The High Performance Plongeur v2.0 Georgia Tech RoboSub 2019

Praneeth Eddu, Eric Phan, Rahul Rameshbabu,
Jonny Tan, Anthony Velte, Joshua Dulle
Patrick Meyer, Michael Steffens

***Abstract*—This paper describes the design of Georgia Tech's platform for the 2019 Robosub competition. This years theme was iteration. The main vehicle design has stayed largely consistent for the last two years, with small updates for improved performance. The major updates for this year include additional ease of use features, such as LED status lights, and further improvements to the modular internal electronics tray. This consistency in hardware allowed for a greater focus on software. The autonomy framework has largely been rebuilt entirely after integrating lessons learned during the 2018 team's trial of the ROS middleware system. This undertaking has led to a more modular autonomy framework that can be quickly adapted to changing competition conditions. As in the past, our strategy for the competition is to focus on consistency. This is born out in our focus on a limited set of tasks that can be accomplished mainly through solid navigation and perception, with minimal additional actuation required. Our focus on navigation and perception has shown promising results in current testing, and we look forward to building on our performances in previous years.**

I. Competition Strategy

The Georgia Tech Marine Robotics Club is excited to once again be participating in the annual RoboSub competition. As we have continued to learn from our past competitions, we have taken home the lesson that consistency is what really leads to results at these competitions. As such, the theme for this year was iteration and improvement. This led to the choice to leave the main vehicle platform largely unchanged from the previous year. Minor updates to the hardware side of the competition include an improved dropper mechanism and additional ease of use features. These new features include an LED status display, greatly improving understanding of the vehicle's performance during testing.

To utilize our pre-competition development time efficiently while attempting to maximize the score, the team chose tasks that primarily required vision coupled with navigation and control. The key objectives are:

- 9.1 Journey to the Undead Realm
- 9.2 Enter the Undead Realm (Gate)
- 9.4 Path
- 9.5 Slay Vampires (Buoy)
- 9.6 Drop Garlic (Bins)
- 9.8 Expose to Sunlight

This strategy negates the added complexity of actuators for firing and picking things up and does not rely on a robust hydrophone system to locate pingers. (See *Appendix A* for full strategy.)

The choice to develop strong navigation and control over other hardware/software was easily justified, because a solid moving base is needed to implement any more advanced hardware/software. This was assessed by bench tests of motor responsiveness followed by and weekly pool tests. During pool tests the PID controllers were tuned and autonomous navigation was verified, culminating in an attempted autonomous gate run for pre-qualification. Six weeks were dedicated to the reliability of the navigation and control hardware and software.

Software development on the detection and classification tasks was started simultaneously, with the goal of having a functional perception stack at the eight-week mark. The backbone of the perception stack treats perception as a service, which fits with the design goal of a modular software system by allowing the back-end (neural net, OpenCV, or other tools) not to matter. Multiple approaches are being tested such as having a single model for all objects or splitting the objects up between different neural network models. These approaches will be compared using various metrics.

While designing the neural-network architecture, the team considered a few crucial factors: complexity, reliability, and training speed. These factors were quantified and compared by using metrics such as time taken for set-up to assess complexity, a confusion matrix to express reliability, and time elapsed to indicate training speed. Due to the nature of neural networks, the size and quality of the training set more greatly influences reliability more than any other factor, so the complexity was not sacrificed to attain improved reliability. To balance development of the software with in-water testing time, precursory test images to train the neural net were manually collected during pool tests to tune the PID and test RC control, allowing the time spent to provide dual benefit.

II. Vehicle Design

The design of the submarine included multiple, creative decisions on the fronts of hardware and software. The hardware considerations included decisions on the static stability and degrees of freedom of the physical vehicle. The software decisions included architecture choices to achieve a balance of modular design and effectiveness for the RoboSub specific tasks.

A. Hardware

Among the most creative aspects of the system is the unorthodox tall frame design, with the main pressure vessel at

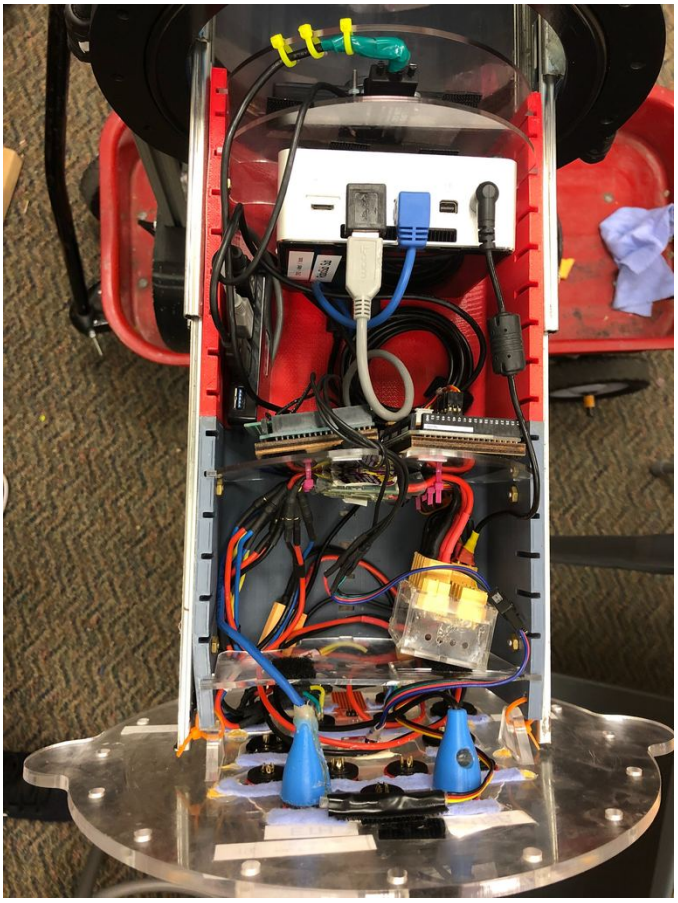


Figure 1: 3D Printed Modular Electronics Tray

the top of a rectangular aluminum frame and all ballast placed at the bottom. As previously discussed, this places the center of mass significantly below the center of buoyancy for the vehicle. This leads to vehicle dynamics that have significant static stability in both pitch and roll. This decision was made to reduce the degrees of freedom that would need to be controlled by the submarine. This simplification of the dynamics of the vehicle was helpful in a matching simplification of the control.

Another major feature of the vehicle is the slotted aluminum frame. This frame is what allows for the tall design and makes modifications to sensor and actuator packages much easier. The slotted rails are commonly used in different applications, so adapters and connectors of varying forms are readily available. This makes the integration of new systems significantly easier than if a custom manufactured frame were used.

Another creative feature of this year's vehicle is a custom designed modular electronics tray. This can be seen in Figure 1. Previous years' vehicles had used assemblies of laser cut wood for electronics. However, these did not make for an efficient use of the space. To maximize the space, a 3D printed rack was designed that could accept trays with components attached. These trays were designed specifically for the components they would hold, and allowed for improved cable management and heat distribution. Additionally, if certain components malfunction in testing either in water or on land it is much easier to identify which components or wiring may be

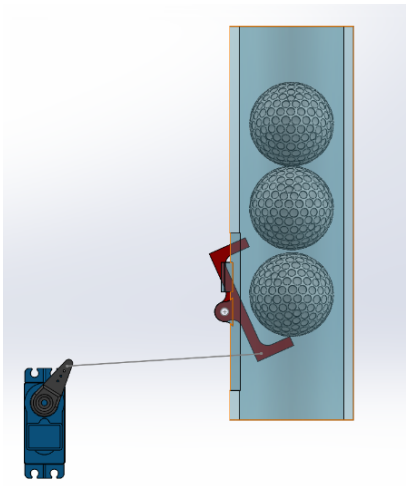


Figure 2: Final Dropper Design

causing the problem and address them directly. Having a much cleaner and less-cluttered components tray made connections easier and reduced the likelihood of plugging wires into wrong places or missing a connection during setup.

Due to budget constraints, many sensors that would aid in navigation are not included on board the vehicle. As such, some creative approaches must be taken to ensure accurate state estimation. The main navigation sensors are a Microstrain 3DM-GX3-25 IMU and two monocular cameras, one forward facing and one downward facing. While the IMU is highly accurate in providing pose information, double integration of accelerometers causes significant errors to compound with time. To alleviate this, the use of visual odometry is being explored. By understanding the shift in location and angle to key points of reference in a cameras frame, a change in position and orientation can be estimated. This can be done for both the forward facing and downward facing cameras. Combining this information from the cameras with information from the IMU and an estimated vehicle dynamic model, a relatively accurate estimate of location relative to the starting location can be found. This is useful in mapping the area and allowing the vehicle to return to a task if it believes it can get more points in a second attempt. This can also be useful in bounding classification of objects within the course and the triggering of their associated behaviors.

For the dropper design, the custom markers the team carries for the garlic drop tasks are golf balls with the team colors on it. Golf balls were chosen because of their ability to sink directly down without weighing down the vehicle and accessibility for the team to have extras when they inevitably get lost during competition runs. The dropping mechanism was designed to be 3d printed for rapid prototyping as development continued. The final design consists of a main cylinder that houses the two markers and an arm that swivels to release a single marker at a time. The arm is actuated using a waterproof servo and returned to the original position by an elastic band. Figure 2 shows the dropper design.

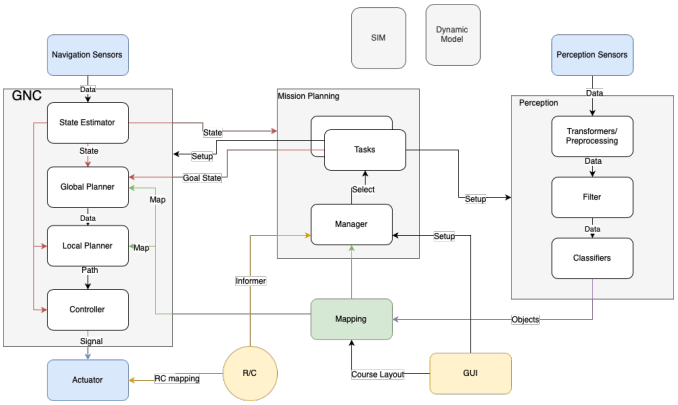


Figure 3: ADePT-ROS Architecture

B. Software Architecture

The ROS architecture, called ADePT-ROS, has a focus on modular design. An overview of the architecture can be found in Figure 3. The architecture is broken into three main components: the navigation stack, the mission planning stack, and the perception stack. Other components include a cross-platform QT-based GUI and drivers to interact with our system hardware. Arduino micro-controllers are also integrated into the ROS architecture. Each of the three major components of ADePT-ROS will be discussed below.

1) Navigation Stack: The navigation stack consists of four nodes: the state estimator, the global planner, the local planner, and the controller. The state estimator node produces estimates of the vehicle's position and orientation as it moves through the water. This is done through a combination of taking in sensor information and a model of our submarine's motion. The global planner provides a path the vehicle can follow to achieve a goal state provided by the tasks. The local planner is used to follow this path as closely as possible, accounting for disturbances to the vehicle and potential obstacles in the environment. Finally, the controller takes in the plan from the local planner, and converts this to actuation signals to be sent to each motor.

In the spirit of modularity, each of these nodes is implemented as a wrapper that then instantiates and calls on specific implementations. This structure serves two purposes. First, it allows for different implementations across multiple vehicles. This is important because the structure was also used in both the RoboBoat and RobotX competitions, which are surface-based. Surface-based vehicles only need to plan and estimate in a 2D space, while the sub must plan in 3D. Second, it allows for multiple algorithms to be implemented for each node and used interchangeably. Not only does this allow the team to implement a basic implementation before integrating more complex algorithms, it also allows the vehicle to choose the best algorithms for each task.

An illustrating example would be the Slay Vampires task from this year. For most other tasks, it is ideal to avoid contact with the task elements. This would lead model-based state estimators to work well. However, the Slay Vampires task is built around obvious contact with an obstacle. This may pose a problem for model-based state estimators, while simpler dead-reckoning estimators could do well over the short time periods of this task. By designing the architecture in a modular way, the vehicle is given the flexibility to change how it's state estimation and other navigation is done to best meet each task. Furthermore, this modularity has allowed us to test methods from as simple as integrating acceleration measurements to obtaining an estimate of position to as complex as visual SLAM techniques that generate maps as well as localization by using the camera feed.

For vehicle controls, yaw and depth are controlled using decoupled PID controllers. Translation control is done with an open loop approach by applying specific commands that map to specific velocities. These velocities are based on an empirically derived motion model. This motion model was built by testing linearly spaced motor commands and hand-timing the velocities. The velocities in between these points are linearly interpolated.

2) Perception: The perception stack consists of three main nodes: pre-processing, filtering, and object classification/detection. For our purposes, the distinction between object classification and detection is based on the output of the node. If the node identifies both the type of object (or the type of multiple objects in one image) and where in the image it is, it falls under object detection. If it only returns the type of object present in the image, it falls under object classification. These nodes are implemented similarly to those in the navigation stack. That is, they are wrapper nodes that allow for different implementations to be mixed and matched in a modular fashion depending on the task being performed. The stack is also formatted in such a way that different nodes can be skipped entirely, thanks to ROS's publish/subscribe architecture. This can be useful for some computer vision architectures that combine filtering and classification/detection into a single function.

For vehicle perception, we utilize a forward and downwards facing camera. By running the forward camera feed through a trained neural network, we can determine what the sub is currently seeing. For pre-processing and filtering, the same implementations can largely be used throughout the competition. This includes common approaches such as image gray scaling and blurring in pre-processing, and edge detection and feature extraction in filtering. However, the classification/detection node requires specialized algorithms trained specifically for each task. Each task will require a specific implementation to either classify or detect the objects that are associated with it.

The object detection node implements Faster R-CNN. This is a neural network architecture that learns image classes and bounding boxes. This allows us to navigate to buoys and coffins for various tasks. A single model will be trained to detect the gate, buoys, and coffins. All buoys will be considered one class, because less training examples will be required to get a better prediction and it is more efficient to have a separate classifier model to classify specific buoys once they are detected.

The downward camera feed is processed by first filtering for the orange color of the path. The the path segment is captured with edge detection. A bounding box is then measured from the detected edges from which a centroid and angle with

respect to the vertical axis can be calculated and the vehicle uses this information to follow the path.

3) Mission Planner: The mission planning stack consists of two major nodes: the task server and the task client. The task server is an actionlib based server that acts as a repository of all possible tasks for the vehicle. It is responsible for the proper setup and shutdown of each task the client requests. The task client is the main way the vehicle interacts with the task server. It can provide the set of tasks to be completed, their parameters, and the interactions between tasks for a successfully completed mission. The task server is also in charge of setting up the remainder of the stacks of the architecture based on its current task. Following the example above, when the Slay Vampires task is started the task server will signal the navigation stack to change from a model-based to dead-reckoning state estimator. The task server will also signal the perception stack to use the Slay Vampires object detection algorithms to find the buoys and choose the correct face.

III. Experimental Results

A general approach to testing was taken that transitions from basic proof of concept to bench test to in-water testing. This approach has been incredibly valuable in catching errors in implementation before the vehicle is in the water. This also maximizes the debugging capabilities early in the process when changes are easiest to make. Our desired amount of in water testing is once per week, which allows us to see the results of any changes made that week. If any issues come up during testing, they can be resolved before moving forward with the sub. We feel that testing once a week is a good balance between progressing with the design and actually validating our work.

Lab bench tests have included the debugging of motor control software, intra-process communications, and basic sanity checks before putting the sub in the water. Old log files from previous competitions have also been used to simulate the competition experience and observe the response of new algorithms. The bench testing setup with the entire assembled sub can be seen in 4.

In addition to bench testing, the team has conducted multiple in-water, pool tests as shown in 5. In-water testing began with testing basics such as strength of the seals of the main and other pressure vessels, and calibrating ballast for the natural buoyancy of the sub. A slight positive buoyancy such that in the case of connection or power failure the sub would ultimately return to the surface was achieved through attaching small diving weights to the bottom of the sub. In addition to these preliminary tests underwater tests were also conducted for control systems.

The testing procedure for the sub control systems was to tune the gains for our depth control and our yaw control separately. This was done by first tuning heading, then tuning depth without heading, then heading without depth to ensure torque from the depth motor was being countered properly.

The control system was tested on the bench and in the water. When the PID control was first implemented on heading and depth control, it was accompanied by two test tasks: one for maintaining heading and one for maintaining depth. Essentially, these tasks took the current submarine heading/depth from the IMU readings, and tried to command them to zero. Through ROS, we were able to see that the commands to fire the motors were indeed being sent and the motor commands would cease as the desired heading/depth were reached by moving the submarine on the bench. Next a similar test was conducted in a swimming pool, with the change being that the overshoot from the un-tuned gains was apparent in the water. The gains were tuned in real-time by utilizing the ROS dynamic reconfigure functionality. The results were accurate control in depth and yaw with minimal overshoot and oscillations. One lesson learned from the initial trial was that heading needed a small dead-band (a small tolerance) so that oscillations would not occur when heading was approximately reached.

Additional testing related to navigation and control includes driving the sub in a box pattern to verify the robustness of the controllers when performing a maneuver in which both

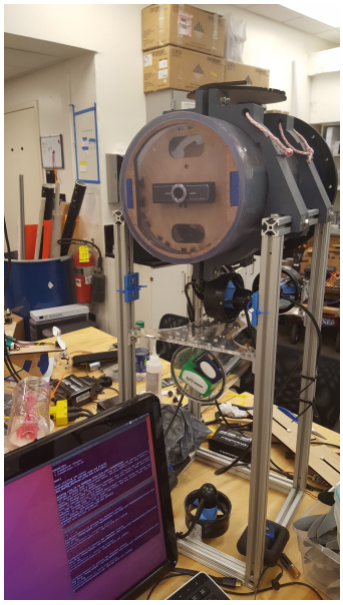


Figure 4: Bench testing setup, allowing hardware in the loop simulation of vehicle performance.

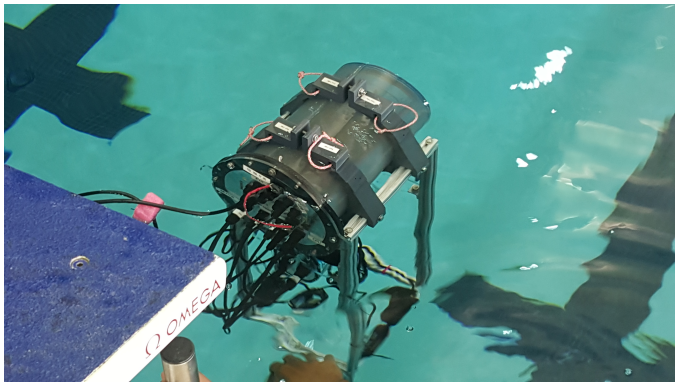
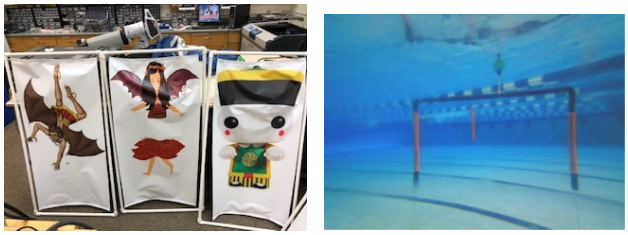


Figure 5: Pool Test

controllers are active. This simultaneously tested the ability to hold heading while maneuvering.

So far, the testing process applied to our vision systems have been proof of concept. The systems that have been validated are gate detection, path detection, buoy classification, and buoy detection. To accomplish these proof of concept tests, the team reconstructed the buoys, gate, and path from the competition, took pictures of them in water, and tested our vision algorithms on them. The buoy and gate props are shown in 6.



(a) Buoys (b) Gate

Figure 6: Props

For object classification, we evaluated the MLP classifier on images of the Vetalas buoy. The neural network was tested both offline and online. Offline, the trained model was validated against a testing set, which provided metrics such as accuracy and precision. The results of the test are shown in 8a. To test online, buoys were placed in the water alongside the sub with a loaded neural network to verify that it is classifying the buoys properly.

Similar to the MLP classifier, a faster-R-CNN model for object detection was tested using a test data set. The metric used for evaluation of the object detection model is mean average precision (mAP). The mAP value resulting from testing the faster-R-CNN model trained on 343 images on a test set of 80 images was 0.986 (mAP is bounded from 0 to 1) [1]. As illustrated in 8b, one lesson learned from the test set was that a method needs to be developed to ignore reflections of buoys.

Initial testing for the "Follow the Path" section consisted of taking pictures with the vehicle of an orange path segment underwater from different angles and distances. The code was then written with the help of those images and once implemented on the vehicle, live testing was done to make adjustments. A sample of the path detection is shown in Figure 7.

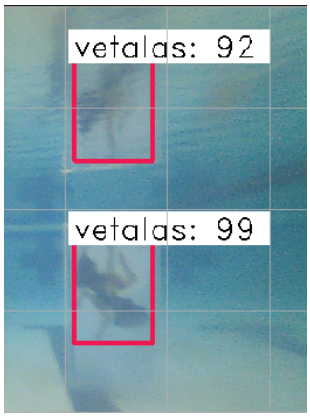
In summary, the results of the current computer vision are positive, with successful edge and color detection for the orange path and successful buoy classification and detection. While all results so far are proof of concept, additional testing will be done in-water in a variety of conditions such as high sun glare, mottled pool floor coloring, and sharp angle of features with respect to the submarine. Additional future testing plans include recreating task scenarios using the props. Also, other neural network architectures will be tested for object classification and detection to try and improve speed and performance.

Figure 7: Path Detection

```
Training results
============================
Confusion Matrix:
 [[317  26   0]
 [  0 479   0]
 [  2   0 287]]
Accuracy:  0.9747974797479748
F1 score:  0.9747974797479747
extracting features

Cross-Validation results
============================
Confusion Matrix:
 [[198  34   3]
 [  8 309   0]
 [  5   0 185]]
Accuracy:  0.9326145552560647
F1 score:  0.9326145552560647
```

(a) MLP Classifier Accuracy



(b) Faster R-CNN Detection

Figure 8: Neural Network results

IV. Conclusion

Our main goal in approaching this year's competition was to develop a platform capable of consistent performance. This took shape through the iteration and improvement of the vehicle hardware and a major rewriting of the platform software. This software rewrite prioritized modularity, simplifying the path to implementing new and improved algorithms in each of the competitions three key areas of perception, navigation, and mission planning. A base level of navigation has been integrated to ensure the vehicle is capable of completing our targeted tasks. New ideas for perception have been integrated into the rest of the autonomy stack and have shown promising results in live testing. After a long year of work improving on past performances, we're excited to see our results at this year's competition!

Acknowledgments

We would like to thank the GT Aerospace Systems Design Lab for supporting our work on this project. Within the ASDL, we would also like to give special thanks to the ADEPT Lab for use of space and manufacturing tools. We would also like to acknowledge the GT Campus Recreation Center

staff for helping us schedule testing times in the diving well and understanding the sometimes frustrating parts of robotics (sorry for the no-shows!). Finally, we would like to thank all the previous team members here at Georgia Tech, your experience and guidance has been invaluable throughout this process.

REFERENCES

[1] "Faster r-cnn (object detection) implemented by keras for custom data from google's open images dataset v4." https://towardsdatascience.com/faster-r-cnn-object-detection-implemented-by-keras-for-custom-data-from-googles-open-images-125f62b9141a. Accessed: 2019-07-08.

APPENDIX A
EXPECTATIONS

Subjective Measures			
	Maximum Points	Expected Points	Points Scored
Utility of team website	50	50	
Technical Merit (from journal paper)	150	150	
Written Style (from journal paper)	50	50	
Capability for Autonomous Behavior	100	100	
Creativity in System Design (static judging)	100	100	
Team Uniform	10	10	
Team Video	50	50	
Pre-Qualifying Video	100	100	
Discretionary Points (static judging)	40	40	
Total	650	650	
Performance Measures			
Weight	See Table 1 / Vehicle		
Gate: Pass Through	100	100	
Gate: Maintain fixed heading	150	150	
Gate: Coin Flip	300	300	
Gate: Pass Through 40% section	400	400	
Gate: Style	+100 (8x max)	800	
Follow the ”Path” (2 total)	100 / segment	200	
Slay Vampires: Any, Called	300, 600	900	
Drop Garlic: Open, Closed	700, 1000 / marker (2 + pickup)	700	
Drop Garlic: Move Arm	400	400	
Expose to Sunlight: Surface in Area	1000	1000	
Finish the mission with T minutes (whole + fractional)	Tx100	1500	

APPENDIX B
COMPONENT SPECIFICATIONS

Component	Vendor	Model/Type	Specs	Cost (if bought new)
Buoyancy Control	Sea Pearls	Vinyl Coated Lace Thru Weights	Miscellaneous sizes, 1-5 lbs.	Varies, up to $30 for a 5 lb. weight
Frame Rails	McMaster-Carr	T-Slotted Framing	Single	$17.68 per 5ft. section
Waterproof Housing	Custom Made	Clear PVC Tube	7” ID	N/A
Waterproof Connectors	Amazon	Generic IP68 Waterproof Connector	2-8 pin	Roughly $3 / connector
Thrusters	Blue Robotics	T200	No ESC	$169
Motor Control	Hobby King	Afro ESC	30 Amp	$11.36
Mid Level Control	Arduino	Mega	N/A	$38.50
Battery (Motors)	Hobby King	Turnigy Multistar 10000 mAh	4S, 10C	$47.64
Battery (Electronics)	Hobby King	Turnigy Multistar 5200 mAh	4S, 10C	$29.04
CPU	Intel	NUC, i7 (discontinued line)	N/A	From $212.83
External Comm Interface	Microhard	VIP2400	N/A	N/A
Programming Language (Navigation)	Python	3.7	N/A	N/A
Programming Language (Vision)	C++ / Python	11 / 3.7	N/A	N/A
Inertial Measurement Unit (IMU)	Lord Microstrain	3DM-GX3-25 (discontinued)	N/A	$1615.00
Camera (forward)	Genius	Widecam F100	120 degree FoV	$49.99
Camera (downward)	ELP	Fisheye Lens 1080p Wide Angle		$45.00
Open source software	ROS	OpenCV	Scikit-learn	
Team Size	15	AE/ME/ECE/CS	Rotated throughout year	Priceless?
HW/SW expertise ratio	1:2	N/A	All team members had basic HW proficiency	N/A
Testing time: simulation	20 hours			
Testing time: benchtop	15 hours			
Testing time: in-water	30 hours			