise and counterclockwise rotating thruster's.

4 Simulation Feedback And Results

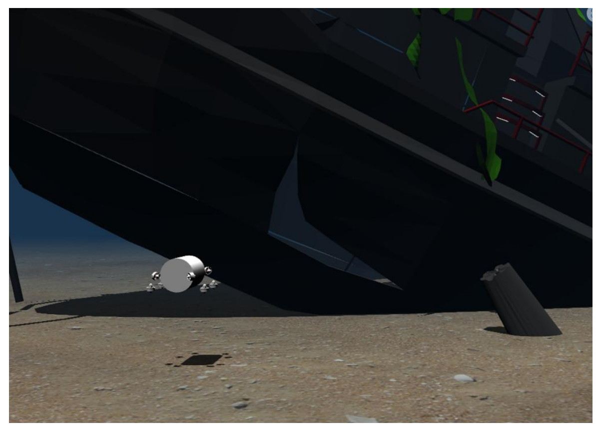
(a) A visual of the simulation environment [3].



(b) Output from the depth camera.



(c) Output from the RGB camera.



(d) Setup configuration.

Figure 3: Simulation environment and visual sensor output.

```
---
header:
  seq: 1966
  stamp:
    secs: 205
    nsecs: 311000000
  frame_id: "base_link"
orientation:
  x: -1.3216809145935267e-12
  y: 3.890716986426611e-12
  z: -0.00012220295934427638
  w: 0.9999999925332185
orientation_covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
angular_velocity:
  x: 0.00020458337652187178
  y: -1.8166084483533592e-07
  z: -1.6343130605027897e-11
angular_velocity_covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
linear_acceleration:
  x: -6.528359239809801e-07
  y: -0.0025161700432883363
  z: 0.0013678702134534582
linear_acceleration_covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
---
```


(a) IMU sensor output in ROS


```
header:
  seq: 1206
  stamp:
    secs: 172
    nsecs: 768000000
  frame_id: "base_link"
twist:
  twist:
    linear:
      x: -6.250659576047698e-06
      y: -9.409789397573143e-06
      z: -0.00013949745536518795
    angular:
      x: 0.0
      y: 0.0
      z: 0.0
  covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
---
```


(b) Output from the DVL.

Figure 4: IMU and DVL readings for localization

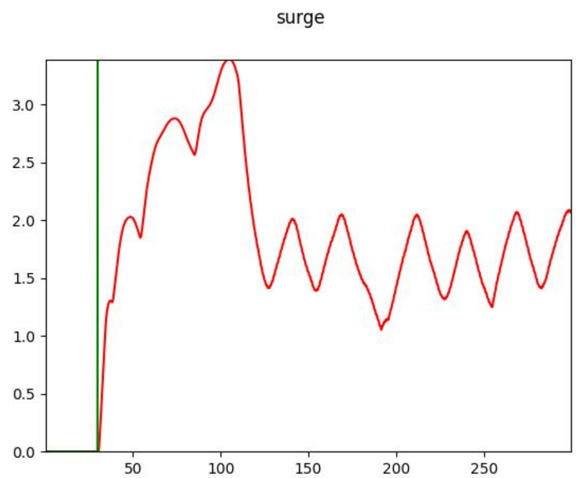
```
---
linear_position:
  x: 0.0
  y: 0.0
  z: 0.001
angular_position:
  x: 3.081250647227238e-05
  y: 1.0046198365752721e-07
  z: -0.019738229143993636
---
```

(a) Position data from localization package

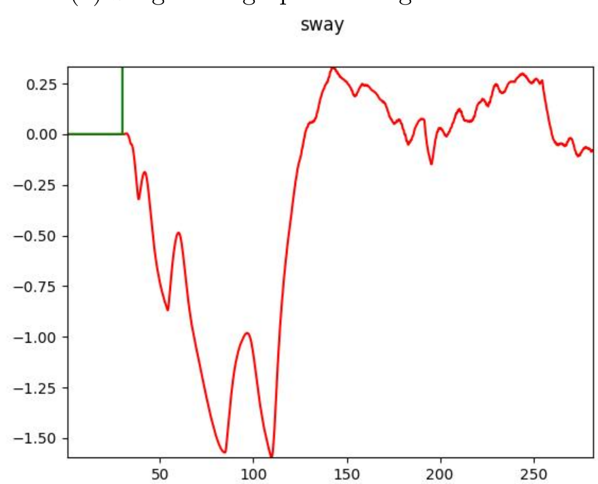
```
---
linear_position:
  x: 0.0
  y: -0.0
  z: -0.0001
angular_position:
  x: 0.0
  y: -0.0
  z: 0.0
---
```

(b) Velocity data from localization package

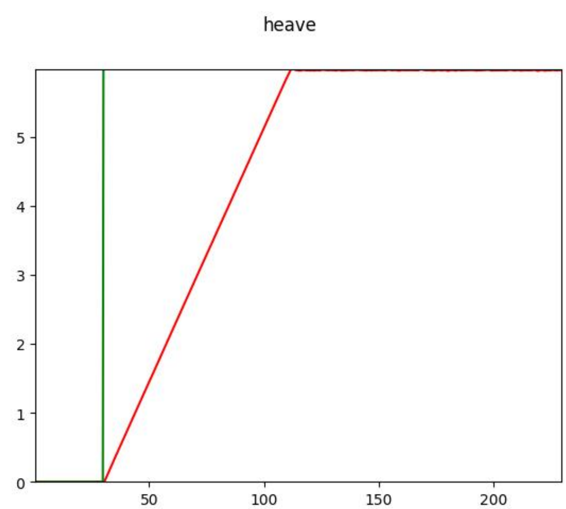
Figure 5: Robot localization package output



(a) Surge PID graph levelling off at x = 2



(b) Sway PID correcting to y = 0 after movement .



(c) Heave PID levelling off at z = 6

Figure 6: PID controller visualization

5 Future Work

The project is currently in a functional state, but several features are still missing. Tele-operation has not yet been implemented, which could limit manual control during certain testing scenarios. Visual odometry is also not included; however, it is not considered essential for the current goals of the project. In future updates, a mapping node is planned to allow the underwater environment to be mapped as the vehicle moves. At present, the DVL sensor does not provide a visual representation from its sonar beams, but this is expected to be addressed in later stages. Additionally, underwater currents are not yet simulated. These will be introduced using the waves plugin available in Ignition Gazebo to make the simulation environment more realistic.

6 Resources

https://github.com/Dafodilrat/RobotVisionProj.git

References

[1] Watanabe, Thomio and Neves, Gustavo and Cerqueira, Rômulo and Trocoli, Tiago and Reis, Marco and Joyeux, Sylvain and Albiez, Jan. (2015). The Rock-Gazebo Integration and a Real-Time AUV Simulation. 10.1109/LARS-SBR.2015.15.

[2] M. M. Zhang et al., ”DAVE Aquatic Virtual Environment: Toward a General Underwater Robotics Simulator,” 2022 IEEE/OES autonomous Underwater Vehicles Symposium (AUV), Singapore, 2022. 10.1109/AUV53081.2022.9965808

[3] M. M. M. Manhães, S. A. Scherer, M. Voss, L. R. Douat and T. Rauschenbach, ”UUV Simulator: A Gazebo-based package for underwater intervention and multi-robot simulation,” OCEANS 2016 MTS/IEEE Monterey, Monterey, CA, USA, 2016. 10.1109/OCEANS.2016.7761080.

[4] E. Potokar, S. Ashford, M. Kaess and J. G. Mangelson, ”HoloOcean: An Underwater Robotics Simulator,” 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 2022. doi: 10.1109/ICRA46639.2022.9812353.