Simulation of Semantically-Aware Obstacle Avoidance Algorithms for Underwater Robots

Chris Walaszek

Faculty Mentor: Junaed Sattar

irvlab.cs.umn.edu

Motivation

- Field trials for underwater robotics, while necessary to test algorithms, can be time-consuming and costly to perform
- Using a 3D graphics engine to test algorithms beforehand has many benefits, including
  - Easily modifiable environments
  - Realistic physics simulation
  - No risk of time lost due to unforeseen hardware/software issues on day of trial

Approach

- Decided to use Unity[2] for 3D graphics simulation. Advantages of Unity include
  - Native compatibility with ROS and ROS2
  - Extensive, straightforward scripting API using C#
  - Can create realistic underwater environments through simulated lighting and 3D assets

Contributions

- A demonstration of Unity's potential to serve as a platform on which to test underwater navigation algorithms using simulation of Aqua robot[1]
  - Unity can publish stereo image data from simulated cameras onto ROS topics
  - Left and right images can then be used to calculate disparity/deduce depth information
  - YOLACT image segmentation model can provide object detection capability using image data
- A Unity-compatible version of Semantic Obstacle Avoidance for Robots (SOAR) algorithm[3] using ROS2

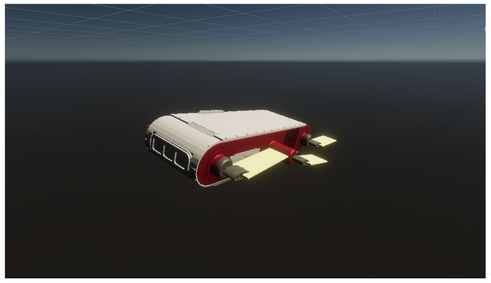

Fig. 1 The simulated Aqua robot. The model includes two front-facing cameras, a rear-facing camera, and IMU and depth sensors

Architecture

- **Aquasim:** Developed by Independent Robotics; provides simulation of Aqua in Unity engine (**Fig. 1**)
- **RoboDevel:** Developed jointly by CMU, UMich, and McGill University; provides hardware controller and autopilot for Aqua. Allows for control using roll, pitch, yaw, and speed commands through `/aqua/ap_command` ROS2 topic
- **Unity C# Scripts**: Publish camera, IMU, and position information onto `/front_{left,right}_camera/compressed`, `/imu`, and `/aqua_global_pose` topics, respectively
- **ImageConv ROS2 Node:** Converts image data from .png format to raw format, publishes onto `/front_{left,right}_camera/image_raw` topic
- **Other ROS2 Nodes:** Use information from topics above to perform image segmentation, disparity calculation, and robot navigation

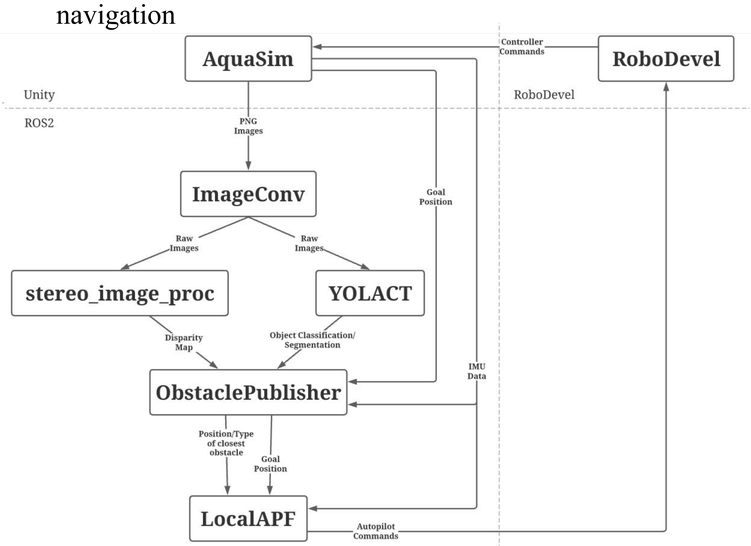

Fig. 2 Graphical outline of architecture used to run SOAR algorithm in Unity

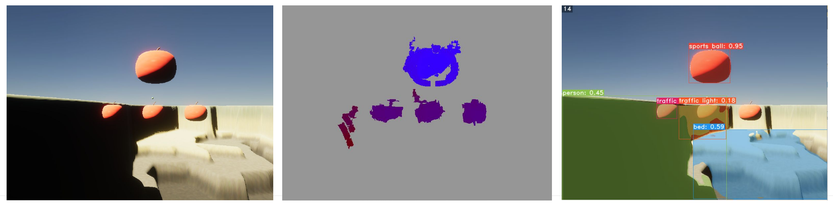

Fig. 3 From left to right: raw image taken from Aqua's left camera, disparity map between left and right cameras, YOLACT instance segmentation of image. Note the stray speckles in the second image and the false positive results in the third image

Results

- Unity/ROS2 implementation of SOAR can avoid obstacles in simple cases (**Fig. 4)**
- Image classification and segmentation often provides inaccurate object classification due to use of default YOLACT model (**Fig. 3**). Can be improved by retraining the model on more applicable datasets and/or providing more photo-realistic obstacles/environments
- Simulation has low performance due to high GPU usage of `ReadPixels()` and `EncodetoPNG()` methods in the Unity script that publishes image data

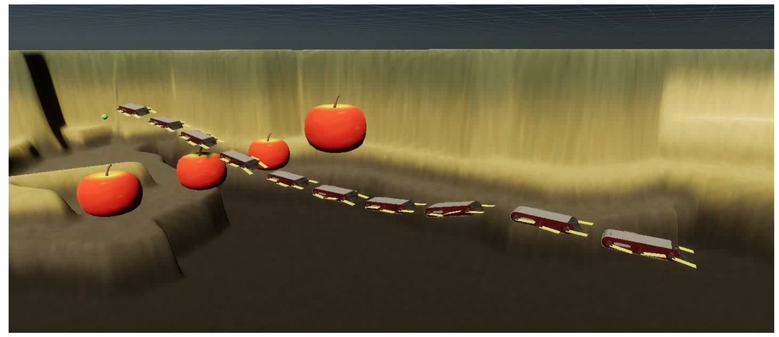

Fig. 4 Image sequence showing motion through obstacle field using SOAR in Unity simulation

Future Work

- Evaluate and improve SOAR algorithm through testing in various simulated underwater environments (e.g. pool, lake, ocean)
- Further optimize simulation for performance
- Evaluate usefulness of simulation within alternate research areas in robotics (e.g. dataset generation, HRI, virtual reality)

References

1. J. Sattar et al., "Enabling autonomous capabilities in underwater robotics," 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008, pp. 3628-3634, doi: 10.1109/IROS.2008.4651158.

2. Unity Technologies. Unity. https://unity.com.

3. J. Hong, K. de Langis, C. Wyeth, C. Walaszek, and J. Sattar, "Semantically-Aware Strategies for Stereo-Visual Robotic Obstacle Avoidance," *International Conference on Robotics and Automation (ICRA)*, 2021.

University of Minnesota
Driven to Discover℠

Interactive Robotics and Vision Laboratory (IRVLab)
Department of Computer Science and Engineering