RoboSub: Autonomous Underwater Vehicle

Team

Jin Chiu – Control Systems
Erika Estrada – Computer Vision
Kunal Juneja – GUI and Computer Vision
David Krystall – Computer Vision
Jesus Lopez – Team Leader

Advisor

Mark Sargent

Liaisons

Diego Santillan
He Shen
Mark Tufenkjian

Table of Contents

1.Introduction

The Robosub project is aimed toward building and programming an AUV (autonomous underwater vehicle) that will be able to autonomously guide itself through an obstacle course at the end of July 2019. The competition takes place at the US NAVY base in San Diego at the Trans Deck over the course of five days. Our project is sponsored by the Cal State LA AUV club and BlueRobotics, and the competition is sponsored by RoboNation and the US NAVY Office of Naval Research. This competition has been held every year for the last twenty years, and is an international competition that high school and college teams may participate in, with the majority of each team being comprised of a student body.

The course guidelines were released at the end of April 2019, and new obstacles for the course have been provided. In order to raise the success rate of our school's team we have enlisted a joint effort between engineering disciplines. Our team this year is composed of Electrical, Mechanical, and Software Engineers. With this team in place we have been able to control various aspects of the current AUV, titled 'GY4R4D05'. The code base software controlling the AUV is titled PENGUIN, which is built on top of the Osprey code base developed last year. The AUV is equipped with many different sensors that can be utilized to complete the various tasks along the course.

We have learned much from the challenges the previous team had faced during the competition last year. The biggest challenge was the computer vision's contour detection system performance in the murky water inside of the Trans Deck. During our own development we have also faced many challenges. Our team has had very limited

experience in software design for use with robotics, and our first semester we all familiarize ourselves with this new concept. We had a subdivision of our team learn and refactor several sections of code that used the Robot Operating System (ROS). No one had much previous experience with this platform either, so there was a long learning period for us to get familiarized with this system.

The main source of our robots autonomy comes from the computer vision module that has been in development throughout the second semester. This module is responsible for sending messages to the state machine to instruct the AUV on which tasks it needs to perform. The computer vision module is using a neural net for detection, which we had to train ourselves using previously gathered images from the Trans Deck. Over the summer we will continue to develop this, which will require a large amount of testing. Our testing experiences has been arduous, as we can only test our code when we have access to a pool, and the AUV hardware itself.

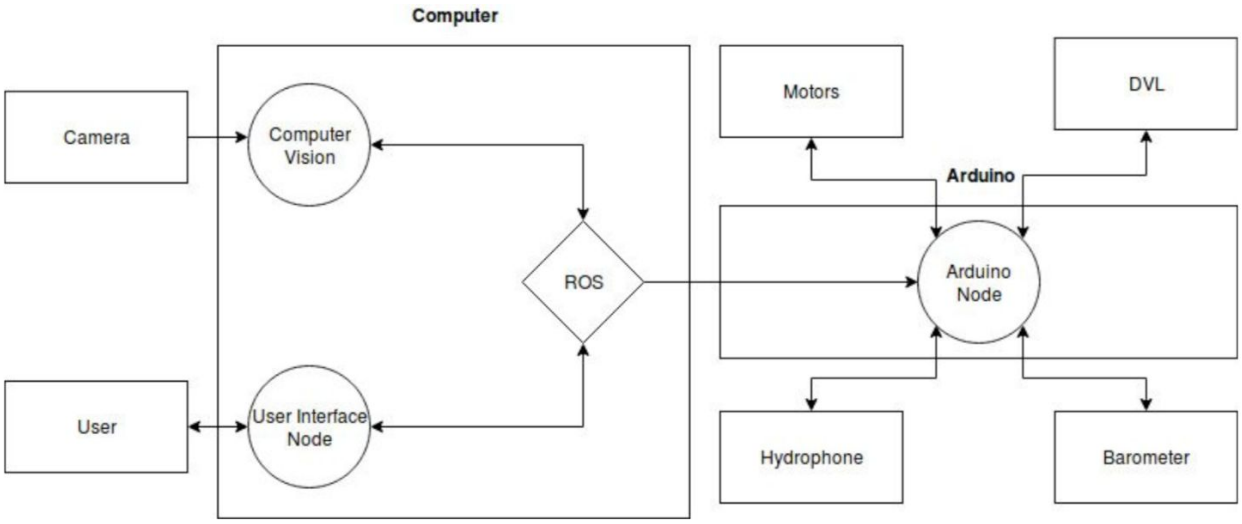
2. Related Works and Technologies

This project was an extension of the project that was worked on last year, so we had a fairly good existing resource to work from. Before the development process began, we reviewed our options for different possibilities that we could use to implement our robot. The majority of us had not had experience using ROS or any other robotics frameworks, so we decided to keep it in our toolkit after reviewing it. We were also tasked with researching how other previous teams implemented their AUV.. The largest common factor we could find was the use of neural networks, and among these implementations the most successful one was YOLO (You Only Look Once).

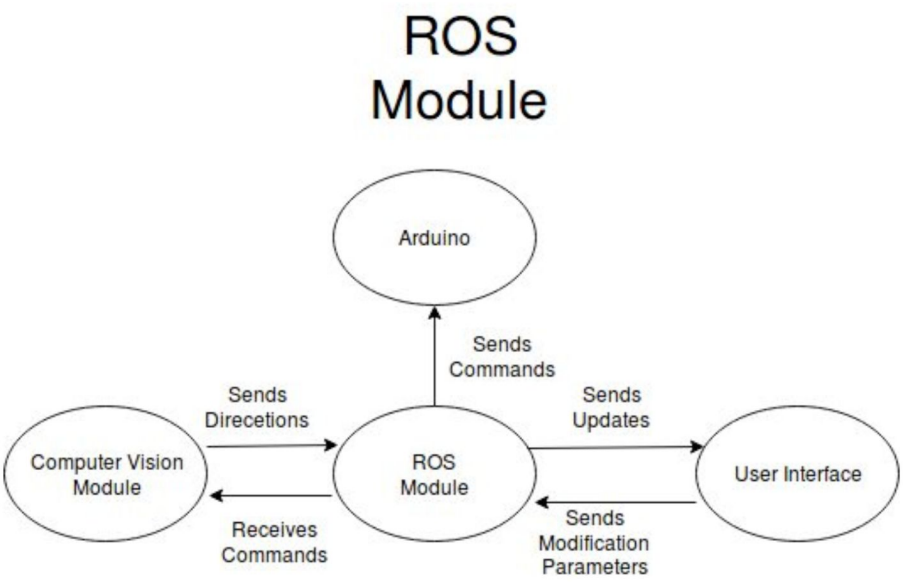
Throughout the project we brought in and removed many different technologies to our project. Our final project involved the use of ROS, Python, Linux Scripting, C++, Arduino, Ubuntu, Convolutional Neural Networks, and many other software libraries. The most significant tool used would be ROS, which is how we interface our python code with the different modules and hardware nodes on the AUV. Second to ROS would be our use of Darknet, which is the implementation of a Convolutional Neural Network. To get to the point where we were using Darknet, it was hard to get started at first. Before we decided on Darknet, we were using Tensorflow. We were using Tensorflow because the coursera course we were teaching ourselves how CNN's worked was using it. The setup process required use of CUDA and cuDNN, which was running on Ubuntu which we had to use because it was required by ROS, was rather difficult to build.

3. Systems Architecture

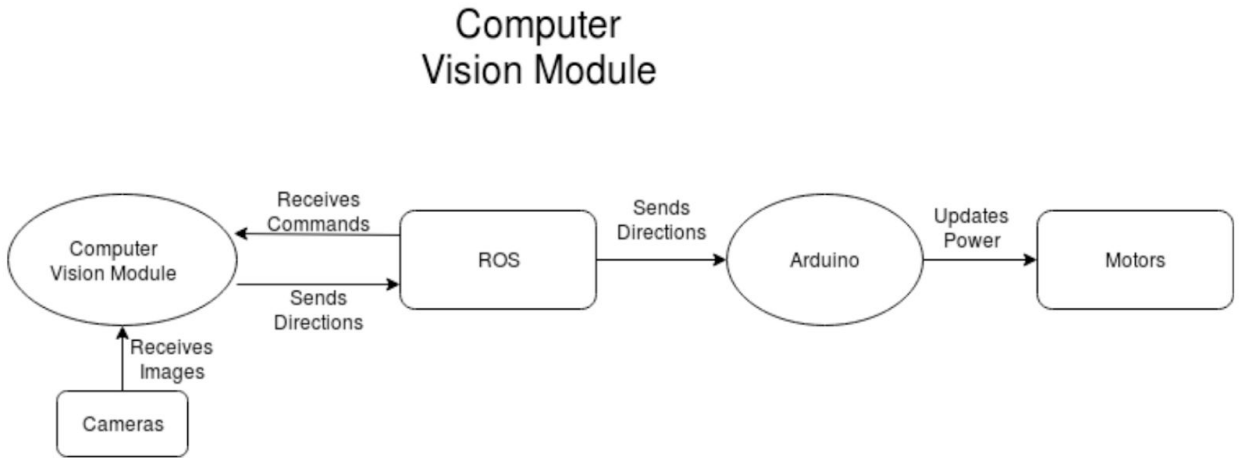
The majority of this project is built on top of the ROS platform, which uses the publisher/subscriber design pattern. The middleware was programmed using Python which helps the computer vision modules communicate with functions of the AUV. The computer vision module relied on OpenCV to gather an image stream which was then sent to be processed by YOLO. YOLO is trained with labelled image data that is then processed with a CUDA enabled GPU, and then is output to a weight file which can then be used by OpenCV. The hardware used for raw data streams to be processed by different ROS modules were the Cameras, the Doppler Velocity Log(DVL) and an Inertial Measurement Unit (IMU); we may also implement a Hydrophone system if time and resources allow.



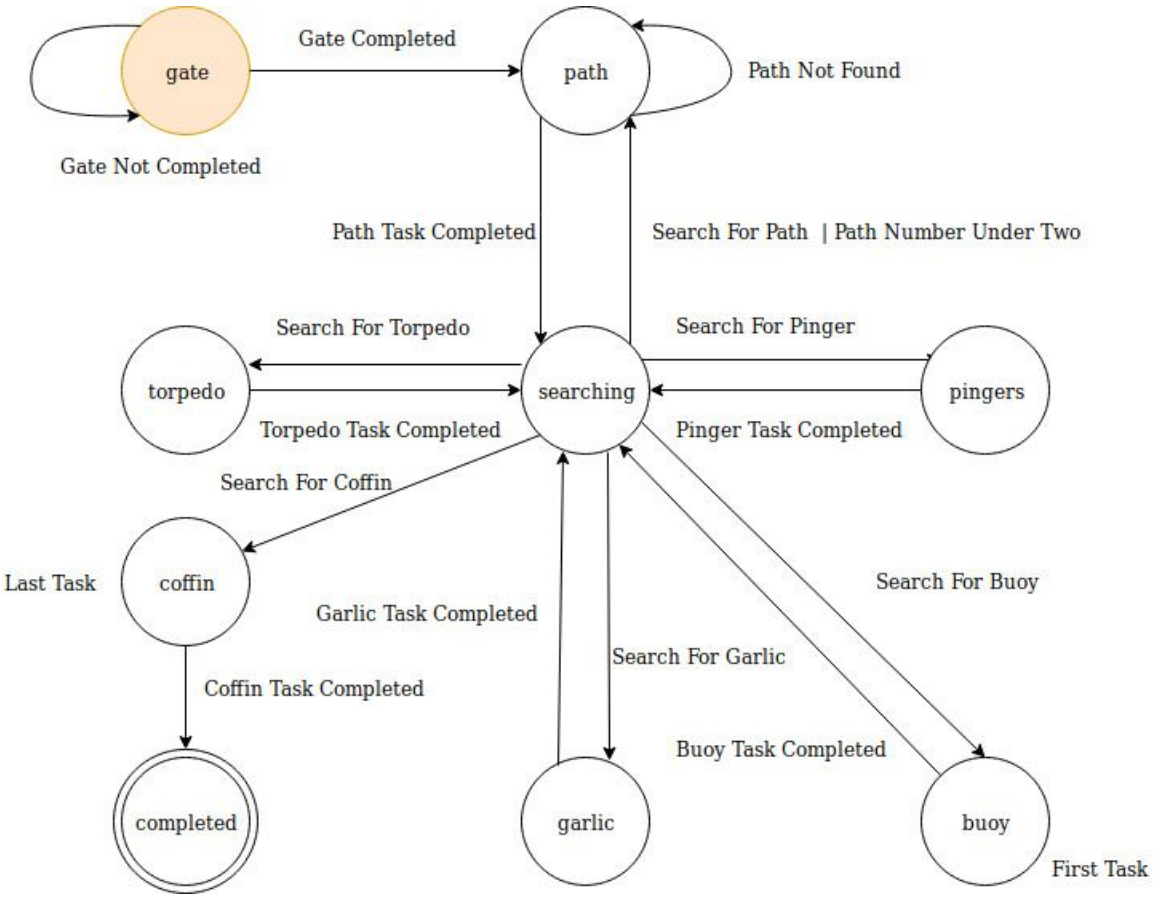
These figures display how ROS manages different publishers and subscribers. The Arduino Nodes gather data from the different sensors, which ROS is subscribed to. When the node publishes information ROS will send data to the Computer vision or User Interface which will then drive our state machine to tell the AUV what to do.



This is a diagram of a ROS module, and how it sends and receives messages from different parts of the platform.



This is a diagram of how the computer vision module will help drive commands deciphered from detected objects, which will call different states within ROS.



This is the current implementation of our state machine. It is based on the objectives that we will have to perform during the competition, and all states besides the exit state will return back to the searching state. We decided to use a state machine to avoid a dead state in the AUV, which will increase our success by never being stuck somewhere. If there is a case where we do end up stuck, our searching function will be able to still guide the AUV around, in a search path, and will keep the computer vision module open.

4. Results and Conclusions

We were able to accomplish much during the school year, but most of this time was utilized more as a ‘ramp-up’ period, during which we were learning how to use the many frameworks and libraries required by this project. We were able to clean up the previous code base, and implement a state machine to be used as mission control logic. We were able to label and train our own neural networks to detect previous objectives. We were able to achieve 90 to 100% precision with previous objectives, with a mean average precision of 97.73%, and an intersection over union score of 86.68%. We implemented a queue for the waypoint system the AUV uses. We are able to interact with all of the components on the AUV. We still have to finish training our next model with the new objectives we will have to perform. We also have to program the new maneuvers the AUV will have to perform during the competition. All of these features will continue to be developed over the summer until the Robosub Competition (July 29, 2018)

5. References

Website address referenced

1. Robosub Competition
    - http://www.robonation.org/competition/robosub
2. Operating System
    - http://wiki.ros.org/lunar/Installation/Ubuntu
3. OpenCV Library Documentation
    - https://docs.opencv.org/trunk/index.html
4. Robot Operating System Documentation
    - http://wiki.ros.org/Documentation
5. Arduino
    - https://www.arduino.cc/en/Guide/Introduction
    - https://www.arduino.cc/en/Reference/Wire
6. Tensorflow
    - https://www.tensorflow.org/
7. YOLOv3
    - https://pjreddie.com/darknet/yolo/
8. Convolutional Neural Networks
    - https://www.coursera.org/learn/convolutional-neural-networks

Documents Referenced

1. Robosub Tasks 2019
    - https://www.robonation.org/sites/default/files/2019%20RoboSub%20Mission%20and%20Scoring_v1.0.pdf
2. Robosub Software Requirements Specification
    - https://csns.calstatela.edu/department/cs/project/resource/view?projectId=6675265&resourceId=6841675
3. Robosub Software Design Document
    - https://csns.calstatela.edu/department/cs/project/resource/view?projectId=6675265&resourceId=6896499