clearwater robotics

a clear company

ReefBot 2.0

Robot Autonomy Final Project Report

Wednesday 9 May, 2018

Basel Alghanem

Aaron Chong

Georgia Crowther

Karthik Paga

Edward Terry

Table of Contents

Project Overview and Objectives **2**

High-Level Goals and Motivation 2

Hardware Platform Description 2

Key Challenges 2

System Architecture **3**

Cyber-Physical Architecture 3

Software Architecture 3

Subsystem Descriptions **3**

Controls Subsystem 3

Algorithms and Methods 4

Implementation Challenges 4

Vision Subsystem 4

Algorithms and Methods 5

Implementation Challenges 5

Evaluation **5**

Evaluation of Individual Components 5

Evaluation of Complete System 6

Results 6

Description of Achievements 7

Discussion of Limitations and Future Work 7

Project Reflection **8**

Division of Work 8

Scheduling 8

Challenges 9

Lessons Learned 9

If you had to do the project again, what would you change? 9

Summary Video 9

References **10**

Project Overview and Objectives

High-Level Goals and Motivation

We would like to use and modify the ReefBot platform to perform fish detection and fish following. Robot-animal interaction is a largely untapped field of research. In the case of marine research, being able to react to and interact with fish is paramount. There have been examples of sufficiently convincing robotic fish integrating into fish schools for limited periods of time, typically via teleoperation in controlled experiments (Marras et al, 2012). The ability for a robotic fish to integrate into a school primarily depends on its positioning and orientation relative to neighboring fish (Faria et al, 2010). Though there is some research on the design requirements for robotic school-following fish (Huth et al, 1992), most of these have been simulations or teleoperated robots to research parameters like intra-school positional preferences. For applications that require a robot to interact directly with fish, unaided by scientists, untethered, and for long time periods, it is critical to be able to mimic and follow fish behavior autonomously. Our goal is to demonstrate the foundations of autonomy on ReefBot for such a potential future mission, with applications such as invasive species tracking and fish behavioral research.

Hardware Platform Description

The team has been given access to the VideoRay Pro4 platform, dubbed ReefBot during its time of service as a teleoperated educational exhibit in the Pittsburgh Zoo. The system, which remains tethered during operation, includes three thrusters, headlights, and various sensors (shown in Figure 1).

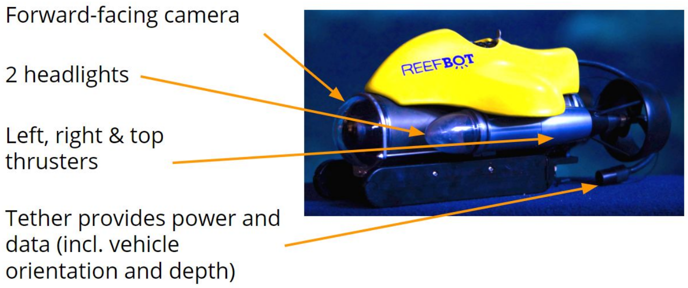
Forward-facing camera
2 headlights
Left, right & top thrusters
Tether provides power and data (incl. vehicle orientation and depth)
REEFBOT

Figure 1: ReefBot Overview

Key Challenges

In order to track and follow a target, ReefBot needs two main systems: a vision system and controls system. The intention of the vision system is to find, identify, and localize the target fish. It must be able to do this in the variable lighting the tank is subject to, and using the narrow field of view of our final camera. This system must report data on the position and distance of the target relative to ReefBot.

The second major system needed is the controls system. This system needs to interpret information from the vision system as thruster commands. This system must be able to compensate for the different power threshold of each truster, and the often unpredictable forces from the tether. Combined, these two systems will work together to autonomously identify and track fish-like targets.

System Architecture

Cyber-Physical Architecture

The vehicle comprises a number of subsystems, outlined in the following sections and summarized in the cyber-physical architecture in Figure 2. Both data and power are transferred to the vehicle via a tethered connection to the base station, with a laptop as the control center. The vehicle was equipped with an IMU, which outputs roll, pitch and yaw, as well as a depth sensor. Unfortunately, the depth sensor has been damaged and was not working. In the original configuration, the camera image data was transferred over the WiFi network along with the robot status data and commands. We have replaced it with a direct wired Ethernet connection.

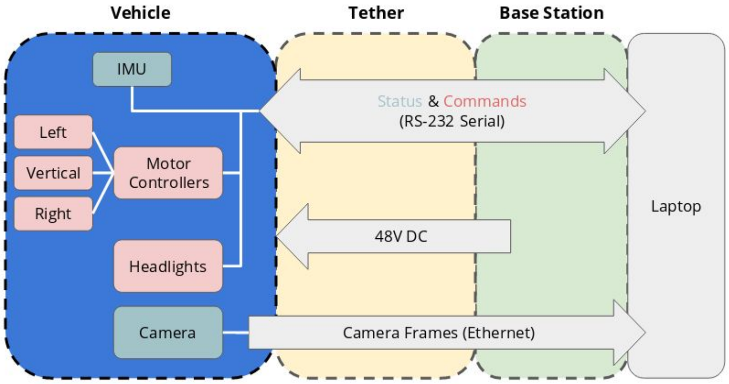

Figure 2: Cyber-Physical Architecture

Software Architecture

Our software architecture is described in Figure 3. First, we are using the legacy ReefBot codebase that interfaces with the platform. Next, we have developed a software package that interfaces with the on-board camera on the platform, detects fish , and determines the location of that fish. Finally, we deployed two control modes for navigating the robot. For the purpose of teleoperation, we take in USB joystick (using the ROS joy package) and interpret them as commands to the platform's thrusters and lights. In autonomous operation, we use the information from the vision package to continuously move towards the detected fiducial.

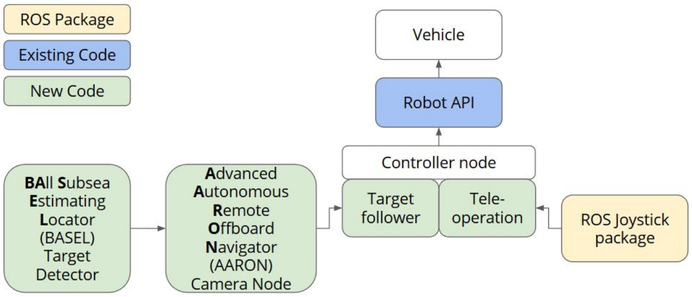
ROS Package
Existing Code
New Code
Vehicle
Robot API
Controller node
Target follower
Tele-operation
ROS Joystick package
BAll Subsea Estimating Locator (BASEL) Target Detector
Advanced Autonomous Remote Offboard Navigator (AARON) Camera Node

Figure 3: Software Architecture

Subsystem Descriptions

Controls Subsystem

ReefBot has three thrusters, which enable three degrees of freedom: longitudinal movement, vertical movement, and azimuth rotation. We have designed a notional control scheme to separately control each of these degrees of freedom, as described in Figure 4. We first use the vertical and horizontal distances of the target center compared to the camera frame center, and the size of the target compared to the goal size, to determine setpoints for our controllers. This information is collected and controls sent on a frame-by-frame basis.

Algorithms and Methods

We used proportional control for each of the three degrees of freedom. As the underwater environment is highly viscous, our system's movements are substantially dampened. As such, we found that proportional control is suitable for this application. For the left and right thrusters, we add the controls determined by the distance to the target and horizontal displacement of the target in the camera frame, and the distance to the target, determined by the size of the target. This gives ReefBot azimuthal and longitudinal control using the same thrusters. For simplicity of debugging, we started with classical control methods; in the absence of a more complex objective function, PID control is sufficient for our needs.

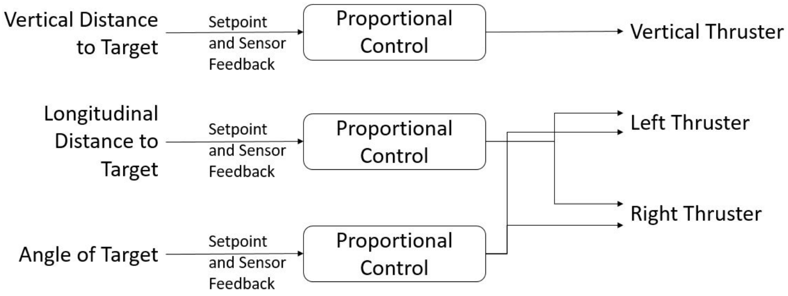
Vertical Distance to Target
Setpoint and Sensor Feedback
Proportional Control
Vertical Thruster
Longitudinal Distance to Target
Setpoint and Sensor Feedback
Proportional Control
Left Thruster
Right Thruster
Angle of Target
Setpoint and Sensor Feedback
Proportional Control

Figure 4: Control Architecture

Implementation Challenges

Each controller used analysis of our slow camera feed as sensor feedback. Further, simply due to the nature of how our thrusters worked, it was difficult to achieve slow forward thruster motion because of a considerable deadband. Water damage to the right thruster caused an even higher deadband. These factors, combined with substantial communications latency, made it difficult to tune our controllers for minimal rise time and steady-state error. If we had had more time for parameter tuning, we might have added a derivative control as well to reduce azimuthal oscillations.

Vision Subsystem

While the ultimate goal is to be able to track and follow underwater objects of arbitrary appearance, the starting point for verifying control algorithms is to use a blob detector to identify a red sphere that we use as our simulated 'fish.' We connected this ball to a long pole so we could move it around the tank. We then developed a ball detection algorithm to find the ball in the camera frame and return its size and position relative to the center of the frame, as seen in Figure 5.

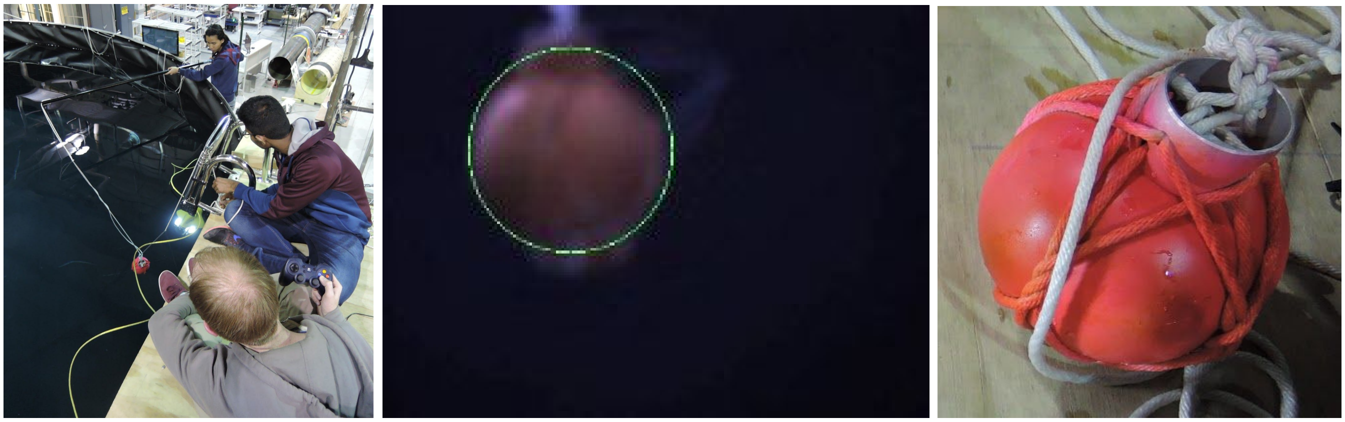

Figure 5: Underwater Ball Detection Set-up, Sample Detection and Target

Algorithms and Methods

Using OpenCV as the framework, we performed 6 processes sequentially on each frame/image to obtain the size and position of the ball:

1. Creating a binary image using HSV color filtering for reddish pixels
2. Applying several erosions and dilations to eliminate stray pixels
3. Finding contours and generating the contours' convex hulls
4. Filtering the hulls by area
5. Identifying the hull lowest in the image (a higher one is likely to be a reflection)
6. Estimate the position and size of the ball by the hull's center of mass and area, respectively

While neural network-based approaches are proving themselves to be capable of detecting arbitrary objects, we felt that a classical image processing approach was sufficient for our needs. The calibration of ball distance to pixel radius was carried out by measuring the detection response at a variety of separation distances.

Implementation Challenges

We strayed from our original plan to follow April Tags, as they were difficult to control and there was high glare under different lighting conditions made them hard to read. We also had to replace our original camera with an Allied Vision Manta unit. This camera did not have a fish-eye lens, like the Point Grey, so our field of view was constricted. Also, its power requirements forced us to reduce power to the rest of the system. Finally, designing a CV algorithm that could account for both over-exposed, under-exposed, and slightly occluded balls was challenging and required a comprehensive dataset of test images. Even with the final algorithm, if the ball were occluded along a line (e.g. if a rope cut through the image of the ball), the algorithm was not successful in finding the overall ball. It also struggled with detecting the correct size of the ball if it was on the edge of the field of view. Finally, if the ball were so close to the camera such that it occupied most of the frame, the algorithm would often identify a smaller object within the ball, and thereby vastly underestimate the size of the ball.

Some of the major lessons we learned include:

- Edge cases in object detection are quite important and need to be tested and tuned for explicitly
- Above water camera focusing was ineffective and, while focus was not critical, we could have determined the difference in necessary focal length and gotten sharper images. However, the blurry image actually facilitated the task of object segmentation.
- The nature of illumination is a major factor in the success of detection. The most problematic source of light was the large bay window next to the tank, which was most apparent at shallow angles relative to the water surface. In these conditions, when the vehicle lights were turned on, even at low intensity, the object in the frame became overexposed.

Evaluation

Evaluation of Individual Components

Because we were able to integrate our components into a complete system, most of our subsystems were evaluated in that context. However, a few components were tested individually. Our visual fiducial detection algorithm was tested against a dataset of example camera images and qualitatively assessed for successful detections, approximate size and relative position predictions of the ball in 100% of the images in our dataset. The fish tracking was evaluated by collected data in bag files and analyzing the drift in position of the target over time, in order to properly tune the gains of the different thrusters.

Evaluation of Complete System

Our evaluation of the complete system involved testing ReefBot in a series of increasingly complex tracking situations. The results for two of the basic sequences are presented below. A more complex sequence combining these maneuvers was run, and while the results are too complex to show here, the reader is invited to view our accompanying video.

Results

We were faced with a few challenges in collecting data. Naturally, the preference for evaluating a control system such as this is to use a reliable ground truth. However, given the complex manual control involved in moving the target, the only test case meeting this benchmark is the stationary tracking case. For the vertical and fore-aft tracking, the operators did the best they could to move the ball at a steady, repeatable rate. Furthermore, as the depth sensor is broken, this takes away a potentially valuable source for evaluating the robot's vertical tracking. In the absence of a 6DOF robot pose as a setpoint, the location of the ball in the frame at (0,0) and a separation distance error of 0.0m is used as a proxy for a static tracking objective.

Figure 6 captures many of the fundamental dynamic behaviors we observed:

- The period of lateral oscillations is between 7-10 seconds, and the period of fore-aft oscillations is approximately 25 seconds.
- The vertical tracking is very stable, with the ball center in Y remaining within 25% of the frame height. This can be attributed to the low motor inertia and quick response time of the vertical thruster as well as the highly damped dynamics of direct vertical motion.
- The repeated tendency for the ball center in X to fall below 0 (i.e. target on the left of center) represents the left thruster being more powerful than the right. This behavior is complicated by the fact that twists in the tether set up a steady state external torque at certain orientations. We have observed that when both thrusters are manually set to full forward thrust, the robot travels straight, which confirms our suspicion that the culprit of this asymmetry is low-thrust deadband.

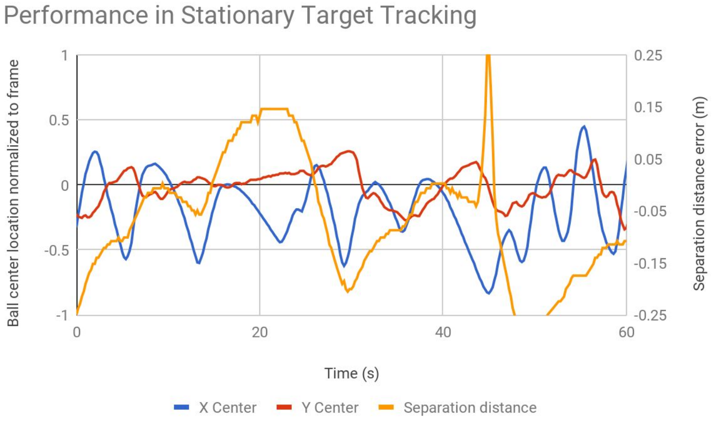

Figure 6: Performance in Stationary Target Tracking

Figure 7 shows the main thruster commands resulting from a more complex fore-aft tracking case. In this case, the target was first moved towards the robot for 10 seconds (50s-60s), held in place for 5 seconds (60s-65s), and then moved away from it for 10 seconds (65s-75s). It is apparent that even in a nominally symmetrical motion case (to the best of the target operators' ability), the thrust commands are oscillatory. However, given the moving ball remains at a relatively constant position in the robot's frame, one can infer that it can maintain a constant separation distance quite well, despite some lateral oscillations.

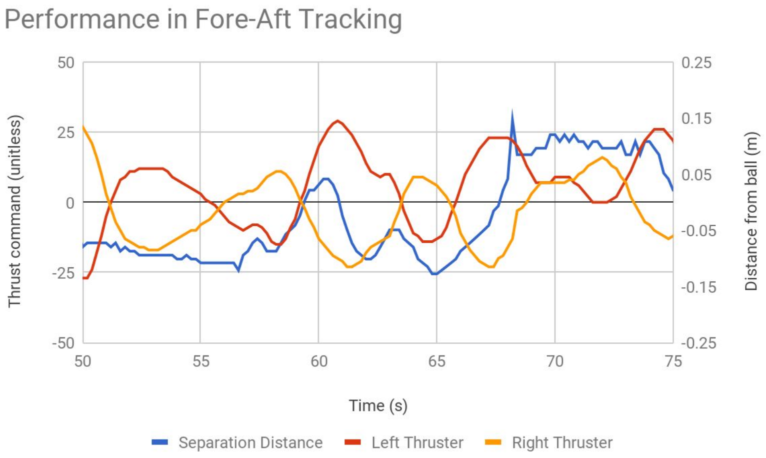

Figure 7: Performance in Fore-Aft Tracking

Description of Achievements

ReefBot had several non-functional parts when we started on this project, with an outdated codebase as well as onboard camera, we put in significant effort to revitalize it to a working state as well as refactoring the codebase to run on newer systems.

Other than that, we have also made sure to document our work properly such that any future team will be able to hit the ground running when doing another project with ReefBot.

Post initialization on the surface (tracking established), the robot was commissioned to operate in autonomous mode while the ball was submerged into the tank. Several trials were conducted and marked successful if the robot autonomously tracked the visual fiducial while maintaining a minimum distance throughout the entire length of the trajectory.

Similarly tests were conducted to assess the tracking performance when the marker was surfaced from the bottom of the tank, the tracker was moved along a straight line at a given depth.

Finally, the continuous tracking capabilities were tested for smooth unstructured paths e.g. 3D spline. Observations were in line with the initial structured path tests, for example, the robot maintained better tracking with variation in depth compared to lateral motion.

Discussion of Limitations and Future Work

If we were to continue our work on ReefBot, we would implement more advanced tracking and controls methods, in addition to attempting to track and object more complicated that a ball, with directionality, which would require more advanced planning.

Currently, whenever the ball leaves the narrow view of our camera, ReefBot no longer knows where to go and simply sets thrust to zero. A better, more robust method would be to track the path of the ball and do some prediction about its direction. This would require us to integrate IMU data to isolate what motion in the camera frame is caused by the target's motion, and what is caused by ReefBot's motion. If ReefBot truly loses the ball, or if it has just started searching for the ball, we would integrate some ball search state, where ReefBot rotates to search for an object to track.

A second step might be to track an object that is more ‘fish-like’ than a ball. This would require more advanced computer vision methods to determine not only where an object is, but what direction its facing compared to the robot. We would then have to do some path planning and trajectory estimation to determine how to orient the robot compared to the target. This would be particularly difficult given ReefBot’s power limitations and wide turning radius.

Hardware upgrades would also be an important part of future work. We initially planned to use a depth sensor and an IMU to provide sensor feedback to our control algorithms at faster rates than our camera processing. Adding those sensors and re-tuning the controllers for the faster sampling rate would enable the ReefBot to follow targets more quickly and consistently.

Project Reflection

Division of Work

Work was divided into three main teams:

- Vision: Basel and Aaron worked on camera data acquisition and communication, in addition to our computer vision pipeline for ball detection.
- Controls: Ed worked on the proportional control scheme for ReefBot and re-mapping and documenting manual controls on the joystick controller.
- System: Georgia and Karthik worked on preparing ReefBot, testing subsystems, reading prior documentation, documenting current functionality, and building the ball and tracking infrastructure.

Scheduling

We used a Gantt Chart to plan our project schedule that was mainly adhered to. Green blocks represent our intended task schedule, yellow blocks indicate tasks that were performed later than scheduled, and red blocks indicate aspirational tasks that were not achieved.

Week	1	2	3	4	5	6	7	8	9	10
Date	**26 Feb**	**5 Mar**	**12 Mar**	**19 Mar**	**26 Mar**	**2 Apr**	**9 Apr**	**16 Apr**	**23 Apr**	**30 Apr**
Vehicle maintenance and modifications	🟩	🟩	🟩	🟩	🟩	🟩				
Gain experience in teleoperation				🟩	🟩	🟩				
April Tag detection testing					🟩	🟩				
Control scheme development					🟩	🟩	🟧		🟨	
Target detection testing							🟩	🟩		
Target tracking testing							🟩	🟩	🟨	🟨
Multiple target tracking testing									🟥	
Final evaluation										🟩

Challenges

Though individual component challenges have been previously covered, we did have some broader challenges with the overall project and platform. While there is an extensive codebase which has been used in the past, there is scant documentation about the structure of the code, the operation of the original camera and the connections on the circuit board. Without proper technical support and documentation, interfacing with the robot required more time than necessary, as the network protocols were poorly documented.

The platform had an insufficient power supply as it was incapable of supplying power to lights, thrusters and the camera simultaneously. When both thrusters were at full power, the ReefBot experienced a ‘brown-out’ and we lost connection to the camera feed. Even with the motor power capped at 30%, applying power to both at once caused the connection to drop.

Lessons Learned

ReefBot taught us importance of teamwork and setting a reasonable project scope. We anticipated that the difficulties of working with older hardware would take some time and patience. Sealing and testing the water-tightness of the robot before each deployment was laborious, but patience and attention to detail here ended-up saving us a lot of time in the long run. Dealing with older documentation and debugging legacy systems is often complicated and dense, but it is also an important skill applicable to many revived or poorly documented projects and platforms.

If you had to do the project again, what would you change?

As a team, we were happy with our project goals, scope, and achievements. If we were to re-do the project, the main changes would be the way we approached the technical challenges with the benefit of hindsight. For example, we would have spent less time attempting to interface with sensors that never ultimately worked.

Summary Video

Our summary video can be found here: https://vimeo.com/268881066

References

Marras, S., & Porfiri, M. (2012). Fish and robots swimming together: attraction towards the robot demands biomimetic locomotion. Journal of the Royal Society Interface, 9(73).

Faria, J.J., Dyer, J.R.G., Clément, R.O. et al. Behav Ecol Sociobiol (2010) 64: 1211. https://doi.org/10.1007/s00265-010-0988-y

Huth, A., & Wissel, C. (1992). The simulation of the movement of fish schools. Journal of Theoretical Biology, 136(3).

Image of ReefBot: ReefBot. (2013, October 19). Retrieved February 15, 2018, from http://hivepgh.sproutfund.org/project/reefbot/

Ahrnbom, M., et al. (2017). Improving a real-time object detector with compact temporal information. *IEEE Xplore*. Retrieved from https://ieeexplore.ieee.org/document/8265241/.