CTHULHU: The Design and Implementation of the Duke Robotics Club's 2020 RoboSub Competition Entry

Samuel Rabinowitz, Eric Jiang, Nikhil Chakraborty, Trevor Fowler, Muthukurisil Arivoli, Shaan Gondalia, Hyungbin Jun, Eric Chang, David Miron, Estelle He, Maverick Chung, Alan Bi, Kevin Yang, Vincent Wang, Reed Chen, Kara Lindstrom, Jake Heller, Christopher Cameron

***Abstract*— In 2020, the Duke Robotics Club returns to the RoboSub Competition with a vastly improved robot, Cthulhu. Cthulhu is the brainchild of dozens of Duke University engineers who collaborate across three subsystem teams (mechanical, electrical, and software). The new robot features a compact yet feature-filled chassis, a reliable, high-performance electronics stack, and a flexible ROS-based software architecture. Our focus this year was on improving Cthulhu's autonomous capabilities – adding more actuators, stereo vision, custom controls, and computer vision via machine learning – and robust virtual and in-water testing. We are also an active member of the Durham, NC community, mentoring high school FTC and FRC teams.**

I. Competition and Design Strategy

Last year, we built Cthulhu from the ground up with a focus on modularity. This year, we capitalized on that flexibility by extending functionality to complete all of the competition tasks. We also replaced some of our off-the-shelf components with more customizable, in-house solutions. When building these new systems, we always keep future extensibility in mind.

A. Mechanical Subteam Strategy

Our standard mechanical design process consists of whiteboarding, computer-aided design (CAD), finite element analysis (FEA), 3D printing, and finally machining a part with computer numerical control (CNC). But since we already had Cthulhu to build upon, we were able to iterate more quickly, tightly coupling CAD with physical testing, forgoing FEA where unnecessary, and not CNC-ing parts where 3D printing sufficed.

To give Cthulhu the capability to complete the tasks, we created a new claw mechanism and improved the torpedo launcher and marker dropper. Before the pandemic, we had gone through a few iterations of working prototypes of all three actuators, and we also elongated and strengthened the robot's frame to make space for these components. Remotely, we have continued our CAD work, completing pre-mortem analyses on any new parts we design.

B. Electronics Subteam Strategy

The electronics subteam kept with overall goals of modularity, robustness, and reliability. To make our robot more modular, we replaced the off-the-shelf Pixhawk flight controller with our own in-house solution, allowing us to have more sensor choices and connections and more control over the code that runs on the robot. To improve robustness, we expanded Cthulhu's computational power with an NVIDIA Jetson to support new computer vision algorithms and multiple camera feeds. Lastly to improve reliability, we improved the robot's depth perception by adding a second front-facing camera to Cthulhu to enable stereo computer vision that calculates the distance to objects in the vicinity. Working remotely, we have continued developing low-level code, PCBs, and speccing out new connectors.

C. Software Subteam Strategy

The software subteam followed team goals of modularity and extensibility by writing standalone code packages that can be iterated on more quickly in separate teams. Frequent communication kept everything working well together, and frequent testing in the pool with last year's robot starting early in the fall yielded immediate results. Last year's investment into using Docker and Robot Operating System (ROS), two industry standards, continued to pay off by providing a common modular framework and a wealth of premade packages. Since last year was a transitional year for the software subteam, this year the subteam focused on building out more features in-house, such as the controls algorithm, task planning, simulation with physics, and computer vision using machine learning. Moving to remote work has gone swimmingly, testing using our simulation and improving our code architecture and documentation.

II. Vehicle Design: Mechanical System

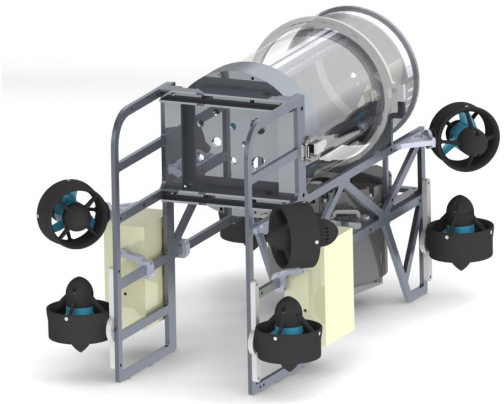

Fig. 1. A rendering of Cthulhu's core frame and elements.

A. Capsule

The capsule is designed to be easily removable. Just pop two latches, remove two thumb screws, and slide it off. We mounted the electronics stack on the robot so that connectors can stay plugged in even as the capsule is removed, allowing us to debug more easily. SolidWorks FEA informed structural changes to support the cantilevered stack, and polycarbonate was used for increased capsule strength. When needed, the whole stack can be removed for extra maintenance by removing only four screws.

B. Frame

One of the most integral mechanical changes was an improvement of robot's structural frame design. Because the robot's frame is not laid out symmetrically around the cylindrical electronics capsule, Cthulhu did not initially float level, but rather pitched forward excessively. Correcting this pitching with the vertical thrusters was possible but was a constant drain on the battery and limited our additional vertical thrust.

To solve this, we redesigned new elongated side frame pieces. We could now mount Cthulhu's heavy Doppler Velocity Logger (DVL) further back, decreasing the horizontal distance between the center of mass and center of volume. In the front, we mounted actuators and buoyant foam. These mounts allowed precise adjustment of the position of the robot's center of volume. The mounts also facilitated wire management for the cables coming out of Cthulhu's frame-mounted capsule end cap.

C. Electrical connections

Underwater thruster connections were sealed not with the traditional potting epoxy but with heat shrink after thorough research revealed its use in Navy submarines and other undersea vessels for increased reliability and handling. [1], [2]

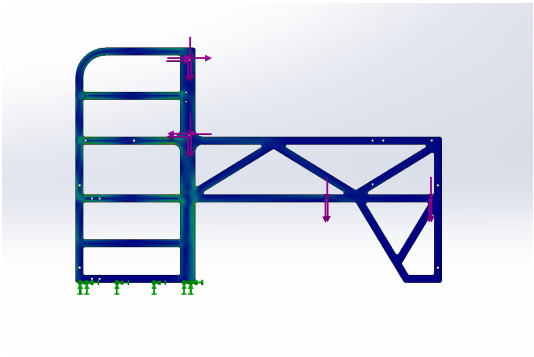

Fig. 2. One of two frame sides undergoing FEA.

D. Actuators

1) Marker Dropper: The marker dropper system was redesigned for Cthulhu this year. It uses two preloaded steel balls, and it works by rotating a cross-shaped part which moves the marker to the output tube one at a time. The system has a detachable cover that allows for easier reloading of markers. When viewed from above, the system takes up much less space due to its tubular shape – this leaves a lot more space for other downward-facing components like the DVL. The system is mounted right next to the downward-facing camera, which greatly improves its accuracy.

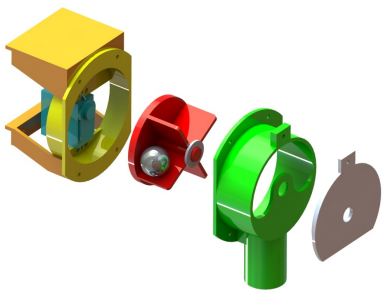

Fig. 3. Exploded view of marker dropper.

2) Torpedo Launchers: Cthulhu's torpedo launchers reduce complexity and failure points by opting for a spring-loaded, servo-controlled design, inspired by BBAUV 3.5's torpedo design [3]. The design features a double barrel setup that allows both spring-loaded

torpedoes to be released via a single servo. A gear attached to the servo moves the latches holding the torpedoes on each side, allowing it to release the torpedoes one at a time. The device mounts parallel to the frame keep its long barrels out of the way. We found that front-weighted torpedoes with one fin travelled the straightest and longest.

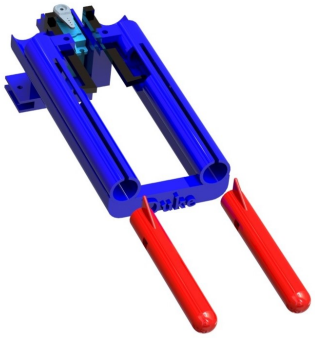

Fig. 4. A rendering of Cthulhu's torpedo launcher. The torpedoes are spring loaded, and each is fired separately via one servo.

3) Claw: The claw is completely new to Cthulhu this year. The 1-DOF claw system allows the robot to grab and move various objects including PVC pipes. A single servo attached to a control link adjusts the open angle of the claw. The mount for the system is extended far upfront so that the claw and the object it grabs do not interfere with other robot elements.

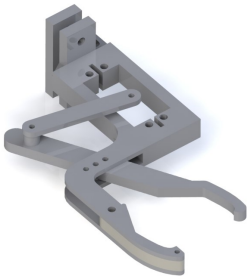

Fig. 5. Cthulhu's 1-DOF claw.

III. Vehicle Design: Electronics System

A. Electronics Stack

The electronics stack within the capsule provides the critical infrastructure that allows the robot's various subsystems to interface effectively. The stack serves as a central hub that routes power and data to and from a suite of various sensors, computing hardware, and thruster and actuator controllers. This year, we improved the internal wire management and are currently working to design PCBs. We also switched Cthulhu's central computer from an Intel NUC to the more GPU-focused NVIDIA Jetson to enable the real-time calculations of our new computer vision algorithms that use machine learning and multiple camera feeds. The Jetson's ARM architecture dissipates less heat and takes up less space.

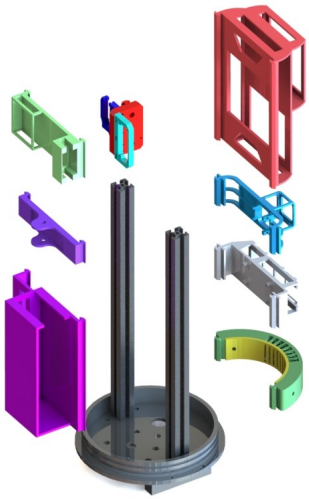

Fig. 6. Exploded view of electronics stack. A customized network of mounts holds all of the electronics, including specific channels for clean wiring.

B. Cameras

This year, we added two front-facing cameras side-by-side for synchronized stereo vision. This allows for better object detection and depth estimation.

C. Acoustics

Cthulhu uses two separate arrays of four omnidirectional hydrophones to locate acoustic pingers. The location algorithm involves two major steps.

First, we use one array obtain an initial guess for the octant in which the pinger is located relative to the robot. The four hydrophones are generously spaced 0.3 meters apart in a right triangular pyramid configuration. We obtain the time differences between the pings in the x, y, and z axes as they reach each hydrophone by using a Butterworth bandpass filter for the desired frequency followed by a cross-correlation. From the time differences, we can derive the octant, which is our guess.

Second, we use our guess and the other hydrophones to precisely locate the pinger. The other array consists

of four miniature hydrophones spaced just millimeters apart in a square. Taking synchronized readings from the hydrophones, Short Time Fourier Transform is used to obtain a magnitude and phase list for each. After several window selections, a final small window is determined based on the stability of phase difference between each hydrophone pair, and from there an average phase difference. Those phase differences and the guess from the first step are input into our Time Difference of Arrival algorithm to find exact horizontal and vertical bearings for the pinger.

D. Microcontroller

We removed the Pixhawk this year in favor of an Arduino and PWM Multiplexer combination and a discrete IMU to split the high-level controls algorithms and the low-level thruster code that moves the robot. Custom controls is more reliable and allows for better system introspection, and the new IMU uses a quaternion-based Kalman filter for real-time accuracy.

IV. Vehicle Design: Software System

A. Controls

Last year, the Pixhawk granted easy stabilization and directional controls, but it was a restrictive black box. Opting to replace it this year with an in-house controls system based on ROS's PID package allowed for custom algorithm improvement and various thruster placements.

Since we wanted the code to be usable for future robots, the controls algorithm is designed to be robot-agnostic, meaning it will work with any placement of any number of thrusters.

We are able to load in configuration files that fully define the n thrusters' placements relative to the robot. We calculate vectors for the torque and unit force each thruster exerts on the robot and gather those into a matrix T, where each column represents a thruster.

$$T = \begin{bmatrix} f_{1x} & \cdots & f_{nx} \\ f_{1y} & \cdots & \cdots \\ f_{1z} & \cdots & \cdots \\ \tau_{1x} & \cdots & \tau_{nx} \\ \tau_{1y} & \cdots & \cdots \\ \tau_{1z} & \cdots & \cdots \end{bmatrix}$$

Given a current and desired position and orientation, our PID loops generate six outputs (one for each degree of freedom), which comprise the vector $\vec{p}$. We solve the equation $\vec{p} = T\vec{t}$ for $\vec{t}$, the amount of power to allocate to each of the n thrusters.

Our algorithm also can operate on a desired local velocity ("power control" instead of "position control").

B. Task Planning

We designed the task planner for Cthulhu with flexibility in mind. We build on top of straightforward tasks, such as movement, to build more complex sequences for the robot to follow autonomously, and ultimately the tasks required for competition (gate, buoy, etc).

C. Simulation

Given that much of our work was done virtually this year, it was particularly important to develop a concrete simulation platform that could serve as an alternative to in-person pool testing. This year, we developed a brand new simulation built upon the physics simulator CoppeliaSim. Our work largely centered upon realistically simulating the underwater environment. In every frame, gravity and linear and rotational drag are applied on the center of gravity, while buoyant force is applied on the center of buoyancy, and thruster forces are applied on the position of each respective thruster relative to the center of gravity. Pseudo-computer vision is used to generate fake bounding boxes based on what the robot would see in a 2D grid. The simulation interfaces with the rest of the CS stack via ROS in the Docker container, taking in the thruster powers from controls.

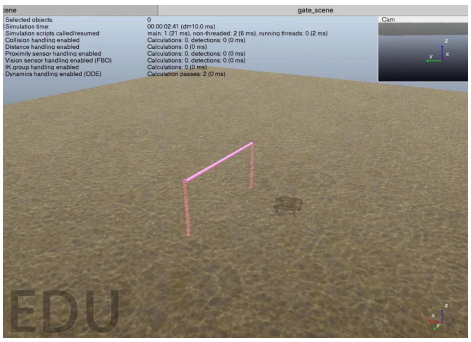

Fig. 7. Custom CoppelliaSim scene with pseudo computer vision.

D. Computer Vision

In 2020, we completely revamped our computer vision, transitioning from inaccurate conventional techniques to a machine-learning based approach. We chose the Faster Regional Convolutional Neural Network (FRCNN) machine learning architecture[4]. This network is made specifically for detecting the location and type of objects in a picture. The underlying weights have already been pre-trained on millions of image classifications and thousands of image segmentations – we grabbed these weights from PyTorch's

model zoo for use on our robot. We then trained these weights further on our own dataset of underwater images. This gave us custom models that could detect buoys, gates, etc. We packaged our data preprocessing and model training process into our own open-source Python library called Detecto[5]. After training our models, we then loaded them into our ROS package, which allows our robot to run bounding box predictions on multiple camera feeds.

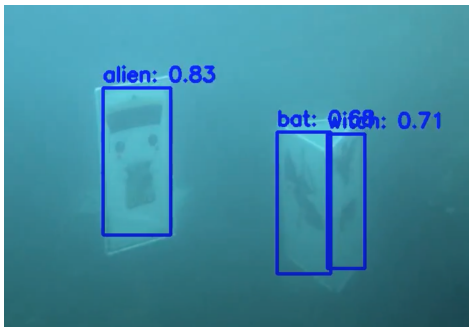

Fig. 8. Computer vision identifying buoys in murky water.

V. Experimental Results

A. Mechanical

The mechanical system was tested thoroughly throughout the construction process. We moved the frame parts and DVL until the robot floated level in the pool. We designed adjustable foam mounts to account for future shifts in the robot's center of buoyancy. The torpedo launcher, marker dropper, and claw were also all pool-tested under varying conditions. It turns out the torpedo can be shot straight even with just one fin (but had to be made more dense to maintain altitude), the marker dropper is accurate, and the claw has sufficient grip strength to lift competition props out of the water. Remotely, the mechanical team was able to continue testing by conducting pre-mortem analyses. We identified the most likely failure points in our actuators and modified the designs in CAD to prevent them. This led us to improve the grip surface on our claw, change the cover of the marker dropper, and alter the structure of the torpedo launcher. We will test the changes we made in-person when possible.

B. Electronics

Given that the electronics team's primary goal was to further improve modularity rather than implement entirely new features, testing focused on ensuring that new additions improved performance. We found that replacing the Pixhawk ultimately paid dividends, allowing for very granular control of the robot. The newly-added NVIDIA Jetson proved powerful enough to handle computer vision calculations with real-time video streams. Finally, all full-system pool tests showed that these major component changes did not break existing functionality.

C. Software

Controls and task planning were tested extensively. Through weekly pool tests, we fine-tuned our custom controls algorithm and ensured that static behavior and basic tasks (up/down, left/right, etc.) work. Once we started working remotely, we switched to a custom simulation environment that mimicked the water conditions and forces. Our virtual tests strongly support that, under ideal conditions, our robot is able to stabilize indefinitely on any translational position and reliably move to a specified global coordinate.

For computer vision, we used real footage of the gates/buoys from RoboSub 2019 to train and test our machine learning model. We were able to achieve 95% classification accuracy, with very accurate bounding boxes and rare false positives. Furthermore, the model was able to output bounding box predictions in near real time on a 1 Hz dummy image ROS topic.

D. Lessons Learned

- **Have multiple modes of testing.** Working remotely has shown our strength in using simulated and analytical testing methods when a pool is inaccessible.
- **Communicate and delegate work.** Delays were frequently not engineering-related, but rather due to miscommunication. Real progress was made when we worked as a cohesive unit.
- **Promote knowledge transfer.** Having just a few people understand certain subsystems has been a failure point in past years. Empowering more members to make an impact benefits the team overall.

Acknowledgments

Duke Robotics Club is housed within Duke University's Pratt School of Engineering. We gratefully acknowledge the Duke faculty and administrative staff who have helped and continue to help make the club a success. Special thanks to Professor Michael Zavlanos, Pratt's Jennifer Ganley, and the Engineering Alumni Council, especially the club's liaison, Christopher Rowland. Lastly, we owe our continued success to our long-time sponsors: The Lord Foundation, Duke Student Government, General Motors, and SolidWorks.

REFERENCES

[1] K. Brezinski and E. Taylor, "MIL-STD-461/MIL-STD-704 Investigation," SAE Technical Paper Series, 1993.

[2] United States. Department of the Navy. Naval Sea Systems Command. *Updates to MIL-STD-2003*. By Christopher Nemarich. March 9, 2017. https://nsrp.org/wp-content/uploads/2017/03/04-Updates-to-MIL-STD-2003-NSRP-9-Mar-17-Final.pdf. (accessed July 2, 2019).

[3] Goh, E., 2018. Design And Implementation Of BBAUV 3.5. [ebook] Robonation, p.5. https://robonation.org/app/uploads/sites/4/2019/10/NUS_RS18_TDR-min.pdf (accessed August 9, 2020).

[4] S. Ren, K. He, R. Girshick, J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *NeurIPS*, Dec. 2015, arXiv:1506.01497v3.

[5] A. Bi. "Welcome to Detecto's documentation!". ReadTheDocs.io. https://detecto.readthedocs.io/en/latest/ (accessed August 8, 2020).

Appendix A: Component Specifications

Component	Vendor	Model/Type & Specs	Cost (if new)
Buoyancy Control		Ridgid Foam Blocks	
Frame	8021 Aluminum, custom		∼$500
Waterproof Housing	Polycarbonate, custom		
Waterproof Connectors	Subconn + Seacon	Wet-mate Connectors	
Thrusters	Blue Robotics	T200-Thruster-R1-RP	
Motor Control	Blue Robotics	Basic ESC	
High Level Control	Arduino	Nano w/ PWM Multiplexer	$35
Actuators	Hitec	D646WP Waterproof Servos	
Propellers	Blue Robotics	Included w/ T200 Thrusters	
Battery	Turnigy	HC 5S 12C 16000mAh Lipo	
Converter	Kohree	DC/DC 36V/12V	
Regulator	N/A		
CPU	NVIDIA	Jetson TX2	Robosub 2019
Internal Comm Network	TP-Link	5 Port Gigabit PoE Switch	
External Comm Interface	NETGEAR	Nighthawk R7000	$143.75
Programming Language 1	Python	2.7	
Programming Language 2	C++ + Lua		
Compass	Built into IMU + DVL		
IMU	VectorNav	VN-100	
Doppler Velocity Log (DVL)	Teledyne	Workhorse Navigator 1200	
Camera(s)	Edmund Optics RMA Electronics	Allied Vision Mako G-234C Tamron M112EM08	
Hydrophones	Teledyne	TC4013 omnidirectional	4 x ∼$1000
Manipulator		3D-Printed ABS Plastic	
Algorithms: vision		Machine Learning: Resnet 50 Architecture based on Faster RCNN	
Algorithms: acoustics		Butterworth Filter Cross Correlation Time Difference of Arrival Short Time Fourier Transform	
Algorithms: localization and mapping		Extended Kalman Filter SLAM w/stereo cameras	
Algorithms: autonomy		In-house task planner	
Open source software	Docker + ROS	ROS Melodic	
Team size (number of people)		32	
HW/SW expertise ratio		40/60 (13 HW, 19 SW)	
Testing time: simulation		60 hours	
Testing time: in-water		30 hours	

Appendix B: Community Outreach

Mentoring: The Duke Robotics Club recognizes the importance of working with the local community and inspiring future generations of STEM students. Over the years, we have worked with both local middle schoolers and high schoolers, often on robotics teams, to do so.

In 2018-2019, members of the Duke Robotics Club mentored a brand new team comprised of local Durham high school students, guiding them in creating a robot for the FIRST Tech Challenge, and also helping to bring them to the finals of the regional competition for the FIRST Robotics Competition. The members joined this FRC Team 6426 Robo Gladiators multiple times per week to help them strategize, build, and program their robot. Advancing to the finals of the regional competition as a rookie team, they outperformed many veteran teams. Looking forward to next season, the Robo Gladiators hope to extend their successes and become national champions, and the Duke Robotics Club looks forward to once again helping them succeed.

In 2019-2020, we continued the mentorship program with the same team. Despite the season getting canceled early due to the pandemic, it was still an amazing experience for all involved, and the team did get to attend their first competition.

Even though each member's time could have been spent improving Cthulhu, we acknowledge how much more robotics can advance with each class of students. The Duke Robotics Club wants to encourage as much innovation as possible, and we are proud to be able to spark ideas in generations of students to come.

Fig. 9. The local high schoolers' FIRST Robotics Competition team, mentored by two Duke Robotics Club members, after reaching the regional finals in their pilot year.

Presenting: In January 2020, two of our leaders presented at local school Bethesda Christian Academy's science week. We were honored to be the conclusion to the 160 elementary and middle school students' week-long dive into different STEM topics. We introduced them to the field of robotics, showed them various examples of robots all over different industries, told them about our club, and ended with ways they can get started with robotics and STEM now. There was a great Q&A after the presentation. They had so many great questions about different robots and tips they could use to learn more.

Fig. 10. Two of our leaders presenting at Bethesda Christian Academy's science week.

Within Duke: The Duke Robotics Club has also spread robotics and STEM through outreach both within Duke's Pratt School of Engineering and the university as a whole. We have run a Bot Battle competition to allow students of all disciplines to give robotics a try. We host information sessions at the beginning of each season, open to the entire university, attracting over 100 students each time. We have reached out to interest and affinity student groups on campus within STEM fields to coordinate cross-promotion and facilitate activities between members of related clubs.

Lastly, this year we developed a brand new intro project (forming into small teams to make Raspberry Pi-powered mini bots) to make our club more accessible to students of all skill sets and backgrounds, provide free basic skill training, and create a space for new freshman engineers to meet each other. The project was extremely well received, with over 70 students turning out initially, and about 30 stayed in the club well after the project ended with increased confidence about their abilities.