Michigan Robotic Submarine: Huron

Alexander Steinig, Kobi Wettstein, Thomas Brunner, Kathryn Wakevainen, Emi Yuki,
David Reihl, Andrew Huston, Ben Manley

***Abstract*—Michigan Robotic Submarine is a student project team at the University of Michigan entering its first year in the RoboSub competition. Our autonomous underwater vehicle, named Huron, is characterized by a simple design with a large free surface area to allow for modularity in mounting additional components. It was designed to have rapid manufacturing and testing capabilities in addition to flexibility in incorporating new systems. We identified three tasks to focus on – the buoy task, gate task, and torpedoes task – which allowed us to lay the groundwork for attempting more challenging tasks in the future. The mechanical team was responsible for designing the frame, torpedo launcher system, and electronics mounting components. The software team designed and implemented a system architecture that included navigational programs, object detection software, and acoustic signal processing. With the limited accessibility of facilities during the COVID-19 pandemic, some in-water testing was conducted in the Marine Hydrodynamics Lab, supplemented with simulation to prepare our AUV for the course.**

I. COMPETITION STRATEGY

Our Autonomous Underwater Vehicle (AUV) is equipped and has been designed to complete three competition tasks: passing through the gate, bumping into the buoy, and shooting torpedoes. We decided to focus on these tasks since they can be completed primarily using visual identification through computer vision (CV) and a hydrophone system, allowing us to focus our efforts on these two primary sensor systems. We are also in the midst of a two-year design cycle to develop a gripper mechanism for use in future competitions.

In order to allocate more time to testing the AUV's algorithms, managing complexity in the mechanical systems was essential. Unfortunately, after we had designed and manufactured the carbon fiber hull, COVID-19 shutdowns prevented our team from completing its construction. Given our limited ability to manufacture in-person, we reevaluated our team goals and decided to shift focus to an alternative design with an emphasis on ease of manufacturing and assembly.

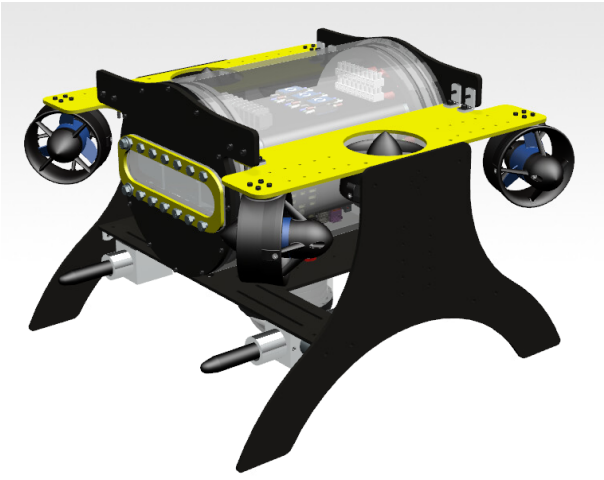

Fig. 1: CAD rendering of Huron

The software team developed our software architecture design, initial navigational, and object detection algorithms remotely. As with the mechanical system, our Robot Operating System (ROS) - based architecture was developed to be expandable, as well as quick to deploy and debug. The software team utilized simulation and a BlueROV as an initial testbed, while the mechanical team completed the construction of Huron. In order to enable our software team to focus on developing our camera and hydrophone systems, we elected to use many off-the-shelf electrical components similar to those in the BlueROV to retain some continuity when switching to Huron.

II. Design Creativity

A. Mechanical

1) Frame: Limitations in our ability to manufacture in person due to COVID-19 restrictions played a large role in how we approached the design of Huron's frame. To reduce our manufacturing load, we prioritized simplicity, while still allowing the frame to be expandable for future mechanical systems. With these priorities in mind, we designed a frame consisting of several 0.16" thick 6061 aluminum plates, which serve as the AUV's primary structural components. The frame contains sufficient mounting space for every other necessary piece of hardware, including electronics, torpedo launchers, thrusters, a bottom facing camera, and hydrophone. The design allows for the addition of other systems as well as weights and flotation devices to manage the AUV's hydrodynamic stability. Because each plate is two dimensional, only a waterjet was needed to manufacture these parts. As a result, our entire frame needed only a few hours to be manufactured, considerable time manufacturing in person. We also developed a 3D-printed electronics chassis within an acrylic, watertight tube to provide mounting points for every piece of electrical hardware. This tube interfaces with the front and rear plates of Huron via dual o-rings.

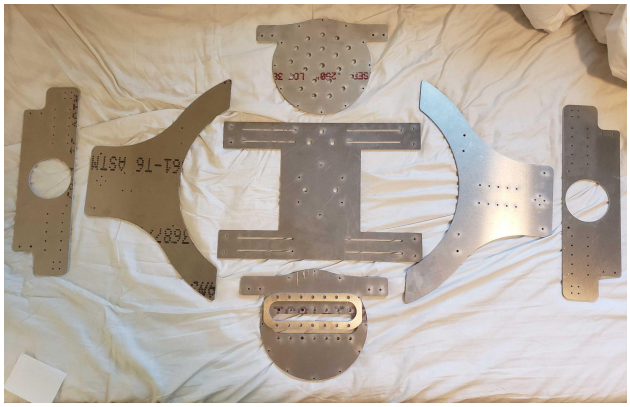

Fig. 2: View of disassembled frame

2) Electronics Chassis: Our desire to make the electronics easily accessible guided the design of our 3D-printed electronics chassis. The front plate can simply be removed and the electronics chassis can be slid off of four rods to access the internal components, as shown in Figure 3 along with the Zed 2 stereo camera and 3D-printed camera mount. The mounting of the electronics on each of four exterior faces of the chassis allows these systems to meet the dimensional constraints defined by the size of the acrylic tube without needing to create stacks of electronics which make them less accessible. The large surface area of the chassis also enables the incorporation of additional electrical systems in the future.

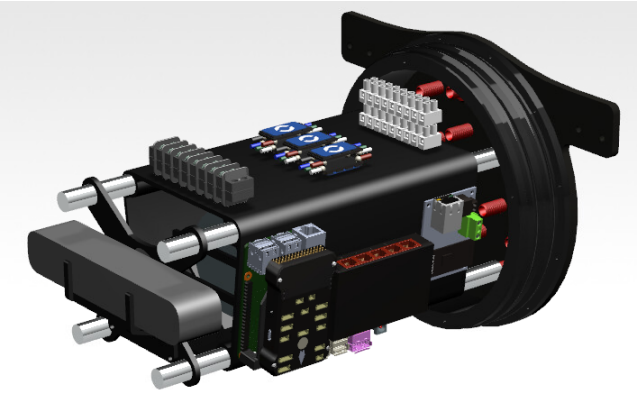

Fig. 3: CAD rendering of electronics chassis

To efficiently use the available space within the acrylic tube, we decided to place the battery inside the chassis. We designed two interior supports which serve to constrain lateral movement of the battery. Two removable 3D-printed supports also prevent the battery from moving forwards or backwards. Figure 4 illustrates the locations of these supports. Because the battery has a significant weight, ensuring that it is properly constrained helps prevent undesirable pitching during operation.

Fig. 4: CAD rendering of interior of electronics chassis

3) Torpedo Launchers: We opted for a spring-actuated torpedo launcher rather than a pneumatically-controlled mechanism, because springs provide greater reliability and simplicity. The release mechanism primarily consists of a solenoid, the plunger of which holds the torpedo in place against the compressed spring via a notch in the torpedo until it is ready to be fired. We also wanted to design this mechanism to be easily accessible and easy to use. Because the notch covers the entire circumference of the torpedo, each torpedo can be loaded in any radial orientation. A simple part that interfaces the spring with the torpedo also ensures that it is centered in the barrel.

After completing the initial design, we made several design decisions with the intention of improving the manufacturability of the torpedo launchers. This involved making the barrel and solenoid mounting cube separate components, each of which were significantly easier to manufacture than they would have been in the original design. We also replaced the end of the barrel with a snap ring to make the inside of the barrel accessible from more than one side. This eliminated the need to use an unrealistically long grooving tool to manufacture grooves located at the end of the barrel. Because the solenoid can be exposed to water, the second design iteration did not incorporate a housing for the solenoid. This new system design makes the launcher simpler and easier to manufacture. The top image of Figure 5 shows an exploded drawing of the original torpedo launcher design, while the bottom image is an exploded drawing of the second design iteration of the launcher.

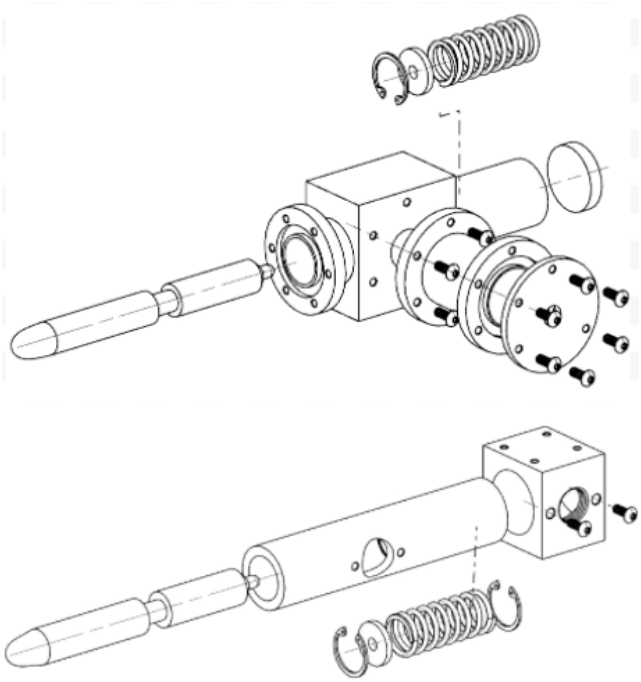

Fig. 5: Exploded view of initial torpedo launcher (top) and modified torpedo launcher (bottom)

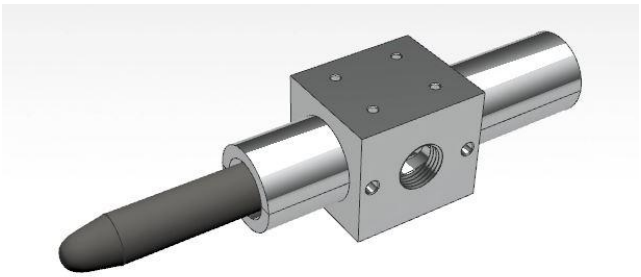

Fig. 6: CAD rendering of final torpedo launcher version

4) Gripper: Due to the inherently complex nature of a gripper mechanism, our priorities were that the end effectors could move under the power of a single motor, and that the gripper would have a high reliability. After researching several options, we decided that linear actuation would best fulfill those priorities. Linear actuators contain relatively few parts compared to other types of actuators, and allow us to use a single motor to move both end effectors. A CAD rendering of our gripper mechanism is shown in Figure 7.

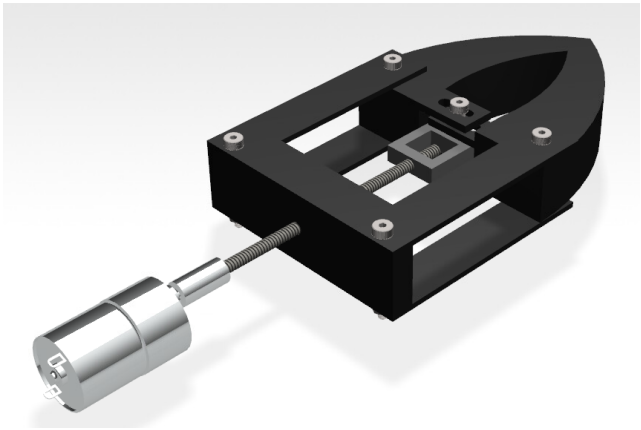

Fig. 7: CAD rendering of gripper

B. Software

1) Architecture: We developed our software architecture with flexibility, expandability, and ease of debugging in mind. To meet these requirements, we chose to build our architecture on the ROS framework which provides a relatively simple interface for multiple programs to run and communicate together at once. We began with a central algorithm, dubbed "captain", to start and monitor nodes based on a set task list. Each competition task has its own node which utilizes our various support nodes as necessary. These support nodes include our object detection and distance node, which use our image processing (further described in section 3) and our stereo camera. Another one of our support nodes, “centering and depth”, is utilized by our task nodes to maintain a designated depth and center the AUV on an object detection bounding box using tuned Proportional Integral Derivative (PID) control. Additionally, an error handling node monitors critical thresholds, such as depth being exceeded, leaks, and inter-node communication loss. A critical error results in an abort and the AUV surfacing.

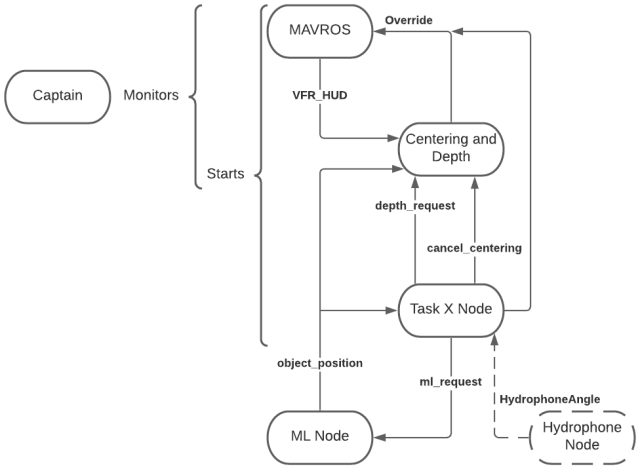

Fig. 8: Top-level ROS architecture with major connections (error handling node not shown). MAVROS is the node on our PixHawk flight controller.

2) Navigation: Given that the gate and buoy tasks were within visual distance of their respective starting points, we opted for a simple approach that could be improved upon in the future. We decided to avoid the complexity of absolute localization with respect to the environment. Instead, we used our computer vision input to constantly reassess relative position to complete and navigate between these tasks. Additionally, for the torpedo task, we developed our hydrophone system to allow us to acoustically navigate by calculating the desired heading using an arrangement of three hydrophones. This approach to navigation allowed us to focus our development primarily on developing our computer vision and hydrophones system. Those systems will be explained in more detail in the following sections.

3) Image Processing and Object Detection: To implement real-time object detection on our AUV, we needed to upgrade our system by adding an additional computer and camera. After researching different computers, we added a Nvidia Jetson Nano to our system to utilize its GPU for image processing. This resulted in a greater than 3 times increase in frame rate when compared to running the image processing on our Raspberry Pi – a necessary improvement to enable reliance on our object detection with a moving AUV. For the camera, we integrated a Zed 2 stereo camera into our system. We selected a stereo camera to be able to estimate object distance from the vehicle.

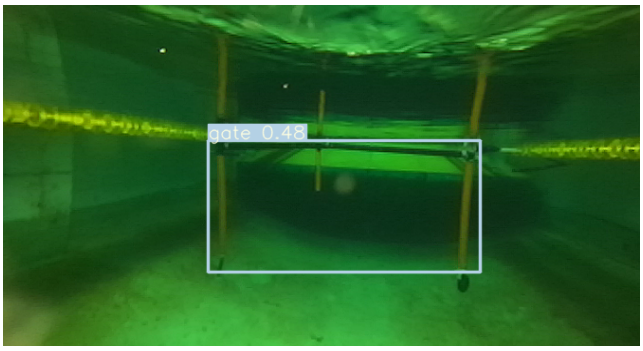

Fig. 9: Bounding box drawn around the gate as detected by our model

For object detection, we decided to use a convolutional neural network because team members had experience in this area. After researching different frameworks, we determined that an off-the-shelf, pruned version of a YOLOv3 (You Only Look Once, version 3) model was ideal [1]. Due to the pruning (removal of the least important weights from the model), this version allowed us to detect objects at a sufficient rate, while still being large enough to handle the number of classes required for the competition. We ran this model using the OpenCV library to process an average of 8 frames per second. If an object is detected, a bounding box is drawn around it, and that information is sent to our primary control algorithm.

To train the model, we used images of objects that we wanted to detect (e.g. the gate) that we collected and labeled. These images were chosen in an attempt to capture the object in a variety of angles and lighting, allowing the model to recognize it from as many positions as possible. We updated the model through an iterative process as new training data was accumulated, giving the model a better understanding of what an image of the object looks like each time.

To further improve the generalizability of our model, we explored using an algorithm that adjusted the white balance of each image [2]. This algorithm reduced the effects of more extreme colors, which is ideal for our primary testing environment that provides predominantly green images. We tested training our model on the white balanced images as well as white balancing each frame during our real-time object detection. Our results showed an insignificant average increase in object detection confidence (4%) with a significant decrease in our framerate (50%). Because the cost currently outweighs the benefit, we do not utilize white balancing in our primary algorithm. Despite this, we plan to further investigate similar algorithms that may achieve better results.

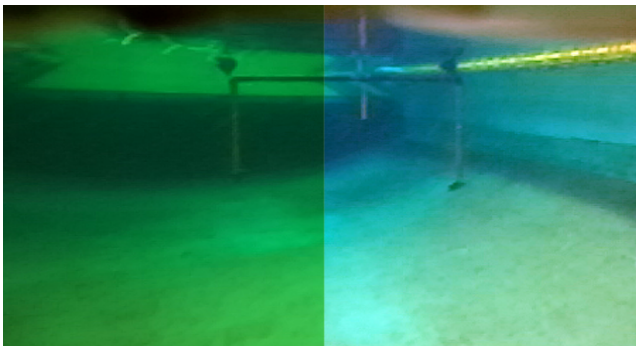

Fig. 10: Original image (left) compared to a white-balanced image (right) in a predominately green environment

4) Hydrophone System - Algorithm: In order to find the acoustic pingers that locate the torpedo task, we developed our hydrophone system. The system utilizes three omnidirectional hydrophones mounted in a triangle configuration on the bottom plate of Huron. Our algorithms relied on a difference in time of flight to determine the ping direction. A full diagram of our system, which will be discussed in the next two sections, can be found in Figure C-1.

When comparing two acoustic signals, the easiest way to do this comparison “on paper” is to identify their phase offset (assuming it is less than 360 degrees). However, in practice, identifying the exact points of a sine wave and where it starts is difficult with data collected from a sensor. Therefore, we instead subtracted one signal from another and took the root mean square (RMS) of the difference. When the two signals perfectly align, this creates a minimum value whereas a 180 degree offset creates a maximum value. We chose to use the RMS as it provides a good average for analog, periodic signals. Additionally, MATLAB and Simulink simulation showed a mostly linear relationship between both delay time vs RMS and RMS vs offset angle (See Appendix D). A linear relationship is desired here as we can easily map the input to a unique output, allowing us to have confidence in our mapping from RMS to offset angle.

The offset angle and the geometry of our hydrophone's position gave us two possible headings for each hydrophone. Comparing these headings with the other pairs of hydrophone readings, we established the correct heading. We then used a PID algorithm to maneuver and approach this heading.

5) Hydrophone System - Filtering: We designed a custom digital and analog signal processing chain to ensure our algorithm receives quality data. The signal from each hydrophone is passed through our custom printed circuit board (PCB), which cleans the signal and digitizes it. The PCB includes an amplifier, first order low and high pass filters, buffers, and a 16-bit analog to digital converter (ADC). The signal is then communicated over the digital Inter-IC-Sound (I2S) protocol from the ADC to our main computer.

Once the signal is received on the main computer, it is passed through a tight, variable-adjustable band pass filter. From there, the signal is tested and optimized based on testing with other teams' data acquired through the data sharing program. Next, we either apply a Fourier transform for debugging purposes or start our calculation process to calculate the heading of the acoustic pinger.

6) Simulation: With our limited access to pools during the pandemic, we began to look into simulation as an alternative to in-person pool testing. After some initial research, we started with an open source Gazebo simulation and some libraries to connect it to ArduSub and our software over ROS. As a robotics simulator, Gazebo accurately responded to our programmed movements. For simulating computer vision, we experimented with Blender, a modeling and rendering tool, to represent our pool environment and feed images to our vision program.

One of the greatest challenges with our simulation utility was accessibility for team members as the entire package needed to run in an Ubuntu environment with ROS. With the processing requirements of Gazebo, running the simulation in a virtual machine proved too slow to be useful if not impossible. On Windows, we investigated using Windows Subsystem for Linux (WSL), especially with the release of WSL2. This worked better than a virtual machine but was still difficult to set up and did not support Blender. The best solution was a pure Ubuntu desktop environment, but we wanted to avoid installing a new operating system on all our members' computers. The solution was to use Amazon Web Services (AWS) and host the simulation in the cloud. Instances could be spun up in less than five minutes and any member could access the simulation environment over Secure Shell Protocol (SSH) and an Xserver for graphics forwarding. For some computers, Xservers were not an easy option so we also added a Virtual Network Computing (VNC) server client system and a minimal desktop to our AWS simulation environment so that anyone could access them.

III. EXPERIMENTAL RESULTS

The COVID-19 pandemic forced us to modify some of our testing procedures in order to facilitate social distancing practices. The limited accessibility of facilities also reduced the volume of testing that we were able to achieve. To account for this, we simplified the design of our AUV and relied on testing our software in simulation. Most in-water testing for Huron was conducted in the University of Michigan's Marine Hydrodynamics Lab (MHL).

1) Leak Testing: Before conducting in-water tests of the full vehicle in the MHL, we employed a vacuum pump to test for leaks. Even if this test did not show the presence of a leak, we still submerged the AUV in a sink in the Wilson Student Team Project Center to verify the results of the vacuum pump test. During in-water testing, we relied on our leak sensor to alert us of

an unforeseen leak and placed a piece of paper inside of the acrylic enclosure to determine where water was entering the enclosure. If a leak was detected, the leaking area would then be fixed and the enclosure would be tested again for water tightness.

2) Torpedo Launcher: Because the aiming system has not been completed due to challenges related to COVID-19 and prioritization of the gate and buoy tasks, tests of the torpedo launcher focused on the release mechanism. The barrel of the launcher and torpedo itself were initially 3D-printed to validate the functionality of the design. Due to the force created by the spring decompressing, the final launcher will be machined out of aluminum to make it sturdy while keeping it relatively lightweight. Machine shop closures prevented this version of the launcher from being completed. After finishing the manufacturing of the launcher, we plan to test it underwater to measure launch velocity, launch trajectory, and travel distance.

3) Gripper: As the gripper is still in the design phase, we have not yet performed any experimental tests. In the future, we plan to 3D-print its end effectors in order to verify that they open and close expectedly before incorporating a linear actuator.

4) *In-water Testing and Simulation:* In-water testing was conducted at the MHL with social distancing protocols in place. We connected to the AUV via an ethernet tether to enable us to control the AUV and make quick changes to our code as needed. In order to capitalize on the limited time we had, we formalized test plans that detailed each test we aimed to complete along with the procedure and expected results in advance of each session. Additionally, we created test-specific programs to systematically and consistently evaluate components of our software and to simulate or "mock" parts of our software that were still in development. For example, our mock machine learning node used our compass and dead reckoning estimates to place a virtual gate at a consistent location in our pool. To ensure we did not waste time debugging these programs, we ran them in our simulation environment prior to our in-person testing to verify that they would behave as expected.

5) *Object Detection Model Development*: The development of our object detection model was delayed until the latter part of the year due to access restrictions to our testing environment. The inability to test led to a lack of training data to begin training our model. Once we were able to begin testing, we saved images from the camera on the AUV to craft our training and testing datasets. During each successive training session, we captured additional images to add to our model-training suite.

One of the main goals of our model was to make it generally applicable. That is, we wanted the model to detect objects in a variety of locations and conditions. Thus, we made it a point to gather images outside of our primary training facility so we could diversify our datasets. After training our first model, we found that it accurately detected objects in our primary testing facility with high confidence, but it was not generalizable to other environments. There was a large variance in confidence, making it appear that Model 1 was overfitted to the training data (which largely consisted of images from our primary testing facility). We expanded our training dataset for the model iterations that followed by capturing a variety of additional images, resulting in significant improvements in model generalizability. The increase in generalizability is reflected in a 20% median

confidence gain, and our full results can be found in Appendix E.

6) *Hydrophone:* As a critical component to completing our targeted torpedo task, we knew that testing our hydrophone system would be vital. However, due to shipping and manufacturing delays, chip shortages, and COVID-19 restrictions on our facilities, we faced significant delays in developing our custom PCBs. With limited test time available, we documented plans and results for continuity testing and testing of the digital interfaces. In the future, we will use our oscilloscope and signal generator to test our analog components.

IV. CONCLUSION

Despite a year of challenges, Michigan Robotic Submarine has worked hard to design, build, and test an effective AUV. Our mechanical team designed our chassis to be modular and easy to manufacture. The software team took a similar approach and developed a robust and flexible architecture ready for future expansion. Additionally, the software team iterated through five machine learning models to optimize for diverse environments. With this development and research, we have set the framework for superior iterations of our AUV.

ACKNOWLEDGEMENTS

The Michigan Robotic Submarine team would like to thank our sponsors for their support: University of Michigan College of Engineering, Keysight, Parker Hannifin Pneumatic, Boeing, University of Michigan CSE Department, General Atomics, Molex, Wayfair, Ford, Raytheon, UTS, GIGAVAC, Beagle Board, Ewert Energy Systems, and Alro. In addition, we would like to thank our advisor Dr. Katie Skinner.

We are greatly appreciative of the Marine Hydrodynamics Lab and Wilson Student Team Project Center facilities for hosting our team and supporting our endeavors.

We are also thankful for hydrophone data provided through the new RoboNation data sharing program.

REFERENCES

[1] Joseph Redmon and Ali Farhadi, "YOLOv3: An Incremental Improvement," arXiv:1804.02767 [cs.CV], 2018.

[2] Andrew Awad, Ben Manley, Matthew Shannon, Michael Yufa-Zimilevich, "Feature and Weight Pruning in YOLOv3," Not Published, 2021, https://drive.google.com/file/d/1lhxxnq7U1FCWXwVz6KLV6QWGGYlDjKWF/view?usp=sharing.

Appendix A: Component Specifications

Component	Vendor	Model/Type	Specs	Cost (if new)	Status
Buoyancy Control	N/A	N/A	N/A	N/A	N/A
Frame	Custom	Custom	6061 Aluminum	$432.00	Installed
Waterproof Housing	Blue Robotics	8” Series	11.75” Cast Acrylic	$164.00	Installed
Cable Penetrators	Blue Robotics	Potted cable penetrators	25mm long M10x1.5 for 6mm cable	$100.00	Installed
Thrusters	Blue Robotics	T200 w/ Propellor	7-20V	$1074.00	Installed
Motor Control	Blue Robotics	Basic ESC	7-26V	$172.00	Installed
High Level Control	Custom	Captain	ROS-based	N/A	Installed
Actuators	DigiKey	Tubular Solenoid STA Pull 1x1	24V DC 6.2W	$30.60	Purchased
Battery	Blue Robotics	Lithium-ion Battery	14.8V, 18Ah	$350.00	Installed
Converter	N/A	N/A	N/A	N/A	N/A
Regulator	Blue Robotics	Power Supply	5V 6A	$22.00	
CPU	Nvidia	Jetson Nano	Quad-core ARM A57 @ 1.43 GHz, 4 Gb RAM	$110.00	Installed
	RaspberryPi	Raspberry Pi 3B	1.4GHz 64-bit quad-core, 1Gb RAM	$35.00	Installed
Internal Comm Network	Open Robotics	ROS Kinetic		N/A	Installed
External Comm Interface	-	Ethernet	-	-	Installed
Compass	PixHawk	PX4	Accel/Gyro: ICM-20689 with Magnetometer	$257.00	Installed'
Inertial Measurement Unit (IMU)	PixHawk	PX4	MPU6000 9-axis	(see above)	Installed

Doppler Velocity Log (DVL)	N/A	N/A	N/A	N/A	N/A
Vision	Stereolabs	ZED2	stereo vision, ROS, depth detection	$449.00	Installed
Acoustics	Aquarian Audio	H1C Hydrophone	omnidirectional, no amplifier	$139.00	Purchased
Algorithms: vision		miniYOLOv3			Installed
Algorithms: acoustics		custom (see Hydrophone - algorithms section)			Installed
Algorithms: localization and mapping	N/A	N/A	N/A	N/A	N/A
Algorithms: autonomy		Custom			
Open source software		OpenCV			
Team Size (number of people)	24				
Expertise ratio (hardware vs. software)	6 mechanical, 13 software, 3 media, 2 business				
Testing time: simulation	20 hours				
Testing time: in-water	40 hours				
Inter-vehicle communication	N/A				
Programming Language(s)	Python				

Table. A-1 Components

Appendix B: Outreach Activities

Michigan Robotic Submarine planned to assist the University of Michigan's Department of Naval Architecture and Marine Engineering (NAME) with their annual SeaPerch competition for middle and high schoolers in the greater Ann Arbor area. However, this in-person event was canceled due to the COVID-19 pandemic.

In the future, we plan to assist with this event as well as other outreach events through the NAME department.

Additionally, we participated in the RoboNation Data Sharing committee with one of our members joining the initial group. Through piloting the program, we provided feedback and input on the program and how it would provide the best value to teams.

Appendix C:

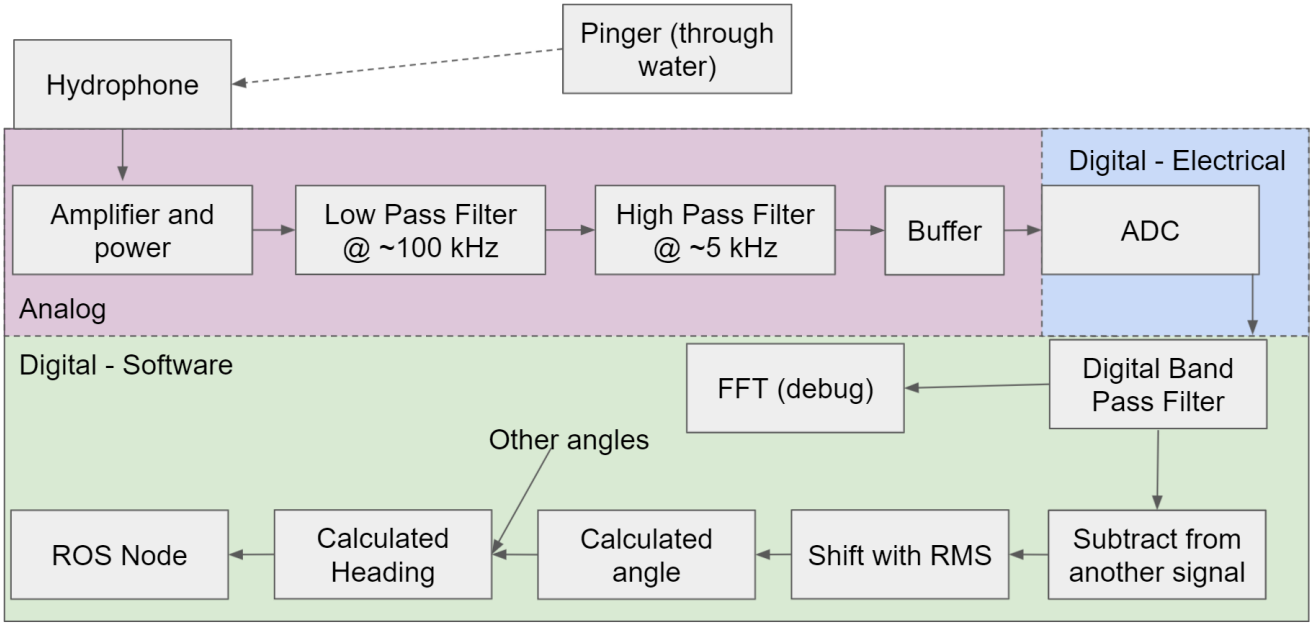

Fig. C-1 Hydrophone digital and analog signal processing chain

Appendix D: Hydrophone Simulation Results

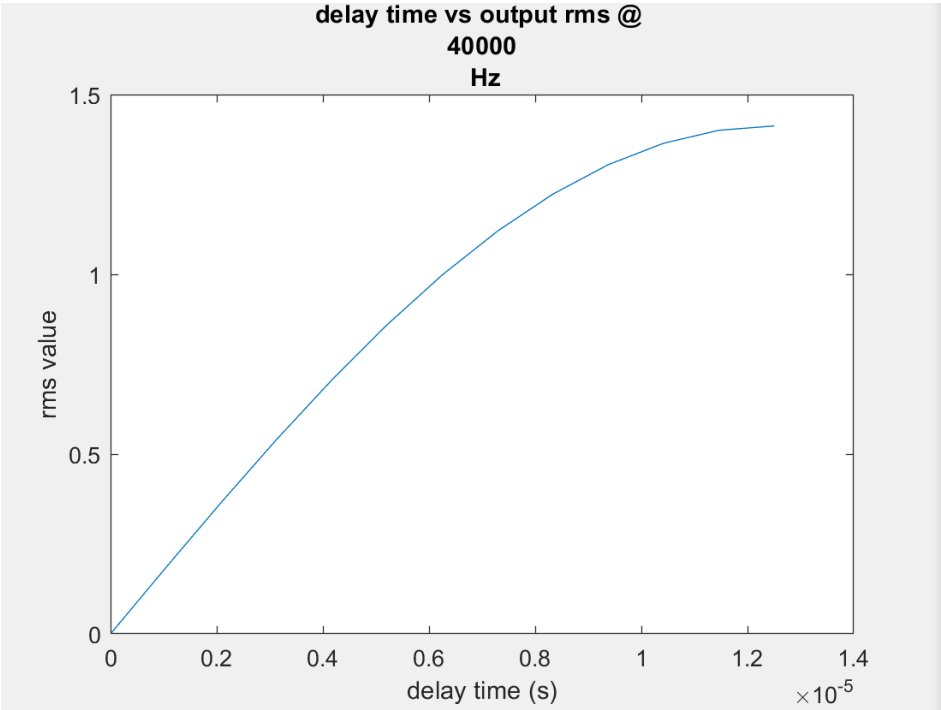

Fig. D-1 Simulation results calculating corresponding RMS value of delay

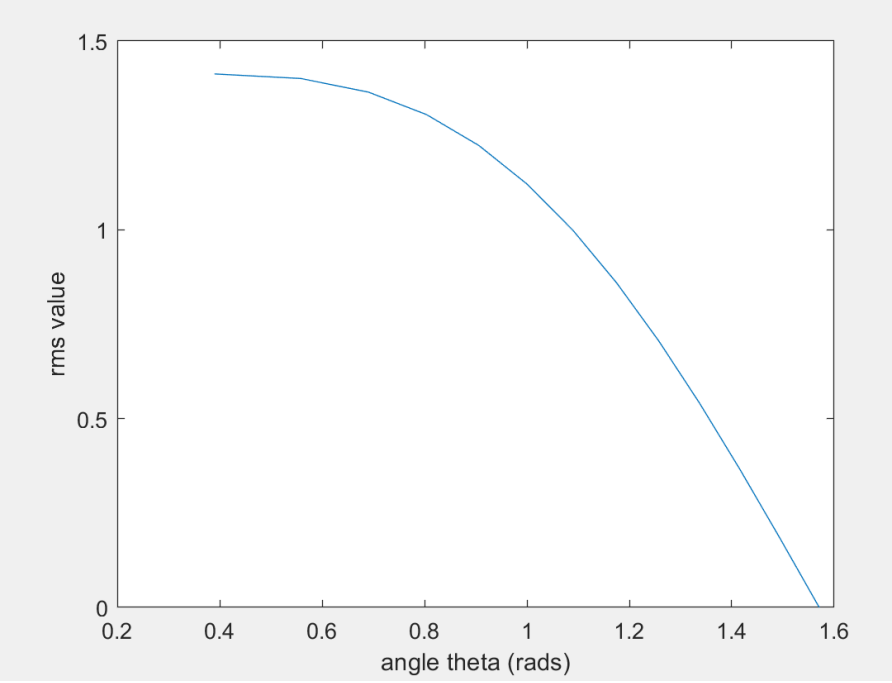

Fig. D-2 Simulation results of corresponding offset angle of the RMS

Appendix E: Object Detection Model Statistics

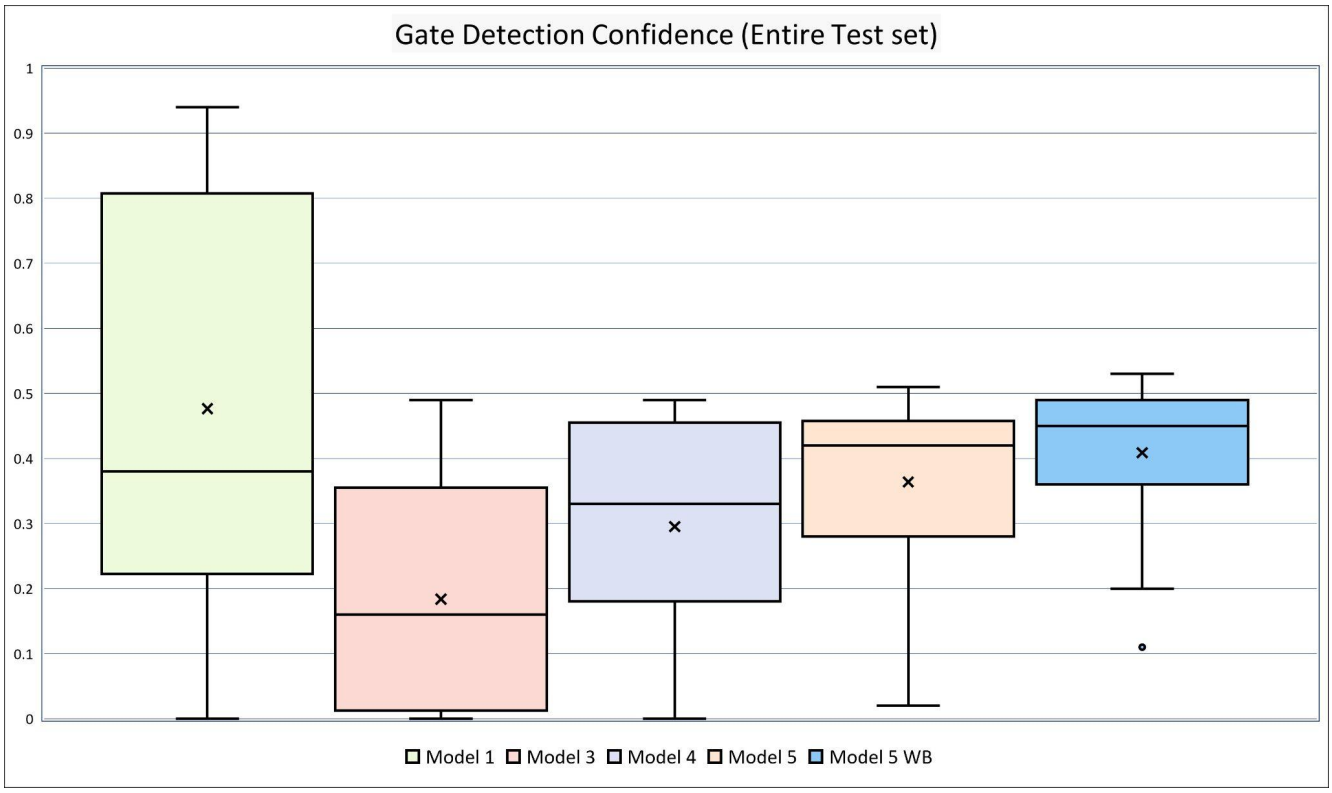

Fig. E-1 Model Progression Box and Whisker of the entire test set (80% MHL - 20% other)

	Model 1	Model 3	Model 4	Model 5	Model 5 WB
Minimum	0	0	0	0.02	0.11
Q1	0.2225	0.0125	0.18	0.28	0.36
Median	0.38	0.16	0.33	0.42	0.45
Q3	0.8075	0.335	0.455	0.4575	0.49
Maximum	0.94	0.49	0.49	0.51	0.53

Table. E-1 Model Progression Statistics for the entire data set (80% MHL - 20% other)

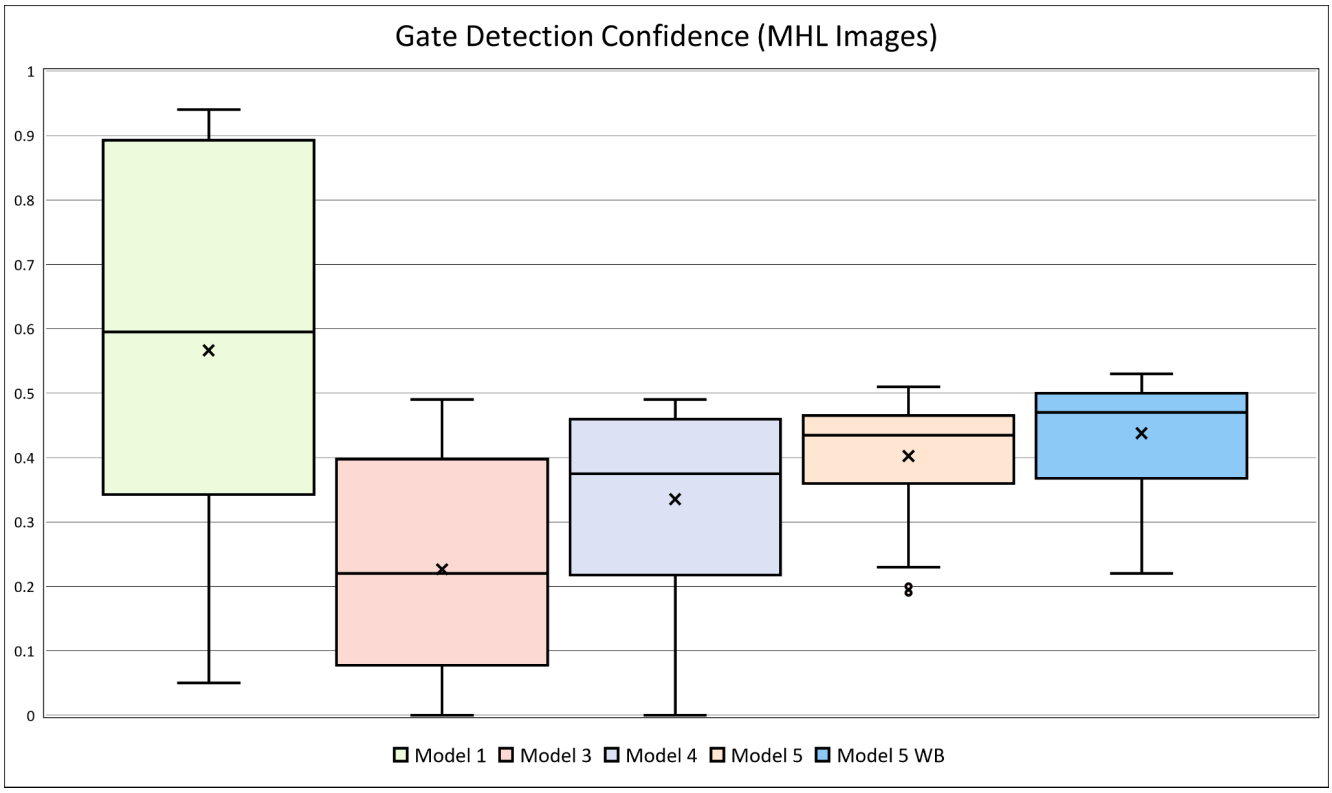

Fig. E-2 Model Progression Box and Whisker of just the MHL images

	Model 1	Model 3	Model 4	Model 5	Model 5 WB
Minimum	0.05	0	0	0.19	0.22
Q1	0.3425	0.0775	0.2175	0.36	0.3675
Median	0.595	0.22	0.375	0.435	0.47
Q3	0.8925	0.3975	0.46	0.465	0.5
Maximum	0.94	0.49	0.49	0.51	0.53

Table. E-2 Model Progression Statistics for MHL images

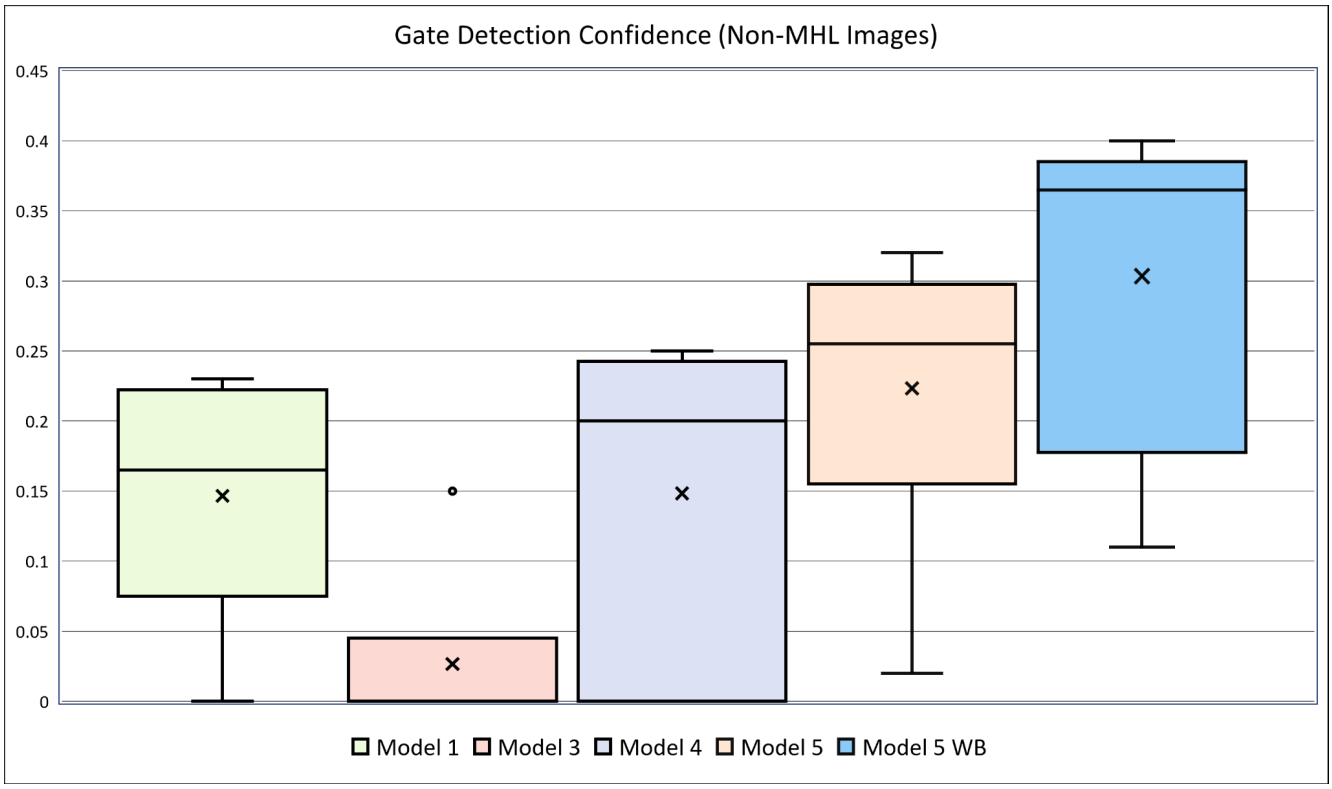

Fig. E-3 Model Progression Box and Whisker of just the non MHL images

	Model 1	Model 3	Model 4	Model 5	Model 5 WB
Minimum	0	0	0	0.02	0.11
Q1	0.075	0	0	0.155	0.1775
Median	0.165	0	0.2	0.255	0.365
Q3	0.2225	0.045	0.2425	0.2975	0.385
Maximum	0.23	0.15	0.25	0.32	0.4

Table. E-3 Model Progression Statistics for non-MHL images