RoboSub: Autonomous Underwater Vehicle

Team
Olin-Mao Alvarez – Team Lead, Computer Vision Developer
Danny Padilla – Computer Vision Developer
Jonathan Song – ROS Lead
Kevin Train – ROS Developer
Henry Ho – ROS Developer

Advisor
Mark Sargent

Liaison
Diego Santillan
Table of Contents

1. Introduction

2. Related Works and Technologies

3. System Architecture

4. Results and Conclusions

5. References
1. Introduction

The sponsors for this project were RoboNation and the Office of Naval Research. The essence of the RoboSub project has been the development of software which will control an autonomous underwater vehicle (AUV). The particular AUV the RoboSub team has been building has been dubbed the Eagle 2, and the software controlling Eagle 2 is called Osprey. The principal responsibility of the Osprey software is to guide Eagle 2 throughout a competition course, and to complete a set of tasks in order to receive as many points as possible and win the competition. The way this will be done is by using a series of different hardware sensor systems for gathering data, then interpreting that data with software directing Eagle 2 on what the proper action to take is, then having the software send signals to motors, and other hardware systems such that Eagle 2 properly completes each task.

The competition is broken down into two sections, qualification and timed runs. In order for the team to qualify we must have Eagle 2 pass through a gate, find a pole, circumnavigate the pole then find the gate again and pass back through. Next there are a series of timed runs where Eagle 2 must traverse a course and complete set tasks. Some of these tasks include touching buoys, dropping colored balls in areas of corresponding color, and a few others. There are a variety sensory information types which will be used including, but not limited to, images, sonic frequencies, gyroscopic heading, etc. The most critical signals that the AUV will be dealing with are the images. Every single task to be completed in the competition is heavily dependent on computer vision for both location and execution of each task.

There have been a number of challenges in regards to developing the software. The first challenge was understanding the source code the team had inherited from the previous year's competition. One of the things that made working with the old code difficult was that a large portion was written in matlab, which none of the developers on the team were familiar with. Another significant challenge was that the majority of the code which controlled the AUV was stored in one giant file. This file was extremely complex, did not really follow any real design architecture and was filled with redundancies. Because of this, the team decided to change to an object oriented approach for the development of the new software. Additionally, we knew that the competition would continue and we wanted it to be easier for the future teams to read the code.

Another roadblock for the team was understanding what we were working with in terms of the software and libraries we were going to be using. The subdivision of the team which was
responsible for the development of the maneuvering of the AUV were going to use Robot Operating System (ROS). None of the developers on the team had previously worked with ROS so there was a large learning curve. Additionally, the computer vision subdivision of the team had a large number of Python libraries to learn. Some of these libraries are OpenCV, Numpy, Pandas, Keras, and Matplotlib to name a few. Though the team had limited experience with some of these libraries they were mostly starting from scratch.

Despite these challenges the team had a number of accomplishments. Primarily the team was able to organize the new code base into easy to follow organized directories. Additionally, the new source code has been designed in such a manner that it is easy to understand and build on. Also the team has made sure that there is a clear object oriented approach wherever possible. Aside from this there is also a clear data flow architecture in place which is understood easily. In regards to actual implementation, the team developed fully integrated code that can receive various types of information from different sensors and maneuver itself. On top of that the team has developed a number of different classifiers for object detection. The computer vision can also provide directions to the AUV.
2. Related Works and Technologies

Given that this was such a large and complex project, there were a variety of different existing resources and technologies used in the implementation of this project. The main purpose of using previous works and technologies was to save time and find solutions which have already been found for specific problems and finding ways in which to incorporate them into the project. Furthermore there were a number of different libraries and resources which prove to be invaluable for the implementation of the code.

The most significant technology used for the project was ROS. ROS is a middleware that works between the operating system and the hardware. ROS provides ways for the software to communicate messages, control devices, and power motors. This was indispensable for the team because without ROS the team would have been stuck writing code to maneuver the AUV from scratch. ROS provided a simple solution to powering motors and other necessary hardware as well as a simple way for different pieces of hardware to communicate with one another.

On the computer vision side of the project there were quite a few critical Python libraries that were used. The most significant of which was OpenCV. OpenCV provided the bulk of the preprocessing and image fetching capacities to the software. Some examples of what OpenCV was used for include; fetching images and video, providing different color filtering to the images, edge detection, gaussian blurring and other transformations, color space transformations. OpenCV allowed the computer vision team to save countless hours of implementing complex algorithms on image files.

Furthermore the computer vision team used other Python libraries such as scikit-learn, Numpy, Pandas, Keras, and Matplotlib. These libraries were used to provide further functionality on top of the image manipulation. Functionalities include machine learning algorithms, optimized matrix and vector manipulation tools, data analysis, abstracted neural network constructions, data plotting and graphing, etc.
3. System Architecture

The vast majority of the project was written in Python to provide uniformity between the ROS side of the project and the Computer Vision side. The machine learning algorithms were created with various Python libraries including SciKit Learn, Keras, and OpenCV. The Computer Vision module relied heavily on OpenCV for image processing and filtering. ROS was used to interface with the Arduino Module (which controls the hardware) because it provides a robust set of functions that would otherwise have to be written from scratch. Data will be collected by the Arduinos and the Cameras and interpreted by the ROS module. The list of tasks completed and to be complete will be handled by the ROS module. The hardware used for signal processing includes Hydrophones, three cameras, a Doppler Velocity Log (DVL) and an Inertial Measurement Unit (IMU).
The ROS module will be used as an interface between the Main Computer and the Arduinos. ROS has 2 main functions. One of the ROS functions is to publish data that that is formulated once it receives values from the Computer Vision Module. The purpose of publishing the formulated data is so the data can be retrieved by the Arduinos which will then send commands to the corresponding piece of hardware. The 2nd function of ROS is to subscribe to the data which is broadcasted onto ROS by the Arduinos. In other words, it is the opposite of publish, which will retrieve data that is sent by the Arduinos.
The Arduino Module is used for the main computer to communicate with each piece of hardware. Each Arduino has different functions which as receive commands from the main computer, send commands to other Arduinos, receive data from hardware and send data to hardware. The barometer, and hydrophone will send data to their corresponding Arduino, while the motos will receive instructions from the Arduino.

The Computer Vision works as follows. First the CV receives a command from ROS indicating what type of object or task it is looking for. Next the Computer Vision selects a corresponding Detector Class. Each Detector is designed for a specific task. The Detector implements a Preprocessor interface and a Classifier interface. The preprocessor interface implementation determines what type of filtering will be done with each image and it also returns general regions of interest that are likely to include the target object. Then the CV sends each of these regions to a classifier. Classifiers used include, but are not limited to, Support Vector Machines and Convolutional Neural Networks. The classifiers then return positive matches. The CV takes the regions with a positive match and provides directions to ROS to guide the AUV towards it. ROS relays this information to the Arduinos which then trigger the motors to navigate towards the direction of the detected object.
4. Results and Conclusions

Over the course of the two semesters, we were able to organize a new code base into an easy to follow organized directories. We were also able to design the new source code in such a manner that it is easy to understand and build on for future AUV team members. We have also made sure that there is a clear object oriented approach whenever possible. Most importantly, we were able to get the AUV working with the new code. All of the hardware, cameras and Arduino are responding nicely to the new software. We are able to control the sub with the current software. Although we do not have a finished product just yet, it will continue to be worked on over the summer until the RoboSub Competition (July 30, 2018).
5. References

Website address referenced
1. Robosub Competition
   - http://www.robonation.org/competition/robosub

2. Operating System

3. OpenCV Library Documentation
   - https://docs.opencv.org/trunk/index.html

4. Robot Operating System Documentation
   - http://wiki.ros.org/Documentation

5. Arduino

6. Scikit-Learn

7. Numpy

8. Pandas

Documents Referenced
1. Robosub Draft Tasks 2018

2. Robosub Software Requirements Specification
   a. http://csns.calstatela.edu/department/cs/project/resource/view?projectId=6059788&resourceId=6301427

3. Robosub Software Design Document
   a. http://csns.calstatela.edu/department/cs/project/resource/view?projectId=6059788&resourceId=6550225