Design of the Minion Research Platform for the 2018 Maritime RobotX Challenge

Jamie E. Barnes, Nate D. Bloom, Stephen P. Cronin, Grady C. Delp, Juan L. Halleran, Matthew R. Helms, James J. Hendrickson, Nicholas R. Middlebrooks, Nicholas D. Moline, James B. Near III, Jefferson S. Romney, Marco A. Schoener, Nicholas C. Schultz, David J. Thompson, Timothy A. Zuercher
Dr. Charles F. Reinholtz, Dr. Eric J. Coyle, Dr. Patrick N. Currier, Dr. Brian K. Butka, Dr. Christopher J. Hockley

***Abstract*— Embry-Riddle Aeronautical University (ERAU) has made significant improvements to their fully autonomous research platform, Minion. To complete mission tasks, Minion uses sophisticated sensory and perception algorithms fusing data from a suite consisting of four LiDARs, two wide-angle cameras, and a high precision GPS/INS. This data feeds path planning and decision-making algorithms that include neural network visual detection and tracking, 3D Multi-Variate Gaussian classification, and dynamic path planning.**

Taking lessons learned from the 2014 and 2016 competitions, the Minion platform was developed emphasizing refinement of existing systems. This allows it to meet the objectives of the 2018 RobotX Challenge and the demands of the team's research and teaching interests. This emphasis on refinement led to major improvements in controls, vision, and propulsion. This also allowed easy integration of other mission requirements, such as the racquetball turret and the autonomous underwater vehicle (AUV) deployment system.

All of Minion's systems are rated to survive operations in adverse weather conditions, including high temperature, high humidity, and heavy precipitation, and they have been tested in these environments. In the course of development, Minion was thoroughly tested using simulations, recorded data, and over 100 hours of in-water testing. The result of this is an advanced platform that is robust, reliable, and readily upgradable.

I. Introduction

A. Background and Vehicle Overview

Embry-Riddle Aeronautical University's (ERAU) Team Minion includes students ranging from undergraduates to Ph.D. candidates with backgrounds in Software, Electrical, and Mechanical Engineering. The team draws from experiences with many autonomous platforms, including entries in the AUVSI Foundation's RoboSub and RoboBoat competitions, as well as the previous two Maritime RobotX competitions.

From its inception in 2014, Team Minion has worked to create a platform that is rugged, customizable, and easily upgradable in order to meet mission requirements. All components are designed to withstand harsh environmental conditions including precipitation, humidity, and heat. In 2016, the Minion autonomous surface vessel (ASV) showcased the MAST (Minion Autonomous Systems Tray), which allowed mission-critical hardware to be suspended under the deck, enabling greater modularity.

For the 2018 Challenge, Team Minion further worked to improve the performance of the ASV by upgrading the propulsion system to allow for holonomic maneuverability. This allows Minion to sway (i.e., move sideways) to fully control position and heading simultaneously. These upgrades, along with software package updates, increased testing time, and a combination of improved custom and commercial-off-the-shelf (COTS) hardware complete a system ready for competition.

B. Software Overview

Software onboard Minion is broken into individual process modules that execute in parallel and communicate asynchronously using a publisher-subscriber messaging system. This enables modules to run at different rates and be selectively activated and deactivated, improving overall system efficiency. The competition software architecture is shown in Fig 1.

Sensing is handled primarily by a combination of a LiDAR-based Perception module, a camera-based Vision module, and a GPS/IMU-based State module. These modules leverage the strength of each sensing modality to detect and classify objects to create a world map for the autonomy modules.

The MinionTask mission tracker aggregates data from various modules and determines the best current objective to complete the mission. It communicates the objective to the Path Planner which calculates the optimal path, which the Controls module then executes. MinionTask also communicates the objective to the sensory modules, enabling and disabling processing algorithms based on the current objective.

Additional modules enable Minion to localize underwater acoustic targets with hydrophones, deploy and control an autonomous underwater vehicle (Anchor), and interface with a stand-alone racquetball turret (Bodyguard II). A custom Ground Station operator control unit enables efficient mission control over low-bandwidth datalinks.

The modules interact asynchronously through the MinionCore inter-process communications suite. The function of each module is discussed in Vehicle Design and in the Appendices.

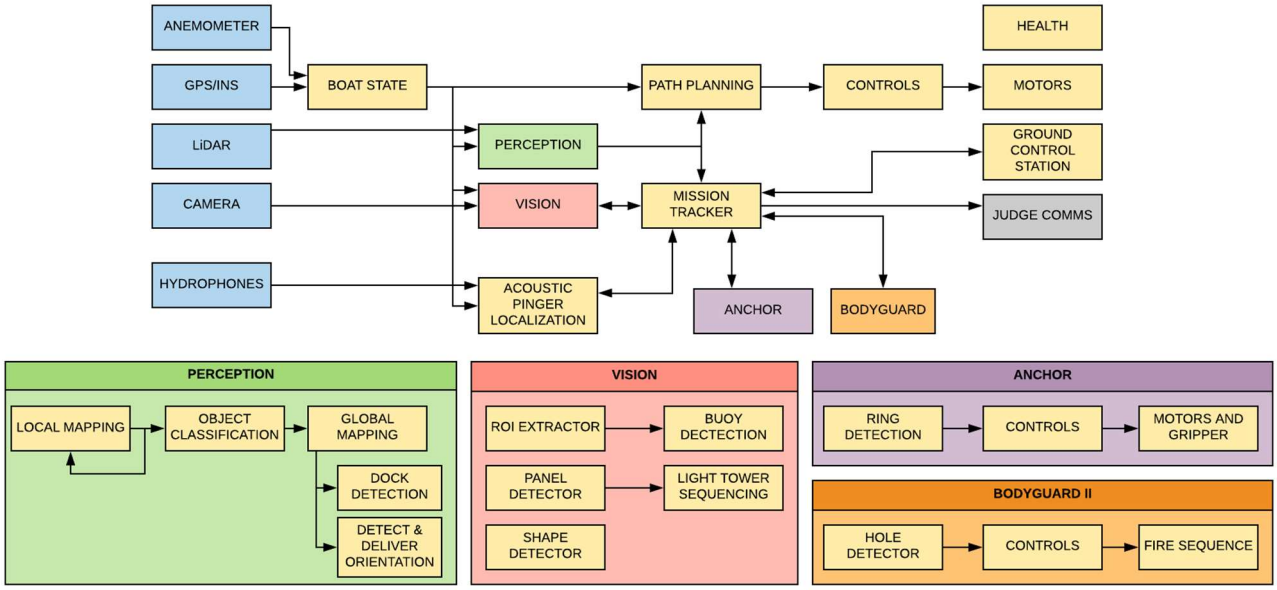

Fig 1. Minion ASV Software Architecture

II. Design Strategy

Learning from the experiences of the 2016 competition, several goals were created to refine the platform's hardware and software to deliver a more robust system that shaped the redesign of the 2018 platform.

One key goal was the implementation of a simulation environment. This simulation allowed integration of the MinionTask, which drives the actions of the ASV, in a virtual environment. While a version of MinionTask was created for the 2016 competition, without the simulation capabilities, it was not robust.

Robust controls were a requirement for 2018 competition. The 2016 competition revealed problems in interfacing a controls system with a path planner and path following system. For example, the prior implementation did not have the ability to follow paths in reverse, posing issues with the docking challenge. The 2018 implementation needed to address these issues while also improving overall system robustness through failover techniques. In case of motor or azimuthing servo failure, the controls module was designed to adjust to the scenario to retain positive control of the system.

The prior perception suite primarily relied on LiDAR sensors for object detection and classification. For color and shape-related tasks such as Scan the Code, the LiDAR sensors provided the camera system with a bounding box to consider for sequence detection. This created a heavy reliance on the correct LiDAR classification of objects and an accurate transformation between LiDAR and camera sensors. In a maritime environment, the constant motion requires the accuracy of the transformation be very high. While one goal was to improve this transformation, another was to use neural networks to allow the cameras to detect the Scan the Code sequence from raw images.

A. Simulation Environment

For the purpose of testing mission and path-planning software without access to the boat, a rudimentary simulation environment was developed during the 2016 Maritime RobotX Challenge. This software was developed using hard-coded maps and variables, but it proved its usefulness and showed the utility of having a more robust and versatile software package in developing autonomy for the 2018 competition.

While a simulation environment running in Gazebo was available for all RobotX teams to utilize, the RobotX Gazebo model was released late in the competition cycle and did not match the capabilities of Minion, including the extensive perception suite and azimuth-capable thrusters.

MinionSim was developed through a series of intermediary milestones, allowing it to become immediately useable by the other modules in the software stack while slowly increasing its usefulness in further developing the other modules. Some of these milestones include producing a synthetic state (position and pose), interpreting the received control messages, sending objects the vessel discovers in the virtual environment, and describing the objects' visual features in ways that are useful to the other software modules.

The simulation software for the 2018 RobotX Challenge includes a mapmaking module for generating fields of virtual objects, and a simulation engine that interprets the files exported from the map maker that contain the object fields. Arbitrary maps can be created that contain configurations of objects and tasks expected in 2018 in order to test the ability of MinionTask. The simulation engine generates and distributes synthetic versions of the messages that other modules expect to receive from the physical ASV, allowing those other modules to be tested without needing the boat to be operating on the water. Other functions, including a hardware-in-the-loop mode,

allow the simulation of virtual objects in the surroundings of the physical boat, which can be used to test path-planning and controls software.

B. MinionTask

Minion's MinionTask provides a unique capability designed to push the system towards the realm of true autonomy. The MinionTask does not script missions; instead the missions are designed through a series of tasks, each of which has defined start conditions, point values, and times to complete. The tasks also encode the object classifications and rule requirements for each mission in a common modular format that can be called by the mission engine. Tasks exist as separate compiled code and can be modified, added, or subtracted without modifying the mission engine.

Minion begins each mission in a search state with no knowledge of the course element locations and no predetermined task order. The search area encompasses the entire operating area and is searched in a pattern seeded by priority locations. As objects are discovered, a ready check function is run for each task to determine if the necessary conditions for initiating the task have been met along with an estimated execution time and point value. Tasks are selected and dynamically launched by the mission engine in real time to maximize the points scored per second of operation time.

C. Controls

Minion's 2018 control algorithms allow for robust handling of the new azimuthing system and improves upon 2016's path following algorithm. The control module is set up in a system that cascades control from the mission objective down to actuator allocation. This cascaded system, in combination with a new nonlinear optimizer, provides a far more robust control system with multiple advantages to the 2016 design. The most significant advantage is the ability to introduce failure, or "limp" modes. These modes enable the platform to operate on a reduced set of actuators and still accomplish the mission objectives. For example, if the port azimuthing were to fail, the system could account for full operation of the starboard actuators and operation of the port actuator at an arbitrary fixed angle.

D. Vision

Minion's vision module, which uses the visible imagers on both Minion and Anchor, supplements the information supplied by the perception module, which relies on LiDAR sensing. In 2016, Minion relied almost solely on the LiDAR system due to efficiency and reliability concerns with the Vision module. As many competition tasks require vision, it was a key to the 2018 strategy to address this deficiency.

Improving vision was addressed by using convolution neural networks (CNNs), which are a type of deep learning network, to increase the speed and accuracy of the vision classification and detection networks. The computational burden of these networks is also offloaded to the system GPUs to prevent slowdown of the other critical systems on Minion.

Ultimately, CNNs are trained using Tensorflow V1.5 to accomplish the Scan the Code task, identify buoy colors, and identify the shape and color of the target on the dock. These CNNs were all created from retraining already constructed networks. This design feature allows crossover of code between different networks and tasks. For the 2018 competition, the Mobilenet V1, Inception V2, and Inception V3 networks are all used to allow for a trade-off between speed and accuracy as well as classification and detection. Using this approach allowed for the easy implementation of a new network for the sub deployment task as well, since the code to run the networks is already compiled and all that is needed is retraining a network.

E. Electrical

To power and control the improvements to the propulsion, sensor, and payload systems, much of the electrical system has been improved from 2016.

The largest changes to the electrical system are the changes made for the new propulsion system. New RDPs demanded a change in motor controllers; new degrees of freedom in the propulsion and vehicle controls system required additional actuators and their associated power distribution and communications circuits. An off-the-shelf motor controller was integrated with the existing safety systems and a custom circuit board was designed to power and drive both the azimuth degrees of freedom and the thruster retraction actuators.

Each payload system, including the turret and the AUV deployment system, also required control circuitry. To improve maintainability of the whole system, the circuit board designed for the azimuth and retraction actuators included some extra peripherals that allowed its common use for the control of the payload systems, increasing field maintainability.

A new feature that has been added to the electrical systems is an upgraded external indicator system. Supplementing the basic light tower of the previous competition, a system of large LED arrays that is an order of magnitude brighter has been added to allow bystanders and operators to easily know the status of the system.

A problem that was found in the previous competition was that the vehicle's motor noise far exceeded the amplitude of the pinger signal, rendering the hydrophones useless while the motors were in operation. This problem has been remedied with the addition of a 5-stage analog filter and gain circuit that attenuates the motor noise.

Please see Appendix H for more electrical information, and Appendix F for more information on the hydrophone.

III. Vehicle Design

A. Design Process

The design process for Minion incorporates techniques from AGILE [1] for both hardware and software development. A 2-week sprint cycle was adopted, as well as weekly stand-up meetings. Stand-up meetings require every

member of the team present their progress in a fast-paced manner with few technical details. This permits every member of the team to be versed in the progress of the entire project. A 2-week sprint cycle concludes with an in-water test on the final day. This provides a visible metric of progress for all team members, as well as incentive to accomplish all goals within a sprint cycle. A 2-week sprint may also conclude with a design review for projects that can't be tested within a 2-week sprint.

B. Major Changes

1) Propulsion Redesign

Following the 2016 RobotX Competition, Minion's propulsion system was redesigned to address weaknesses that were limiting the maneuverability of the platform. Two "motor pods" mount to existing hardpoints on the WAM-V, one at the aft of each pontoon. The pods serve a dual purpose of housing the propulsion system and providing required buoyancy to the WAM-V. Each motor pod houses a RDP thruster. These are a type of marine electric motor in which a brushless motor is built around the propeller to improve efficiency and reduce noise compared to electric trolling motors. It also minimizes the risk of tangling with seaweed or anchor lines and ensures the platform does not disrupt wildlife. Minion's 2016 Torque-Jet RDP thrusters been replaced with Copenhagen VM Asymmetric thrusters in 2018 for improved efficiency, thrust, and reliability.

The focus of the propulsion system redesign was to add azimuthing and beaching capabilities to the selected thrusters. Allowing the thrusters to rotate, or azimuth, improves maneuverability because it allows control over the magnitude and direction of force from each thruster, rather than just magnitude. Independent azimuthing also improves robustness as it allows the platform to operate and maneuver even if only one thruster is functional. To limit the impact of the additional complexity on reliability, the azimuthing thrusters can be mechanically locked, returning the platform to differential thrust. Azimuthing is achieved with Volz DA-30 servos, allowing the thrusters to rotate ± 85°. These powerful servos were chosen due to their environmental ratings and ability to rotate at 150°/s while producing 70.8 lbf-in of torque to smoothly and quickly azimuth the thrusters, even under maximum thrust.

The thrusters can also be retracted from the water, reducing the human intervention required when launching, retrieving, and beaching the ASV. Linak LA-36 linear actuators raise and lower the thrusters. Due to their worm drive, these actuators lock in position when not powered. Retraction and deployment each take under 10 seconds. Appendix D further details the design and analysis behind the new propulsion system. The final propulsion system is shown in Fig 2.

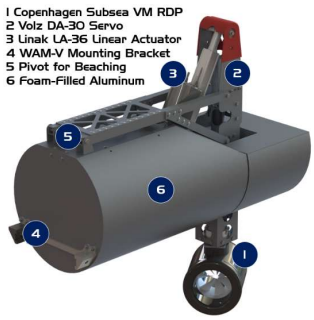

Fig 2. Azimuth and beaching enabled propulsion system.

2) Racquetball Turret

For the purpose of completing the Detect and Deliver task, Team Minion opted to develop a logistically simpler but more functionally robust solution than the air-powered turret used in 2016. The turret, Bodyguard II, is independently powered from the rest of the vehicle by a 6-cell lithium-polymer battery in a waterproof housing, rather than requiring a source of compressed air. This simplifies the beach operations that will be necessary to recharge the turret, and is accomplished by changing the method of firing from a compressed-air cannon to two sets of counter-rotating wheels.

The 2018 turret, shown in Fig 3, is mechanically simpler than in 2016. Bodyguard II operates on two four-bar linkages, powered by HiTec D845WP Waterproof servos. These servos, when operating at 6V, can produce 35 in-lb of torque through their 180 degrees of motion. The mechanical advantage of the four bar linkages, which reduces the azimuth range of motion to 60 degrees, and the altitude range of motion to 20 degrees, amplifies the torque, offering smoother and stronger responses to the changing positions and angles of the WAM-V deck and target.

The active targeting system built into Bodyguard II is enabled by a Microsoft Lifecam, with processing taking place onboard the turret using a Nvidia Jetson TK1. Once triggered by a task running on the primary Minion computer system, the video feed from the LifeCam detects the center of the Detect and Deliver target. The servos then move to position the target in the center of the camera view, and the turret fires.

Fig 3. Bodyguard II Racquetball Turret.

3) Underwater Capability

a) AUV

Anchor, Minion's deployable AUV, is a BlueROV2 from Blue Robotics, and was selected because of its robust design, compact size, and open-source software. The BlueROV2 uses two Blue Robotics T200 thrusters to control depth and an additional four thrusters for holonomic control in the horizontal plane. Anchor is controlled by Minion via a Mavlink stream by processing a video stream from Anchor's onboard low-light 1080p USB camera.

b) Deployment

Anchor is linked to the ASV with the Blue Robotics Fathom Slim Tether that carries 2-wire ethernet and includes Kevlar, so it can act as both a physical link and data link. A custom ratcheting winch spools the tether to deploy and retrieve the BlueROV2. The winch itself is a large spool driven by an Ampflow A28-150-F48 48V motor through a 3-stage gearbox. The winch is mounted on Minion's modular under-deck payload tray, the MAST, shown in Fig 4.

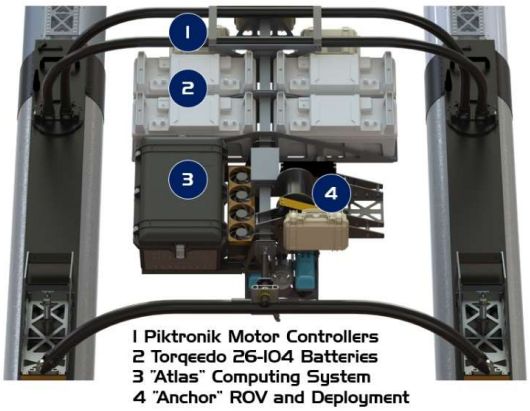

Fig 4. Submarine deployment module mounted on the MAST.

Since the deployment system's location on the MAST is port of the ASV's centerline, a spring-loaded swing-arm transfers the sub from the deployment tray to the centerline of the platform. This minimizes the risk of the sub hitting Minion's pontoons during retrieval, seen in Fig 5. More details on Anchor and its deployment system are in Appendix B.

Fig 5. BlueROV2, Anchor, and deployment showing swing-arm.

4) Path Planning & Controls Approach

Two modules are responsible for acting on autonomous behaviors: the Path Planning and the Controls modules. The Path Planning module is responsible for taking objectives from MinionTask and producing a trajectory that accomplishes the objective. The Controls module takes those trajectories and commands the vehicles actuators. For details on controls and path planning, see Appendix L.

The Path Planning module takes input of different objectives (waypoint, path with constant heading, station keeping, circling, docking) that produce trajectories for the Controls module. The other four, (stop, heading hold, point hold, direct) are special cases that do not produce trajectories and disable parts or all of the Controls module.

The Path Planning module coverts the nine tasking objectives into three possible controls modes: stop, direct, path. In the stop mode the vehicles actuators are disabled. In the direct mode the Controls module is receiving surge, sway, and yaw targets directly. The path mode causes the Controls module to follow a trajectory to an objective.

The trajectory controller in the Controls module is time-based leader-follower technique. Trajectories are smooth, continuous functions of time. The controller first calculates an error using the current vehicle state and the target state at time t. Then, the controller calculates an error between a future predicted state and the target state at time $t + t_{lead}$. A weighted average of the resulting control outputs is used to command the body controller.

The body controller consists of a set of gain-scheduled PID algorithms to control the following states: yaw, yaw rate, sway speed, and surge speed. The PIDs also have ramped inputs and output rate limits. The output of the body controller stage are a set of desired forces and moments. These are given to the actuator allocation stage.

The last stage of the Controls stack is actuator allocation. The allocation stage attempts to produce those forces and moments with the available actuators. The allocation problem is solved with a nonlinear optimization using quadratic sequential programming. The nonlinear optimizer considers the command limits of both the azimuthing actuators and the thrusters, as well as minimizing the amount of change in setting for the actuators. If an exact solution cannot be found, then a best fit solution is produced instead using a relative weighting of the objectives. The currently set modes are differential (no azimuth), full (complete azimuth), crutch mode (only one available azimuth), limp mode (only one motor/azimuth pair), and "twerk" mode (only a single, non-azimuthing thruster).

IV. Experimental Results

A. Test Approach

1) Simulation & Playback

In between tests, the software team uses simulation and playback tools to develop and improve algorithms.

2) Vehicle Shakedown

A vehicle shakedown takes place the day before a test to verify changes and ensure compatibility between modified code throughout the past sprint. This procedure helps ensure that time is not wasted during a test debugging incompatibility between the software modules.

3) Test Days

A test day occurs the final day of each 2-week sprint. In a standard AGILE method, this would be the member demo. During a test, the platform is deployed in the river with a chase boat and may be run in tele-operated or autonomous modes. These tests may be used to find discrepancies between simulation and the real-world environment. Similarly, it offers a good opportunity for logging data from the sensors to improve algorithms in the next sprint.

4) Logging

The Minion platform has multiple methods of logging to ensure all the relevant data is captured while on the water. Each module automatically logs all MinionCore messages or incoming sensor data while open. However, for some modules like vision, these logs are compressed and may not provide the best information for new algorithms. There is also a method for taking uncompressed manual logs.

5) Test Debrief & Sprint Planning

After a test, there is a debrief for all team members. This debrief covers everything that was accomplished during the test, as well as anything that needs to be accomplished for the next test. These debriefs are used to begin planning for the next 2-week sprint cycle.

B. Capabilities Testing

1) Perception

The Perception module is responsible for the detection of waterborne objects, mapping those objects, and determining if any of the detected objects are a competition object. The goals for this system are:

1. Detect objects within 25m to bow, port, and starboard, 10m to stern
2. Identify object location and size to within 0.5m
3. Map an area up to 1 sq. mile
4. Classify competition objects within 5 seconds of detection
5. Classify objects with over 90% accuracy and less than 20% false positive rate.

The detection, mapping and classification methods of the perception module are all detailed in Appendix E.

The accuracy of the detection and mapping system was found to be approximately 20cm for stationary objects within 20m of the vessel. This accuracy can be seen in Fig 6 where the pier pylons can be easily distinguished. However, all competition objects are floating, and can therefore have their point clouds distorted by the wave induced motion. The point clouds can still be easily recognized as the associated object, as evidenced by the TaylorMade Buoy and Light Tower object in Fig 7. Both Fig 6 and 7 were created using empirical data collected from on-water testing and then re-played through Minion's perception module.

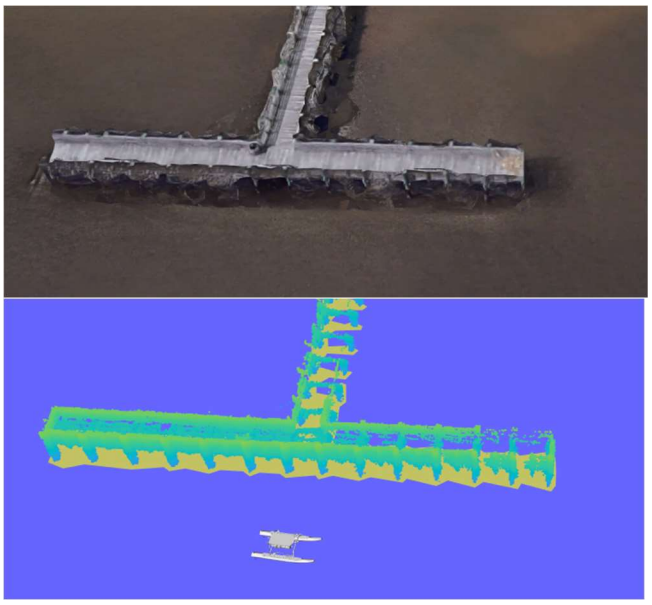

Fig 6. Satellite view of a pier compared to the 3D point cloud captured by Minion. The yellow polygon surrounding the bottom of the pier represents the mapped object boundaries used by the path planner.

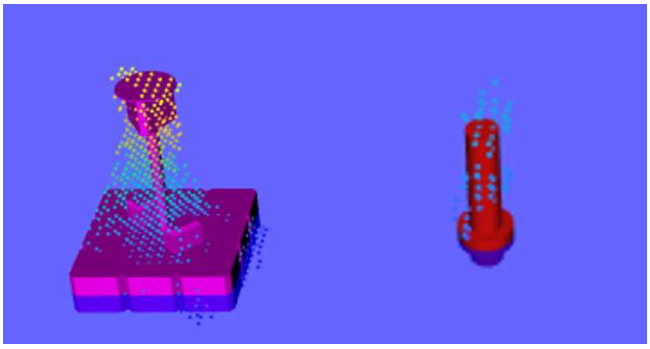

Fig 7. Point clouds for a floating TaylorMade Buoy and Light Tower. The CAD Model for each object has been placed on the figure in the location the object was detected. The CAD model for the light tower does not include the guide ropes on each corner of the base which are connected to the pole just below the panels.

Object features used in classification were collected from four competition objects during in-water testing, which are unique in spatial capabilities or near infrared reflectivity. The confusion matrix of Table I shows the accuracy of classifying these objects across a total of 1863 samples. It should be noted that a tall buoy with reflector refers to the green and red TaylorMade buoys, while the general Tall buoy class is a white TaylorMade buoy or totem.

Table I

CONFUSION MATRIX OF CLASSIFICATION RESULTS

		Test Class				
		Tall Buoy	A3 Buoy	Light Tower	Tall w/o Reflector	Unknown
Predicted Class	Tall Buoy	97.7%				17.4%
	A3 Buoy		97.2%	1.1%		9.2%
	Light Tower			95.8%		3.2%
	Tall w/o Reflector				98.5%	20.9%
	Unknown	2.3%	2.8%	3.1%	1.5%	49.3%

The results show that the goal of over 90% accuracy has been achieved with the lowest classification accuracy being 95.8%. Similarly, the false positive rate is under 20% for every class with only unknown objects even exceeding a 2% false positive rate. While not a specific goal, false negative rates are also below 4%. These results, combined with the history-based filtering discussed in Appendix E, allows the MinionTask to trust the class label given by the perception module.

2) Vision

The vision module is responsible for identifying the color of objects and shapes used in competition. Color classification is required to complete Scan the Code, while shape and color detection are required for the Detect and Deliver task as well as the Docking task.

a) Light Tower Task

A combination of three different CNNs was used to determine the sequence of the tower. The first network ran using the Faster RCNN Inception V2 (Coco) detection model trained with 50 proposal regions to crop the raw image from the camera down to the light tower. From the tower image, the single-shot detector (SSD) mobilenet V1 (Coco) was used to crop the image down to the light panel. Once the light panel image was obtained, a final network uses the Inception V3 classification framework to output the color of the panel.

This task ran at approximately 7 frames per second to obtain this number of samples per sequence. Next, to determine the sequence, a detector was implemented which used a moving mode to determine the color of each panel. Using this method, the sequence was output successfully in all cases except when the sun washed out the colors on the panel. In this case, the boat would rotate around the light tower and attempt to classify the panel in better lighting conditions. Fig 8 shows an example of the proposed regions (blue bounding box) of the light tower and panel.

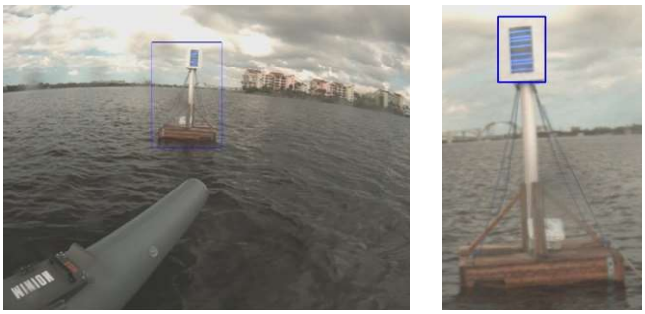

Fig 8. Light tower and panel prosed areas of interest.

The results in Table II show a robust system that can effectively crop down to the light panel when the light tower is in range during the task. These networks worked in bad lighting conditions as well since the CNNs did not rely heavily on color as a determining factor in the detection process. This was not true, however, for the panel color detection as this CNN was almost solely based upon color of the panel, since that is the only difference between red, green, and blue.

Table II

COMBINED SPEED AND ACCURACY OF LIGHT TOWER NETWORKS

Network	Purpose	Type	Speed [ms]	Accuracy [%]
Inception V2	Crop Tower	Detect	100-150	100
Coco Mobilenet V2	Crop Panel	Detect	20-40	99.7
Inception V3	Identify color	Classify	20-40	98.2
Combined	All	Both	140-230	97.9

An overall results and confusion matrix is shown in Table III for an 855-image set over two different sequences. This data shows that the color detector is robust as well and only fails to work well when the sun washes out the panel.

Table III

LIGHT PANEL COLOR CLASSIFICATION NETWORK RESULTS

		Test Class			
		Black	Blue	Red	Green
Predicted Class	Black	100.0%			
	Blue		96.4%		3.0%
	Red			100.0%	
	Green		3.6%		97.0%

The final step in predicting the sequence is the use of the sequence detector, which uses the results of the CNNs in real time. The detector uses a moving mode of the last five predictions to vote on the actual prediction. This is repeated until a sequence is detected; a black panel, 3 non-black panels, and a final black panel. A sequence is output once it has been voted for three times in the allotted time. If no sequence is detected in this time, the boat will circle around the tower and try once more.

b) Object Classification

The other use of vision for Minion was classifying the different colored buoys and the signs for docking. Both tasks were also completed using the Inception V3 classification network. These networks were retrained individually for the different tasks they were applied to.

For the buoys, the network was segmented into classifying the buoy as either red, green, blue, yellow, white, or black. Only color was of importance, so the type of buoy did not matter and was not accounted for in the classification. This was possible since the LiDAR data was accurate enough to classify the type of buoy.

A confusion matrix is shown in Table IV for the general performance of this network. For this task, if a color could not be identified with over a 70 percent confidence and a majority vote over a range of 20 images, the color is returned as unknown. This was done because a false positive is almost always worse than having an unknown

color, as other logic can be applied to determine the corrective action to take.

Table IV

CONFUSION MATRIX FOR BUOY COLOR CLASSIFICATION

		Test Class					
		Red	Green	Blue	Yellow	White	Black
Predicted Class	Red	99.9%					
	Green		99.5%				
	Blue		0.2%	100.0%			
	Yellow				100.0%		
	White	0.1%	0.2%			100.0%	
	Black		0.2%				100.0%

The second classification network that was trained was for the docking signs. This network was used to detect the shape and color of the different docking signs. The same process was used for this task as was for the buoys. The network identified the color (red, green, or blue) and the shape (cruciform, circle, or triangle). Due to time limitations, there was no available data for testing this network.

3) Acoustics

Localization of the pinger was accomplished utilizing an ultra-short baseline array of four hydrophones and a multilateration processing algorithm. These sensors were arranged in a tetrahedron, allowing for the position of the source to be calculated as opposed to the bearing, which was employed in the prior implementation on the platform. This approach allows for more robust means of rejecting invalid returns, as the location can be compared to the buoys defining the gate. The specifics of the system as well as a detailed discussion of the algorithms implemented are in Appendix F.

At a high level, the goals of the system were to provide a more robust system than the prior iteration of the technology that could leverage the positioning information with enough accuracy. This translated into requirements would be:

1. Bearing accuracies (on surface plane) of ±5 deg
2. Positional accuracy within ±1m on (surface plane)

To validate whether or not the algorithm achieved these results, two forms of testing were employed. Signal simulation models allowed for rapid validation of the behavior of the full array; pool testing allowed for a controlled environment where the source could be easily moved and its location measured. These two forms of testing allowed for a confident deployment of the system on the platform.

To process the data, the raw waveform, Fig 9, is taken and the pulse, Fig 10, is extracted using frequency analysis. This allows the clean signals at the front of the incident pulse, Fig 11. At this point, phase analysis can be performed in order to compute the position.

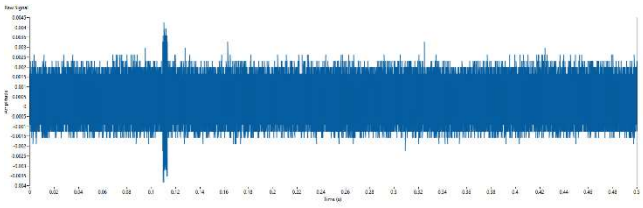

Fig 9. Raw waveform – 0.5 second capture.

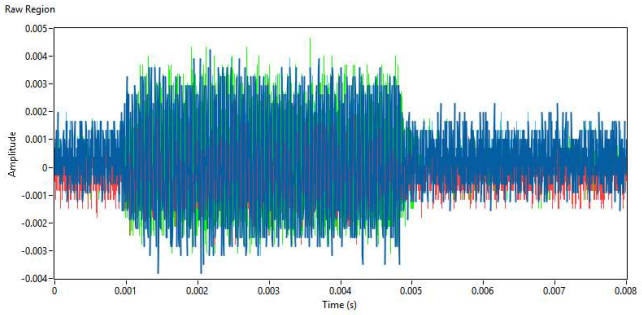

Fig 10. Extracted pulse (seen at t-0.15s above)

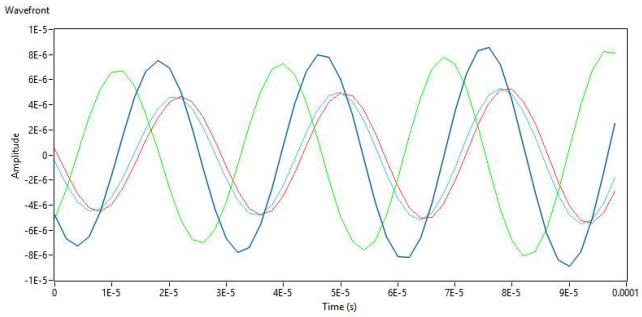

Fig 11. *Wavefront Signals (seen at t=0.001s above)*

Evaluating the position involves a numerical solution to the location of the pinger in XYZ space. For the signals above, doing so yields a position offset of (10,11,1) m. Based on the measurements of the pinger and arrays location, this results in a positional error of 0.2 and 0.18 m and an angular error of -0.1 deg in the surface plane. These performance results were repeatable in subsequent testing, achieving errors less than that of both the angular and positioning requirement.

C. Mission Testing

The break-down of each mission's testing, both in simulation and in-water, can be seen in Table V. The result of this extensive simulated testing was 58 total tests run across 5 different tasks, including: Navigation Gates, Acoustic Gates, Scan the Code, Obstacle Field and Totem Circling, and Docking. This extensive simulation testing helped ensure that the limited in-water test time was used to tune task run times and thresholds. As a result, the approximately 13 hours of in-water testing were used to refine the task parameters and to find edge cases to further test in simulation.

Table V

MISSION TESTING RESULTS

Mission	Simulation Test Hours	In-Water Test Hours
Navigation Gates	20.	5
Scan the Code	20	5
Acoustic Gates	10	2
Obstacle Field and Totems	8	-
Docking	-	1
Total	**58**	**13**

1) Simulation Tracker Results

Of the five tasks that were tested, four of these were able to be tested in MinionSim. The tasks that were tested in the simulator were the Navigation, Scan the Code, Acoustic Gates, and Obstacle Avoidance and Totem Circling tasks.

The Navigation task was the most heavily tested task in the simulator. Since it is the entry key to all testing that will need to be done on the course, it was critical that this challenge would be able to be robustly and reliably completed. In the simulator, the team was able to test several edge cases such as the gates being severely out of spec compared to the listed dimensions in the task outline. This included gates that were upwards of 40 meters apart and in skewed configurations, such as the case shown in Fig 12. Testing in the simulator also allowed testing of the platform's ability to complete this challenge both with and without color classification information being applied to the buoys. Through over 20 hours of simulator testing throughout the logic development phase, this task was proven to be highly reliable.

Testing of the Scan the Code challenge was fairly limited in MinionSim. Without the ability to simulate the sequence in a way that would allow the vision module to be tested, simulation was limited to checking the movement routines of the platform throughout the task. However, this did allow for around 20 hours of behavior and waypoint testing in simulation.

Similar to the Scan the Code task, limitations in MinionSim prevented the team from simulating pinger data, which would be used to determine the start gate to cross through. As a result, the platform would randomly guess and then transit through one of the three gates before completing the rest of the task.

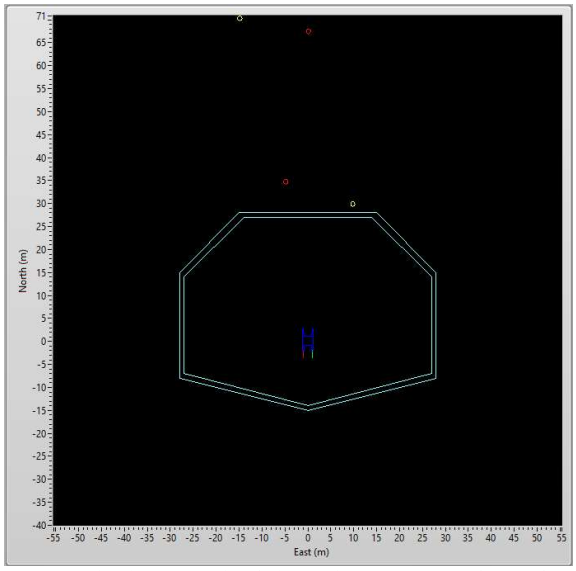

Fig 12. Navigation gate simulation using 40+ meter long, skewed gates

Although the pinger could not be simulated, MinionSim was more than capable of testing of edge cases and of situations that would be out of specification for competition requirements. Some of the edge cases tested included unevenly spaced gates, gates with the start buoys positioned above and below the line of fit (see Fig 14) and elongated start gates. The robustness of the module was also tested by simulating objects with incorrect or missing classifications as well as missing color identifications.

While testing this task, the circling of a specified totem was also tested, which allowed for behavior validation for the circling that would be done for the totem task. This also included testing the behaviors of the platform when only one buoy was detected in the region where a totem would be expected for circling. As a result, nearly 10 hours of successful simulator testing for the acoustic gate task was performed.

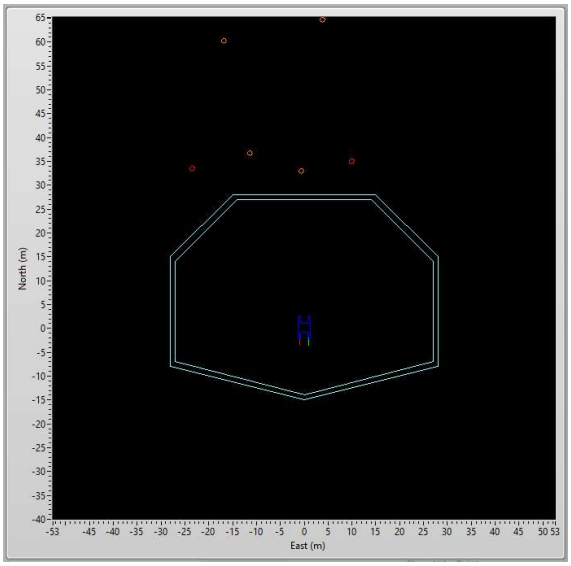

Fig 13. Acoustic gates with skewed, unevenly spaced entry gates.

2) In-Water Tests

As was to be expected, in-water testing revealed logic bugs and edge cases that were not initially considered when testing each of the tasks. During the time that was available for testing each challenge, four of the five tasks were attempted on the water. These included the Navigation gates, Scan the Code, Acoustic gates, and Docking challenges. Through all this testing, an impressive 13 hours of tasks testing was accumulated.

In-water testing of the Navigation gates challenge showed that several edge cases needed to be accounted for in the task logic. The first of these edge cases discovered was when the start gates were skewed. This was solved by adding a check in the logic for possible skewed gate conditions, further described in Appendix I. However, after this fix was implemented, it was found through in-water testing that it causes the boat to plot waypoints away from the end gate location to correct for what it thought was a skewed gate when only one of the end gate buoys had been classified. This problem was then solved by making the skewed gate case toggleable in the configuration file. Real world testing also showed that, due to potentially slow classification times, there was a need for this task to be able to find both the start and end gates with limited, and in some cases incorrect, classification information from Vision and Perception. After these changes were made, the navigation gates challenge was again attempted in the water. The platform was able to successfully navigate through the gates during 10 hours of in-water testing.

Testing of the Scan the Code challenge in real world conditions proved to be highly successful. This testing showed that scan angles, which allowed the sun to appear in front of or behind the ASV, hindered the sequence accuracy. The configuration files were changed to allow the platform to approach the tower at more ideal angles. Testing of this task also showed that the ideal scanning distance to get fast, reliable sequence returns was anywhere from 10-15 meters from the light tower, which was also edited in the configuration file. Testing of this task proved to be extremely successful with around 10 hours of successful in-water testing.

The Acoustic Gates task in-water testing revealed that the details about the ASV's real-world handling characteristics, primarily Minion's turning radius, were not accurately modeled in the simulation environment. As a result, it was determined that the circling radius for the end buoys was too tight, so this parameter was added into the configuration file so it could be tuned. It was also discovered that the intermittent waypoint generated after the ASV crossed through the gate, but before it went to search for the buoy to circle, was too close to the gate buoys. This would cause the ASV to make large, circular paths that would often put the ASV back through one or more gates while attempting to achieve the intermittent waypoint. This was then corrected in the Acoustic Gate task code by making the parameter for how far out the waypoint was placed past the gates a tunable parameter. Unfortunately, the only element of this task that was unable to be tested on the water was the gate detection via the hydrophones. However, even without this element, the ASV was able to successfully detect the gates, navigate through a randomly selected gate, find the required buoy to circle, and circle that buoy in the correct direction. As such, there were only 5 hours of successful attempts at this task on the water.

The Docking challenge was one that was only able to be simulated on the water through hardware-in-the-loop simulation of the dock. However, this did allow the logic for this challenge to be refined and tested. Through the in-water testing that was done, it was determined that the object growth that was done by the path-planner in order to ensure the ASV did not ram into obstacles prevented the ASV from being able to successfully complete the docking challenge. Thus, it was noted that there needed to be a direct mode in the Path Planner that would ignore obstacles in the way and simply drive to a point. It was also noted that a fall-back option that would be able to directly command the controls module, regardless of obstacles in the path, would need to be developed or revived. This resulted in emergence of Direct Mode, which overrides the path-planner and sends direct messages to the controls module. Both additions would allow the ASV to successfully complete the docking challenge. Unfortunately, this resulted in only around an hour of in-water testing for the docking challenge.

V. CONCLUSIONS

ERAU Team Minion has improved on its 2016 RobotX entry by improving the electrical, propulsion, controls, tasking, simulation, and vision capabilities to address the 2018 challenges. Additional new capabilities include azimuthing control and a deployable AUV. The system has been extensively tested with hundreds of hours of simulation development and over 100 hours of in-water testing. The result is a highly capable autonomous maritime system capable of competing successfully in RobotX 2018.

VI. ACKNOWLEDGEMENTS

ERAU Team Minion acknowledges the RobotX sponsors, AUVSI and ONR. The team also wishes to acknowledge its sponsors: Glenair, Velodyne Lidar, TORC Robotics, Volz Servos, Teledyne Dalsa, Solidworks, Mathworks, ERAU COE, ERAU SGA, Copenhagen Subsea, Pelican, the Paul B. Hunter and Constance Charitable Foundation and all other generous sponsors. Finally, the team would like to thank all those individuals who have helped us in this project.

VII. REFERENCES

[1] Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J. & Thomas, D. (2001). Manifesto for Agile Software Development Manifesto for Agile Software Development.

VIII. APPENDICES

A. Situational Awareness
B. Autonomous Underwater Vehicle
C. Racquetball Turret
D. Propulsion Systems
E. Perception Systems
F. Acoustic Systems
G. MinionCore Inter-process Communications
H. Electrical Systems
I. MinionTask Mission Planner
J. MinionSim Simulator
K. Vision Systems
L. Controls & Path Planner

Appendix A: Operator and Onlooker Situational Awareness

Grady C. Delp, Marco A. Schoener

I. Introduction

With the increasing presence of autonomous vehicles in everyday life, it is necessary to make the intentions and actions of a vehicle known, not only to the operators and occupants, but to onlookers in the surrounding environment. Team Minion has developed a set of novel solutions to this problem, making use of the modular nature of the Minion platform, and its software stack. These solutions are three-fold:

- MinionTab Tablet Interface
- MinionG Ground Station Interface
- Lighted Indicator Panel Modules

II. MinonTab Tablet Interface

Making a return from 2016, the MinionTab Tablet Interface has a new and more fully-realized user-interface, giving it a more streamlined appearance and providing greater ease-of-use.

MinionTab fulfills the 2018 Maritime RobotX Challenge requirement of a "Judges Display". The tablet enables the user to remain updated on the status of the Minion platform during a mission run. By displaying information about Minion's physical location, health, and modes, as well as information about mission tasks in progress vehicle hardware and software The interface (Figure Fig 1) is composed of two panes: the navigation panel and the boat monitor.

The navigation panel displays the different boat monitoring panels, gives current general mission state and statuses of the boat, and contains user settings for the application. The buttons are set to display the available status of the boat. The general information underneath the buttons give updates of mission tasks; boat speed and bearing; the current wind speed and direction; and safety and control states. And the user-settings allow the user to change unit systems for data display.

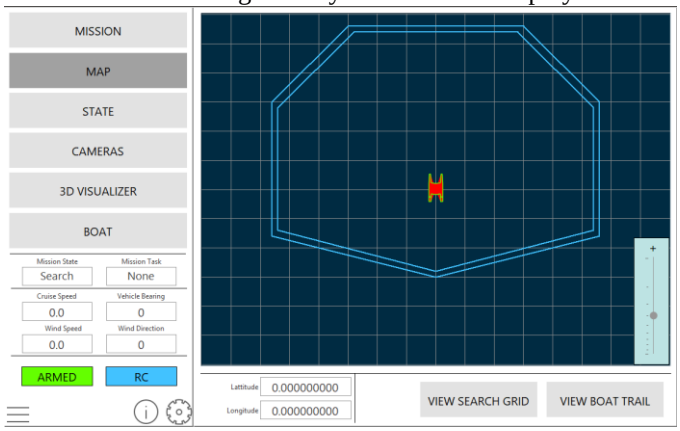

Fig 1. The MinionTab interface overall display.

The boat monitor comprises of the up-to-date status of the boat. This panel contains interactive elements to view certain aspects of the boat. For example, the boat tab can be zoomed in or use buttons to display the search grid as the mission runs.

A. Mission Tab

The Mission tab (Fig. 2) displays all of the scoring elements required for the "Judge's Display", the mission and run-block time, and a list of the compiled tasks. Mission information comes directly from the Mission Planner running off of the boat to keep up-to-date with each run. The tasks are as follows: Scan the Code color sequence; the detected active entry and exit pingers; the two colors and shapes both for the Dock Signs and Detect and Deliver targets; and which totems are supposed to be circled.

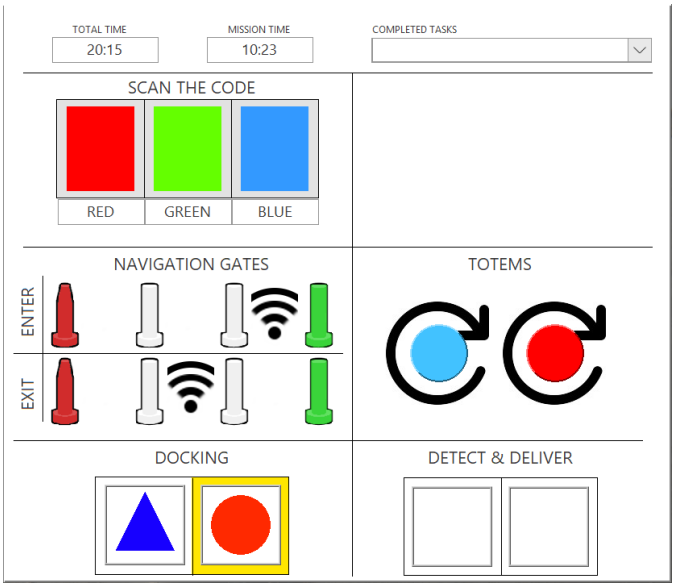

Fig 2. Mission Tab of the MinionTab UI displaying example results for the scoring elements.

B. Map Tab

The Map tab (Fig. 3) displays the boat's environment which includes the boat (red), the visibility horizon (light blue), objects in the environment (colored by class), the trail of the boat's movements, the search grid (not shown), and the current object path. The boat's position is displayed as latitude and

longitude.

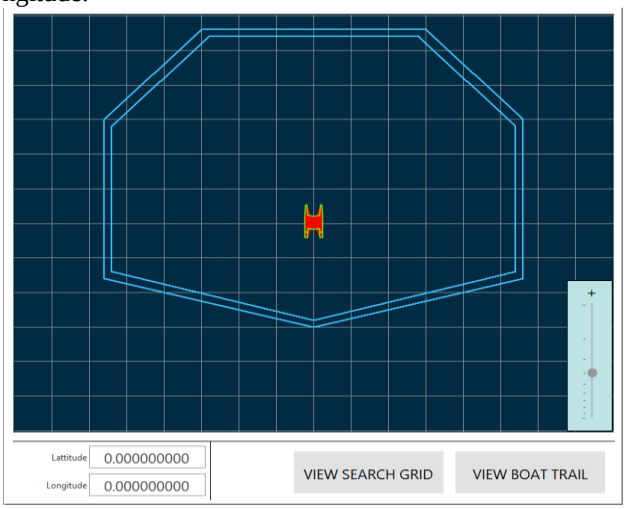

Fig 3. The Map tab of the MinionTab UI that shows the boat moving about its environment.

The map can be zoomed in and out using the zoom slider (on the right) or a pinch gesture. The search grid and the boat's trail can be toggled on and off.

C. State Tab

The State tab (Fig. 4) displays the states of the boat in a set of history graphs. The states that can be viewed are translational speed, rotational speed, and the angles. The history can range between the previous 10 to 60 seconds with the ability to pause them.

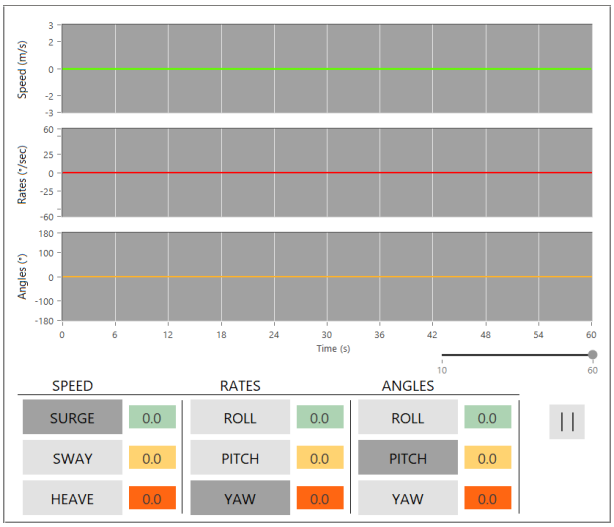

Fig 4. The State tab of the MinionTab UI that displays the measured speeds and angles off of Minion for up to 1 minute of history.

D. Cameras Tab

The Cameras Tab (Fig. 5) displays the images that the boat streams out. Each available camera can be toggled to show up to two of the available cameras.

The boat is equipped with cameras on the base and auxiliary platforms (submarine and turret). However, the GigE cameras on the platform transmit too much data to be handled over a wireless link. To alleviate this issue, the GStreamer [1] package

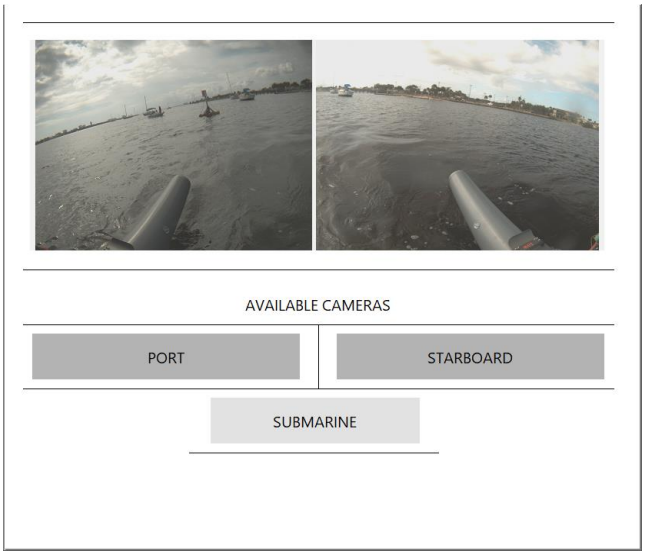

Fig 5. The Cameras tab of the MinionTab UI that displays the boat, submarine, and turret cameras.

was used to enable both end-to-end streaming and hardware encoding/decoding.

The boat hosts a server awaiting connection requests from viewer devices via MinionCore message. This permits any number of users to request individual video streams at any bitrate. The server multi-casts the streams such that no duplicate streams are used, minimizing the bandwidth overhead of the video.

E. 3D Visualizer Tab

A 3D Visualizer tab (Fig. 6) was desired to display an accurate representation of the world as viewed by Minion. This visualizer was built on the PCL [2] library to handle large numbers of point clouds and polygons. On the platform, the visualizer can view the raw point clouds and the processed occupancy grids. However, for remote users, the visualizer may show a CAD model of Minion, polygons representing each object with class information, and the planned path of Minion.

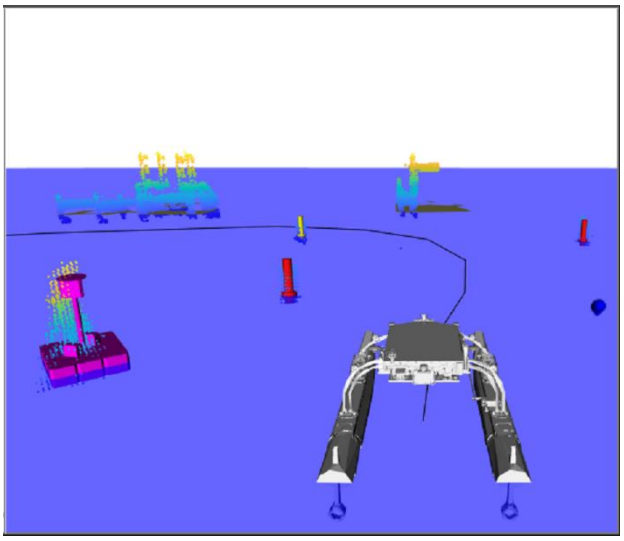

Fig 6. The 3D Visualizer Tab in the MinionTab UI that displays the boat, path, and objects in 3D.

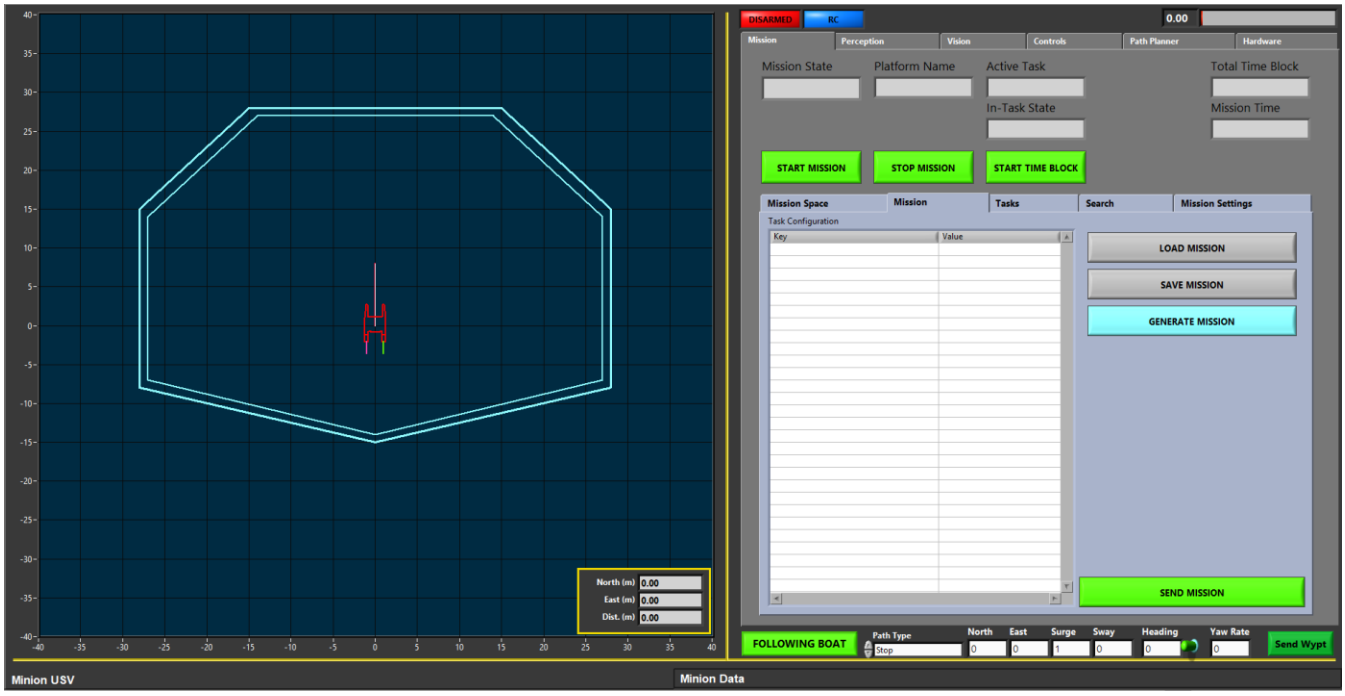

Fig 7. A screenshot of the opening panel of the MinionG Ground Station. On the left is the environment monitor. On the right the mission module monitor is currently selected. The Mission module monitor allows the user to communicate and generate missions from the Ground Station for each run.

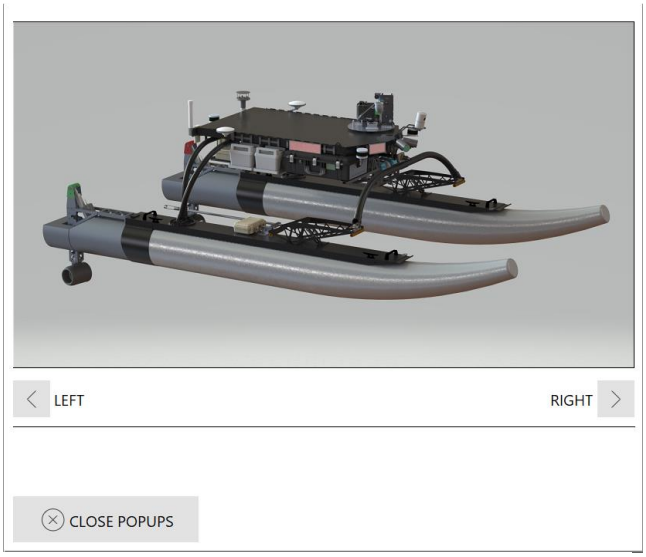

Fig 8. The Boat tab in the MinionTab UI that displays the status of the boat hardware as popups.

F. Boat Tab

The Boat tab (Fig. 8) displays the hardware statues and operations for the boat. A picture of Minion is displayed with colored components of the boat to determine which component statuses can be viewed. The color of these components also determine a general idea on its connectivity and operation status. When the user clicks on the component, a popup displays that specific information.

III. MinionG Ground Station Interface

Minion's Ground Control Station (GCS) has been revamped to focus on providing for a full interface for the vehicle and its software, removing the need to use Remote Desktop to interface with on-board modules. The 2018 GCS has the ability to monitor the vehicle, the environment, and each active module in a concise interface (Fig. 7).

The GCS interface is split into two panes: the environment monitor (left) and the module monitor (right). The environment monitor displays the vehicle (red), the visibility horizon (light blue), objects in the environment (colored by class), the current objective path (not shown), the current and desired heading extending from the boat (white and red respectively), the azimuth angles from the port and starboard motor pods (red and green respectively), control target (not shown), and the search grid (not shown).

The environment monitor has a set of ease-of-use features for the user to interact with. The user can zoom in and zoom out, follow the boat, or snap the map to show all existing objects. Distances on the map can be measured via two user-selected points. The user can use the environment pane in order to input values for the monitor panes (i.e. drawing the search grid bounds by clicking on the environment pane).

Above the module monitor is the safety monitor that indicates the user on battery voltage, safety state, and autonomous state.

A. Mission

The Mission monitor (Fig. 7) displays the boat's current mission tasks, mission and run-block time, mission generation and loading, and general mission parameters. The task generator can import pre-existing missions or create new missions to send to the Mission Planner module. The user can generate a new search grid boundary by clicking on the environment monitor.

B. Perception

The Perception monitor (Fig. 9) displays the LIDAR health status, mapping parameters and filters, request to log specific LIDAR files, and send general perception parameters. This monitor is intended for quick health check on the perception algorithms and sensors and monitor the applied settings.

Fig 9. Communicates and sets perception settings from the Ground Station to the Perception module.

C. Vision

The Vision monitor (Fig. 10) displays the currently streamed images from the boat and submarine cameras, requests for specific logs to be taken, and general camera and Vision settings. The image incoming will be from the Vision module's Gstreamer tool to send compressed GiGE images over the network. The images displayed will be either raw or processed with overlaid LIDAR points to verify the cameras are calibrated. The images are also selectable to be displayed individually or both port and starboard cameras simultaneously.

Fig 10. Displays the streamed images from Minion's Vision module.

D. Controls

The Controls monitor (Fig. 11) displays the history of all states, displays desired and current histories of specific controllable states, set gains and autonomy modes, request specific logs, command setpoints for control tuning, and view the propulsion system's commands, feedback and detected faults.

This monitor is primarily a verification of control stability on the water. This tool will show what the boat wants to do versus what it is currently doing. The history keeps the previous 140 seconds of data to track patterns and give time to verify data.

Fig 11. Communicates settings and objectives to the controls module and displays Minion's desired movement.

E. Path Planner

The Path Planner monitor (Fig. 12) sends path settings to the Path Planner module, sends waypoints via user-selection, and displays the path calculation and error statuses. This monitor is primarily for verifying the correct settings are in the planner before the planner begins calculating paths. And during testing and controls verification, clicking on the environment monitor places a waypoint location and waits until the user hits the

"Send Wypt" button below the module monitor.

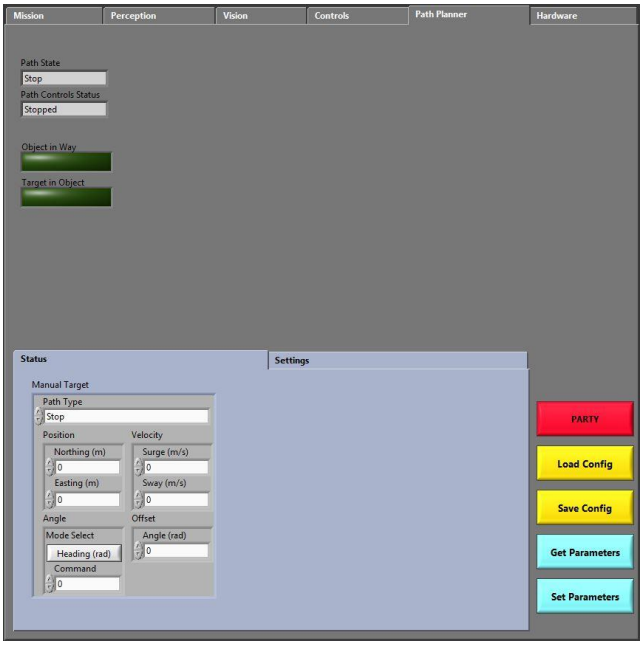

Fig 12. Communicate settings and statuses to the path planner module.

F. Hardware

The Hardware monitor (Fig. 13) displays histories of the propulsion system commands and feedback. The commands displayed are the thruster throttle commands and the azimuth angles. The feedback histories are the motor RPMs, temperatures, and angles; the motor controller temperature, voltage, and current draw, and the RC controller's PWM channels.

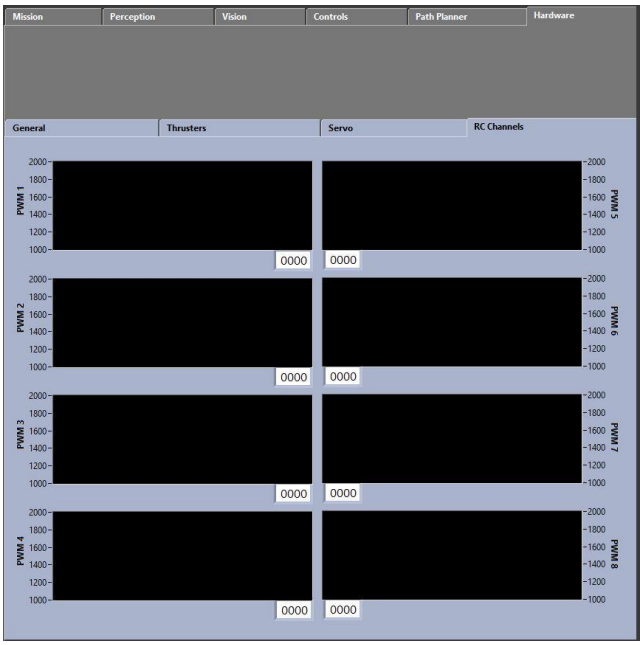

Fig 13. Indicates the statuses and commands of the propulsion system.

IV. Lighted Indicator Panel Modules

When operating in the early interim between the 2016 and 2018 Maritime RobotX Challenges, it was evident that the lighted status indicator, as specified by the competition organizers, and built into the Minion ASV since 2014, was insufficient for operation in bright daylight. Lighted indicator stacks are typically designed for use in factory-floor environments. As such, they are not readily available in configurations that would make them adequately visible in outdoors usage scenarios. Team Minion made the decision to develop a purpose-built indicator system that would be daylight-visible and provide enhanced situational awareness for onlookers and persons in the environment immediately surrounding the Minion ASV.

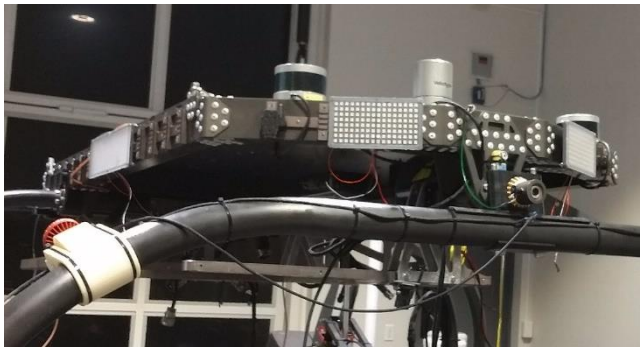

Fig 14. Photograph of the indicator panel modules dry-fitted to the Minion ASV during a sensor mount overhaul.

A. Construction

The lighted indicator panel modules are constructed of widely available WS2812B individually addressable RGB LED modules. These were sourced in the form of pre-assembled flexible arrays, which are low-cost and easy to implement for this application. These arrays are available in a variety of sizes and pixel pitches; the team opted for the narrowest array that would still provide ample numbers of pixels for display of simple text and graphics. This minimum size leads to panels that are less than four inches tall, and vary between seven and thirteen inches wide.

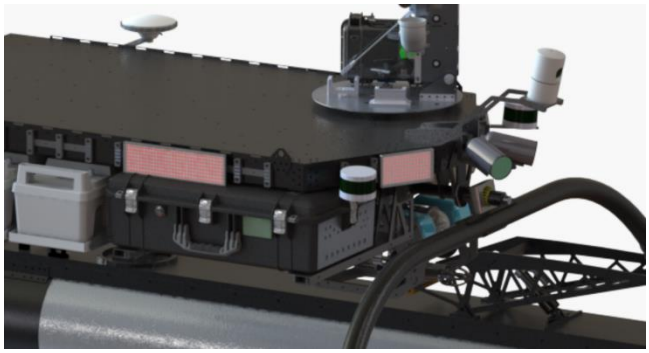

Fig 15. Render depicting indicator panel modules mounted to the starboard side of the Minion ASV.

These LED arrays were built into water-resistant housings that include thermally conductive material to sink the heat dissipated LEDs into the frame of the housing. The housings are mounted to the Minion ASV utilizing the picatinny-style

attachment points positioned around the periphery of the deck (Fig. 14, Fig. 15). The front face of the housing is a matte-finish acrylic, which serves to diffuse the light from the LEDs as well as to prevent glare from the sun that may affect nearby onlookers.

Power for the LED arrays is provided through a 5V regulator. Signal for the panel is driven by an LPC1768 microcontroller, which is seated into the SPI expansion port that is built into the BUGS safety system and acts as a daughterboard to the BUGS system. This provides it with direct information as to the safety state of the system and also allows BUGS to pass along information from the software running on the Minion ASV primary computers. This information determines which of the three modes the indicator panels operate in.

B. Safety Mode

Safety Mode is the primary operating mode of the indicator panels, is the default if it detects the system changing to an active, non-emergency-stopped, state. This default behavior ensures that the vessel will, when operating on water, display the most relevant information to onlookers in its vicinity. This information can be displayed as solid colors or as black text with colored background, based on configuration. The information displayed in this mode includes:

- Boat is non-engaged ("STOP", red illumination) (Fig 16.)
- Boat is under manual control ("R/C", amber illumination)
- Boat is under autonomous control ("AUTO", blue illumination)

Fig 16. The indicator panel display when the boat is in the "STOP" state, shown as black text on red background.

Fig 17. Indicator panels in team spirit mode

C. Static-Display Mode

Static-Display mode is a mode that allows user-set text, images, or animation to display on the indicator panel modules. These images can be used to demonstrate team-spirit or provide an energetic light-show. Static-Display mode is only available when the boat is in the emergency-stopped state for greater than ten seconds, and the user purposefully opts to enter Static-Display mode after that time has passed. The BUGS safety system then serves as a passthrough between the Minion computer systems and the indicator panel controller. If the ASV switches to an active state, Static-Display mode will be disabled until the conditions of safe-duration and user-input are met again (Fig. 17).

D. Enhanced Operation Mode

Enhanced Operation mode serves as an extension of the safety mode that provides further information to onlookers who may not have express knowledge of the boat's intention or purpose. In this configuration, the safety system will actively listen to the ASV computer systems to determine and display the autonomous operations the vessel is performing.

For the purpose of the Maritime RobotX Challenge, this may include the task the ASV is in the process of performing. In a more general purpose, this can include what maneuvers the vessel may be attempting. This mode can be likened to the reverse lights or turn-signals in an automobile that serve to provide the autonomous intent of the vessel to those in its immediate vicinity.

REFERENCES

[1] Gstreamer. (n.d.). Retrieved November 23, 2018, from https://gstreamer.freedesktop.org/documentation/

[2] R.B. Rusu, S. Cousins, "3d is here: Point cloud library (pcl)." In *Robotics and automation (ICRA), 2011 IEEE International Conference on* (pp. 1-4). IEEE, May 2011.

Appendix B: Submarine

Nate D. Bloom, James J. Hendrickson, Matthew R. Helms, David J. Thompson

I. Introduction

The 2018 Robot X Challenge introduced an underwater task, which requires teams to recover stationary rings on an underwater "tree". While this task would be feasible with an attachment to the surface vessel, developing such a system would have very little relation to real world research and development opportunities. Therefore, the team elected to pursue the more challenging but more relevant path of deploying an AUV.

Minion's deployable AUV, Anchor, is a Blue Robotics BlueROV2. Anchor acts as an extension of Minion and is controlled as if being driven by a human operator. Minion processes the incoming video stream and generates the appropriate Mavlink messages to guide Anchor toward the rings.

Since regulations require the underwater vehicle to be tethered to the surface vehicle, a simple winch with a single cable that serves as the mechanical tether and communication line was designed.

Fig 1. The Submarine Deployment System with the BlueROV2, Anchor

II. AUV

Anchor is constructed from a series of milled HDPE plastic and six Blue Robotics T200 thrusters. A Pixhawk autopilot on-board handles controls and a Raspberry Pi manages the video stream. The two computers and the other hardware they need to operate are mounted in a waterproof acrylic enclosure. Two of the T200 thrusters are used to control the depth of the AUV while the remaining four thrusters allow for holonomic movement in the horizontal plane. The onboard camera assembly provides clear high-definition video to Minion in low-light conditions with the added capability of a servo-based tilting mechanism. The Blue Robotics Newton subsea gripper is mounted to the AUV to provide the ability to obtain the ring. Anchor is powered by a Blue Robotics 4s Li-ion battery that is comprised of 16 Samsung 30Q 18650 cells. The thrusters, lights, camera tilt, and gripper are all directly controlled by the Pixhawk autopilot running the Ardusub firmware.

Integration of the BlueROV2 system with Minion was made simple with the open-source nature of the Ardusub project. Minion controls Anchor by sending a Mavlink message that is intended to contain joystick information. Minion is capable of relaying these joystick messages directly from the ground station computer to Anchor for remote manual control. When attempting the ring retrieval task, Minion processes the incoming video stream and adjusts the joystick values in the Mavlink message. The video is streamed from Anchor to Minion using GStreamer.

A. Deployment

1) Requirements

1) The system shall deploy and recover Anchor from Minion to the water.

2) When Anchor is in the water, the system shall direct the tether to the center of the boat on the y-axis.

The intended deployment location from Requirement 2 is shown in Fig 2. The intent of Requirement 2 is to minimize the likelihood of the AUV hitting Minion's pontoons during retrieval.

Fig 2. Anchor at the Y-Axis Center of Minion

3) When completely retrieved, Anchor shall sit on the MAST (Minion Autonomous Systems Tray) to the port of Atlas (Minion's Computer System).

Requirement 3 specifies that the AUV deployment system shall attach to Minion in the previously empty spot on the MAST. This is the empty area at the bottom right of Fig 3.

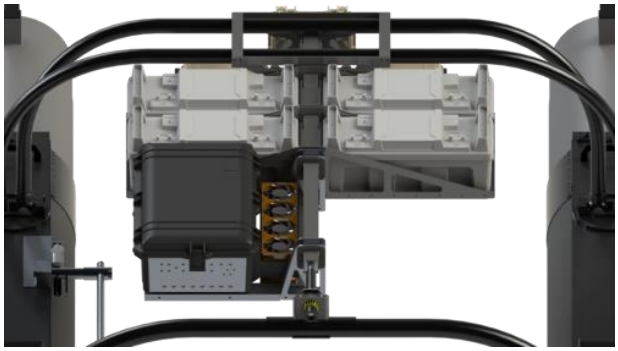

Fig 3. The MAST with an Open Payload Bay

4) The system shall be able to retrieve Anchor when 40 m of tether is unspooled in less than 60 seconds.

5) The system shall be able to contain 50 m of spooled tether.

6) The system shall be compatible with 20-60V systems.

Requirement 6 ensures that Minion may continue to operate at either 25 V or 50 V nominal.

2) Winch Mechanism

a) Spool and Tether

The winch mechanism essentially consists of a geared motor and a spool with a tether. The tether is a BlueRobotics Fathom Slim, a two-wire cable with internal Kevlar strands for strength. This allows it to serve as both the data and physical connections to Anchor. The tether has a minimum bend diameter of 1 inch, a working strength of 80 lbf, and a breaking strength of 350 lbf. This meets the working load of 50 lbf discussed in the Motor and Gearbox section.

To ensure the tether is always contained, regardless of how much slack is in the line as it is spooled up, the spool is significantly oversized. If the tether was perfectly wrapped the tether could hold 173 m of cable, but it is only required to hold 50 m. From here, the tether is spooled through a pulley to a guiding eyelet, discussed below.

b) Motor and Gearbox

The motor and gearbox were selected based on the voltage and time requirements. To determine the torque needed to meet time requirements, a constant retrieval velocity was assumed. The force on the tether was simplified to the weight of Anchor and the force of drag on Anchor. In reality, Anchor's speed would be constantly varying, even with a constant motor rotation per minute (rpm), due to the layers of tether on the spool changing the effective spool diameter. Therefore, torque to pull Anchor at constant speed was determined at the minimum and maximum pulley diameters and averaged.

Since both the motor and gearing ratio needed to be selected, an Excel tool was developed to quickly compare different motors and find the optimum gearing ratio. The tool requires the spool/tether specifications, and two points on the motor rpm/current/torque curve. For any given gearing ratio, it assumes the motor curve is linear, finds the motor rpm that matches the required torque, and finds the spool time at minimum and maximum pulley diameters based on that motor rpm and the gearing ratio. It also finds the linear speed to calculate drag on the tether. This drag can be iterated to get more precise results for a given gearing ratio. Finally, it finds the motor's current draw at that torque to determine if it may be putting too much stress on Minion's electrical system.

A motor capable of 48 V nominal was desired, as it could operate at both of Minion's voltages without the need for a voltage regulator. With these considerations and iterations of different motor and gearing combinations with the Excel tool, the AmpFlow A28-150-F48 was selected. Its specifications are shown in Table I.

Table I

AMPFLOW A28 SPECIFICATIONS [1]

Specification	Value
Max Torque (continuous)	7400
Max Torque (stall)	141.6 lbf-in
Supply Voltage (nominal)	48 VDC
No Load Amps	2.5 A
Kt (in-lbf/Amp)	0.544

At a gearing ratio of 38:1, the AmpFlow can retrieve Anchor in an average of about 37.6 seconds, giving it a factor of safety of 1.6 to the requirement. This was determined using the Excel tool and is verified in the following steps.

1. The drag force of pulling the sub at 4 ft/s is about 26 lbf, and the weight of the sub in air is 24 lbf with all ballast. Since the sub is positively buoyant in water and there is minimal drag in the air, the tether will never see both of these forces at the same time. Therefore the force on the tether will be assumed to be the average of the drag in water and weight in air, 25 lbf, and doubled to 50 lbf for a factor of safety of 2.

$$T = Fd$$
$$T = (50\ lbf) * (1.75\ in)$$
$$T = 87.5\ lbf * in$$

2. At a gearing ratio of 38:1, this torque is reduced to 2.3 lbf-in at the motor shaft. The motor current draw and rpm may then be found with linear interpolation as follows.

Table II

TORQUE-SPEED-CURRENT SPECIFICATIONS

Torque (lbf-in)	Speed (rpm)	Current Draw (A)
0	7400	2.5
156.25	0	290

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$

$$y - 7400\ rpm = \frac{0 - 7400\ rpm}{156.25\ lbf * in - 0}(2.3\ lbf * in - 0)$$

$$y = 7293\ rpm$$

$$y - 7400\ rpm = \frac{0 - 7400\ rpm}{156.25\ lbf * in - 0}(2.3\ lbf * in - 0)$$

$$y = \mathbf{7293\ rpm}$$

$$y - 2.5\ A = \frac{290\ A - 2.5\ A}{156.25\ lbf * in - 0}(2.3\ lbf * in - 0)$$

$$y = \mathbf{6.65\ A}$$

3. Since the current draw of 6.65 A is acceptable, the spooling time can be calculated from the motor rpm of 7293 rpm. The 38:1 gearing reduction means the spool rotates at 191.9 rpm. The amount of tether let out will be assumed to be 90% of the total tether length. Since we have 50 m (1968 in) of tether, 90% is 1758 in. In this case it is assumed that the spool constantly reels in at the minimum diameter. Therefore, the spool time is:

$$\frac{1758\ in}{\pi * 3.5\frac{in}{rev} * 191.9\frac{rev}{min} * \frac{1\ min}{60\ s}} = \mathbf{49.9\ s}$$

Doing the process again for the maximum spool rpm, it is found that the motor draws 10.8 A, and spools in 25.37 seconds. This confirms the average spool time of 37.6 seconds.

Gearing is achieved with VexPro aluminum gears. These gears were selected due to their availability and off the shelf usability. Most off the shelf gears have plain bores that need to be broached to a drive pattern. These VexPro gears are sold with a 0.5 inch hex bore, reducing our manufacturing time. Since the gears are aluminum, they are also less prone to corrosion than steel gears. The gearbox has three reduction stages. The first reduction is 48:84, followed by 18:84 and another 18:84. This results in an overall reduction of 38.1:1.

To hold load while the motor is not powered, a brass ratchet and pawl are used. These are driven off the motor shaft to minimize the load they must hold. A waterproof servo disengages the pawl during deployment, and a torsional spring forces pawl engagement during retrieval and storage.

3) Deployment Arm

To prevent the submarine from hitting Minion's pontoons or frame, the system pulls it from the middle of the platform, as shown in Fig 2. However, the empty payload bay on the MAST is on the port side. Therefore, a spring-loaded swing arm directs Anchor to the center during deployment and guides it to the port payload bay during retrieval. Fig 4 illustrates the motion of the deployment arm.

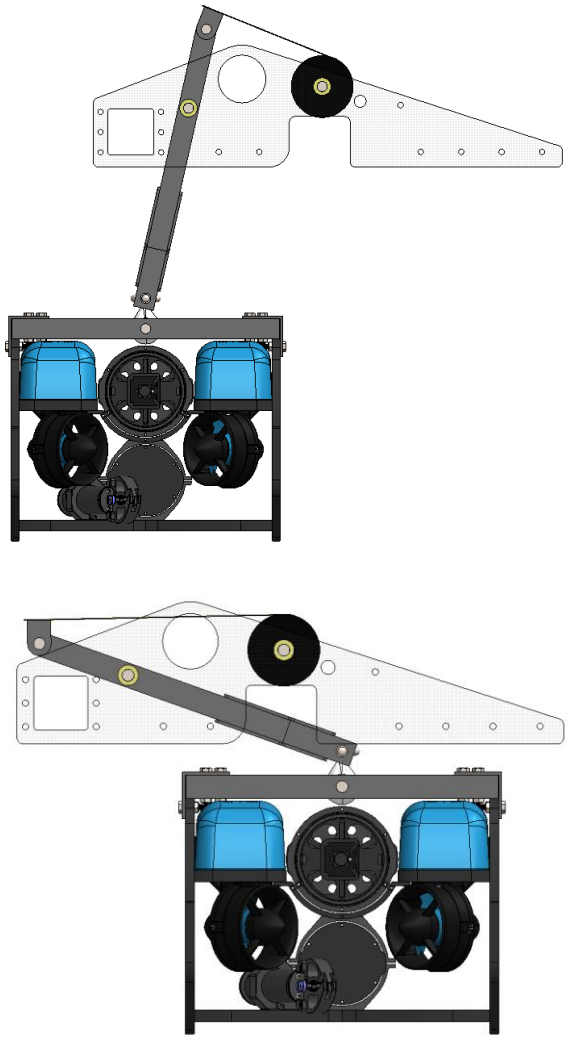

Fig 4. Deployment Arm

The arm is passive and pulled to center by a constant force spring. The tether is fed through an eyelet at the base of the arm. When Anchor is lifted to the base of the arm, it catches on the eyelet, and the tether pulls against the spring to retract the arm.

The desired force from the constant force spring when the arm is deployed is enough to keep the tension in the tether from pulling the arm away from the center. Determining force from the tether on the arm begins with finding the force the tether exerts on the eyelet. Refer to Fig 5.

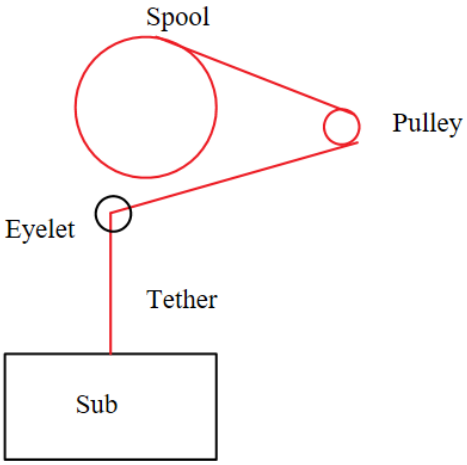

Fig 5. Tether Routing Visualization

Here, the tension in the tether is the weight of the sub, about 24 lbf. The tether to the sub will be assumed to be vertical, and the tether angle between the pulley and the eyelet is 40 degrees from horizontal. Therefore, the horizontal force trying to pull the eyelet away from center is:

$$24\ lbf * \cos(40°) = 18.38\ lbf$$

This force acts as a moment on the deployment arm, which must be cancelled by the spring to prevent rotation

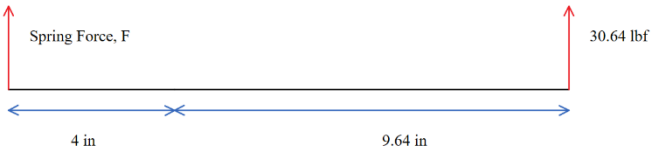

Fig 6. Deployment Arm as Simple Beam

$$F * 4\ \text{in} = 18.38\ \text{lbf} * 9.64\ \text{in}$$
$$\Sigma M = 0$$
$$F = 44.3\ lbf$$

The most powerful constant force spring readily available has a loading force of 40.9 lbf and was chosen for the design.

4) Structural Analysis

a) Deployment Arm

Maximum loading on the deployment arm occurs when the tether and spring are simultaneously pulling on the arm, but the arm is not moving. Maximum force from the spring occurs when the arm is in the deployed position, as the angle between the spring and the arm is nearly 90 degrees (see Fig 4). Since the chosen constant force spring creates 40.9 lbf, that load is applied to the arm at a conservative angle of 90 degrees. This moment is equal to the perpendicular component of the force from the tether. To approximate the load from the tether, the bearing hole for the eyelet was set as a fixed support. The bearing hole for the arm pivot was set as a cylindrical support with free tangential movement. Refer to Fig 6 and Fig 7.

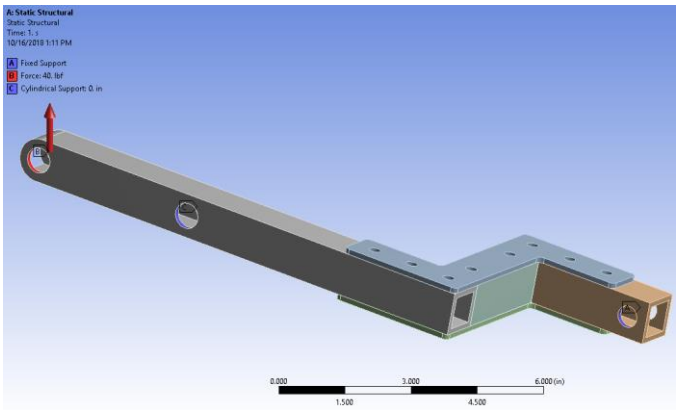

Fig 7. Static Loading of Deployment Arm

This loading results in very minimal deflection for the arm, and a maximum stress of 2,375 psi, giving a factor of safety of 16.8 to 6061 Aluminum's yield strength of 40,000 psi. [2]

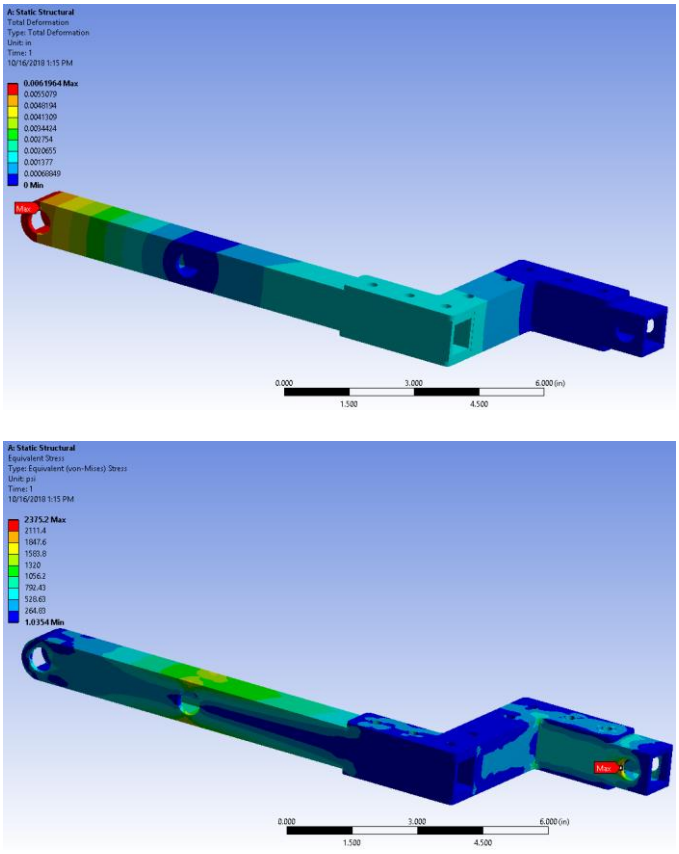

Fig 8. Deflection and Equivalent Stress of Deployment Arm

b) Pawl Shaft

When Anchor is in the payload bay, it is held up by a ratchet and pawl on the motor output shaft. This pawl rests on a shaft that spans the gearbox as shown in Fig 8.

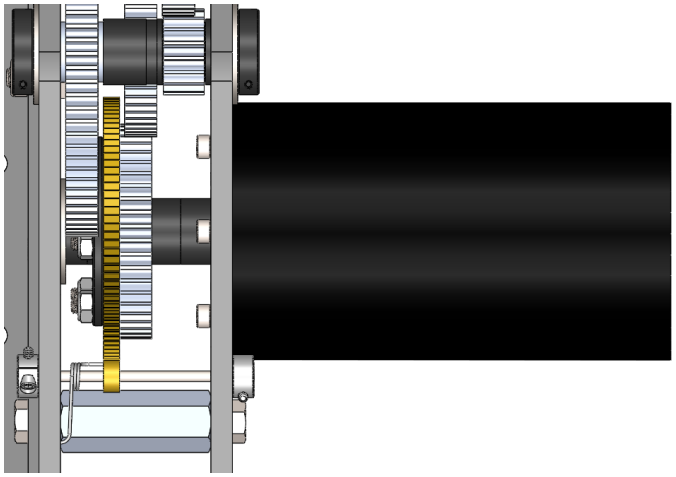

Fig 9. Gearbox Showing Ratchet and Pawl

The max force this shaft could ever see occurs at the breaking point of the tether. The tether's breaking force is 350 lbf. [3] Assuming the tether is at the maximum diameter of the main spool, this is multiplied by a 3.5-inch moment arm for 1,225 in-lbf of torque at the main spool. This is reduced by 38 times due to the gearbox reduction, creating 32.2 in-lbf at the motor shaft. The pawl shaft is offset about 1.5 inches from the motor shaft, so this torque is divided by that moment arm, resulting in 21.5 lbf on the pawl. This loading was added to a simple FEA model as shown in Fig 10.

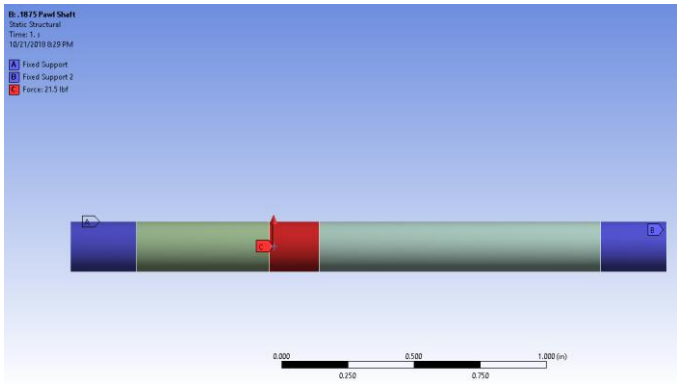

Fig 10. Simple Beam Loading of Pawl Shaft

This loading does not yield the shaft, creating a max stress of 14,417 psi, a factor of safety of 4.18 to 302 stainless steel's yield strength of 60,200 psi. [4] This is very minimal deflection in the shaft. Refer to Fig 12.

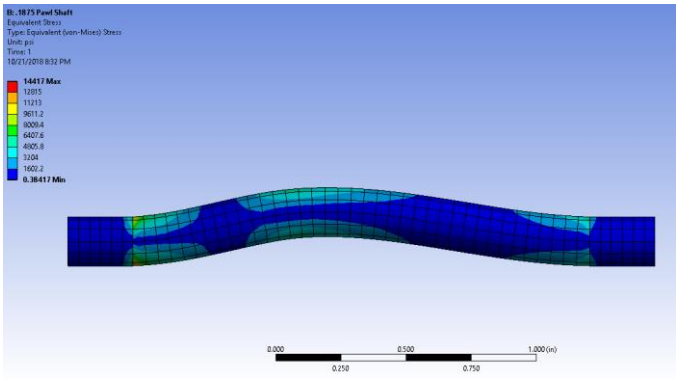

Fig 11. Equivalent Stress in Pawl Shaft

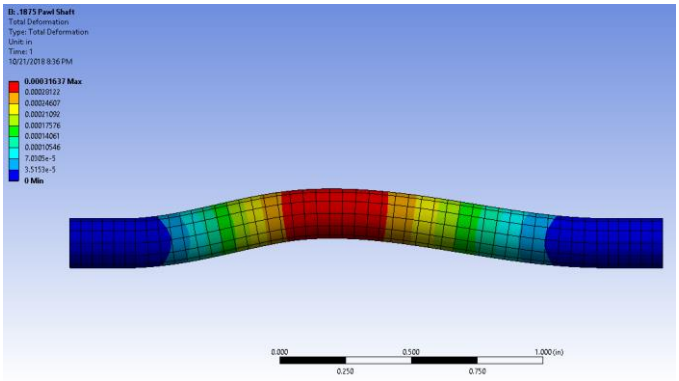

Fig 12. Deflection in Pawl Shaft

B. Vision

The submarine utilizes a low-light, 1080p camera with a 110° horizontal field of view to capture images. The camera stream is then sent from Anchor to Minion for processing. The stream is retrieved from the sub in the same manner as vision data is sent from Minion to the ground station, as described in Appendix K**.**

After the sub has been deployed, the submarine's vision module will begin detecting rings in the camera frame. This is accomplished using a Faster R-CNN Inception V2 [5] network. A further description of this network's structure can be found in Appendix K. This network was trained on an image set of 2400 samples at various angles, distances, and lighting conditions. The results of this training were superb. In a validation set of 490 images, the retrained Faster R-CNN network achieved a mean intersection over union (mean IoU) score of 0.901. An additional detection validation criterion was set such that the proposed bounding box could take up no more than ¼ of the overall size of the image and that the detection confidence needed to be over 70%. Even with this additional bound, the network was capable of detecting the ring in over 90% of the validation images. This was also able to be achieved while running at a framerate of ~10 FPS on a GTX 1080. This network was very robust to most conditions including distance

extremes, oblique angles, and lighting changes, with some positive detections shown in Fig 13.

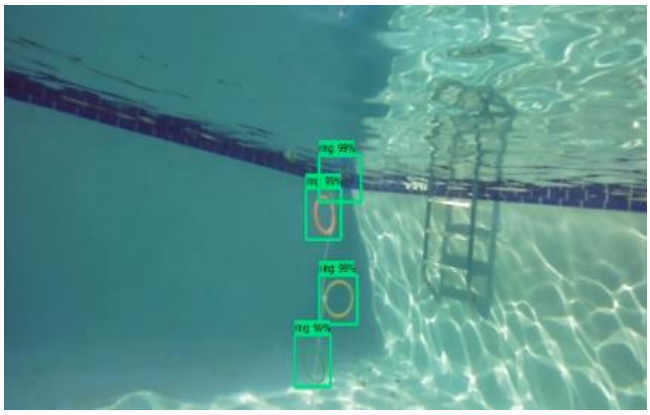

Fig 13. Positive detection of four rings in the same frame

False positives were seen with the network confusing a yellow ring with the yellow tether from Anchor. This may be seen in Fig 14. However, this problem only surfaced when the tether was wrapped in a loop in front of the camera. This problem was solved with the addition of the spool tensioner. While the tether could end up in the camera frame while attempting the ring recovery task, it would be unable to wrap into a ring-like shape due to the constant tension kept on the tether.

Fig 14. Incorrect detection due to the tether looping in a way that looks like a ring

After positive detection of a ring is obtained, the sub uses the rings position in the camera as feedback for a position controller. The area of the bounding box is utilized to estimate the distance from the sub to the ring. The aspect ratio is used to estimate the offset angle of the sub, in both the vertical and horizontal directions from the ring. If the ring drops out of frame, the sub is commanded to maintain the commands that were previously given. This is maintained until positive detection is regained or after several empty frames have elapsed. If the system has had too many empty frames then the platform will being searching for the ring again.

REFERENCES

[1] *"Three Inch High Performance Motors," AmpFlow, [Online]. Available: http://www.ampflow.com/three_inch_high_performance_motors.htm.*

[2] MatWeb, "Aluminum 6061-T6; 6061-T651," [Online]. Available: http://www.matweb.com/search/DataSheet.aspx?MatGUID=b8d536e0b9b54bd7b69e4124d8f1d20a.

[3] BlueRobotics, "Fathom Slim Tether Specifications," [Online]. Available: http://docs.bluerobotics.com/fathom-slim/#specifications.

[4] MatWeb, "303 Stainless Steel," [Online]. Available: http://matweb.com/search/DataSheet.aspx?MatGUID=4041fb2dafde40549e59f4018b7571b8.

[5] Ren, S., He, K., Girshick, R. and Sun, J. (2017). "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6), pp.1137-1149.

Appendix C: Bodyguard II Turret

Grady C. Delp, Juan L. Halleran, David J. Thompson

I. Introduction

For the 2018 RobotX challenge Team Minion has completely redesigned the Bodyguard racquetball turret from the 2016 RobotX Challenge. Taking the undesirable behavior and maintenance requirements of the previous design and eliminating them in the updated version, this new design uses counterrotating flywheels to propel the racquetball to the target and is able to pan and tilt independently from the rest of the boat. The system has an onboard Jetson TK1 for vision processing and active aiming adjustment, as well as an isolated power system, which makes the subsystem capable of operating independently from the boat, save two easily pluggable connections to the boat's computer network and E-Stop circuit.

II. Hardware

Even when only considering the hardware components of the system, Bodyguard II serves as a vast improvement over the Bodyguard turret from 2016. This can be seen through numerous aspects of the design and manufacturing strategy, the selection of components, and the overall abilities of the system.

A. Design Strategy

Bodyguard II serves as a departure from its predecessor in nearly every way, with a render displayed in Fig 1. Bodyguard I utilized a compressed-air cannon to fire racquetballs, with a relatively tight tolerance on a gravity-fed reloading mechanism, and brushless gimbal motors for stabilization and aiming.

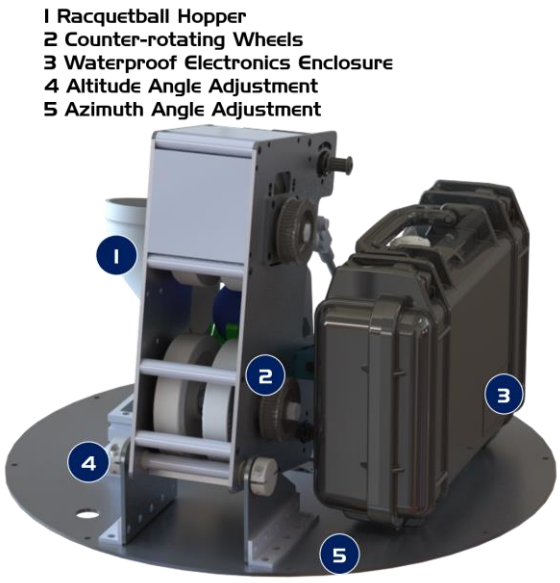

Fig 1. Bodyguard II Racquetball Turret

The compressed-air cannon has been replaced with sets of counter-rotating wheels, through which racquetballs are fed by a HiTec D646WP waterproof servo. This eliminates many of the previous internal complaints of the system, including that it required an auxiliary control box to regulate compressed air between multiple pressures, and that it required a cumbersome method to recharge the compressed air tanks when on-site at the competition. By contrast, the counter-rotating wheels are powered electrically, and the on-board power system even makes it independent from the ASV power supply. This reduces the number of connections required when mounting Bodyguard II to the payload tray of the ASV to one. While the counter-rotating wheels do not have a barrel through which the ball would achieve increased accuracy, like a compressed-air cannon would, it is the experience of several of our team members, who have built similar mechanisms in prior robotics competitions, that they perform on-par with compressed-air methods of firing.

The brushless gimbal motors used to stabilize and aim the 2016 iteration of Bodyguard have been replaced with servo-controlled four-bar linkages. The major unexpected downside of the gimbal motors, that caused Bodyguard I to underperform, was that the system would need to be almost exactly balanced around the center of rotation to operate competently. As implemented, there was a large unbalanced load placed on the gimbal by the tubing connected the auxiliary control box with the turret, which would require more torque to maneuver than the gimbal motors can provide.

To mitigate the issues that were seen with the previous gravity-feed mechanism, the hopper that stores racquetballs prior to firing is looser and allows them to fall more freely. The hopper is illustrated with a cutaway view in Fig 2.

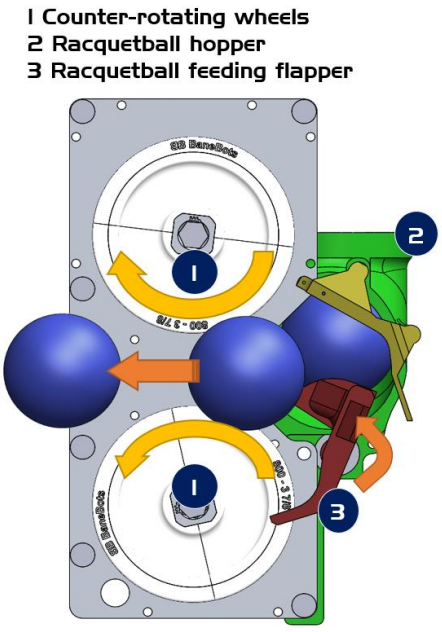

Fig 2. Cutaway view showing operation of Bodyguard II

In the initial design, Bodyguard II was designed to use the same Volz DA-30 servo motors utilized by the azimuth assembly on the 2018 propulsion system. This was to allow better synergy within the Minion ecosystem: one type of servo would allow interchangeability of the components and decreased quantities of spares. However, due to the long lead times that came with those components, they were swapped with a high-torque model of servo by HiTec, the D845WP. This required the design of 3D printed adapters to affix them to the predesigned mounting holes.

These servos do not necessarily provide the necessary torque to drive the turret directly through its ranges of motion. However, by synthesizing four-bar linkages to adapt the 180-degree stroke of the servo to the 60 degrees of stroke in the azimuth direction, or the 20 degrees of stroke in the altitude direction, the servos possess a mechanical advantage over the turret.

The angle of 60 degrees of azimuth control is to accommodate one of the design requirements for Bodyguard II. When designing this turret, it was determined that the ASV would be able to maintain a heading hold within 30 degrees of the desired angle. The center of the 60-degree stroke lies 30 degrees offset from forward, off the port bow of the vessel. With this angle, and the designed location of the turret, the racquetballs fired from the turret would have to veer 15 degrees off of the firing plane to strike the forward sensor array, which may cause physical damage to the sensors. This amount of curve was deemed unlikely to occur over the 25cm that stretch between the firing position and the forward sensor array.

The electrical system was designed to be robust, but capable. On-board computer vision is accomplished using a Jetson TK1 SOC development board, with video capture coming from a Microsoft LifeCam. This computer was chosen due to the availability, and the camera was chosen due to the team's experience in making water-resistant housings for this model in the past. Off-the-shelf motor controllers and inexpensive 24V DC brushed motors are used for spinning the counter-rotating wheels.

B. Construction

Bodyguard II was designed to be largely constructed of three categories of components:

- commercially available off-the-shelf components
- 3D-printed components that can be produced in-house
- Laser-cut 1/8" aluminum plate

The small remainder of the components are simple geometries and hole-patterns that can be cut or drilled by hand, or pieces of bar-stock that can be cut to length by hand.

All 3D printed components were manufactured in ABS by an Ultimaker 2+. ABS was chosen primarily due to its ability to withstand weathering in the outdoors.

C. Electrical Design

Bodyguard II was designed to be electrically independent of the ASV's power systems. This was to reduce the cost required to source components that would be capable of powering the turret, including the motor controllers, relays, and voltage regulators for the servo and on-board computer and controller systems. Based on the battery configuration, the Minion ASV may operate between 24-30 and 48-60 volts. Typically, we specify regulators that can handle both of these ranges without issue, but for the high stall currents that the motors selected may draw, specifying relays and motor controllers that could handle this wide input voltage range proved to be an expensive endeavor. Due to this, power is supplied independently from the ASV's other power systems via a 6S lithium-polymer battery, which typically supplies voltages between 22.2 and 25.2V. Due to the ability to limit power draw from the motors, and the fact that the turret will only be searching for the Detect and Deliver target when triggered to do so by Minion, idle power draw will be minimized, and having the small independent power supply is not considered to be an issue.

Power supplied to the motor controllers first passes through a set of relays that are controlled in parallel, with one relay operating each of the two motor controllers. The relays' coils are driven by a low-voltage signal that is in parallel with a safety control on the Bodyguard control board and is in series with a parallel line of the E-Stop circuit on the Minion ASV. This configuration allows the Minion ASV to continue to operate if the turret suffers a fault but prohibits the turret from functioning if the boat enters an emergency-stop state. It also requires the boat be in an active state, or else power will not be supplied to the motor controllers. This is similar to how the primary motor controllers for the ASV's thrusters are supplied power.

The HiTec D845WP servos are driven by an mBed LPC1768 microcontroller, integrated into a PCB similar to what drives the servos for the propulsion azimuth servos.

All of the electronics are housed within a watertight Pelican 1170 case. Team Minion regularly uses Pelican cases for creating watertight electronics housings, and this is a fitting application for that style of enclosure.

III. SOFTWARE

The software of the turret is all on an onboard Nvidia Jetson TK1 where it uses a camera to sense the target and then commands the servos to place the target in the center of the camera view. To calculate the elevation, angle a Minion Core message is sent to the perception module to find the distance to the target using the onboard boat LIDAR.

A. Vison

The turret system uses a Microsoft LifeCam to collect images and pass them to a Jetson TK1 that is running OpenCV. A sample image from a mockup target is shown in Fig 3. The vision uses collects images when the boat aligned with the target and will threshold and greyscale the images. It then uses a trained Histogram of Gradients (HOG) to find the location of the square in the frame. Since we are asking the boat LIDAR for distance the vision system is solely dedicated to the x-angle of the turret.

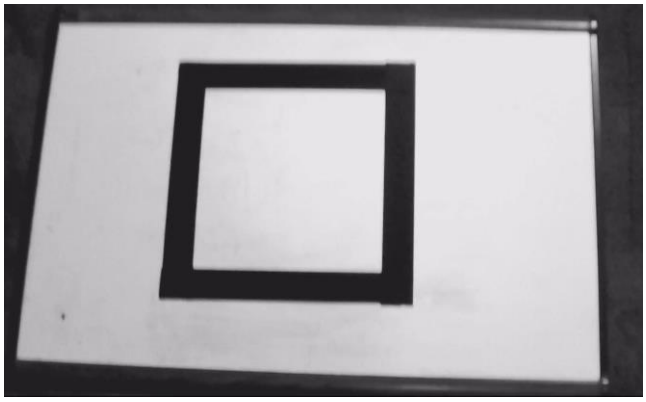

Fig 3. Testing mockup

B. Controls

The control software to align the turret to the target is dependent on the angle found by the vision and the distance given by the boats LIDAR. The distance is directly plugged into the y-angle of the turret because that distance value is corrected for boat pitch and roll in the Perception module. This will keep the turret elevation in the correct position to compensate for varying distances. The pixel error found on the vision side is then used to move the x-angle of the turret center on the target. Since the turret is driven by servos all error values can be directly plugged into the servos since they have an internal position controller.

Appendix D: Propulsion

Nate D. Bloom

I. Introduction

Following the 2016 RobotX Competition, Minion's propulsion system was re-evaluated, finding three key weaknesses. First, the RDP (rim-driven propeller) thrusters were nearing their end of life. Second, Minion's maneuverability was inadequate. Third, deploying and retrieving Minion required team members to manually raise or lower the RDP thrusters. The propulsion system was redesigned to address these weaknesses.

The overall layout of the new propulsion system is similar to the old propulsion system. There are two "motor pods," one at the aft of each pontoon. The layout of each pod is shown in Fig 1. Each motor pod has one azimuthing Copenhagen Subsea VM RDP Thruster. This thruster is the newest iteration of the Torque-Jet thrusters on the previous system. Azimuthing is accomplished with Volz DA-30 Servos. These servos produce enough torque at high rotational speeds and are IP67 rated with additional testing for saltwater spray. Beaching is accomplished with a Linak LA-36 linear actuator, a component the team has prior experience with in hydrophone deployment.

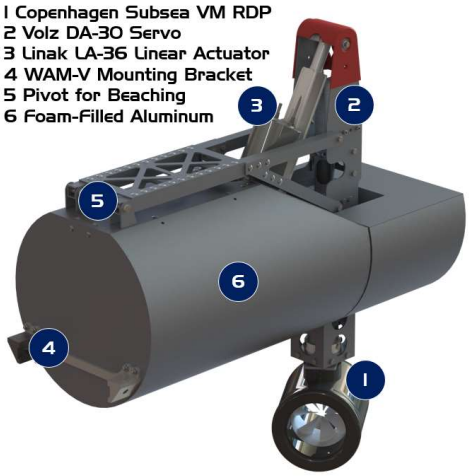

Fig 1. Azimuth and Beaching Enabled Propulsion System

II. Requirements

Requirements for the updated propulsion system were based on the strengths and weaknesses of the 2014-2016 propulsion system. The requirements and justifications are as follows.

1) The system shall attach to the WAM-V boat using the attachment points detailed in the "RobotX Guide WAM-V Propulsion Examples" paper's section on "Other Alternatives." [1]

Since the WAM-V does not include a propulsion system, there is a pivot at the aft of each pontoon. These pivots serve as the mounting point for custom propulsion systems and allow these systems to pitch relative to the WAM-V pontoons. This pitching motion helps to ensure that the propulsion motor remains submerged. The motion is illustrated in Fig 2.

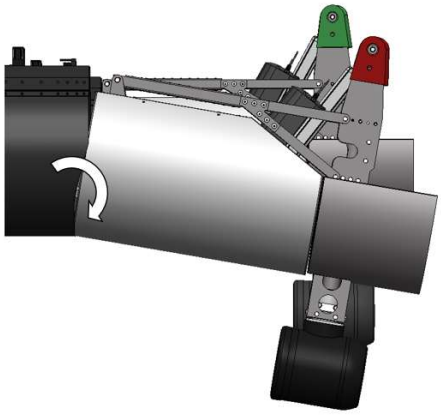

Fig 2. Propulsion Pitch Relative to WAM-V

2) The system shall be positively buoyant.

Requirements 1 and 2 together ensure that Minion fulfills the 2018 Maritime RobotX Challenge Rule 3.1.2. [2] The WAM-V requires additional buoyancy aft of the pontoon, or it risks capsizing.

3) The system shall be compatible with the Torque-Jet RDP thrusters, as well as suitable replacement thrusters.

As mentioned in the introduction to this appendix, the Torque-Jet RDP thrusters were to be replaced. However, the team wished to ensure that the propulsion system could run if there were delays in acquiring the new thrusters, or if one or both new thrusters were to be damaged.

4) The system shall be able to deploy the topmost point of the thrusters to a minimum depth of 4 inches below the bottom of the WAM-V pontoons.

This depth requirement helps to ensure that the thrusters always remain submerged.

5) The system shall be able to deploy the centerline of the thrusters to a minimum of 24 inches aft of the mounting points.

The intent of Requirement 5 is to ensure a sufficient moment arm from the thruster to Minion's CG. This moment arm is key to yaw acceleration.

6) The length of Minion including the system shall not exceed 16 feet.

Florida maritime law requires that vessels larger than 16 feet require a floatation device for each person on board, plus an additional floatation device [3]. This would require that Minion always have a personal floatation device despite being unmanned. To avoid complications with this regulation, 16 feet was accepted as Minion's maximum length.

7) The system shall be able to pan the thrusters +/- 85 degrees from parallel with the WAM-V pontoons.

170 total degrees of rotation with thrusters that can turn in forward or reverse means that thrust can be applied in nearly any direction from a top looking down view. While a range of motion of +/- 90 degrees would be more advantageous, limiting this requirement to 85 degrees improves the availability of off the shelf actuators that meet the required specifications.

8) The system shall be able to retract the thrusters from the water such that the bottommost point of the thruster is above the bottom of the WAM-V pontoons.

Requirement 8 protects the thrusters when trailering Minion or landing Minion on a beach. This prevents the thrusters from striking the ground or the trailer in these scenarios.

9) The system shall be compatible with 20-60V systems.

Requirement 9 ensures that Minion may continue to operate at either 25V or 50V nominal.

10) The system shall be able to operate continuously in the marine environment.

The marine environment is especially harsh on mechanical systems. Actuators need to be waterproof, and all materials need to be exceptionally corrosion resistant.

III. Updated Thrusters

For 2018, team Minion elected to continue with RDP thrusters. RDP thrusters have numerous advantages over electric trolling motors. First, the removal of a central shaft improves efficiency and reduces the likelihood of tangling with seaweed, anchor lines, and other debris. RDP thrusters are also inherently covered in a shroud and typically produce little noise. This minimizes their potential impact on marine wildlife, which is particularly important to the team since dolphins and/or manatees occasionally approach the ASV.

The dated Torque-Jet thrusters Minion used in 2016 were replaced with Copenhagen Subsea VM asymmetric thrusters. These thrusters are based on a similar design but are thoroughly updated. Improvements include inlet and outlet shrouds, revised propeller profiles, and revised internals. Due to these improvements, the Copenhagen Subsea thrusters offer improved efficiency, durability, and thrust capability compared to the Torque-Jets.

IV. Bearing Material Selection

Since all bearings in the propulsion system need to function both in and out of a saltwater environment, all bearing surfaces are static bushings. Bearings with moving pieces are susceptible to salt buildup over time, and underwater bearings do not function well out of water.

Multiple bushing materials were selected, and specific bushings were chosen from these selected materials based on the availability and cost of each.

The selected materials are Igus Iglide T-500, Iglide H-370, and Iglide J. Their applicable properties are shown in Table 1.

TABLE 1

IGUS BUSHING MATERIAL PROPERTIES [4] [5] [6]

Specification	Iglide T-500	Iglide H-370	Iglide J
Water Absorption (% Weight)	0.5	< 0.1	1.3%
Permissible Static Surface Pressure (psi)	21,760	10,880	5,075
Effective Coefficient of Friction	0.09 - 0.27	0.07 - 0.17	0.06 – 0.18

V. Azimuthing

I. *Geometry*

As specified in the requirements, the thruster must sit 4 inches below the nominal waterline. However, there are few high torque electric motors that are made to be used continuously underwater. Therefore, the azimuthing actuator must sit above water and some drive system must connect the azimuthing actuator to the thruster. To maximize efficiency, a direct drive was chosen. An exploded view of the drive shaft is shown in Fig 3.

Furthermore, the thruster must be retractable from the water for beaching. To simplify the direct drive system, the thruster and azimuthing assembly function as a single piece that can be raised and lowered for beaching.

Fig 3. Azimuth Direct Drive Exploded View

To route the thruster cable, the direct drive is achieved with a hollow drive shaft. Adapters at the top and bottom of the shaft couple the actuator to the thruster. The drive shaft is supported by bearings at the top and bottom of the shaft to minimize radial force on the actuator. A section view of the top and bottom azimuth bearing assemblies is shown in Fig 4.

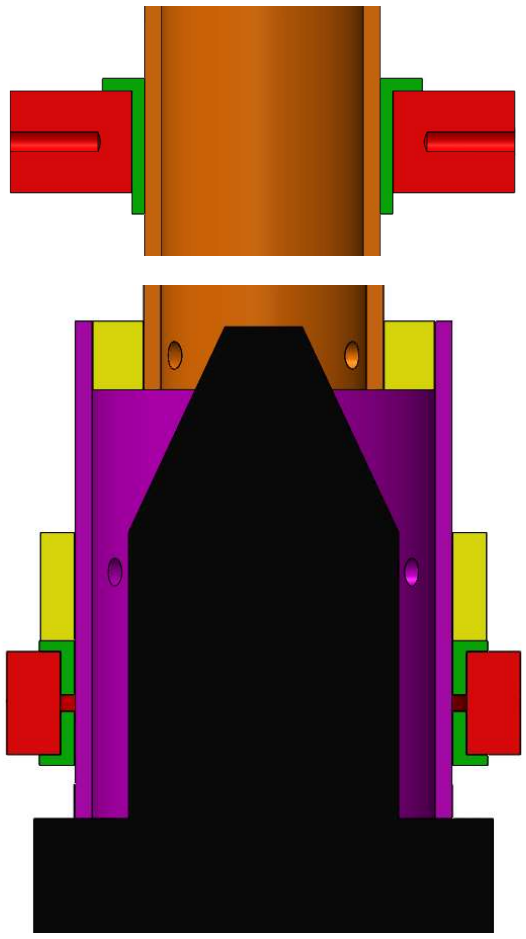

Fig 4. Azimuth Bearing Assemblies

The pieces shown in Fig 3 and Fig 4 are as follows:

- The orange tube is the hollow drive shaft.
- The black pieces are the thruster and azimuthing actuator.
- The green pieces are bushings.
- The red pieces support the bushings.
- The yellow pieces prevents the drive assembly from moving axially.
- The purple pieces couple the thruster and actuator to the driveshaft

II. *Structural Analysis*

To find the torque requirements for the actuator, the worst-case scenario must be considered. For azimuthing, that occurs when there is a maximum force of friction on the bushings. This maximum force of friction most likely results from the thruster's maximum load. To find the friction, the force reactions at the bearings during maximum thrust are found. These are multiplied by the bushing's static coefficient of friction and the radius of the shaft to be converted to the frictional torque opposing azimuthing. This frictional torque is what the azimuthing actuator must overcome. The torque was found using two methods to verify results: hand calculations modeling the system as a simply supported beam and an FEA model.

1) Method 1: Simply Supported Beam

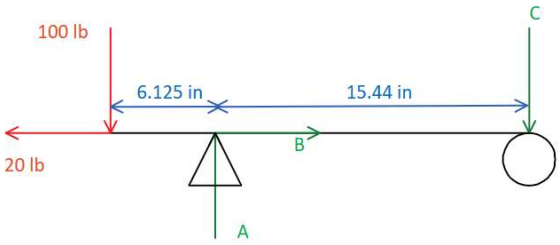

Fig 5. Azimuth Modeled as Simply Supported Beam

$$\Sigma F_x = 0$$
$$\mathrm{B} = 20 \text{ lb}$$

$$\Sigma M = 0$$
$$(100)(6.125) = C(15.44)$$
$$\mathrm{C} = 39.7 \text{ lb}$$

$$\Sigma F_y = 0$$
$$100 + 39.7 = A \rightarrow A = 139.7\ lb$$

$$\Sigma N = 20 + 39.7 + 139.7 = 199.4\ lb$$
$$F_f = \mu N = 0.1 * 199.4 = 19.94\ lb$$
$$\tau = Fr = 65.8\ lb * 1.1\ in = \mathbf{21.9\ in * lbf}$$

2) Method 2: FEA Reactions

For this FEA, a full thruster load of 100 lbf was applied to the azimuth tube model as shown in Fig 6. The red arrow indicates the force, which was applied at the distance of the motor centerline to the motor attachment plate. The 3 Igus bushings were constrained and monitored for their reaction forces.

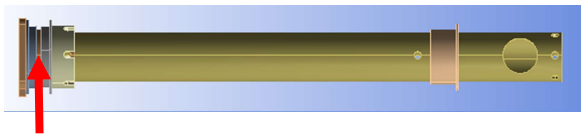

Fig 6. Azimuth FEA Load

This load case resulted in a net 180 lbf reacting from the bearing surfaces. With the 0.1 coefficient of friction mentioned above, a normal force of 18 lbf is present at the outer diameter of the bearings. To overcome this, the thruster must produce enough torque to overcome the normal force and the moment arm of the shaft radius (1.1 in). Therefore, the actuator must produce **19.8 in-lbf** of torque.

3) Evaluating the Methods

Since the methods have very close results, a nominal operating torque of 22 in-lbf was used.

III. *Actuator*

The selected actuator is the Volz DA-30 servo, shown in Fig 7. A summary of the applicable features of this actuator are in Table 2.

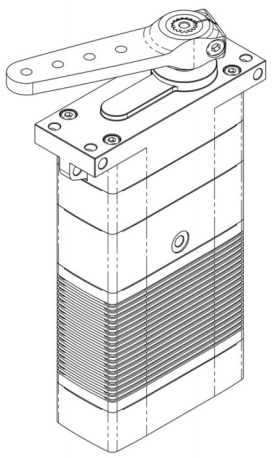

Fig 7. Volz DA 30 [7]

Fig 8. Primary Load Bearing Structure

TABLE 2

VOLZ DA 30 APPLICABLE SPECIFICATIONS [7]

Specification	Value
Max Torque (continuous)	70.8 lbf-in
Max Torque (stall)	141.6 lbf-in
Supply Voltage Range	24 – 32 VDC
Max Travel Angle	±85° = 170° total travel
Environmental Rating	IP67
Salt Water Resistance	>100 hrs salt water spray

The environmental rating and supply voltage clearly meet the requirements, and the built-in position sensing, limited travel angle, and rated salt water resistance are massively advantageous features.

The Volz also meets the torque requirements, with a factor of safety of 3.2 for continuous operation and 6.4 if stalled. While this may appear excessive at a glance, there are many factors which could increase the resisting friction including drag, imperfect alignment, and the surface finish of the driveshaft.

VI. CHASSIS

I. *Design*

Construction of the propulsion system was largely driven by the methods easily available to our team. Embry-Riddle has CNC mills and lathes, but very limited capability in welding thin aluminum or bending sheet metal. Therefore, CNC machining was the primary method of manufacture for parts to minimize outsourcing and associated costs. Most parts are anodized 6061 or 5052 aluminum to prevent corrosion. Where higher strength material is required, 316 stainless steel is typically used. Parts are joined using aluminum rivets or 316 stainless steel bolts.

The main frame of the propulsion system is an assembly of gusseted 1 x 1 x .125’’ square 6061 aluminum tube. This construction was chosen to allow easy mounting of the beaching and azimuth systems. It was also designed to allow any number of construction methods to the pod skin. The chassis is intended to take the greatest part of all loads, to minimize the stress on the skin assembly.

II. *Structural Analysis*

1) Chassis

As stated, the motor pod chassis is designed to take the loads of the system, independent of the skin structure. FEA was run under two full thrust loading cases, assuming full thrust reverse and full thrust sideways. Gravity was also included, and buoyant forces were added where the pod skin meets the chassis. The model and loads are shown in
Fig 9.

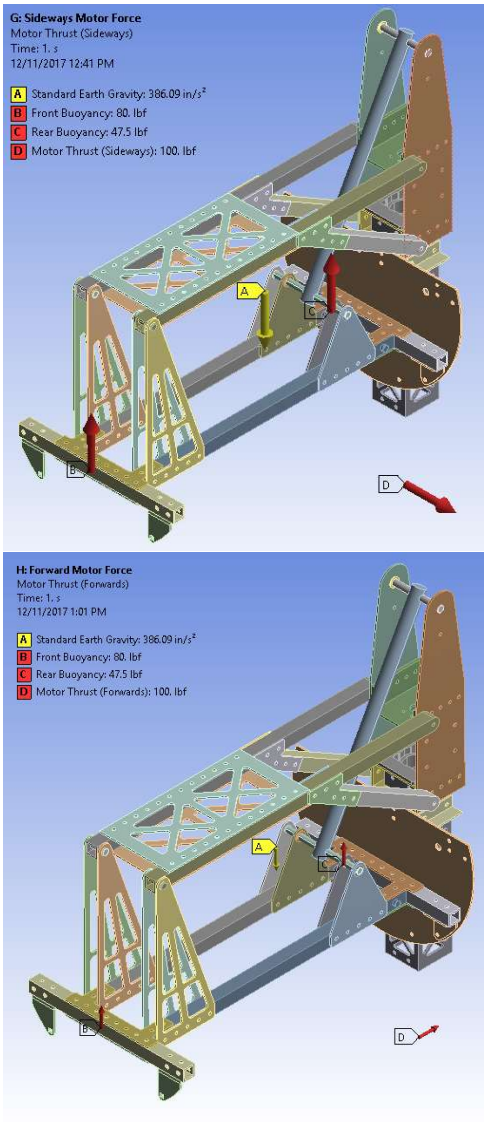

Fig 9. Model and Loads on Propulsion System Chassis

The sideways loading resulted in a max stress of about 25 ksi, in the bottom pin where the Linak attaches. These stresses are well under 316 Stainless Steel's yield strength of 34.8 ksi. [8] The 78.6 ksi stress is a false concentration at an interference where the azimuthing tubing meets the upper arm assembly. Results are shown in Fig 10 and Fig 11.

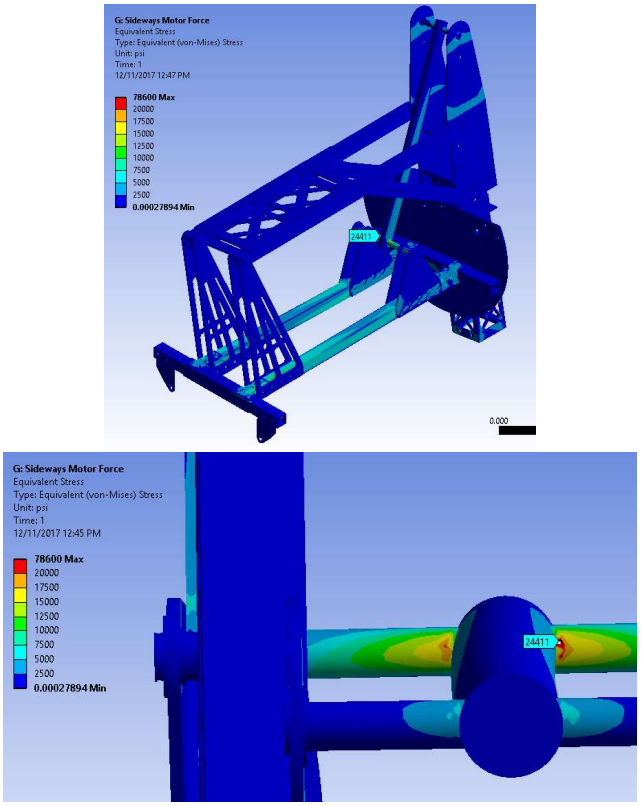

Fig 10. Full Sideways Thrust Chassis Stress

The full reverse thrust load case resulted in max stresses in the 15-20 ksi range where the Linak mounting plates meet the chassis. These stresses are well under 6061-T6's yield stress of 40 ksi. [9]

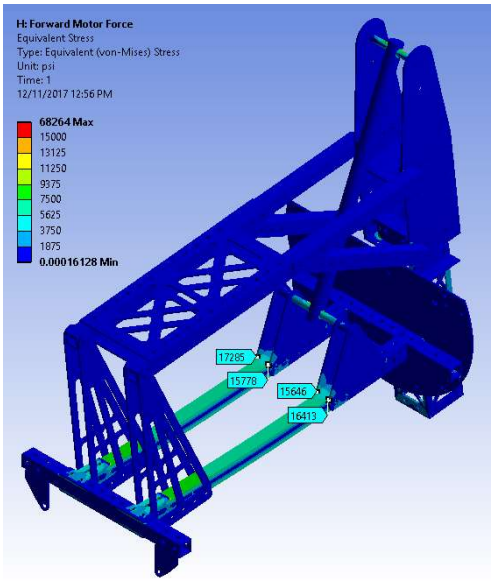

Fig 11. Full Reverse Thrust Chassis Stress

2) Azimuth Structure

The azimuthing portion of the system was independently evaluated under full forwards thrust and full reverse thrust to see the loading the Linak would experience.

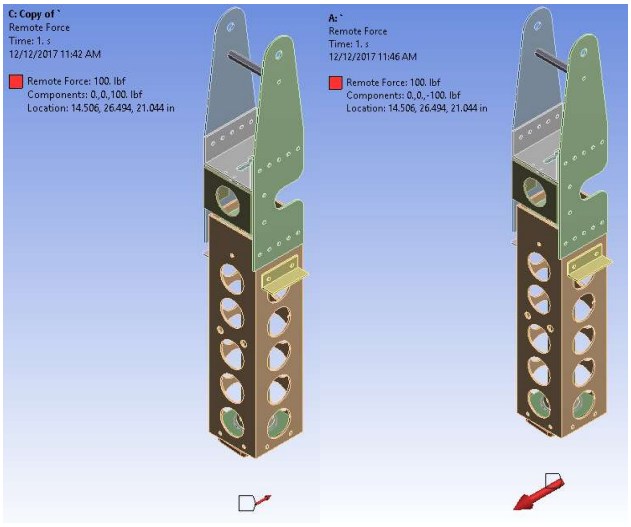

Fig 12. Full Forward Thrust (Left) and Full Reverse Thrust (Right)

Under full forward thrust, the Linak sees about 100 lbf, or 445 N of force. Under full reverse thrust, the Linak sees about 150 lbf, or 667 N. This is significantly less than the Linak's static holding force of 3400 N. This analysis also further proves the structural integrity of the azimuthing assembly as shown in Fig 13 and Fig 14.

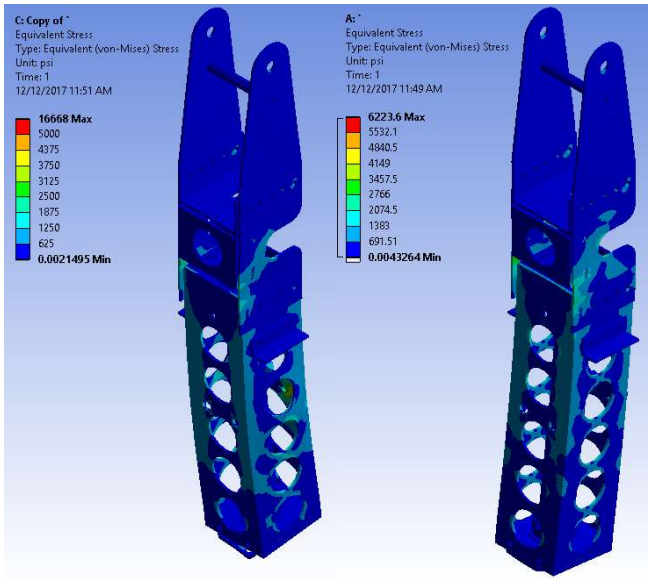

Fig 13. Forward Stresses (Left) and Reverse Stresses (Right)

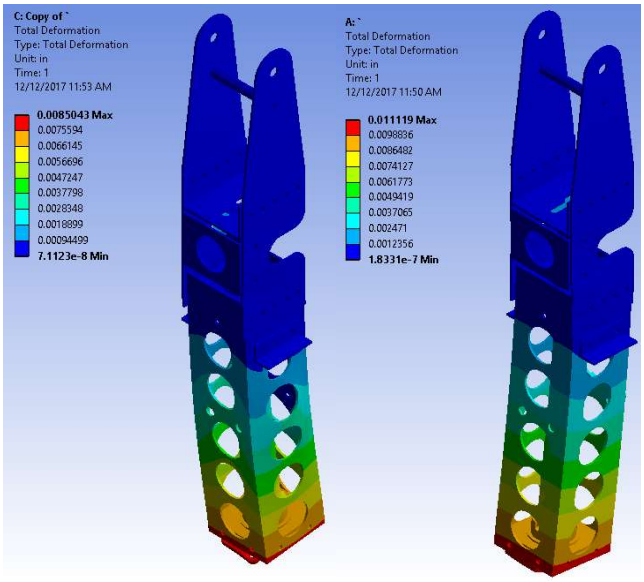

Fig 14. Forward Deflection (Left) and Reverse Deflection (Right)

VII. Beaching

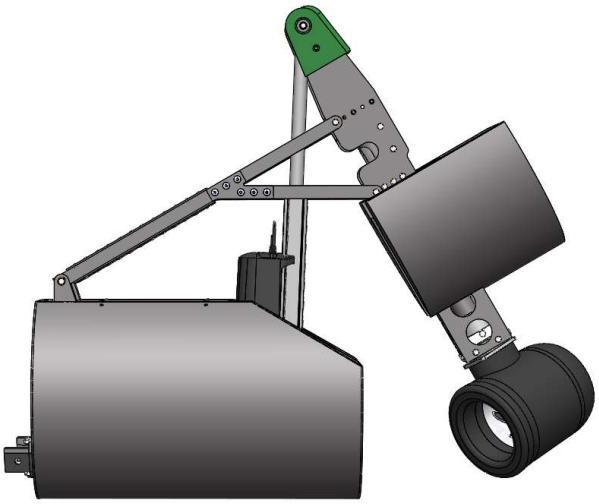

Fig 15. Motor Pod in Beached Configuration

The beaching motion is accomplished with a Linak LA-36 actuator. The Linak was selected due to an IP-66 dynamic and IP-69K static rating, and the team's experience using a smaller Linak in very close proximity to the water. The selected LA-36 also has 400mm of travel, end stop signals, a max actuation force of 2600N, and a static holding force of 3400N. [10] These forces are more than sufficient as proven in the azimuth structural analysis above.

I. *Skin*

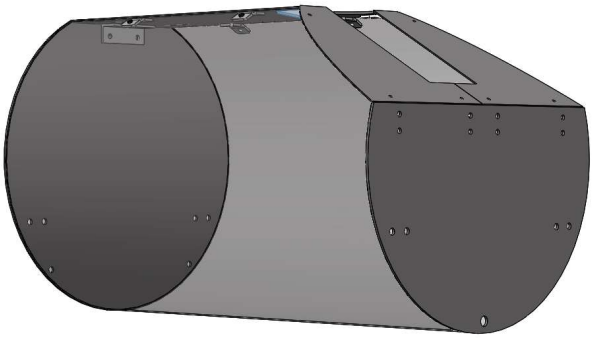

Fig 16. Skin

To contain the motor pod's buoyancy and give visual continuity from the WAM-V's pontoons, most of the motor pods are surrounded by a 16.25-inch diameter aluminum skin. Aluminum was chosen over fiberglass (which the previous propulsion system used) due to fiberglass's tendency to crack and shatter on impacts. This skin is epoxied to the main chassis using 3M EC2216 Scotch-Weld Epoxy. Other fastening methods such as brackets with rivets were not chosen due to increased likelihood of leaks through additional holes.

Buoyancy is accomplished using a closed cell expanding foam fill with Dow Froth-Pak. This ensures that in the case of any leaks, the pod will not fill with water.

REFERENCES

[1] "Maritime RobotX Challenge Primer WAM-V Propulsion Examples," 21 March 2014. [Online].

[2] "2018 Maritime Robot X Challenge Rules and Requirements," 19 September 2018. [Online].

[3] Florida Fish and Wildlife Conservation Commission, "Minimum Required Safety Equipment for Class 1 Recreational Vessels: 16 to less than 26 ft/ 4.9 to less than 7.9m)," [Online]. Available: http://myfwc.com/boating/safety-education/equipment/vessels-16-to-259-feet/.

[4] Igus, "Iglide T500," Igus, [Online]. Available: https://www.igus.com/contentData/Products/Downloads/iglide_T500_FM_USen.pdf.

[5] Igus, "Iglide J," [Online]. Available: https://www.igus.com/_Product_Files/Download/pdf/h370.pdf.

[6] Igus, "Iglide H370," [Online]. Available: https://www.igus.com/_Product_Files/Download/pdf/h370.pdf.

[7] "DA 30 Technical Specifications," Vikz, [Online]. Available: DA 30 Technical Specification. Volz Servos, www.volz-servos.com/English/resources/Downloads/DataSheets/DA-30_Datenblatt_uni_Rev-C.pdf..

[8] "316 Stainless Steel, annealed bar," MatWeb, [Online]. Available: http://www.matweb.com/search/DataSheet.aspx?MatGUID=dfced4f11d63459e8ef8733d1c7c1ad2.

[9] "Aluminum 6061-T6; 6061-T651," MatWeb, [Online]. Available: http://www.matweb.com/search/DataSheet.aspx?MatGUID=b8d536e0b9b54bd7b69e4124d8f1d20a.

[10] "Actuator LA36 Data Sheet," Linak, [Online]. Available: https://cdn.linak.com/-/media/files/data-sheet-source/en/linear-actuator-la36-data-sheet-eng.ashx.

Appendix E: Perception through Object Detection, Mapping and Classification

David J. Thompson and Eric J. Coyle

I. INTRODUCTION

Team Minion designed a perception module to perform the necessary functions of object detection, mapping and classification. Object detection proceeds through the use of four on-board LiDAR sensors whose returns are mapped to an occupancy grid for object extraction. The contours of each object are then represented by a single planar polygon. The Minion platform then maps objects by combining the object contours identified from the occupancy grid with those previously known to the system using Boolean polygon operations. Lastly, objects are classified using unique spatial and reflective properties through a priori training and a Multi-Variate Gaussian (MVG) classifier.

The output of this module is used by three corresponding modules in operation on the Minion platform. The vision module uses object detections to associate object color with the objects, the path planning module utilizes the set of mapped objects in order to plan and navigate safely, and the MinionTask module uses the set of classified map objects as cues to complete the individual challenges.

This appendix contains several excerpts from a pending publication with the Journal of Oceanic Engineering. All sections using this work are denoted as such.

II. OBJECT DETECTION

The following section is an excerpt from a pending publication with the Journal of Oceanic Engineering.

A. Coordinate Transformation

LiDAR returns, such as those on Velodyne LiDAR sensors, are natively provided in a local reference frame in spherical coordinates. The elevation angle is a constant, as it is related to the fixed mounting of each laser. The rotational angle is provided by an encoder built into the sensor, and the radius is the distance measured by time-of-flight. It should be noted that since water absorbs the laser light, only low intensity returns are obtained from the water's surface. Thus, water is easily ignored using an intensity threshold on the 0-255 intensity output given by the LiDAR sensors.

When using multiple LiDAR sensors, it is necessary to convert their returns into a common reference frame for processing. A global frame not only permits the use of multiple sensors but makes mapping more straightforward and efficient by preventing the need to continuously compute point locations in a moving reference frame. To that end, a northing-easting-down (NED) frame will be used here. LiDAR returns are first converted from spherical coordinates to homogeneous coordinates using:

$$p_{VEL} = \begin{bmatrix} R\sin\omega\cos\alpha \\ R\sin\omega\sin\alpha \\ R\cos\omega \\ 1 \end{bmatrix}, \quad (1)$$

where P_{VEL} is a single LiDAR return in the Velodyne's reference frame, R is the distance measurement reported by the sensor, α is the rotational azimuth angle reported by the sensor, and ω is the elevation angle of the laser. This transformation is illustrated below in Fig 1 as provided by the manufacturer [1].

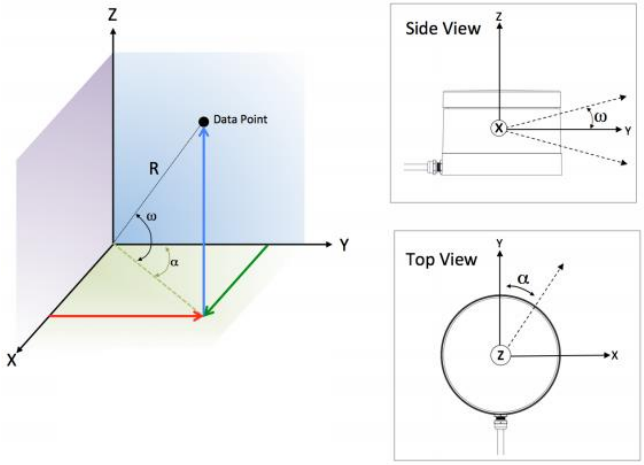

Fig 1. Coordinate Frame used by Velodyne Sensors [1]

Then, using the known mounting location and orientation of each LiDAR, the points are moved into a body Forward-Right-Down (FRD) reference frame using:

$$p_{FRD} = T_{FRD}^{VEL} p_{VEL}, \quad (2)$$

where T_{FRD}^{VEL} is the homogeneous transform from the Velodyne LiDAR's local frame into the local FRD reference frame of the vessel. This frame is represented on the platform in Fig 2**Error! Reference source not found.**. The FRD frame is centered between the Minion's two battery bays, which are not shown in Fig 2.

Similarly, the LiDAR returns are then moved into the NED global reference frame using the TORC PinPoint GPS/INS reported state of the vessel, i.e. the NED location and Euler angles. This is given by:

$$p_{NED} = T_{NED}^{FRD} p_{FRD}, \quad (3)$$

where T_{NED}^{FRD} is the homogeneous transform from the local FRD frame to the global NED frame.

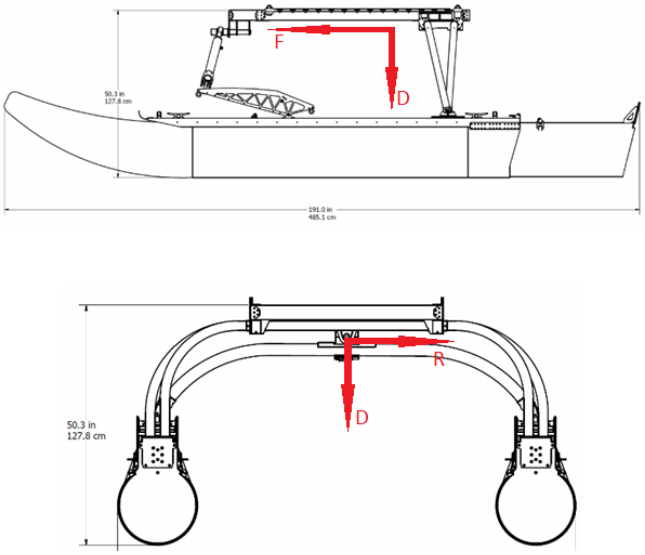

Fig 2. Platform FRD frame. Top – Platform port-side profile. Bottom – Platform rear profile

This approach is only valid for obtaining p_{NED} for a single LiDAR return. However, the acquisition rate of individual LiDAR returns is over 300kHz, while the vehicle state is updated from the GPS/INS at a rate of only 100Hz. Using only the most recent vehicle state to determine T_{NED}^{FRD} could lead to significant errors in the calculation of p_{NED}, particularly if the vehicle is moving at a high linear or angular speed. Since the Velodyne sensors can return a GPS time stamp for each LiDAR return, T_{NED}^{FRD} is uniquely determined for each LiDAR return by linearly interpolating the vehicle state based on GPS time stamps. It should be noted that a full sweep of the scanning LiDAR (i.e. a single LiDAR scan) occurs at a rate of 10 Hz. To reduce the processing load, all points are retained in a memory until a full scan has been completed by each of the onboard Velodyne sensors and then fed into the object extraction and mapping algorithms. The resulting calibration is accurate to approximately 5cm, as shown in the example of Fig 3 where the returns from each sensor is represented by a different color.

B. Object Extraction

Object extraction is performed by first fitting raw LiDAR returns from a single scan of each sensor into a 3D occupancy grid structure. The occupancy grid is referenced to the global frame, but the grid range is limited to tunable area around the vessel. Here, the $MxMxN$ grid matrix has a size defined by:

$$M = 1 + (d/\delta), \tag{4}$$

and

$$N = 1 + (h/\delta), \tag{5}$$

where the distance d is the maximum range covered by the occupancy grid, δ is the resolution of each grid cell, h is the height of the grid, and the vessel is located in the center of the grid at all times. The occupancy grid indices, denoted p_{idx}, of return p_{NED}, are computed by:

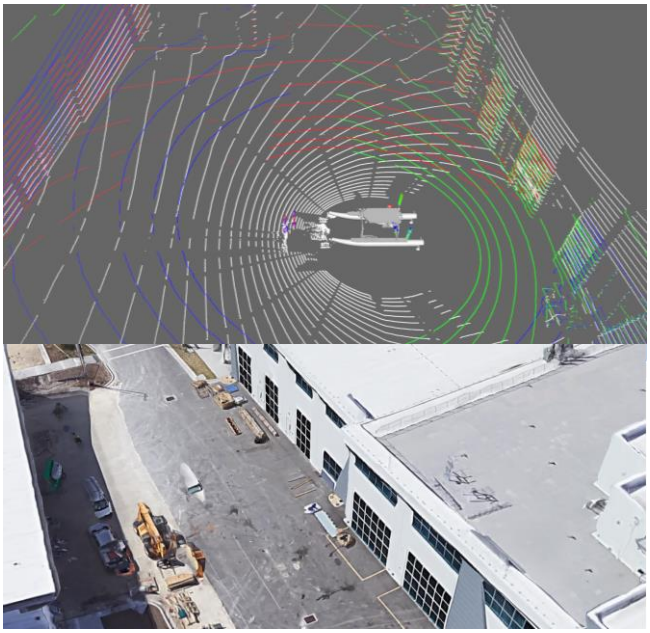

Fig 3. Top: A sample of LiDAR returns from the port (blue), starboard (red), bow (white), and aft (green) Velodyne sensors. This plot shows how accurately the sensors are calibrated to the same NED frame. [2] Bottom: Satellite view of the same area.[3] Note: None of the vehicles or construction equipment were present when the LiDAR data was taken.

$$p_{idx} = round\left(\frac{\acute{p}_{NED} - q_{NED}}{\delta}\right) + \frac{[M \quad M \quad N]^T}{2}, \tag{6}$$

where $\acute{p}_{NED}$ is a 3x1 vector comprised of the first three elements of p_{NED}, $round(x)$ rounds all elements of x, and q_{NED} is the current NED location of the USV. This equation can easily be inverted to give NED location for any indices in the grid.

The approach thus far is limited to using LiDAR returns from the most recent LiDAR scan from each sensor. However, a single scan may not be sufficient to detect all or even most of the geometry of a maritime surface object due to gaps between the LiDAR lasers.

To address this, a temporal decay of grid cells is used, which also allows object locations to slowly change over time. Similarly, the temporal decay alloys false positives from the LiDAR to be removed from the map after a period of time. While false positives are rare, certain conditions can increase the frequency of false positives, such as white caps on the water's surface, as noted in [4], or increased particulates in the water. False positive LIDAR returns could then lead to detected objects in areas that could actually be traversed. This temporal decay is implemented by first tuning η_{max}, which is the maximum allowable age of a LiDAR return in milliseconds. When a LiDAR return is used to fill an occupancy grid cell, it is assigned a current age of $\eta = 0$. For each subsequent scan that is processed, any grid cell that is not updated by a newly received LiDAR return has its age incremented by the elapsed time since the

previous scan was received. Once the age of a grid cell meets the criteria $\eta > \eta_{max}$, the cell is set to empty. The choice of η_{max} is a tuning parameter and for this sensor configuration was selected to be $\eta_{max} = 4000$ ms.

One popular application for LiDAR mapping is the use of Octomaps [5]. This method reduces the size of point clouds with an octree structure such that 3D maps may be retained. However, the retained information of Octomaps is most useful when the environment contains stationary objects. Even a shoreline changes location between visits as the tide may be at a different height. Indefinitely retaining 3D information for waterborne objects can be problematic as these objects tend to have variations in both position and orientation.

Using a 3D representation of the entire environment is likely overly complex for solving the 2D path planning problem of most USVs. Realize that while the third dimension is useful as classification features (to give features such as height and surface area), it is rarely needed for path planning. This is because unlike a ground environment where there are plentiful overhead obstacles such as foliage, signs, lights, and overpasses, a maritime environment generally only has bridges that create overhead obstacles. A 2D map will inherently use less memory than a 3D map, even when using specialized 3D structures such as octrees. Thus, the grid can be flattened to 2D for navigation purposes. To this end, the 3D occupancy grid is first flattened to 2D, resulting in a binary matrix. While object segmentation could then be performed by clustering algorithms such as Euclidean Clustering [6][5] or K-Means [7], these algorithms can be computationally intensive and may require the number of objects to be pre-determined.

Here object extraction is performed using the pixel following method, from computer vision, which is fully described in [8]. Any holes in objects are ignored, so that only the outer object boundaries are retained. The coordinates of these contours are moved into the NED frame using the previously discussed operations in (6), resulting in a list of objects $A = \{a_1, \quad a_2, \quad \cdots \quad a_n\}$. Where each object $a_1 = \{x_1, x_2, \ldots\ x_n, y_1, y_2, \ldots\ y_n\}$ is defined by a set of NED vertices, x is a northing coordinate and y is an easting coordinate. While this paper does not focus on classification of objects, it should be noted that any 2D or 3D occupancy grid cell within the bounds of a_1 and $\eta > 0$ can be used to compute spatial characteristics of the object such as size and surface area.

The pixel following method is not necessary when using the 2D occupancy grid for path planning directly. However, the complexity in planning increases when using voxels/occupancy grids rather than polygons. This approach trades the unbounded problem of path planning using the voxels/occupancy grid for the bounded complexity of combining detected polygons with mapped polygons. This additionally opens up the system to methods that don't utilize grid-based planning.

C. Point Decimation

While the list of objects A has been created, it is beneficial to reduce the number of points that represent the polygon a_1 to improve path planning computations. To accomplish this, the Ramer-Douglas-Peucker point decimation algorithm is used. This algorithm uses an iterative method to reduce the number of points on a curve or polygon to find a similar polygon subject to a perpendicular distance constraint [9]. This distance is treated as a tunable value, but in general should be at least as large as the grid resolution δ. Setting the value too high will result in a loss of object resolution to the point of distortion.

III. Mapping

A. Mapping process

The following section is an excerpt from a pending publication with the Journal of Oceanic Engineering.

Recall that the list of polygon objects extracted from the occupancy grid is defined in the NED frame and denoted $A = \{a_1, \quad a_2, \quad \cdots \quad a_n\}$. This set provides a detailed description of the local area around the vessel, but vessel operations are likely to require an extended map of the area for both path planning and tasking. Thus, it is desired to find a set of mapped polygon objects at time t_k, denoted $B(t_k)$. While it may seem logical to simply union A with $B(t_{k-1})$ to give $B(t_k) = A \cup B(t_{k-1})$, this is impractical for several reasons. First, it is likely that part or all of some objects will be already represented in A and $B(t_k)$. Second, the polygon boundaries of A are more current and likely more accurate than those of $B(t_{k-1})$ for most objects. Lastly, some objects may be present in $B(t_{k-1})$ and not A due to the range of the LiDAR sensors.

To address this issue, a visibility horizon is defined in the FRD reference frame and denoted by a polygon boundary. The visibility horizon, denoted P_{FRD}, is said to contain the area around the vessel where there is sufficient LiDAR return density to trust the current information in A over $B(t_{k-1})$. The vertices of the visibility horizon can then be moved into the NED reference frame using the homogeneous transform T_{NED}^{FRD}. This results in

$$P_{NED} = T_{NED}^{FRD} P_{FRD}. \tag{8}$$

The portion of each object $a_j \in A$ that should be mapped is then defined by the intersection of a_j and P_{NED}. This results in a new set $\tilde{A}$ defined by:

$$\tilde{A} = \{\tilde{a}_1 \quad \tilde{a}_2 \quad \cdots \quad \tilde{a}_n\} = \{a_1 \cap P_{NED}, \quad a_2 \cap P_{NED}, \quad \cdots \quad a_n \cap P_{NED}\}. \tag{9}$$

Similarly, P_{NED} should be removed from each polygon $b_i \in B(t_{k-1})$ using a polygon subtraction. This results in:

$$\tilde{B}(t_{k-1}) = \{\tilde{b}_1 \quad \tilde{b}_2 \quad \cdots \quad \tilde{b}_m\} = \{b_1 - P_{NED}, \quad b_2 - P_{NED}, \quad \cdots \quad b_m - P_{NED}\}. \quad (10)$$

The process to determine the current map $B(t_k)$ is then simplified to the polygon union of all objects $\tilde{A}$ with the objects in $\tilde{B}(t_{k-1})$. This is defined by the following polygon union

$$B(t_k) = \tilde{A} \cup \tilde{B}(t_{k-1}). \quad (11)$$

1) Assigning Class to Mapped Objects

While $B(t_k)$ creates a set of polygon objects, with a tunable number of vertices for path planning, the current formulation does not show how to determine the class of objects $b_i \in B(t_k)$. To do this the class of any object $b_i \in B(t_k)$ and $b_i \in \tilde{B}(t_{k-1})$ should be the same at time t_k and t_{k-1}. Similarly, the class should be the same for any $b_i \in B(t_k)$ as a_j if $a_j \cap P_{NED} = b_i$. The only remaining case is that $b_i \in B(t_k)$ is the union of one or more objects from $\tilde{A}$ and $\tilde{B}(t_{k-1})$. In this case the class of b_i should be set to the most prevalent class among the polygons in $\tilde{A}$ that have a non-zero intersection with $b_i \in B(t_k)$. The result is that any polygon in the map $B(t_k)$ will have an associated class if it has ever been classified before, and that its class information can be re-evaluated only if new sensor data is sufficient to change its classification.

B. Object IDs

For each object in the map, Minion retains object information that includes the known extents of the object, the classification history, the class label, and an object ID that is unique to the object. When a new set of mapped objects are found, the extents of the newly discovered objects are compared to the previous set of mapped objects to determine if the object is new or had been previously identified. Mathematically this is implemented by assuming that object extents will not change from one map iteration to the next by more than 1m in distance. If matching extents are found, the classification history and object ID are passed from the previous iteration of the map onto the object from the current iteration. Object IDs are then used by the task tracker to ensure that Minion continues to interact with the same object throughout the execution of an individual task and for fusing vision data with perception.

IV. Object Classification

A. Minion Classification

Classification is performed on every object in the visibility horizon. This is accomplished by extracting a feature vector from the 3D and 2D occupancy grid cells. The feature vector F is defined as shown below:

$$F = [F_1, F_2, F_3, F_4, F_5, F_6, F_7, F_8, F_9, F_{10}] \quad (12)$$

where:

$F_1 - Max\ Intensity\ of\ Object\ Returns$
$F_2 - Min\ Intensity\ of\ Object\ Returns$
$F_3 - Average\ Intensity\ of\ Object\ Returns$
$F_4 - Standard\ Deviation\ of\ Intensity\ of\ Object\ Returns$
$F_5 - Max\ Height\ of\ Object$
$F_6 - Total\ Number\ of\ Filled\ Cells\ in\ the\ Object$
$F_7 - Polygon\ Perimeter$
$F_8 - Polygon\ Area$
$F_9 - 2D\ Minor\ Axis\ Length$
$F_{10} - 2D\ Major\ Axis\ Length$

To classify these features the team considered the use of two classifiers: a support vector machine (SVM) classifier and a Multi-Variate Gaussian (MVG) distribution. While both of these classifiers are considered to have low real-time computational costs, SVM is typically more robust due to a variety of tuning parameters and the assumptions made by MVG. The research team did in fact find SVM to give better classification performance as evidenced by the table below:

Table 1: Classification accuracies by class, mean classification, and misclassification rate

Class	SVM	MVG
Taylor Made Sur-Mark Can Buoy	99.1%	95.9%
Competition Light Tower	99.8%	99.6%
Competition Dock	100%	100%
Competition Detect and Deliver	100%	100%
Taylor Made A3 Black Buoy	98.4%	90.4%
Taylor Made A7 Black Buoy	94.6%	83.7%
Mean Classification	98.7%	94.9%
Misclassification Rate	1.3%	5.1%

Despite the better classification performance with SVM, the SVM implementation does not natively allow for objects which the system has not classified to be left as unknown. Placing a threshold on confidence and training for unknown objects are both possible solutions to this issue with SVM, but neither could be completed in time for competition.

Instead the MVG classifier, which is itself able to achieve a high accuracy for competition objects, is utilized and false positives are reduced by requiring a minimum confidence before trusting the predicted class label. Furthermore, the team utilizes a classification filter to prevent objects from switching class based on an infrequent misclassification or unknown label. This is implemented by tracking the classification history in the form of a counter on each possible class label. All objects are initially given the class label of "unknown." Once the object has been classified a minimum number of times, and minimum percentage of its classification history is of the same object class, then the class label is updated. Once the object label is switched from "unknown", the system does not allow re-classification of the object. Since identifying objects of a specific class is a common cue used by the tasking software, MinionTask, this prevents the loss of pertinent cues to task completion.

B. Dock Bay Identification

While Minion Classification is responsible for detecting course elements including TaylorMade Buoys, Polyform Buoys, the Light Tower, and the Detect and Deliver Target, the identification of Docks and Docking Bays creates a different challenge. This is due to the size of the dock often exceeding the limits of the visibility horizon, and even if the dock is detected via Minion Classification additional processing is required to identify the location of the Docking Bay.

For this reason, team Minion has developed an algorithm to search mapped objects for potential Docking Bays. The principles of this algorithm are to look for a concave object that can fit the regulation sized docking bay within its contents while ensuring the approach to the docking bay is clear of obstacles. As such, for each mapped objected (which is again represented by a polygon in the map), the algorithm follows the series of steps below:

Initialize w, l;
Template of Bay polygon is defined by point set
$BE = \{\frac{l}{2},\frac{w}{2};\ \frac{-3l}{2},\frac{w}{2};\ \frac{-3l}{2},\frac{-w}{2};\ \frac{l}{2},\frac{-w}{2};\ \frac{l}{2},\frac{w}{2}\}$
For each polygon $b_i \in B(t_k)$
  $W = width(b_i)$
    $L = length(b_i)$
    $if(W > w \ \&\ L > l)$
      $H = hull(b_i)$
    For each polygon $h \in H$
      $center = centroid(h)$
      For each line segment $line \in h$
        $\alpha = angle(line) - \pi/2$
        $Bay = rotz(BE, \alpha) + center$
      $if\big(Bay\ does\ not\ intersect\ any\ b_i \in B(t_k)\big)$
        $Bay \in DB$
      end
    end
  end
end

Fig 4. Pseudo-Code for detecting a docking bay

where w is the width of a docking bay, l is the length of a docking bay, DB is the set of all known docking bays, and $rotz(BE, \alpha)$ rotates the polygon BE by angle α about the z axis.

The team did not have the resources to build or transport a competition sized dock to our test site. Instead, docks at a local marina were used to test the algorithm. A sample result from testing at the marina is shown in Fig 5. As shown, after tuning w and l for this test location, the algorithm was able to detect docking bays. In a 20 minute driving session, all docking bays were discovered that met the tuned width and length criteria, with only three false positive results. It should be noted that the 3 false positive results were from large, partially mapped objects. However, no objects this large will occur on the competition course and these false positives were eventually ignored by Minion once the full dock object had been mapped. Furthermore, the orientation of the docking bays was detected to within +/- 15 degrees.

Fig 5. On top, the satellite view of the Halifax Harbor Marina. On the bottom is a plot of the objects (yellow) and docking bays (blue Minion outlines) detected by Minion. The current location of Minion is shown with a cyan line, while the visibility horizon is plotted in a dashed cyan line.

V. Acknowledgements

The authors would like to thank the Office of Naval Research (ONR) and the Association of Unmanned Vehicle Systems International Foundation (AUVSI) for supporting the Maritime RobotX competition for which the team at ERAU originally developed the methods disclosed here. This work was also partially supported by the Department of the

Navy, Office of Naval Research, grant number N00014-17-1-2492.

REFERENCES

[1] *VLP-16 User's Manual and Programming Guide,* Velodyne Acoustics, Inc. Rev A, August 2015.

[2] R.B. Rusu, S. Cousins, "3d is here: Point cloud library (pcl)." In *Robotics and automation (ICRA), 2011 IEEE International Conference on* (pp. 1-4). IEEE, May 2011.

[3] Google Maps. (2018) Retrieved from https://www.google.com/maps/@29.1833392,-81.0426868,60a,35y,270h,39.6t/data=!3m1!1e3

[4] R. Halterman, and M. Bruch. "Velodyne hdl-64e lidar for unmanned surface vehicle obstacle detection." Unmanned Systems Technology XII. Vol. 7692. International Society for Optics and Photonics, 2010

[5] A. Hornung, K.M. Wurm, *et al*. "OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees," Autonomous Robots, 2013

[6] R.B. Rusu, "Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments," Computer Science Dept. Technische Universitaet Muenchen, Germany, 2009

[7] T. Kanungo, et al. "An efficient k-means clustering algorithm: Analysis and implementation." IEEE Transactions on Pattern Analysis & Machine Intelligence 7 (2002): 881-892.

[8] J. Seo, S. Chae, J. Shim, D. Kim, C. Cheong, & T.-D. Han, "Fast Contour-Tracing Algorithm Based on a Pixel-Following Method for Image Sensors," Sensors, vol. 16, Issue 3, pp. 353, 2016.

[9] D. Douglas, and T. Peucker. "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature." Cartographica: The International Journal for Geographic Information and Geovisualization, vol. 10, pp. 112-22, 1973.

Appendix F – Design and Implementation of an Ultra-Short Baseline Array for Acoustic Localization

Stephen P. Cronin, Nicholas D. Moline

I. Introduction

The RobotX competition requires the AMS to detect and localize underwater pingers at specific frequencies in order to complete the acoustic transit tasks. Minion uses four Teledyne TC-4013 hydrophones in an ultra-short baseline array to detect the acoustic wave fronts.

The process by which the wave front is converted into a source location is known as multilateration. At a high level, this approach looks at the time difference of arrival of the source to multiple sensors in an array. As the geometry of the array is known, this can be used to determine the positional offsets of the source to the array.

II. Algorithm Overview

The process by which a sound pulse is separated from the signal being propagated through the environment to the desired location information is a multi-step endeavor involving both analog and digital signal processing. At a high level, the sound wave is converted to an electrical signal by an array of hydrophones. This signal is then filtered by analog components before being sampled and converted into a digital signal. The signal is then further processed with the goal of extracting the wavefront of the signal, the portion of the signal containing the sound transmitted in the shortest path to the sound source. This portion contains the information needed to extract the time difference of arrival and to calculate the location of the sound source. For the RobotX competition, this source is a periodic sound pulse at a rate of 0.5 Hz, with a possible frequency of 25-40 kHz at each kilohertz increment.

III. Localization Theory

A. Analog Signal Processing

In the 2016 competition, during the find-the-pinger task, Minion had trouble simultaneously locating the pinger and keeping the boat stationary because of the extremely high level of motor noise which caused clipping on the analog to digital converter rendering the data useless. This was remedied with a 5-stage analog bandpass filter on the hydrophone signals before being digitally sampled.

B. Digital Signal Processing

The signal being sampled by the data acquisition device (DAQ) accepts a range of frequencies that are limited on the low end by the analog filter stage. To filter the target frequency out from the signal present, a bandpass Butterworth filter is employed. The Butterworth filter was selected for the flat response it provides within the passband. As the spacing between the frequencies, 1 kHz, is significantly greater than the frequency of the noise around the target, on the order of Hz, this provided the best response for the search frequency while still removing those not desired. To ensure that the frequencies outside of the band were properly removed, a filter order of 10 was used with a band of 150 Hz above and below the target frequency.

C. Wavefront Detection

To extract the sound signal at the wavefront, a spectrogram is taken of the raw signal. The goal of this is to determine the region of the signal that contains the full pulse itself. As can be seen in Figure 1 and Figure 2, at the location of the pulse, a spike occurs in the spectrogram return at the frequency and subsequent harmonics. Extracting the region of signal corresponding to the spectrogram spike results in the pulse itself being extracted from the overall signal. Figure 3 shows the pulse extracted ready for filtering to extract the wavefront.

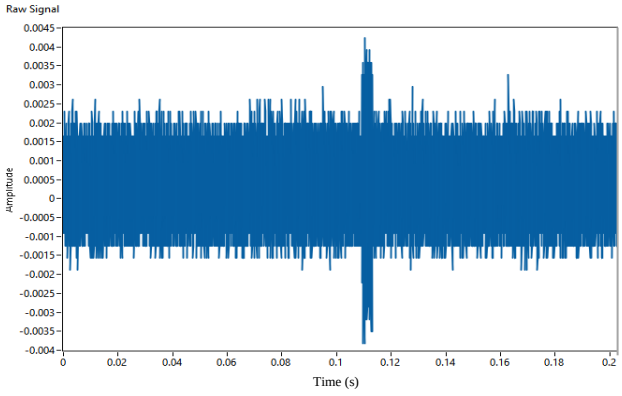

Fig 1. Raw waveform containing pulse (at t = 0.11s).

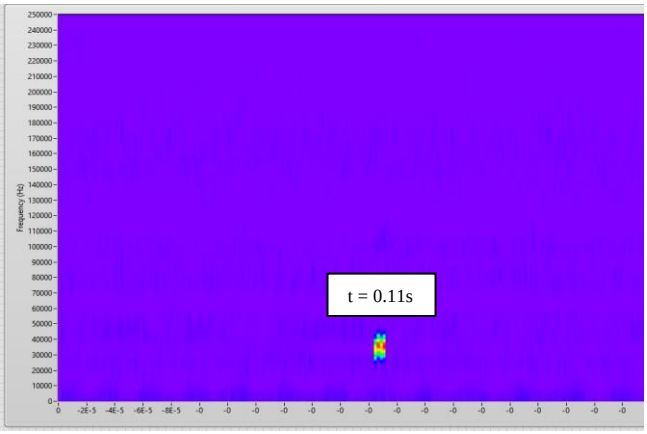

Fig 2. Spectrogram of waveform with pulse.

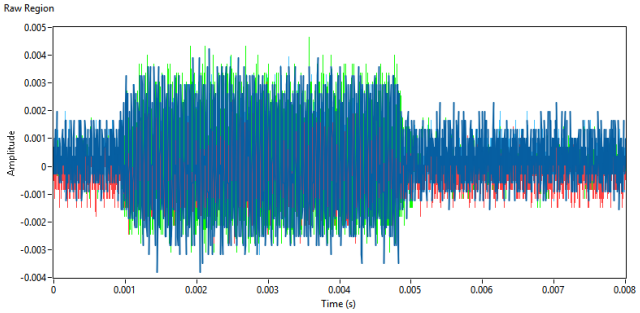

Fig 3. Pulse extracted from raw signal

To extract the signal, a bandpass is first applied to the pulse to remove all spurious frequencies. It should be noted that although the application of a bandpass will produce a phase shift on the data, it will be consistent across all channels, meaning that the end goal of computing the difference of phase will be unaffected.

Applying the bandpass to the pulse results in the signals shown in Figure 4. As can be seen, we the combination of the phase shift and removing of spurious data has left the initial presence of the pulse arriving at the array. The final step in the process is to reduce these signals down to as close to the initial arrival as possible.

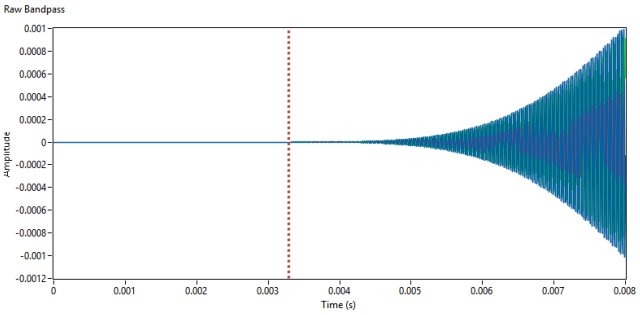

Fig 4. Signal pulse after bandpass application.

To extract the initial rising edge of the signal, the absolute value of the filtered region is taken and then a moving average filter is applied to the signal. This serves to capture the behavior of the signal. An approximation of the second derivative is applied to this signal. When the signal first begins to experience a change in slope, the location is recorded. A fixed length segment is then pulled out of the signal from that location onwards. Figure 5 shows the result of this extraction process, taking place at a time of approximately 0.00325s (at the red line) in Figure 4.

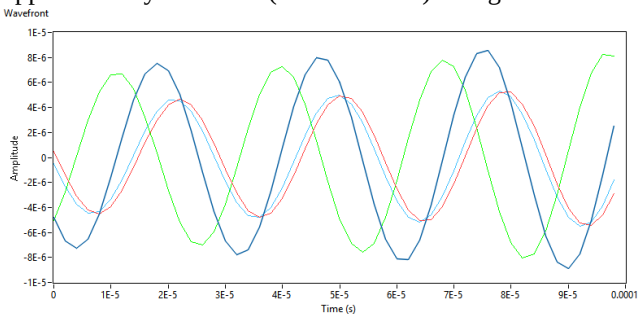

Fig 5. Signals at the wavefront.

D. Phase Calculation

With the wavefront extracted from the signal, the phase of each channel of the signal can be computed. Phase is utilized over the time of arrival as it allows for the calculation of the difference of arrival at any point during the wavefront as opposed to at the instant of arrival, which is significantly harder to detect. More specifically, the phase difference of the signal can be calculated anywhere within the clean portion of the arriving signal, whereas using timing requires one instant of the signal, one that may not actually be exactly captured as the sampling rate of the DAQ is not infinite.

To compute the phase of the wavefront of each channel, begin by taking the FFT of the signal. From this filter the bin corresponding to the frequency in question is of interest. As indicated before when discussing the bandpass filter applied, the order of this filter must be high enough to suppress frequencies near the signal that can be of similar magnitudes. As Nyquist-Shannon states, for the sampling rate of the hardware, 500 kSamples/s, and the size of the wavefront detected, 80 samples, the bin size of the FFT can be computed.

An FFT on 80 samples yields half that many bins:

$$\frac{80\ samples}{2} = 40\ bins$$

Applying Nyquist-Shannon to the sampling rate can resolve up to 250 kHz. With this the frequency resolution of each bin is computed:

$$\frac{250\ kHz}{40\ bins} = \frac{6250\ Hz}{bin}$$

This large amount of frequency per bin is what necessitates the higher order filter to reduce the chance of other frequencies not of interest impacting the bin of interest.

It should be noted that decimating the data will not help in this case, as it produces both a reduction in the number of samples and the maximum frequency resolvable, yielding the same frequency per bin.

The bin of interest is the bin corresponding to the frequency being produced by the sound source. From this bin, compute the frequency of the signal as a means to validate removal of the spurious frequencies from the bin.

E. Ultra-Short Baseline Array

Looking at the mathematics of localization, the fewest sensors that can be used to locate a sound source in 3 dimensions is 4 sensors. The following mathematics shows how to convert the time difference of arrive to location.

Assume: 4 hydrophones at (x_0,y_0,z_0), (x_1,y_1,z_1), (x_2,y_2,z_2), (x_3,y_3,z_3) and Source at (x,y,z)

The distance from the source to the hydrophone can be described by Euclidean distance, where:

$$D_n = \sqrt{(x-x_n)^2 + (y-y_n)^2 + (z-z_n)^2} \qquad (1)$$

For hydrophone 0:

$$\sqrt{(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2} = D_0$$

For hydrophone 1:

$$\sqrt{(x-x_1)^2 + (y-y_1)^2 + (z-z_1)^2} = D_1$$

For hydrophone 2:

$$\sqrt{(x-x_2)^2 + (y-y_2)^2 + (z-z_2)^2} = D_2$$

For hydrophone 3:

$$\sqrt{(x-x_3)^2 + (y-y_3)^2 + (z-z_3)^2} = D_3$$

The time difference of arrival and phase difference of arrival are directly related by a constant speed of sound throughout the water. Therefore, the distances can be related through differences in phase, a more accurate calculation than time.

$$\Delta d_1 = D_0 - D_1 \text{ and } \Delta d_1 = \frac{(\Phi_0 - \Phi_1)}{2\pi}\lambda \qquad (2)$$

Since the phase difference can be measured, all Δd's are known:

$$D_0 - \Delta d_1 = D_1$$

Therefore:

For hydrophone 0:

$$\sqrt{(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2} = D_0$$

For hydrophone 1:

$$\sqrt{(x-x_1)^2 + (y-y_1)^2 + (z-z_1)^2} = D_0 - \Delta d_1$$

For hydrophone 2:

$$\sqrt{(x-x_2)^2 + (y-y_2)^2 + (z-z_2)^2} = D_0 - \Delta d_2$$

For hydrophone 3:

$$\sqrt{(x-x_3)^2 + (y-y_3)^2 + (z-z_3)^2} = D_0 - \Delta d_3$$

Squaring each side:

For hydrophone 0:

$$(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2 = {D_0}^2$$

For hydrophone 1:

$$(x-x_1)^2 + (y-y_1)^2 + (z-z_1)^2 = {D_0}^2 - 2 * D_0 * \Delta d_1 + {\Delta d_1}^2$$

For hydrophone 2:

$$(x-x_2)^2 + (y-y_2)^2 + (z-z_2)^2 = {D_0}^2 - 2 * D_0 * \Delta d_2 + {\Delta d_2}^2$$

For hydrophone 3:

$$(x-x_3)^2 + (y-y_3)^2 + (z-z_3)^2 = {D_0}^2 - 2 * D_0 * \Delta d_3 + {\Delta d_3}^2$$

At this point, the location is represented as 4 equations and 4 unknowns, (x, y, z, D_0). To solve for the location, a numerical solution must be employed as the terms are not linearly separable.

IV. HARDWARE DESIGN

The hardware of the acoustic localization system consists of three main components, the ultra-short baseline array, filtering circuits, and the DAQ.

A. Ultra-Short Baseline Array

The type of array employed is an ultra-short baseline array. This allows for some assumptions to be made about the behavior of the array.

Sizing the array is based on the shortest wavelength the array would process. For the purposes of this competition, this occurs at 40 kHz. The wavelength at this frequency is:

$$\frac{Speed\ of\ Sound}{Frequency} = \frac{1480\ m/s}{40000\ Hz} = 0.037\ m$$

Half this wavelength is used to determine the maximum distance the sensors are spaced from one another to prevent there being multiple locations where the same delta phase can be produced. In practice, a factor of spacing is employed to account for factors such as differences in the speed of sound as well as frequency shifts. A distance of 1.8 cm was used as the spacing between the sensors. The sensors themselves are physically less than half of this spacing with a diameter of 9.25 mm, making this array possible.

B. Hardware Filtering

A hardware filter to removes the 15 kHz motor noise before the signals are sampled. The design consists of a low pass Butterworth filter with a cutoff frequency of 50kHz followed by a high pass Butterworth filter with a cutoff frequency of 20kHz. The gain of the system was designed to be +20dB. The goal of this gain was to align the output of the circuit with the inputs of the DAQ, which accepts a voltage of -10 to 10 volts.

C. Digital to Analog Conversion

Sampling of the signals is done with a four-channel digital

to analog device. These signals are sampled at 500 kSamples/s/channel. This sampling rate puts the resulting signals well above the Nyquist criteria for resolving the frequencies in question. Sampling on each channel is also synchronized, meaning that measurements happen at the same time. This is useful in extracting the wavefront of the signal and ensures that the FFT bins are referring to the same portion of time between sensors.

D. PODS Board

One of the goals of the platform was to reuse technology whenever possible. To this end, the same circuit board used to raise and lower the motors for beaching was used to lower the hydrophone arm.

E. Sensor Guard

As the sensors themselves are relatively fragile, with some of them even experiencing damage during the 2016 competition due to a collision with the dock bay, a guard was constructed to handle accidental contact instead of the sensors themselves. A guard was designed with the ability to prevent damage to our sensors at the full range of operating speeds experienced and at all likely collision angles. Validation of the guard was conducted using finite element analysis (FEA).

V. Design Considerations

A. Multipath

As with any system involving signals being transmitted through and environment, multipath is one of the primary sources impacting the phase measurement. Looking at the physical system, the closest source of reflection to the array describes how long of a usable wavefront will be present. The equation below relates the distance of the shortest reflection path to the number of DAQ samples the wavefront will have sampled. For this system, a distance of at least one meter from any source of reflection was the target, resulting in approximately 350 samples, many full phases at the frequencies in question.

$$N\ samples = \frac{Distance}{Speed\ of\ Sound} * Sampling\ Rate \qquad (3)$$

B. Digital Bandpass

Filtering the data is a balance between the filter order, band size, and time to compute. From these features the best suited filter can be determined. It should be noted that these features are all inter-related in terms of impact but are treated separately as they are distinct from an algorithm perspective.

Looking at band size, first consider the doppler shift, shown below, where c is the speed of sound in the medium, f is the observed frequency, f_o is the emitted frequency, v_r is the receiver speed, and v_s is the source velocity. Note that for this purpose, the operating speeds are significantly less than the medium speed. In testing, the boat travels at a maximum of 3.5 m/s whereas the speed of sound is approximately 1480 m/s. The other frequency shift that would widen the band is from the inconsistencies in the device producing the frequency. Applying doppler shift to the highest produced frequency, 40 kHz, this would result in a shift of approximately 95 Hz. To account for the fact that additional velocity components from water current which can be added, the band was sized at 150 Hz.

$$f = \left(\frac{c \pm v_r}{c \pm v_s}\right) f_o \qquad (4)$$

Filter order effects the falloff of the frequencies outside of the band allowed. Based on the competition rules, there is a separation of at 2 kHz between channels. As discussed prior, this separation is less than the FFT bin width, necessitating a more aggressive filter to prevent other sound sources in the environment from impacting the frequency of interest. Based on experimental testing on the range of frequencies allowed, a filter order of 10 was found to provide the necessary falloff to prevent other frequencies from presenting an issue.

C. Wavefront Detection

The methodology described assumes that a wavefront is present in the collected data. This however, is not a guarantee and as such cases where there is not one present need to be rejected. Figure 6 shows a spectrogram return for this case. In this implementation, a check is performed on the average spectrogram return of the lowest bins, typically containing random noise sources, to the bin of interest. If the difference in return does not reach a threshold, there is considered to be no pulse present. This approach was deemed robust in this implementation as hardware filtering takes place in the filtering process, drastically reducing the presence of random noise in returns.

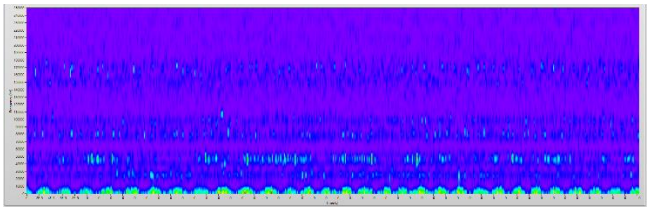

Fig 6. Spectrogram without pulse present

D. Phase Calculation

The signal to noise ratio (SNR) is one of the primary factors in accurately determining the phase of the signal, and as such the location of the source. Various sources of noise can contribute; however the primary concern is in frequency sources in the same FFT bins as the target frequency. This can stem from environmental sources themselves or harmonics of sources, a 32 kHz harmonic of a 16 kHz motor controller for instance would be one of the available localization targets. Through testing, it was found that accurate phases could be computed at signal to noise ratios of 50 or more. Raw signals can possess more noise than this, however after a bandpass filter is applied to the signal noise levels are dramatically reduced with signal to noise ratios in

the 100's. Therefore, noise does not typically prevent the algorithm from functioning as intended.

E. Numerical Method

Each difference of phase that the array can experience correlates to one point uniquely in space within the hemisphere of the array governs. However, this claim is only true in a purely theoretical sense. When a signal is sampled into a finite resolution, 16 bits, and multiple algorithms each with errors is applied to these signals, the phases are no longer unique. As a result, the numerical algorithm can converge to multiple solutions. A property of how the errors equations converge is that solutions along the same vector are the most likely solution that the method will converge to. As such, these solutions have the same bearing to the source, and can be used for navigational purposes. Other spurious results are filtered through outlier analysis over multiple readings.

VI. PERFORMANCE

Experimentation on the performance of the algorithm took place in both field testing as well as simulation. This allowed for easy testing of large numbers of use cases along with accurate representations of multipath in environments, something difficult to model along with the impact that the system may have on the localization hardware.

It should be noted that the data presented is not a comprehensive look at the performance of the system. Further testing to draw statistically significant conclusions is needed. As such, test cases relevant to the competition will be evaluated rather than the generalized performance.

The results presented took place in the university pool. This was used as it is a controlled environment that allowed for ease in producing repeatable results.

To test the relevant cases to the competition the assumption was made that the boat would be near the vicinity of the gate and be able to face hold a heading. Table I shows the results of testing at various bearings to the pinger. This was selected as the results to present as from a measurement perspective it was easier to produce accurately measure. In the competition this is sufficient to be able to navigate the gates. As can be seen, the deviation of error was relatively low, achieving results lower than the accuracy target of ±5 degrees discussed in the main paper. Certain cases have higher standard deviation, but at this time it a conclusion as to the cause of this is unknown. Moving forward more testing in the same test cases and other frequencies and bearings will be conducted to form a better picture of the behavior of the system as a whole.

Table I

EXPERIMENTAL RESULTS PINGER BEARING

Case (deg)	Frequency (kHz)	Mean (deg)	Standard Deviation (deg)	Samples
45	40	45.34	7.85	7
0	40	-0.62	3.93	17
0	35	0.25	4.62	13
80	35	79.29	3.20	12
-90	35	-90.90	5.94	8

Appendix G: Minion Core Inter-process Communications Library

Timothy A. Zuercher, Patrick N. Currier

I. Introduction

Team Minion has historically used National Instruments LabVIEW as a basis for the autonomy software package. Although somewhat unusual in the robotics community, LabVIEW is natively parallelizable, is designed to interact with hardware, is capable of calling libraries written in other languages, offers excellent debugging tools, and is easily accessible to students who are not extremely skilled programmers.

The Minion code base is highly modularized in order to spread the responsibility for programming across the team. To solve the problem of inter-process communications, the team developed the Minion Core communications library for LabVIEW. This library emphasizes ease of use while still maintaining robustness to component configuration and communication interference.

II. Minion Core

Minion Core is designed as a lightweight, master-less, auto-configuring, UDP-based communication structure featuring support for acknowledgement and sequential messages.

A. Interface Design

The primary design goal for Minion Core was to produce a library that was extremely easy for students to deploy and use. Specific interface design goals included:

1. Simple and automated message generation
2. Minimal code components
3. Minimal LabVIEW data wires
4. Automatic network configuration
5. Support for parallel loops
6. Deployment in pre-compiled form

An example of a complete program capable of sending and receiving Minion Core messages is shown in Fig 1.

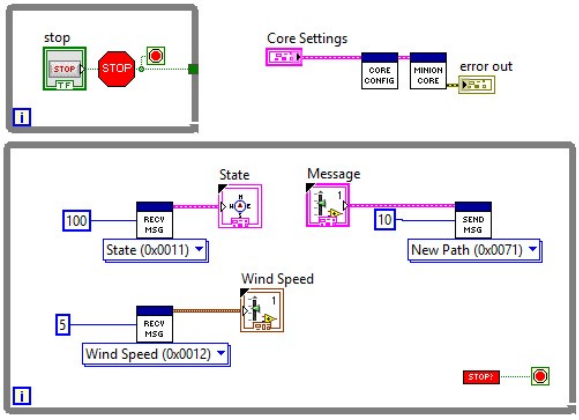

Fig 1: Example of complete Minion Core program

Minion Core is distributed as a packed library (the LabVIEW equivalent of a dynamic linked library or a shared object). Fig 1 shows all of the features of the Minion Core interface that are contained in the library. In the top right is the MINION CORE block. This block contains all of the underlying code that manages message transfer. To use Minion Core, a user must simply drop this block on to the base diagram and create a Core Settings control that contains basic network information. In most cases, default values are used, but this control allows for separate Minion Core networks.

Messages are sent and received using auto-generated polymorphic blocks. To send a message, the SEND MSG block is dropped into the diagram. This block auto-detects the correct message to send based on the data type wired. In some cases, a user may want to override this setting and can do so using the case selector (set to New Path in this example). Once the data is entered into this block, it is passed to the MINION CORE block using a LabVIEW notifier, enters a send loop, and is automatically sent to all known destinations for the message ID. On the first call of the send block for a particular message, the MINON CORE block sends out multicast ping messages to attempt to find a destination component for the message. All returned components are considered valid destinations until the component's heartbeat drops from the network.

Receiving a message works similarly using the polymorphic RECV MSG block. When first called, this block triggers a multicast ping message to find a component send the requested message. Once a message has been received, the RECV MSG block will return data in the correct data type for the message. A separate indicator

reports if new data has been obtained or if the block has timed out. The RECV MSG block is implemented as blocking code, but the timeout can be set to zero if required. In the event that no new message has been received, the block will return the last received data.

Messages are configured using a Message Manager that is contained within the packed library, Fig 2. This component allows a user to define all of the relevant messages by creating a LabVIEW cluster containing the data type, specifying an ID, and setting default options for timeout, acknowledgement, and sequence enforcement. Once these options are specified for each message, the Generate Messages button triggers automatic generation of the SEND MSG and RECV MSG blocks. Individual VIs are generated for each message based on a template and then bundled into the polymorphic blocks. Regeneration of messages affects all components within the project, but messages remain backwards compatible unless data types or IDs are changed.

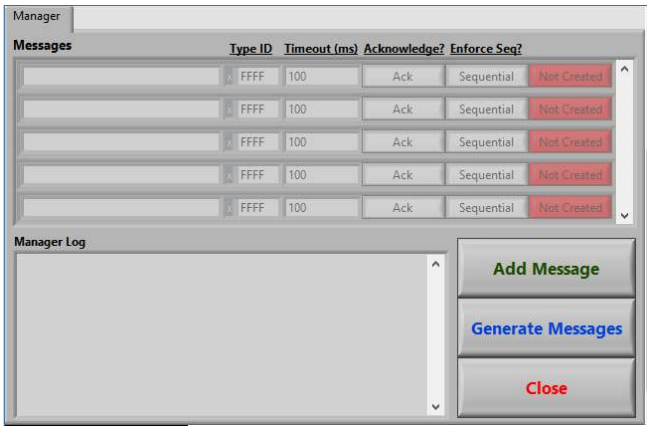

Fig 2: Minion Core Message Manager interface

The final component of the interface design is the STOP blocks that allow for control of multiple parallel loops. This is often a challenging problem in LabVIEW, but is handled in Minion Core via a SEND STOP block (stop sign shaped block) that triggers a global stop notifier. This notifier is the read by the QUERY STOP block and used to signal additional loops to stop execution.

B. Communications Design

The Minion Core communications architecture was designed with several goals:

1. Low overhead
2. Master-less, auto-configuring
3. Robust to low-bandwidth links
4. Robust to intermittent links

To address these design objectives, Minion Core communications are based around a UDP multi-cast auto-configuration and single-cast message sending.

UDP was selected to reduce overhead relative to TCP, which requires retransmission of data to confirm receipt. Minion Core adds only a 17-byte header and standard UDP header to each message. Messages are allowed in sizes up to 65519 bytes of data. Multi-part messages are currently not supported. The potential weaknesses of UDP are addressed through Minion Core features, except for security. MinionCore is not inherently secure and requires the network to be secured at OSI layer 1.

At startup, each Minion Core component opens an available single-cast UDP port and registers a set of sent and received messages from an array of critical messages connected to the MINION CORE block. Additional messages are registered as the SEND MSG and RECV MSG blocks are called in the user code.

The Minion Core auto-detect module attempts to find sources and destinations for each message by sending out multi-cast UDP pings containing its network port and arrays of message IDs that it is seeking. Other Minion Core components listening on the same multi-cast address reply with ping messages.

The component parses these messages and sets up single-cast UDP connections via a heartbeat message. The components each register a new source or destination for the relevant message and continue to exchange single-cast heartbeat messages at a user-defined interval. If the heartbeat is not received for a user-specified amount of time, the connection is considered lost and the relevant message source or destination is de-registered, triggering additional multi-cast pings if no other sources or destinations are registered.

This auto-detect functionality allows for components to be easily moved between computers for testing purposes. Components can be run on developer laptops for debugging with no reconfiguration or additional overhead.

Robustness to poor communications links is provided by the Minion Core acknowledgement functionality. By default, messages are transmitted over single-cast UDP with no guarantee of delivery. For messages that are sent rapidly and where only the latest data is relevant (such as vehicle state), this is a desired functionality to minimize bandwidth requirements. Data validity for all messages is checked using the UDP checksums.

For messages that may only be sent once or that are critical, the user may choose to request an acknowledgement. When an acknowledgement is requested, the Minion Core code will send the message and then add its ID and destination to an acknowledgement queue. The receiving component will detect the requested acknowledgement and reply with an ACK message containing the header of the sent message. The acknowledgement queue will attempt retransmission of the message until either the ACK message is received or a user-specified timeout is reached. Either way, the user will receive a Boolean indicating the status of the acknowledgement.

For messages that may be required to be received in a specific order, Minion Core also allows the specification of enforced sequence. If this option is selected, the receiving component will reject messages that violate the monotically increasing sequence number for that message ID. Since

sequential messages are required to be acknowledged, this functionality will trigger a retransmit of the message from the sending component, ideally arriving after the lost sequential message (which has also not been acknowledged) is received.

This method reduces bandwidth requirements by requiring only minimal ACK messages while allowing for multiple retransmission attempts in the case of intermittent connections. These types of connections are often encountered in long-range wireless environments, such as ship-to-shore. The Minion Core acknowledgement and sequential enforcement provides increased robustness to real-world communications not found in some other packages, such as ROS.

Minion Core also logs all messages sent and received by a component. The disk space allocated to this log is automatically managed subject to user-specified file and total size limits. These messages can then be played back to simulate or replay events.

C. Results

Minion Core has proven highly successful in practice. Students can learn to use the package in only a few minutes of training. The auto-configuration functionality enables rapid debugging by allowing components to move seamlessly between computers and the acknowledgement functionality improves robustness to poor communication conditions experienced in field operations.

Appendix H: Electrical and Power Systems

Jefferson S. Romney, Nicholas R. Middlebrooks

I. Introduction

The design of Minion's electrical system has evolved since its first version in 2014. The design follows three main tenets. The first and most important philosophy is safety first. Safety takes priority over function in all cases. After safety, but closely related, redundancy is a goal of the electrical design. Although not every system has total redundancies, most systems critical to the minimal function of the vehicle have redundancies. The third major design guideline is distributed power regulation. This is a lesson learned from the first power system on Minion; where if multiple systems were tied to the same power source, when one goes down, they all go down, where now if one system goes down the others can still run. Every subsystem regulates power for itself and there is no central power distribution. This follows the principle of redundancy in that no one regulator failure can disable the whole boat. This can be seen in Fig 1 where each system gets direct battery power.

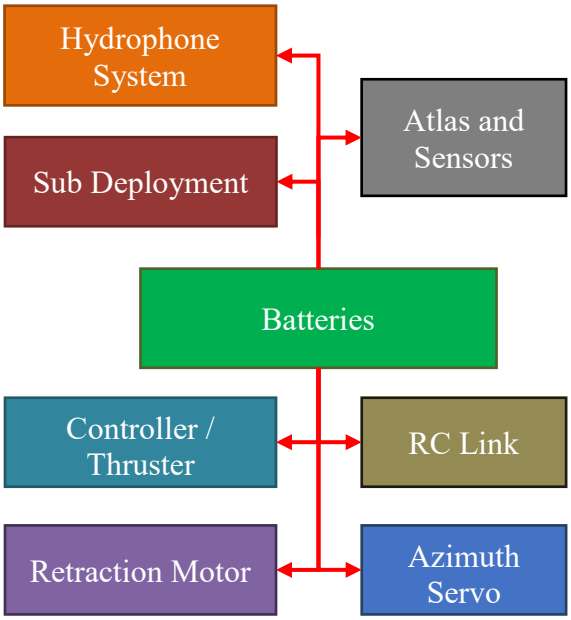

Fig 1. Main Power Distribution on Minion

II. Safety

A. E-Stop system

The most important safety system on the boat is the E-Stop system. Minion's e-stop system is designed for utmost reliability and redundancy. It allows the boat to be e-stopped both remotely and with on-board physical buttons, all without software intervention. It also includes a low-level software watchdog to monitor the system and allow for software-controlled e-stopping as well.

1) On-Board Button system

Minion is equipped with 4 physical e-stop buttons; one at each corner of the boat is located low on the arches for ease of access when on the water. These normally closed buttons operate in series as an input to the estop circuit. Fig 2 shows the circuit of the 4 buttons.

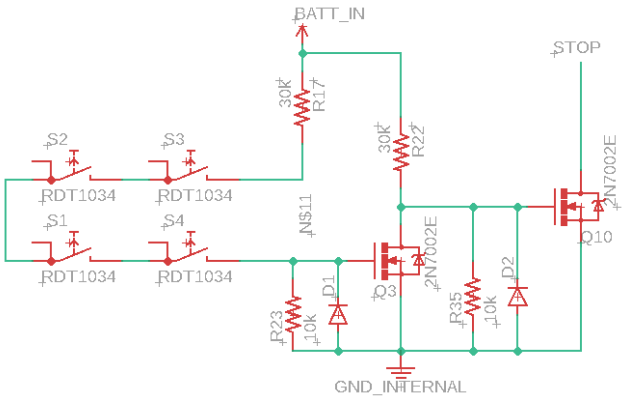

Fig 2. E-Stop Button Loop Circuit

2) R/C Hardware E-Stop

The primary remote e-stop is operated using a dedicated channel on the remote-control system. The signal is received as a standard RC PWM signal and is split to both the safety system micro-controller and a specialized flip-flop circuit to threshold the pulse width, shown in Fig 3. The signal path to the microcontroller allows software to monitor the signal without interfering with its e-stop functionality.

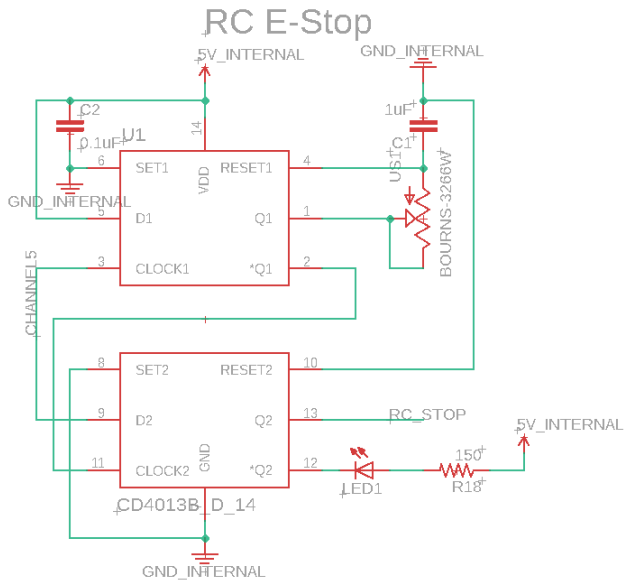

Fig 3. RC Hardware E-Stop Flip-Flop Circuit

3) Backup External E-Stop Solution

A consideration that was included in the design of the e-stop system was the ability to integrate an external independent wireless e-stop system such as a TORC SafeStop. This is done via a simple active high input.

4) Software Oversight

To monitor the hardware e-stop circuit and enable software activation of the e-stop, a microcontroller monitors the overall e-stop state and the RC signals, and is allocated an input to the e-stop system. With these connections the microcontroller software provides RC out-of-range detection, protects against quick oscillations of the signal, and communicates with the higher-level software autonomy systems.

5) Hardware OR-ing Combination

Each of the inputs to the estop system: buttons, RC, external e-stop, and software control are OR-ed using a system of MOSFETs. This circuit allows any one of the inputs to put the boat into a safe state ensuring that no software can interrupt the assertion of e-stop. The schematic of this circuit is shown in Fig 4.

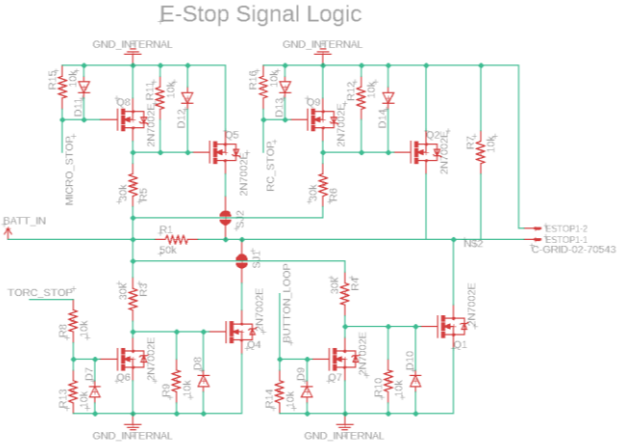

Fig 4. E-Stop Control OR-ing Circuit

6) Dedicated Relay Coil Power Regulators

The output of the MOSFET circuit enables or disables two individual isolated power supplies that each control one of the e-stop relays.

B. Indicators

During the 2016 competition, it was difficult for the light tower indicator to be seen from the shore. To address this issue, a new light indication system was developed using multiple LED panels.

These LED panels are NeoPixel brand LED matrix boards being controlled from an auxiliary microcontroller attached to the control/safety board on Minion. A NeoPixel is an addressable RGB LED, and Minion is equipped with 5 panels of these totaling 1024 pixels, providing an order of magnitude increase in indicator brightness over the previous tower which improves the visibility at nearly all angles. Each panel can draw between 12-25W of power for the LEDs, compared to the .75W draw of the old light tower. While this provides a simple way to display the E-Stop, R/C, and Autonomous modes of Minion, the light panel array can also be used to display other information that 5 simple lights could not. Instead of just showing light colors, the light panel can show colored words to let other vessels know what each light mode means

C. Smart Batteries

There are four Torqeedo Power 26-104 batteries in use on Minion are marine-grade, high-performance, smart lithium-NiMnCoO2 (LiNMC) batteries. Each battery has an energy rating of 2.6 kWh at 25.9V nominal, and are configured on Minion as two batteries connected in series and then connected in parallel, for a total energy rating of 10.7 kWh at 51.8V nominal while running. In this configuration, Minion has a continuous runtime of 6-8 hours. The batteries are ideal for use in a marine environment due to their robust integrated battery management system that protects the batteries against overcharging, over depletion, shorts, overheating, and polarity reversal. The batteries are also IP67 rated.

D. Isolation

Because of the high power-draw of the thrusters, a spike in thruster speed can cause significant fluctuation in the level of the battery ground. These fluctuations are very dangerous to other components on the boat. To combat this, each of the distributed regulators is isolated and communications between all the major systems are also isolated. This is achieved through isolating grounds; using isolated power supplies for stepping down voltage; and isolating data lines through magnetics, opto-electronics, and capacitive isolation.

III. Redundancy

A. Datalinks

To interface with the main autonomy system, a high-speed

5.8 GHz datalink is employed through the use of paired Ubiquity Rocket M5 radios to connect to the ground station and judges' tablet. This datalink can be susceptible to interference and network faults. To ensure that manual control is always available, it is transmitted over a second, more reliable, lower-frequency datalink dedicated to RC control that can be run on either a 433 or 900 MHz frequency. This makes sure that if the connection monitoring the autonomy fails, the manual control link can act as a redundancy rather than failing with it.

B. Degrees of Freedom

With the addition of an azimuth degree of freedom on each thruster, some redundancy has been built into the control system. With as little as one thruster and one azimuthing actuator, the boat can be fully controllable in forward or backward motion. To support this redundancy, each azimuthing servo has an independent isolated drive circuit and power regulator. If any single actuator or thruster fails, Minion can still be driven in all but fully holonomic modes. For more information see Appendix I.

C. Independent Computer Power supply

Inside the Atlas control box are two separate computers that act as the computational brain for the Minion platform. Each computer has a copy of the runtime software, so in the unlikely case that one computer is disabled, the other computer could shoulder the load to keep basic autonomy functional. To complete this redundancy, each computer has an independent custom DC/DC ATX power supply.

D. Uniform Platform for Actuator Interface

To control the additional actuators, including the hydrophone boom actuator, the Submarine deployment system actuators, the racquetball turret, and the motor pod azimuth and retraction actuators, a new circuit board was designed to meet the needs of all these subsystems. This unified design allowed the same circuit board to be used in all these applications. The resulting interchangeability allows for reduced development time, simple replacement, and fewer unique spares needed to keep on hand.

E. Thruster Controllers

Since the old Torque-Jet thrusters and the new Copenhagen VM thrusters are both compatible with similar motor controllers, the custom controller designed in-house and the new Piktronik controller are freely interchangeable.

IV. Changes

A. New Thrusters

1) No longer prototype thrusters

The Minion platform for both the 2014 and 2016 competition used Torque-Jet Rim Driven Propeller (RDP) thrusters. Those were some of the first generation of small-scale RDP thrusters available on the market, and as such reached the end of their operational life during late 2017. Torque-Jet's technology has since been purchased by Copenhagen Subsea A/S, who then further developed and refined, and the product of these improvements has succeeded its predecessor as the propulsion unit used on Minion. Copenhagen Subsea's VM thruster was selected to replace the old thrusters.

Like the previous RDP thrusters, the new VM Thrusters are 12-pair permanent magnet brushless DC RDPs, making them easy drop-in replacements. The new VM thrusters carry many advantages over the older Torque-Jets thrusters, including: optimized blade design for increased hydrodynamic efficiency, more efficient cooling for higher sustained load, improved serviceability and maintainability, and increased thrust.

2) Asymmetric Thrust

The new VM Thrusters are fitted with special nozzles and propellers to give asymmetric thrust for greater thrust in the forward direction, as most of Minion's movement, especially with azimuthing control, is with the thrusters spinning in the forward direction. The asymmetric nozzles can be seen in Fig 5.

Fig 5. Copenhagen VM Thruster with Asymmetric Nozzles

3) 3D Printed Propellers can be Easily Replaced

The Torque-Jet RDP propellers were injection molded in very dense and brittle plastic making them heavy, expensive to produce, and susceptible to chipping. The VM Thrusters mitigated these problems by using 3D printed propellers made out of Nylon PA12, allowing for Copenhagen Subsea A/S to quickly iterate and optimize the propeller design to produce maximum thrust with minimal weight, making the propeller easier to accelerate for better response. The flexibility of the Nylon PA12 prevents chipping of the edges of the blades which can prolong the life of the propeller. A major design improvement is replaceable propellers so that if one of the propeller blades does break, it can be quickly replaced with backup propellers that are on hand. In addition to these improvements, the Nylon PA12 retains a high degree of resistance to moisture absorption a good chemical tolerance. [1]

B. New Thruster Controllers

1) Failure of Old Motor Controllers and Use of New COTS Controller

The new VM Thrusters and the old Torque-Jets are both sensor-less brushless DC thrusters that use the same 3-phase interface, so it was very easy have them interface into Minion's existing motor controllers that had been custom-built in-house for the Torque-Jets. However, due to complications with the existing motor controller relating to higher-voltage operations, the Piktronik SAC1-90A motor controllers recommended by Copenhagen Subsea A/S proved to be the most favorable choice for primary thruster control. The advantages of the COTS motor controller are that it is a proven product that can be easily sourced for replacements, and the bundled SACTERM software for motor interfacing/troubleshooting. The advantages of the custom motor control solution include customizability and increased feedback which are important for research purposes, therefore the compatibility with these is maintained as a backup solution.

The SAC1-90A controllers also include many features built into the custom motor controllers such as thermal monitoring and protection of the controller, current measurement and torque and speed estimation. The included SACTERM interface software allows for easy bench testing and troubleshooting of the SAC1-90A controllers when connected to the VM thrusters, as seen in Fig 6.

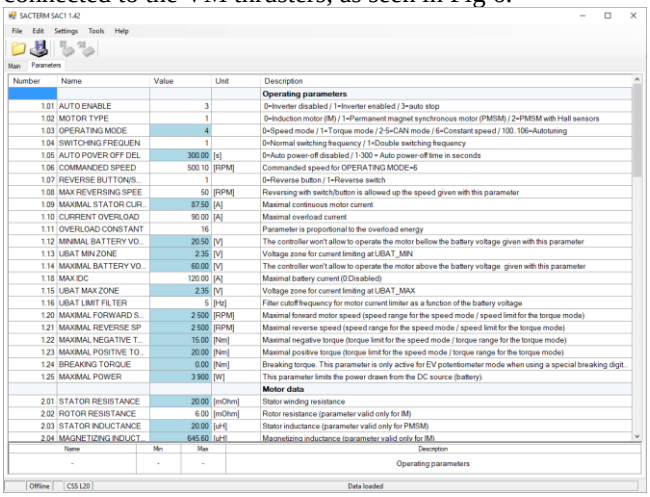

Fig 6. SACTERM interface software displaying VM Thruster config data

2) New Software Interface Over CAN Bus

The Piktronik SAC1-90A controller, in contrast to the older motor controller that was controlled using a simple PWM signal, uses a CAN Bus for control and data feedback. To interface with this, a CAN transceiver was added to the control/safety board which provides motor control signals. Both motor controllers are plugged into this CAN bus and each is given its own address range.

C. Pods

1) Pods Control for Thruster retraction and Azimuthing

Since the new thruster pod design has auxiliary actuators for control of azimuthing and thruster retraction, a new microcontroller board was needed for the low-level interface and control. The retraction actuator is the Linak LA-36, and the azimuth actuator is a Volz DA-30 servo. This led to the development of the Pods board to regulate power, control the actuators, process actuator feedback, and communicate with the safety/control board via an opto-isolated UART connection.

The Linak LA-36 is a 24V brushed motor linear actuator with built-in end-stop switches. The Pods board supplies the 24V power from an on-board converter and controls the Linak through a standard H-Bridge driver circuit.

The Volz DA-30 servos are high-torque servos rated for use in salt water, powered through 24V and controlled over an optocoupled or standard TTL-level PWM interface. The Volz servo draws power from the same 24V regulator as the Linak while being directly driven from the Pods board over PWM. For greater noise immunity over the long cable run near switching motor phase wires, it uses the optocoupled interface with approximately 10mA of current in the on state. The actual position of the servo is reported back via a single-ended analog signal.

D. Hydrophone Filter

In the 2016 competition, during the find-the-pinger task, Minion had trouble simultaneously locating the pinger and keeping the boat stationary because of the extremely high level of thruster noise which caused clipping on the analog to digital converter rendering the data useless. This was remedied applying a 5-stage analog high-pass filter to the hydrophone signals before being sampled by the ADC. For more information on the hydrophones and the filter, see Appendix F.

V. Conclusion

The electrical system, while it is the backbone of the Minion platform, is only there to support the platform while it completes its various tasks. To be that support, the electrical system was updated with respect to its main tenets to be safer, more reliable, and having a distributed power delivery system in order to work with the various new systems added to Minion this competition. It is safer by preventing shorts due to the improved signal and power isolation and more visible using higher powered indicator lights. It is more reliable through increased redundancy with backup datalinks, motor controls, and power; a unified platform for actuator control; and interchangeable control systems for the new thrusters and actuators. It works with more subsystems by interfacing with brand new motor controllers, thrusters, actuators, servos, and sensors; all while making sure they have their own independent power supplies. While from the outside it may seem that the electrical system hasn't changed, it has sustained many substantial under the hood upgrades to keep Minion at peak efficiency.

REFERENCES

[1] M. de Wargney, “Nylon PA11, Nylon PA12 & Multijet Fusion PA12 in 8 questions,” 3D Printing Blog: Tutorials, News, Trends and Resources | Sculpteo, 23-Nov-2017. [Online]. Available: https://www.sculpteo.com/blog/2017/09/06/differences-pa11-pa12-multijetfusionpa12/. [Accessed: 16-Oct-2018].

Appendix I: Intelligent Autonomous Mission Planning and Execution

James J. Hendrickson, Patrick N. Currier

I. Introduction

The MinionTask module is responsible for identifying which tasks can be completed, executing the tasks, and managing the operating mode of the vessel. Both competition and research objectives for the Minion ASV require a robust and reliable system to autonomously determine the operating mode of the robot based on information streaming in from onboard systems and off-board systems.

The MinionTask module differs from most competition approaches in that it utilizes this information to dynamically launch task modules to execute the required tasks as the course elements are discovered. The following sections will first go into detail how the MinionTask module is structured, then analyze the task development, and finally analyze the results of the simulation and in-water testing.

II. General Structure

MinionTask implements a multi-modal mission planner with intelligent tasking. User-specified configurable objectives and geo-fencing information initialize operations for a given task scheme, and the module then searches the defined area and acts on information as task elements are discovered. The tasking code is implemented as an independently compiled library that is launched asynchronously as start conditions are met. An overall progress tracker monitors the mission state, and launches or terminates behaviors to ensure continued mission progress by either accomplishing goals or timing out. The structure of this can be seen in Fig 1.

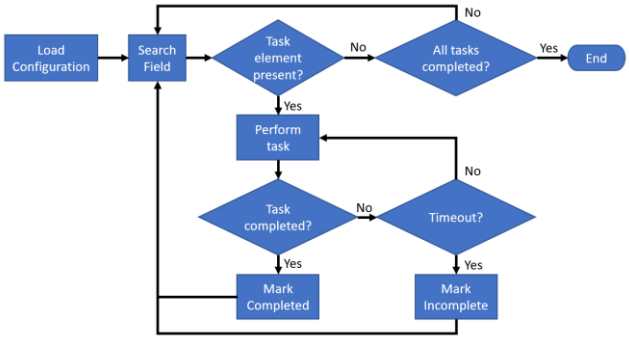

Fig 1. General module run-time structure.

A. Mission Configuration

Configuration files for the MinionTask module come in two forms. The first contains the search configuration that sets up the search pattern, the geo-fence, and the priority search nodes based on the GPS points of interest for each task that are provided by the technical directors. The latter elements are particularly important as it allows MinionTask to prioritize searching areas of the field of known interest before searching the remaining nodes based on their coverage by the perception modules.

The other configuration file is the task configuration. This XML file contains all the information needed to configure each mission. Each configuration file can define any number of allowable tasks and contains the task setup parameters and the path information to the dynamic task functions.

When a new task configuration is loaded, it is stored into the MinionCore Task Database. For the 2016 competition, this database only allowed a single set of task configurations to be loaded into the database, as shown in Fig 2. Unfortunately, if two or more platforms, for instance the ASV and a UUV, were running the same instance of MinionTask, they would both receive the same task list despite having very different operational objectives.

This vehicle deployment issue was solved by restructuring the database to include a sub-heading before the "task configuration" stage whereby the platform could be dictated and accessed by name, as shown in Fig 3. This allows not only for each independent platform on the network to run its own tasks, but also allows for any other platform specific information to be accessed from the database. The revised structure allows multi-vehicle operation from a single instance of MinionTask.

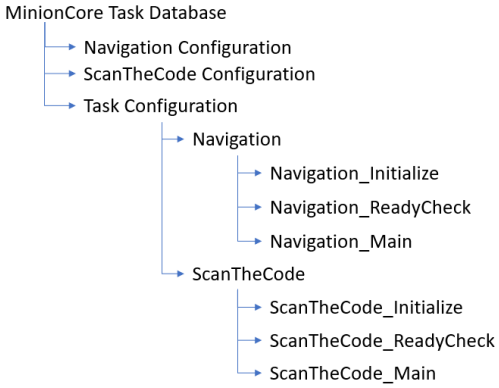

Fig 2. Minion task database for the 2016 competition

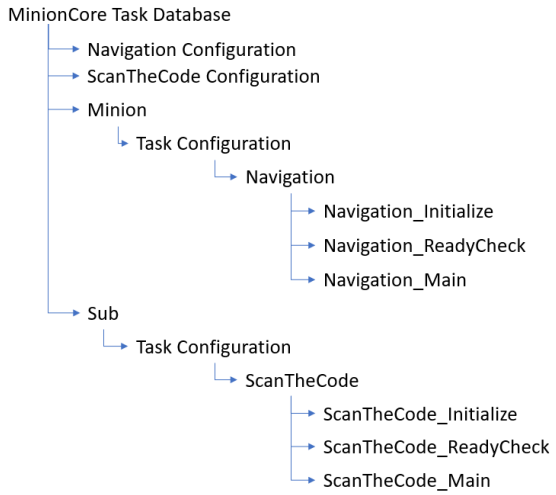

Fig 3. Minion task database for multiple platform configuration.

B. Search Mode

Upon mission start, the module is assumed to have no information of the competition course beyond the search configuration file. Every mission begins by entering a search mode to discover the task elements until a task is ready to be launched.

While the module attempts to find the required start conditions for a task, it will sweep through the field following a lawn mower search pattern in an attempt to find all of the course elements. If there are priority nodes available, then these points would be visited first based on each priority node's Euclidian distance from the boat. As new objects are discovered, new priority nodes are dynamically placed to ensure that the search explores areas near course elements. MinionTask will remain in search mode until a new task is launched and will return to search mode after the completion of the task. Once the search area has been exhaustively explored, MinionTask will return to idle.

C. Mission Structure

After a task configuration file has been loaded, the module calls three separate files for each task: initialize, ready check, and main. An example of how these files are called for each task that is loaded can be seen in Fig 3.

The initialize function for each task is responsible for converting the configuration data into the proper data type. For example, in the Navigation challenge, critical configuration elements that are configured include the tolerance on the start gate buoy spacing, if it is expected that color information about the buoys will be available or not, and the task timeout. This structure is also readily changed to allow for configuration elements to be added or removed rapidly during testing.

The ready check function for each task is responsible for checking the perceived field elements to see if any of the objects in the perception object list meet the requirements needed to start a given task. For example, in the Navigation challenge, the ready check is looking for two tall buoys with reflectors that are within 10 meters, plus or minus the tolerance specification that was loaded by the initialization function. Once the start conditions for a task have been satisfied, the ready check sends back a flag to the main module to declare the task ready as well as an expected task completion time, transition waypoint, and expected score.

The task engine attempts to optimize the overall score by selecting ready tasks based on their projected scores. Specifically, the expected score for each task is divided by the sum of the travel time to the transition point and the expected completion time to create a points-per-second value. This value can take into account information linkages between tasks and probability of completion if this information is encoded in the ready check. The engine also requires a minimum points-per-second value that decreases as overall mission time elapses. The task with the highest points-per-second value is selected, the vessel begins driving to the transition point, and its main function is activated.

The main function for each task utilizes behavior primitives to accomplish the components of a task after the platform has moved into position to begin the task. The logic for individual tasks is discussed in Section III. Inside each main function, there is also a continuous check of the task conditions for validity. In the event that an object used for the task that is running is reclassified, the module will stop running the task, but will not report that the task has been completed. This will allow for the ready check for that task to be run again and to trigger the start for the task if the required start conditions are met again.

Once a task main has either exited or timed out, MinionTask will return to search mode. Once all tasks have been completed, MinionTask will return to idle and await operator commands.

III. Task Structure

A. Navigation and Control Demonstration

The demonstration of navigation and control challenge logic begins with the ready check for the challenge. The ready check function contains the following checks:

1. Check the object list for two buoys that are within 10 meters of each other, plus or minus a tunable threshold.
2. Check if these buoys have color classifications.
    a. If one of the buoys is classified as either red or green, plot a waypoint that is in the center of the two buoys with a heading that is that keeps the red buoy to the port side of the ASV.
    b. Else, assume that it is on the start gate side of the navigation challenge, plot it's starting waypoint between the two buoys, assume the buoy that is furthest left of the

ASV is the "red" buoy, and plot the resulting heading.

The ASV then transitions to the starting waypoint before entering the main function. For the main task function, the ASV performs the following actions:

1. Clear the start gate by placing a waypoint that is 5 meters forward of the center of the start gate.
2. After this point is achieved, plot a waypoint 40 meters in the same bearing used to cross through the start gates.
3. Begin transitioning to the end point and search for the end gates as the ASV transitions to that point:
    a. The end gates must meet the same requirement as the start gate of having two buoys that are within 10 meters of each other, plus or minus a tunable threshold.
    b. The center of the end gate buoys must also be within a tunable distance (typically 35 meters) of the center of the start gates.
4. If the ASV does detect the end gates, then plot a waypoint that is 5 meters past the center of the end gates.
5. If the end gates are not found using the same criteria as the start gates, then a further check is preformed:
    a. Minion then looks if there is at least one object that is more than 10 meters away from the center of the start gates and is classified as a tall buoy with a reflector.
    b. If there is at least one object that is classified as a tall buoy with reflector and another object; classified as a tall buoy with reflector, a tall buoy, or an unknown object, that is within 10 meters plus or minus the threshold, then those two objects are assumed to be the end gates.
6. If there is a single tall buoy with reflector that has been classified, is within 15 meters, and at least 5 meters away from the ASV, Minion will enter into what is known as the skewed gate case:
    a. This case assumes that the end gates are offset to the left or right of the perpendicular bisector of the start gates.
    b. If this case is entered, then the ASV plots an intermediate point either to the right of the detected buoy if it is on the right-hand side of the perpendicular bisector, or vice-versa for the left side.
7. As the ASV transits to this new point, it continues looking for the end gate in the same manner as before:
    a. If the end gates are detected, then the ASV plots the remaining waypoint as was described previously.
    b. If it fails to detect the end gate before reaching the new end point, then the ASV assumes that it has successfully crossed through the end gates and merely failed to classify the other buoy.
    c. If the end gate is not detected in this case, then the final predicted score for accomplishing this task is reduced.
8. If the ASV instead made it to the original end point without detecting the end gates:
    a. Minion assumes it passed through the end gates along the way.
    b. The final predicted score for this task is reduced to a greater extent than the failure state in the skew case.
9. This task concludes by reporting back the expected score and that the task has been completed.

B. Scan the Code

The scan the code challenge logic begins with the ready check for the challenge. The ready check function contains the following checks:

1. Check the object list for detection of the light tower by perception.
2. Set a waypoint that is 20 meters away from the tower, facing the tower:
    a. The bearing of this waypoint with respect to the tower is governed by the configuration file
    b. Typically set to be either north or south to keep the sun from being directly in front or behind the camera

The ASV then transitions to the starting waypoint before entering the main function. For the main task function, the ASV performs the following actions:

1. Move to a waypoint that is 5 meters from the tower:
    a. The heading of the boat at this point is set such that either the port or the starboard camera has the tower fully in frame.
    b. The side that the boat will point at the tower is set in the configuration file.
2. After this point has been achieved, the boat is commanded to heading hold at that waypoint.
3. The Vision module is signaled to begin scanning the sequence. The processes for determining the sequence can be found in Appendix K.
4. The platform will wait for either vision to return a sequence or for a configurable amount of time (usually 2 minutes) to elapse.
5. If the sequence was returned, a check is performed on the result that was sent from Vision.
6. If the sequence contains no repeated values or unknowns, then this sequence is used.
7. If the data contains any unknowns in the sequence:
    a. The module will replace the unknowns with a randomly generated result of red, blue, or green (making sure to not repeat the same color for two consecutive values).
    b. Then the task function will reduce the expected score for the task based on the number of sequence elements that were replaced.

8. If no sequence is returned or the timeout elapses:
    a. The module will report back a randomly generated sequence that has no consecutively repeated values.
    b. The expected score for the task will be set to the minimum value.
9. This task concludes by reporting back the sequence, the expected score, and that the task has been completed.

C. Entrance and Exit Gates

The entrance and exit gates challenge logic begins with the ready check for the challenge. The ready check function contains the following checks:

1. The ASV searches for 4 buoys that meet the following criteria:
    a. 4 buoys are checked to see if they fall within line that is no more than 40 meters long, plus or minus a configurable threshold and within a tunable distance from the line of best fit.
    b. If these conditions are met, then the center point is used as a reference for the start waypoint, which is set a configurable distance (typically 15 meters) back from the start gate.
2. If 4 buoys cannot be found that meet those criteria, then the ASV attempts to see if 3 buoys meet the following criteria:
    a. The same check is followed as before, except the three buoys must now make a line that is no more than 30 meters long, plus or minus a configurable threshold, within the configurable distance from the line of best fit and has one of the end buoys classified as a tall buoy with reflector.
    b. The two tall buoys that are not a tall buoy with reflector are used as the reference for the start waypoint, which is again set back the configurable distance from the center point of those buoys.

The ASV then transitions to the starting waypoint before entering the main function. For the main task function, the ASV performs the following actions:

1. At the starting waypoint, the ASV is first commanded to heading hold at the same heading as the transition point.
2. MinionTask then communicates to the Hydrophones module to begin lowering the hydrophone arm and then to start listening for the pinger.
3. MinionTask will wait for either the Hydrophone module to return the pinger location or for a configurable amount of time (usually 1 minute) to elapse.
4. If the Hydrophone module does return the point where the reading was captured and the NED bearing of the pinger from the reading point, then a check on this information is performed to determine which gate to select.
5. To select a gate, the system compares the NED bearing of the pinger at the reading point to the NED bearings of each pair of buoys that makes up the gates with respect to the reading NED location:
    a. If the pinger bearing falls within any of those ranges, then the boat selects that gate as the one to go through, reports the gate chosen, and sets a waypoint at the midpoint of the line between the respective buoys.
    b. If the bearing reported does not lie within the buoy range of the outer most buoys, plus a configurable threshold, then the module will guess a gate to go through, reports the selected gate, reduces the score for this task, and sets the waypoint to that gate.
6. If the timeout occurs, then the module makes a guess on which gate the pinger is in, reduces the expected score for this task, and sets a transition point to be at the center of the selected gate.
7. Before transitioning to the center of the selected gate, Minion Task signals the Hydrophone module to raise the hydrophone arm and waits for this action to complete.
8. After the hydrophone arm is raised, the ASV transitions to the center point of the selected gate.
9. After the ASV has made it to the center point of whichever gate was selected, an intermediate waypoint is projected forward 15 meters from the center of the start gate.
10. After the ASV has achieved this waypoint, it sets an end waypoint that is 40 meters from the center of the start gate and begins driving towards this waypoint.
11. During this transit, the ASV checks for any objects that are within a cone that is mirrored about the perpendicular bisector to the start gate:
    a. The angular coverage and length of the cone are configurable tuning parameters.
    b. The coverage parameters are typically set to be 90° and 50 meters, respectively.
12. If any tall buoy without reflector that is the correct color to be circled is in the cone:
    a. The ASV immediately proceeds to circle that buoy in the direction based on the configuration file.
    b. Circling of a buoy is done by projecting a configurable number of waypoints (currently 4 waypoints) in succession in either a counter-clockwise or a clockwise pattern that is a configurable distance from the center point of the object to be circled (currently 5 meters in radius).

13. If two buoys are detected within the cone, but no color information is available, then the module refers to configuration file:
    a. A parameter is given for what the left most buoy in the cone's color should be.
    b. Based on this possible color information, the ASV then decides which buoy to circle based on the proposed color information and circles it.
    c. The module then reduces the expected score for this task.
14. If only one tall buoy without reflector is detected in the cone by the time the ASV reaches the projected end point:
    a. The ASV will proceed to circle that buoy in the correct direction.
    b. The module will reduce the expected score for this task further.
    c. This is done to ensure that the platform will at least have chance at circling the correct buoy, even in the event of a failure of the perception and vision modules to properly detect and classify buoys.
15. If the ASV reaches the projected end point without detecting any buoys within the cone:
    a. No buoys will be attempted to be circled.
    b. The module will reduce the expected score for the task to account for the fact that no buoy was circled.
16. This task concludes by reporting back the gate number traveled through, the expected score, and that the task has been completed.

D. Obstacle Avoidance and Totem Circling

The obstacle avoidance and totem circling challenge logic begins with the ready check for the challenge. The ready check function contains the following checks:

1. Check for three tall buoys without reflectors that are in an L-shaped configuration:
    a. This includes checking that any two buoys are within a configurable distance (typically 40 meters).
    b. This also includes checking that the 'L' is formed by three buoys that are 90° plus or minus a configurable threshold (typically 20°).
2. A transition waypoint is set 10 meters away from the buoy that is nearest to the ASV at that time and projected back 180° from where the opposite corner of buoy is or would be (if unidentified).

The ASV then transitions to the starting waypoint before entering the main function. For the main task function, the ASV performs the following actions:

1. A waypoint is set that is 10 meters past the buoy that is opposite the buoy the ASV is currently closest to.
2. As the ASV transitions to this point, it:
    a. Scans for tall buoys without reflectors that are within the area formed by the three buoys that define the obstacle field.
    b. Moves to avoid obstacles as they appear. Re-planning of the path as new objects appear can be found in Appendix L.
3. If a tall buoy with reflector is classified in the obstacle field are and has a color class associated with it, the ASV will do one of the following actions:
    a. If the buoy color is one that is supposed to be circled, then the ASV will proceed to circle that buoy.
    b. Otherwise, the ASV will ignore that buoy.
4. If the ASV detects a tall buoy without a reflector, but it does not have a color classification associated with it:
    a. The ASV will move to point either its port or starboard camera (a configurable selection) at the buoy while facing the buoy from north or south, whichever is closer, to give the best chance at giving classifying the color.
    b. If the color is classified, then the ASV will perform the checks and routines based on its color.
5. If the ASV get to the end point and there are any tall buoys with reflectors that are within the obstacle field:
    a. The ASV will circle the buoys in a random order.
    b. The module will also reduce the expected points for this task.
6. If the ASV gets to the end point and the are no tall buoys with reflectors in the obstacle field, then the ASV will reduce the expected score to account for the fact that no totems were circled.
7. This task concludes by reporting back expected score and that the task has been completed.

E. Underwater Ring Recovery

The underwater ring recovery challenge logic begins with the ready check for the challenge. The ready check function contains the following checks:

1. Check the object list for detection of a tall buoy with reflector that also has been classified with the ring buoy color.
2. Set a waypoint that is 20 meters away from the tower, facing the tower:
    a. The bearing of this waypoint with respect to the tower is governed by the configuration file.
    b. Typically set to be either north or south to keep the sun from being directly in front or behind the submarine's camera.

The ASV then transitions to the starting waypoint before entering the main function. For the main task function, the ASV performs the following actions:

1. Move to a waypoint that is 5 meters from the ring buoy.
2. After this point has been achieved, the boat is commanded to station keep at that waypoint.
3. The Submarine module is signaled to deploy the submarine. The methodology for retrieving the ring once the sub has been deployed can be found in Appendix B.
4. The ASV will wait for either the sub to report that a ring has been recovered or for a configurable amount of time (usually 2 minutes) to elapse.
5. If the submarine reports that the ring has been recovered, then the maximum expected points for the task awarded.
6. If the ring was not recovered, due to the timeout elapsing, the following checks will be done:
    a. If a ring was detected, then the module will reduce the expected points for this task to indicate that no ring was recovered.
    b. If no ring was detected, then the module will reduce the expected score for this task to account for no ring being detected or recovered.
7. Before completing the task, Minion Task will signal the Submarine module to recover the submarine and the ASV will wait to finish the task until the submarine has been reported as being recovered.
8. This task concludes by reporting back the expected score and that the task has been completed.

F. Identify Symbols and Dock

The identify symbols and dock challenge logic begins with the ready check for the challenge. The ready check function contains the following checks:

1. Check the object list for a dock bay detection.
2. The ASV then sets a transition waypoint that is 20 meters out from the dock bay, facing into the bay.

The ASV then transitions to the starting waypoint before entering the main function. For the main task function, the ASV performs the following actions:

1. The ASV sets a waypoint a configurable distance (typically 10 meters) from the entrance to the dock bay and pointing such that the port or starboard camera (configurable) to have the dock sign fully in frame.
2. The ASV is commanded to hold heading at the previous heading and waypoint.
3. Minion Task signals the Vision Module to begin scanning the dock symbol. The methodology for determining the dock symbol can be found in Appendix K.
4. The platform will wait for either Vision to return the dock symbol or for a configurable amount of time (usually 1 minutes) to elapse.
5. If the symbol was returned and is the correct symbol to deliver dock at, then the ASV prepares to dock at that bay.
6. If the returned symbol is not the one to be docked at or the timeout elapses:
    a. If the other docking bay has been detected, then a waypoint that is 20 meters from the bay and pointing towards the bay will be set.
    b. If the second docking bay has not been detected, then an intermediate waypoint will be set 180° from the current docking bay to a point that would be approximately 20 meters from where the bay would be expected.
7. If the ASV has the second bay detected, then it runs through steps 1-4 again.
8. If the symbol was returned and is the correct symbol to deliver dock at, then the ASV prepares to dock at that bay.
9. If the returned symbol is not the one to be docked at or the timeout elapses:
    a. Assume that you are at the correct docking bay to save time in transiting again.
    b. Prepare to dock at this bay.
    c. Reduce the expected score for this task.
10. If the second bay is not detected by the time the ASV reaches the expected second bay location:
    a. Assume the first bay was the one to dock at.
    b. Prepare to dock there.
    c. Reduce the expected score for this task further to reflect the even lower chance of being in the correct bay.
11. To dock in a bay, the ASV will use one of the following modes, which is configurable:
    a. The ASV will set a waypoint within the dock bay and send a specific message to path planner, telling it that this waypoint should be driven to in docking mode. More information on this mode can be found in Appendix L.
    b. The ASV will enter ramming mode, whereby the platform will drive forward for a specified period of time, ignoring all obstacles perceived to be in the way of the ASV's transit.
12. After docking, the ASV will reverse out of the bay This task concludes by reporting back the expected score and that the task has been completed.

G. Detect and Deliver

The detect and deliver task begins at the triggering of the ready check. The ready check contains the following criteria:

1. Detect and deliver sign is in the object list.

The ASV then transitions to the starting waypoint before entering the main function. For the main task function, the ASV performs the following actions:

1. Move to a waypoint that is 5 meters from the tower.

   a. The heading of the boat at this point is set such that either the port or the starboard camera has the tower fully in frame.
   b. The side that the boat will point at the tower is set in the configuration file.
2. After this point has been achieved, the boat is commanded to heading hold at that waypoint.
3. The Vision module is signaled to begin scanning the sign. The processes for determining the sign can be found in Appendix K**.**
4. The platform will wait for either vision to return a sequence or for a configurable amount of time (usually 1 minutes) to elapse.
5. If a symbol was returned, a check is performed on the result that was sent from Vision.
6. If the symbol is incorrect:
   a. A waypoint is plotted to the other side of the tower.
   b. Steps 2-5 are repeated.
7. If an incorrect sign is returned or the timeout elapses:
   a. The expected score for the task will be set to the minimum value.
8. If a correct sign is found:
   a. The module signals to the Turret module to begin searching for the target and to deliver the racket balls when ready. The methodology for doing so is found in Appendix C.
   b. Tasking waits for either the configurable timeout (usually 2 minutes) or a signal that the task has been completed.
9. If the Turret module responded with success:
   a. The expected score is increased.

IV. Experimental Results

The need for reliable mission execution lead to extensive testing of both the MinionTask module and the tasks it would run. This was initially performed in our simulation suite (see Appendix J for details) and on the water. The use of the simulator allowed for behavior primitives as well as full tasks build ups to be tested and debugged before they ever saw the water. It also allowed for edge case testing and mission information testing to see the impact of unexpected conditions on the task completion accuracy. Once on the water, the system was tested in practice where information such as timeout ranges and noise characteristics could be observed and then corrected for to give better completion accuracy.

A. Simulation

Testing for each task began in the simulator. Here all task behavior primitives, such as circling buoys, finding specific objects, looking for gates, etc. were tested to ensure high repeatability and reliability of these behaviors. Following this testing, the missions themselves were tested in the simulator to debug logic errors, to test edge cases, and to determine reliability when data was unavailable (like vision or hydrophones). Additionally, four tasks that were outlined above were able to be tested in MinionSim. The tasks that were tested in the simulator were the Navigation and Control Demonstration, Scan the Code, Start and End Gates, and Obstacle Avoidance and Totem Circling tasks. The following will detail the results of the simulation testing as well as the test cases that were used for each task.

Table I

Results of simulated mission testing

Mission	Testing Hours
Demonstration of Navigation and Control	20
Scan the Code	20
Entrance and Exit Gates	10
Obstacle Field and Totem Circling	8

The navigation task was the most heavily tested task in the simulator. Since it is the entry key to all testing that will need to be done on the course, it was critical that this challenge would be able to be robustly and reliably completed. In the simulator, the team was able to test several edge cases such as the gates being severely out of spec compared to the listed dimensions in the task outline. This included gates that were upwards of 40+ meters apart and in skewed configurations. Testing in the simulator also allowed testing of the platform's ability to complete this challenge both with and without color classification information being applied to the buoys. Due to this extensive testing throughout the logic development phase, this task was proven to be highly reliable. The demonstration of navigation and control task was able to accumulate approximately 20 hours of simulation testing.

Testing of the scan the code challenge was fairly limited in MinionSim. Without the ability to simulate the sequence in a way that would allow the vision module to be tested, simulation was limited to checking the movement routines of the platform throughout the task. However, this did allow for approximately 20 hours of behavior and waypoint logic testing to occur in simulation.

Similar to the Scan the Code task, limitations in MinionSim prevented the team from simulating pinger data, which would be used to determine the start gate to cross through. As a result, the platform would randomly guess and then transit through one of the three gates before completing the rest of the task.

Although the pinger could not be simulated, MinionSim was more than capable of testing of edge cases and of situations that would be out of spec for competition requirements. Some of the edge cases tested included unevenly spaced gates, gates with the start buoys positioned above and below the line of fit, and elongated start gates. The robustness of the module was also tested by simulating objects with incorrect or missing classifications as well as missing color identifications.

While testing this task, the circling of a specified totem was also tested, which allowed for behavior validation for the circling that would be done for the totem task. This also included testing the behaviors of the platform when only one buoy was detected in the region where a totem would be

expected for circling. In total, this task received around 10 hours of simulator testing.

Simulation testing also showed a clear correlation between the amount of sensor data available and the successful completion rates of a given task. For the Navigation challenge, the availability of color data from vision to supplement the perception classes drastically increases the successful completion rate. Without this information available, the module is highly susceptible to incorrectly performing the task. Errors in execution include going through the task backwards (i.e. going in through the end gate and leaving through the start gate) or going in sideways. The latter is an edge case that only occurs when the spacing between the start and end gates is close to that of the spacing between the tall buoys (including the tolerance). The first problem is solved when color information is available because the orientation of the red and green buoys is constant for the start and end gate. The edge case of going through sideways is also solved with color information since it would not be able to be selected as a start gate since it would be a red-red or green-green pair. Due to a limitation in the simulation suite, the only element of the Scan the Code mission that could be completed was having the boat go through the motions of attempting to read the sequence from the tower. It did, however, allow for robustness testing of the failed return case whereby vision did not return a valid sequence. In the event of a bad sequence, the system was supposed to return a sequence that it generated on the spot, which it did every time.

B. In-Water Testing

As was to be expected, in-water testing revealed logic bugs and edge cases that were not initially considered when testing each of the tasks. During the time that was available for testing each challenge, four of the five tasks were attempted on the water. These included the Demonstration of Navigation and Control, Scan the Code, Entrance and Exit Gates, and Docking challenges. Through all of this testing, an impressive 13 hours of in-water tasks testing was achieved.

Table II

RESULTS OF IN WATER MISSION TESTING

Mission	Testing Hours
Demonstration of Navigation and Control	5
Scan the Code	5
Entrance and Exit Gates	2
Docking	1

1) Demonstration of Navigation and Control

In water testing of the navigation gates challenge showed that a number of edge cases needed to be accounted for in the task logic. The first of these edge cases discovered was when the start gates were skewed. This was solved by adding a check in the logic for possible skewed gate conditions. However, after this fix was implemented, it was found through in-water testing that it causes the boat to plot waypoints away from the end gate location in an attempt to correct for what it thought was a skewed gate when only one of the end gate buoys had been classified. This problem was then solved by making the skewed gate case toggleable in the configuration file. Real world testing also showed that, due to potentially slow classification times, there was a need for this task to be able to find both the start and end gates with limited, and in some cases incorrect, classification information from Vision and Perception. After all of these changes were made, the navigation gates challenge was again attempted in the water. In total, the demonstration of navigation and control challenge received 5 hours of successful in-water testing.

2) Scan the Code

Testing of the scan the code challenge in real world conditions proved to be highly successful. This testing showed that scan angles which allowed the sun to appear in front of or behind the ASV hindered the sequence accuracy. The configuration files were changed to allow the platform to approach the tower at more ideal angles. Testing of this task also showed that the ideal scanning distance to get fast, reliable sequence returns was anywhere from 10-15 meters from the light tower, which was also edited in the configuration file. Testing of this task proved to be extremely successful with approximately 5 hours of successful in-water testing.

3) Entrance and Exit Gates

The acoustic gates task in-water testing revealed that the details about the ASV's real-world handling characteristics, primarily the turning radius of the ASV, that were not accurately modeled in the simulation environment. As a result, it was determined that the circling radius for the end buoys was too tight, so this parameter was added into the configuration file so it could be tuned. It was also discovered that the intermittent that would be generated after the ASV crossed through the gate, but before it went to search for the buoy to circle, was too close to the gate buoys. This would cause the ASV to make large, circular paths that would often put the ASV back through one or more gates while attempting to achieve the intermittent waypoint. This was then corrected in the acoustic gate task code by making the parameter for how far out the waypoint was placed past the gates tunable and further out. Unfortunately, the only element of this task that was unable to be tested on the water was the gate detection via the hydrophones. However, even without this element, the ASV was able to successfully detect the gates, navigate through a randomly selected gate, find the required buoy to circle, and circle that buoy in the correct direction. As such, there was nearly 2 successful hours of in-water testing.

4) Docking

The docking challenge was one that was only able to be simulated on the water through hardware-in-the-loop simulation of the dock. However, this did allow the logic for this challenge to be refined and tested. Through the in-water testing that was done, it was determined that the object growth that was done by the path-planner in order to ensure the ASV did not ram into obstacles prevented the ASV from being able to successfully complete the docking challenge.

Thus, it was noted that there needed to be a mode in the path planner that would ignore obstacles in the way and simply drive to a point. It was also noted that a fall back option that would be able to directly command the controls module, regardless of obstacles in the path, would need to be developed or revived. This resulted in the re-emergence of the ramming speed primitive, which overrides the path planner and sends direct messages to the controls module. Both additions would allow the ASV to successfully complete the docking challenge. Unfortunately, this resulted in only around an hour of in-water testing for the docking challenge.

V. Conclusion

Through testing both in simulation and in-water has allowed the tasking behaviors to be not only refined, but also made more robust. Through these countless hours of testing, many logic bugs, edge cases, and timeouts were able to be accounted for in each mission's structure. Additionally, by allowing each task to be dynamically launched as the required start conditions were met, the system was set to be far more robust to emerging data that streamed in than a traditional scripted mission framework would be. Together, this allows the Minion Task module to successfully complete both research and competition challenges with ease.

Appendix J: MinionSim Simulation Suite

Grady Delp

I. Introduction

During the 2016 Maritime RobotX Challenge, Team Minion recognized its need for a software utility that would enable the team to thoroughly tests its higher-level autonomy software, without direct access to the Minion ASV. A preliminary version of such a software module was developed over the course of the competition week, albeit a version that was very limited in scope and ability. Due to the utility and usefulness that the rudimentary version of the software provided, it was determined that a more featureful implementation of the same concept would take priority among new additions to the Minion ASV software stack.

II. Development Strategy

MinionSim was developed with the role of aiding the development of the other various software modules, including but not limited to the path planner and mission planner. For development, the authors for the other software modules were considered as the customers for the MinionSim product, and were polled for how they thought a bespoke simulation engine for the Minion ASV should be implemented.

While other options for creating a simulation environment for the Minion ASV were considered, including making use of the publicly available RobotX Gazebo simulation, it was decided that effort may be better utilized in creating a purpose-built simulator for Team Minion. This was chosen over developing a method to interface the Gazebo ROS environment with the otherwise LabVIEW-developed MinionCore messaging system.

The early consensus among team members is that MinionSim should operate as a virtual source for the types of MinionCore messages the sensors on the Minion ASV generate. These messages include examples such as the State message, containing position, heading, and velocity information, and the ObjectList message, containing all objects discovered by the LiDAR Perception module. By generating the same messages that other modules are expecting to receive from the State and Perception modules, MinionSim would effectively act as a virtual manifestation of the physical ASV. The other modules, MinionTask, Path Planner, and Controls, would then behave as if they were being run on water or in a staged competition scenario.

It was determined that the path of development for MinionSim would seek to fulfill all necessary functionality, and then move on to providing quality-of-life and user-experience improvements. This would serve two purposes: first, function would be prioritized over form, ensuring undue development time and resources were not exhausted on an aesthetically pleasing but ill-performing module. Second, it would ensure that any problems encountered during development could be addressed as core functionality issues, rather than as unintended side-effects of what may have been believed to be cosmetic tweaks. This prioritization was made on the part of the developers who had experienced ill effects from not making such a prioritization in the past.

To accelerate development, code-reuse was encouraged when possible. For example, previous work had been done to issue controls messages to the ASV's thrusters utilizing a USB gamepad plugged into a PC at the ground monitoring station. This was used to simulate the remote-control functionality of the ASV in the virtual environment. Other examples include the use of subroutines that reorder points into the format of the ObjectList message, and blocks of code used for loading XML files from predetermined or user-chosen filepaths.

III. Software Architecture

The software architecture of the MinionSim Simulation Suite contains two separate modules that are both necessary for operation. The first of these modules is MapMaker, and the second is the simulation engine, MinonSim.

A. MapMaker

MapMaker is used to create the maps that serve as configurations of objects and obstacles that get loaded into the environment of the virtual ASV. In the MapMaker, users are able to select a position on a grid, select the type of object that they wish to place there, and add it into the environment. The MapMaker user interface (UI) is shown in Fig 1.

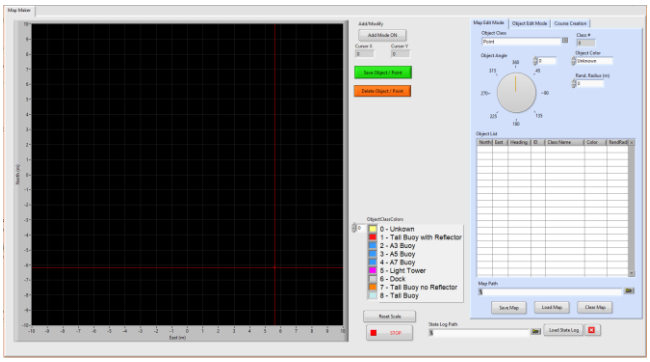

Fig 1: The default display for MapMaker.

Objects are placed into a list that is saved in memory, and only committed to disk when the entirety of the map is saved. This allows the user to modify or delete objects that may have been mistakenly placed or configured. Parameters that may be configured include the object's heading, color, and a randomization radius, which is used when loading the maps into MinionSim. When exporting the objects from MapMaker, they are stored to disk as an XML file containing all the parameters previously mentioned, as well as Northing and Easting position, in meters.

The objects themselves are also created using MapMaker,

using a "Object Class" called "Point". Creating an object in this manner is compliant with the way that the Perception module handles objects, which is as a list of boundary points. The boundary points must be created sequentially, to create objects that are not self-intersecting. In the current implementation, this responsibility lies on the user. Objects are also exported as XML files only containing the center location of the object, the list of bounding points, and the integer that assigns it to an ObjectClass, which is standardized across all modules in the Minion software stack.

A more recent development that was added to the MapMaker module is the ability to generate an entire RobotX Challenge course with a single button. This interface is shown in Fig *2*. Several parameters are available for configuring these courses, including turning specific tasks on and off, the size parameters for tasks that have dimensional parameters (Navigation Gates, Pinger Gates), what object configuration files to use for each task, and the overall course size to place the objects within. Each task is randomly placed sequentially, with subsequent tasks ensuring that they are placed outside of the bounds of a previously placed task. To avoid scenarios with a small course, many tasks, and oversized keep-out zones, these checks can only run a preconfigured number of times, with the user being notified of violations of the keep-out zones.

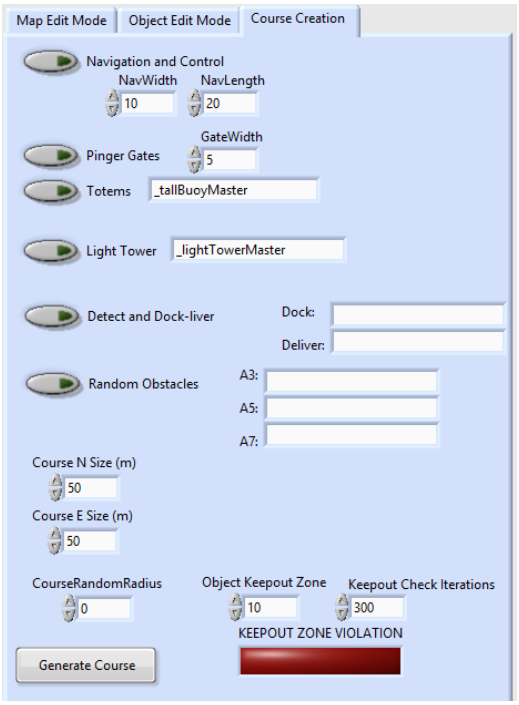

Fig 2: The Course Creation panel built into MapMaker.

B. *MinionSim*

MinionSim, shown in Fig 3, serves as the simulation engine that performs the majority of the functions of the MinionSim Simulation Suite. This software module is itself programmed in a modular fashion, with each submodule running in parallel and handling the processing and generation of different data. This data is shared among the submodules where necessary and is distributed to the other software modules operating in the Minion software stack via the appropriate MinionCore messages.

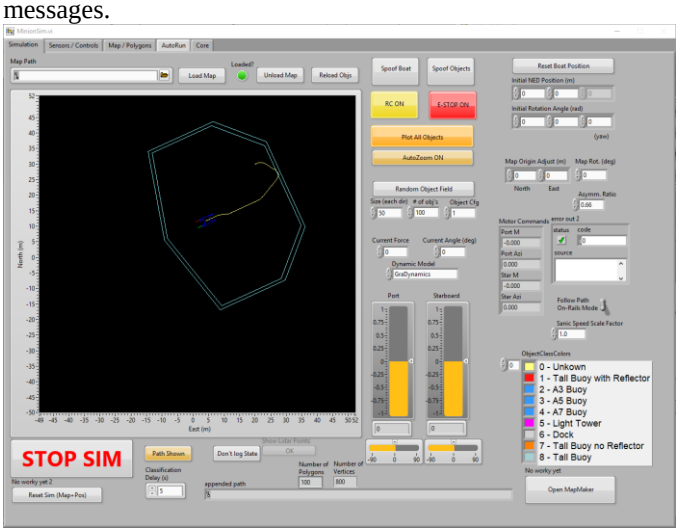

Fig 3: MinionSim Simulation Engine

1) *State*

The State MinionCore message was the first message to be developed as part of the MinionSim module. The purpose of the State message is to broadcast to all other modules the boat's current pose. While this typically includes three-dimensional elements of the vehicle, including pitch and roll, and height above sea level, the majority of customer modules disregard this information as provided by the ASV's GPS and IMU. As such, MinionSim treats the vehicle as a two-dimensional object operating on a two-dimensional plane. This served to reduce the computational load.

The modules that performs generation of the synthetic State message operate at a 100Hz refresh rate, similar to the update rate from the physical ASV's State module. The positional information is generated by integrating the previous position with the current velocity to determine what the new position of the ASV should be. Similarly, the velocity is generated by integrating the previous velocity and the current acceleration, to determine the new velocity of the vehicle. Cartesian (XY) and angular values for position and velocity are both handled similarly in this manner.

The acceleration of the vehicle is determined by summing the forces and moments acting on the vehicle, including thrust and drag, and an overall external force which can be considered a combination of wind and current. The thrust values also take the azimuth angle of the thrusters into account. The external force parameter is treated as a user-configured constant, to see how the vessel would operate in a high-wind or high-current situation.

2) *Perception*

Several MinonCore messages are associated with the broad term 'perception," including the LiDAR object detection and classification module, and the vision used to detect object color. For programming simplicity, these are largely handled in the same way in MinionSim.

When a map file is loaded into MinionSim, the series of points associated with each object is loaded into memory. Sequentially, each set of points is run through a point-in-polygon algorithm, with the polygon being a seven-sided figure

that is configured to be the furthest bounds that the ASV can reliably and confidently detect and classify objects. The polygon is translated and rotated to always match the position and orientation of the virtual ASV.

Objects that fall outside of the polygon are ignored and held in memory to be checked against the polygon in the next iteration as the virtual ASV continues to transit the world. Objects that lie on the edge of the polygon continue to be checked, but the vertices of the shapes that lie within the polygon are forwarded to the next portion of this submodule. Objects that lie fully within the perception polygon are removed from the list of objects to repeatedly check and are also forwarded to the next portion of the submodule.

Following detection within the perception polygon, objects are assigned an unknown classification and color. If the object is new to the list of objects that have been fully within the perception polygon, they are also assigned the current timestamp. Once an amount of time, configurable by the user, has passed, the object is given its preassigned classification and color, which was assigned to the object during the map-making portion of the MinionSim workflow. This delay in classification is similar to what occurs within the Perception module, as a delay occurs before objects are classified, based on the time required to classify the object with a degree of confidence. This amount of time can be reconfigured based on how long the Perception module is typically taking to classify objects; if the Perception module is improved to be more robust and confident more quickly in the future, this delay can be decreased.

Future developments of MinionSim intend to tie the addition of color to an object's classification to the heading of the boat, as well as an independent range of color identification. This will make the simulator more realistic and will not allow the virtual ASV to detect color when the cameras are not facing towards the object.

The randomization radius, applied to each object in MapMaker, comes into play in the section of MinionSim that loads objects into memory prior to checking them against the perception polygon. This serves to mildly adjust the parameters of the map each time the same file is loaded into MinionSim, similarly to how buoys on the water may slightly drift in relation to each other between autonomous runs of the ASV. Functionally, this requires the thresholds that object positions are tested against to be more robust than they might otherwise be.

3) Hardware-in-the-Loop

The hardware-in-the-loop function of MinionSim serves the specific purpose of being able to perform on-water tests with more course elements than may otherwise be possible to test with. This includes allowing the ASV's control module to be tested on-water against multiple virtual docks without needing to deploy large numbers of floating platforms, testing heading- and position-hold functions when working with a stationary virtual object such as the light tower, or creating large virtual obstacle fields to test the robustness of the path-planning and obstacle-avoidance subsystems. Hardware-in-the-loop mode disables the synthesis of state and status messages, and reduces MinionSim to a perception ObjectList message broadcaster. Other information necessary to operate the simulator is received from the applicable sources rather than being generated by the simulation itself.

4) Simulator Automation

One feature of MinionSim that has not seen much use in 2018 is the ability to trigger it to run automatically, with a specific MinionCore message that contains several parameters for operation. These include all the user-configurable parameters that can be set at run-time, as well as the map that should be loaded prior to running the simulator. The intention of this functionality is to permit unattended batch-testing of many simulations that could be analyzed for success rates and determine what parameters of other modules may require tweaking. However, the module necessary to run batch-testing that would trigger this functionality in the simulator was not finished during the 2018 competition cycle. This functionality will likely be revived in preparation for future RobotX competitions.

5) On-Rails Path Following

When testing the mission planning software, it was discovered that the virtual ASV did not behave exactly as the physical ASV did on-water. This is likely due to the simplifications that were made to the physics model that the virtual ASV operates on. Because of this, it was seen necessary to implement a no-error path-follower in MinionSim. This feature, togglable by the user on a per-run basis, causes the virtual ASV to follow the path generated by the path-planner on a node-by-node basis. This speed is also tunable, allowing testing of long-duration operations, such as search patterns, to be condensed, quickening the pace of development.

IV. Experimental Results

Throughout the preparations for the 2018 RobotX Challenge, MinionSim has achieved strong positive feedback and high levels of use, proving it was a worthwhile addition to the Minion software stack.

For the purposes of testing Mission Planner, it achieved close to 250 hours of testing. This resulted in the statistics and success rates listed in Appendix

For the purposes of testing and debugging the path-planning and controls modules, MinionSim achieved close to 50 hours of testing. This resulted in higher confidence in the simulator, more robust path-planning algorithms, and more productive on-water controls and path-planner testing.

Appendix K: Vision

James B. Near III, David J. Thompson

I. Introduction

The 2018 vision module was desired to integrate deep learning in a manner that would allow users to rapidly deploy trained network models to the Minion platform, as well as other GPU-enabled Windows machines. This is due in part to the difficulties integrating deep learning into the 2016 platform, particularly with providing a real-time pipeline for image capture and inference. The prior implementation was also only capable of running a single neural network and could not be easily deployed to other machines. The 2018 implementation utilizes a number of networks for completing the scan the code, circle the totem, detect and deliver, and docking challenges.

A. Design Strategy

The Minion ASV was designed to support multi-modal approaches for object detection and classification. Minion uses the its array of LiDAR sensors for initial object detection. LiDAR is subject to low amounts of noise and has high position accuracy relative to the camera, making it a more reliable sensor for object detection. Furthermore, it is more computationally efficient to detect objects with the LiDAR than using a deep neural network.

However, to support classification, it is necessary to add cameras to retrieve color and shape information. To enable the use of cameras as a secondary sensor, it is necessary to allow multiple neural networks to run simultaneously. For example, the LiDAR may detect a totem in the course, but must rely on the camera to determine the color to be red. One network may be assigned solely to determine color given a cropped image of an object. Similarly, a different neural network may be used to determine the shape and color of the docking bay signs. Lastly, in order to support a failure of the LiDAR sensors, it is desired to have a large object detection neural network for determining object position and classification.

To that end, Team Minion has chosen to train several deep neural networks with the TensorFlow framework to enable object detection and classification. A neural network may be turned on or off depending on the current state of the vehicle in the course or the current failure mode of the LiDAR sensors. The following sections will explain the individual network or networks used to complete each of the tasks in the 2018 challenge.

II. Process

A. Overview

Vision was used on three main tasks: determining the sequence of a light tower, classifying different colored buoys, and classifying instructional signs on the dock. All benchmarked tasks in this section were achieved by using the software and hardware listed in Table I.

Table I

VISION HARDWARE AND SOFTWARE

Software	Hardware	
Cudnn 7.5.1	CPU	Intel Xeon E5-2620v3 6-Core 2.4Ghz
TensorFlow 1.5	GPU	Nvidia GTX 1080
Cuda 8.0	RAM	32GB DDR4 ECC Memory
Opencv 3.4.1	Camera	FLIR Blackfly BFLY-PGE-31S4C-C
Visual Studio C++ 2015	Lens	Theia Technologies ML410M

B. Convolutional Neural Networks

Convolutional neural networks (CNN) are a subset of neural networks that accept an input as an image and are used extensively in image recognition and classification. Conventional neural networks cannot handle the size of an image, as the weights start to increase exponentially as the image size increases. CNNs address this issue by using convolutions to reduce the size of the output layer. At the output layer, the size will have been reduced such that there is one node for each output class. This output layer may then be used to determine final class, most often by using the output node with the highest value. The basic approach is shown in Fig 1.

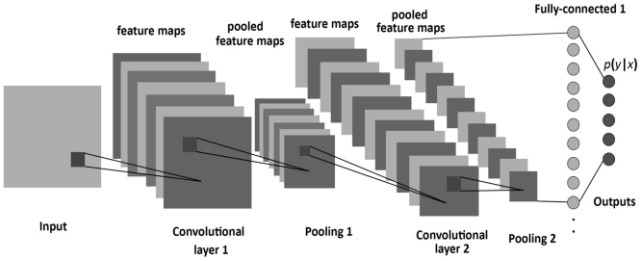

Fig 1. CNN Architecture

A convolutional neural network has many operations to consider. The first of these operations is creating a feature map through convolutions. This is controlled through the tunable parameters of depth, stride, and padding [1]. Once the feature maps are created, a Rectified Linear Unit (ReLU) is applied to the image. This applies an absolute value

function to all values. The next step after the ReLU is pooling. Pooling reduces the complexity of the model. Different methods and sizes of pooling can be used such as max, min and average pooling. The specific kind depends on the network [2].

After the first convolution, ReLU, and pooling phase are completed, the task is run again on the previous layer. This can be once or many times. The exact number depends on the specific model. The final step is to run the final pooled results into an *n* number of fully connected layers that will then predict the class [3]. This architecture allows neural networks to run efficiently and accurately on images of many sizes with numerous features and complexity.

For this project, the CNNs that were chosen were based upon the ability to retrain already tested networks. This was largely because creating specific networks for these tasks would provide little benefit for the time it would take to design and program the network. Since TensorFlow offers the ability to easily retrain and test neural networks, as well as provide a C-API for deployment it was chosen over other frameworks.

Two different types of network training methods were chosen: one for object detection and one for object classification. These methods were selected based upon the number of networks available and the retraining ability. Object detection uses the TensorFlow Object Detection API [4], and object classification uses the TensorFlow Image Feature Extraction Module [5].

C. Training Neural Networks

Once the software used on Minion was determined, the training method had to be selected. The training method had to allow for different networks to be trained using the same data and preprocessing to enable comparing different networks against each other on the water. Because of this need, transfer learning was used for training. Transfer learning is slower than creating custom networks for a specific task but benefits from already being proven to work.

The TensorFlow Object Detection API [4] was used to train detection networks. This method allowed for different networks to be trained and tested using the same data with just changes to a configuration file and a pretrained frozen inference graph. It took around 5-10 hours for a network to converge once training was started. For the classification networks, the TensorFlow Hub classification retainer method [5] was used. This method allowed for quick classification retraining in about 30 minutes on the system specified in Table II.

Table II

HARDWARE SPECIFICATIONS OF TRAINING SYSTEM

CPU	2 Intel Xeon E5-2670v3 12-Core 2.3Ghz
GPU	Nvidia Quadro M6000 24GB
RAM	128GB DDR4 ECC Memory

D. Light Tower

1) Task Description

The light tower task involves accurately detecting and identifying the sequence of a changing light tower (greater than 90 percent accuracy on panel color) in real time (greater than or equal to 5 frames per second) using hardware and software available to Minion. Fig 2 shows the competition light tower below.

Fig 2: Competition Light Tower

The light tower can change between four different colors. These are black, blue, red, and green. The sequence is generated by displaying the black panel for 1 second, then three colored panels (green, red, or blue) for 1 second each, and finally, the black panel again for 1 second. The only caveat for the three middle colors is that a single color cannot be repeated consecutively but may be used multiple times in a sequence. For example, a sequence of Blue-Blue-Green is not allowed but Blue-Green-Blue is allowed. Due to these rules there are only 12 possible sequence combinations, as shown in Table III.

Table III

VALID LIGHT TOWER SEQUENCES

Seq.	Color 1	Color 2	Color 3
1	Blue	Green	Blue
2	Blue	Green	Red
3	Blue	Red	Blue
4	Blue	Red	Green
5	Red	Green	Red
6	Red	Green	Blue
7	Red	Blue	Red
8	Red	Green	Blue
9	Green	Red	Green
10	Green	Red	Blue
11	Green	Blue	Red
12	Green	Blue	Green

From this task description, the below self-imposed requirements were derived:

- The time to complete a single-color classification and loop through the sequence detector shall be no

more than 200ms.

- The sequence detector shall be robust enough to not give a false positive in more than 1 percent of tests.
- The sequence detector shall correctly identify the sequence more than 90 percent of the time in one attempt.
- The sequence detector shall correctly identify the sequence more than 99 percent of the time in two attempts.
- The sequence detector shall identify a sequence in a minute or less.

2) Task Methodology

To start the task, MinionTask and Perception are used to classify the light tower and position one of Minion's cameras in frame of the light tower at a distance between 5 to 15 meters. Once this has been accomplished, vision is activated to determine the sequence. This is accomplished in five main steps:

1. Use the Inception V2 object detection neural network to identify and crop the light tower from the raw camera image.
2. Use the Coco Mobilenet V1 object detection neural network to crop the light panel from the light tower image.
3. Use the Inception V3 object classification neural network to determine the color of the light panel.
4. Feed the color result into the sequence detector to determine the light tower sequence.
5. If a sequence is not found in one minute, position the boat 180 degrees on the other side of the tower and try again.

For step one, a full camera image (~2MP) is given as an input to detect the sequence. This step utilizes the Inception V2 object detection network to find the light tower in the frame and crop it out of the raw image. The Inception V2 network was chosen, along with the other networks in this paper, based on an accuracy and speed tradeoff. This task was determined to be the most important because all other networks depended on this result being correct. Because of this dependency, a larger, more accurate network was chosen for increased accuracy. After testing the speed of different networks running on Minion, the Inception V2, Mobilenet V1, and the Faster RCNN Resent were the only networks that could keep the framerate above 5 FPS. The general results of these networks are shown in Table IV below using the same training data.

Table IV

LIGHT TOWER CROPPING NETWORK COMPARISON

Network	Speed [ms]	Accuracy [%]
Inception V2	100-150	100
Coco Mobilenet V1	20-40	81.3
Faster RCNN Resnet	120-185	99.2

As can be seen from the table, the Mobilenet does not work reliably due to its small size. The inception V2 and the Faster RCNN Resnet both provide near perfect accuracy but the inception V2 is faster, thus it was chosen in this application. The region of interest (ROI) from the Inception V2 network is shown in Fig 3 below.

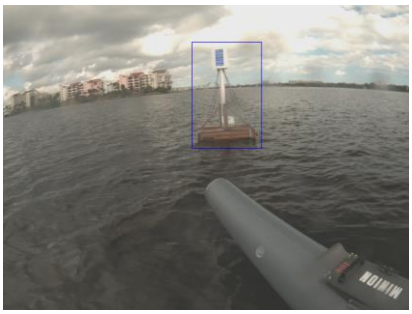

Fig 3: ROI of Light Tower

Once the tower was cropped, the second network could be initiated. This network could be smaller and lightweight compared to the Inception V2 because by this point, every image should be similar with the light panel in the same location, orientation, and size. The only difference between the different images is the background and the light panel color. Due to the similarities between these images the Mobilenet was tested again. The Mobilenet gave close to 100 percent accuracy at a speed of 20-40ms per image, which is the fastest of all available networks. Therefore, the Mobilenet was chosen to further crop the light tower down to the light panel. The ROI from the Mobilenet is shown in Fig *4*.

Fig 4: ROI of Light Panel

After the panel is cropped, the color is determined using a classification neural network as identified in step three. The same process for determining the detection network was used for the classification network. A comparison of networks was conducted. These results are displayed in Table V.

Table V

PANEL COLOR CLASSIFICATION NETWORK COMPARISON

Network	Speed [ms]	Accuracy [%]
Inception V3	20-40	98.2
Mobilenet (Full)	20-35	78.6
Mobilenet (Half)	10-15	76.2
Resnet V2	50-70	87.3

It can be seen from Table V that the Inception V3 network offers the best combination of speed and accuracy and thus was chosen as the classification network for this task. This network feeds the color result into the sequence detector.

Once the sequence detector receives a color from the Inception V3 network, it will attempt to vote on a sequence. The sequence detector works by using a moving mode over a range of the last 5 colors detected. The mode will vote for a color, which is appended to the color list matrix. This matrix is then run through template matching to detect if a valid sequence is present over each range of the last five colors. If a sequence is detected it will vote for that sequence as a possible correct sequence. If a sequence is voted as a possible candidate three times, then that sequence is reported as the correct sequence, and is sent to MinionTask.

In the case where a sequence is not found, or a sequence is not voted for three times in the one-minute allotted time, then the boat will be repositioned 180 degrees from the current position. This is done because most of the time the boat cannot determine the sequence, it is due to the boat's orientation relative to the sun's light. If Minion cannot recognize the sequence on the second attempt, then MinionTask will move onto the next task and come back to the light tower task if time is left in the competition.

III. Results and Discussion

The overall speed and accuracy of the combined networks are tabulated in Table VI.

Table VI

COMBINED SPEED AND ACCURACY OF LIGHT TOWER NETWORKS

Network	Purpose	Type	Speed [ms]	Accuracy [%]
Inception V2	Crop Tower	Detect	100-150	100
Coco Mobilenet V2	Crop Panel	Detect	20-40	99.7
Inception V3	Identify color	Classify	20-40	98.2
Combined	All	Both	140-230	97.9

These results meet the requirements for the accuracy and almost always meet the requirements for the speed. The only time that speed is not met at 5 FPS is when the boat is sharing its resources with other tasks at the same time as the neural networks are running. This causes a drop to 4.34 FPS at the lowest, but the sequence detector was still able to correctly identify the sequence due to the robust nature of the algorithm. Furthermore, to get a sequence wrong, a color must be classified wrong three out of the five times in the moving mode. The probability of this occurring is calculated using Formula 1.

$$\sum_{k=x}^{n} \binom{n}{k} p^k (1-p)^{n-k} \quad (1)$$

The probability of a color being classified wrong at least three times in 5 as less than 1E-4 percent which almost guarantees that a single color will not be misclassified using the sequence detector. If this is extrapolated over the chance of a single sequence having a single-color wrong, there is still less than a 1E-2 percent chance this occurring. The sequence detector, along with the overall accuracy rate of the light tower detector code, can correctly detect a given sequence under the given requirements and constraints. The only caveat for this statistical analysis is when the light panel is being washed out by the sun. In this circumstance the classifier cannot reliably differentiate between green and blue.

The individual performance of the Mobilenet and Inception V2 network results are tabulated in Table VII.

Table VII

LIGHT TOWER AND PANEL DETECTION NETWORK RESULTS

Network	Speed [ms]	Median Proposal Confidence	Mean Proposal Confidence	Actual Accuracy
Light Tower	100-150	100%	100 %	100%
Light Panel	20-40	100%	99.70%	100%

From this table, these two networks were nearly perfect in identifying the light tower and panel, respectively. This was true through all lighting and weather conditions.

The results of the classification network are shown in Table VIII below as a confusion matrix.

Table VIII

LIGHT PANEL CLASSIFICATION NETWORK RESULTS

		Test Class			
		Black	Blue	Red	Green
Predicted Class	Black	100.0%			
	Blue		96.4%		3.0%
	Red			100.0%	
	Green		3.6%		97.0%

This table shows that the classification network worked well in classifying all the colors except for the occasional switch of the blue and green under bad lighting conditions.

These conditions are shown in Fig 5.

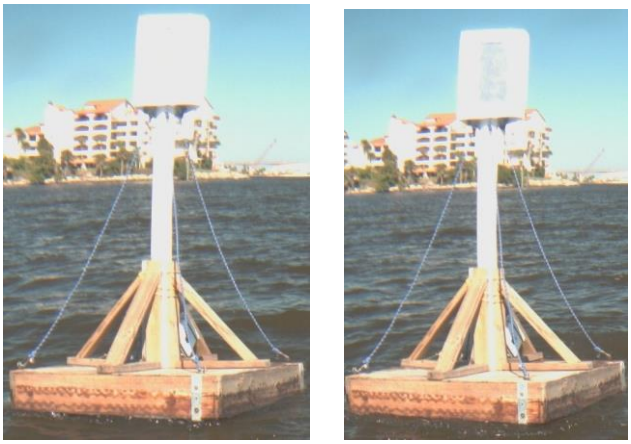

Fig 5: Bad Lighting Condition Examples

The image on the left shows an example of an image that is impossible to classify at all, as the entire panel is whited out by the sun's reflection. The second image is barely discernable as blue. Other than these conditions, which MinionTask corrects for by repositioning Minion, the entire light tower sequence detector task performs at or above the given requirements. This implementation of the task shows a huge improvement over the 2016 implementation. During water testing, the sequence was identified right at over a 90% rate, while in 2016, the sequence was not correctly identified once.

A. Object Classification

The other use of vision for Minion was classifying the different colored buoys and the signs for docking. Both tasks were also completed using the Inception V3 classification network. These networks were retrained individually for the different tasks they were applied to.

The requirements for both these tasks were:

1. The object classification networks shall run at a speed of 10 frames per second or better.
2. There shall be no more than 1 false positive in 100 classifications attempts.
3. A classification attempt shall be attempted every two second or less.

For the buoys, the network was segmented into classifying the buoy as either red, green, blue, yellow, white, or black. For this task, only color was of importance, so the type of buoy did not matter and was not accounted for in the classification because the LiDAR is generally accurate enough to determine the difference between the gate buoys with retro-reflectors and the totem buoys. The data was used on any task that required knowing the buoy color.

A confusion matrix is shown below in for the general performance of this network. For this task, if a color could not be assigned with over a 70 percent accuracy over a range of 10 images, the color would be returned as unknown. This was done because a false positive is almost always worse than having an unknown color, as other logic can be applied to determine the corrective action to take. Table IX shows a confusion matrix for the buoy color classification.

Table IX

CONFUSION MATRIX FOR BUOY COLOR CLASSIFICATION

		Test Class					
		Red	Green	Blue	Yellow	White	Black
Predicted Class	Red	99.9%					
	Green		99.5%				
	Blue		0.2%	100.0%			
	Yellow				100.0%		
	White	0.1%	0.2%			100.0%	
	Black		0.2%				100.0%

As can be seen from this testing data set, the classifier worked within the given requirements. This test was performed in an area where the sun did not wash out or dilute any colors. When the buoys are washed out, the accuracy drops into the 80 percent range. An exact number was not obtainable since all data from these cases was used for training, to increase the accuracy in this scenario.

The second classification network that was trained was for the docking signs. This network was used to detect the shape and color of the different docking signs. The same process was used for this task as was for the buoys. The network identified the color (red, green, or blue) and the shape (cruciform, circle, or triangle) using the Inception V3 network. Due to time limitations, there was no available data for testing this network. Furthermore, as a secondary option to LiDAR, an Inception V3 network was trained to crop down a raw image containing the dock sign in a similar manner in comparison to the scan the code challenge.

REFERENCES

[1] A. M. Saleh Albelwi, "A Framework for Designing the Architectures of Deep Convolutional Neural Networks," *Entropy*, vol. 19, no. 6, 2017.

[2] W. Z. J. Y. W. L. a. L. H. Yuguo Qian, "Comparing Machine Learning Classifiers for Object-Based Land Cover Classification Using Very High Resolution Imagery," *Remote Sensing*, vol. 7, pp. 153-168, 2015.

[3] Schuber R. Carvalho, "A Deep Learning Approach for Classification of Reaching Targets from EEG Images," Immersive Interaction Group (IIG), Ecole polytechnique f ´ ed´ erale de Lausanne (EPFL), Switzerland, 2017.

[4] TensorFlow, "TensorFlow Object Detection API," Github, 31 March 2018. [Online]. Available: https://github.com/tensorflow/models/tree/master/research/object_detection.

[5] TensorFlow, "How to Retrain an Image Classifier for New Categories," 20 November, 2018. [Online]. Available: https://www.tensorflow.org/hub/tutorials/image_retraining.

Appendix L: Controls and Autonomy Operations

Marco A. Schoener, Timothy A. Zuercher

I. Introduction

In RobotX 2016, controls performance was a weakness that Team Minion elected to address for 2018. The development was split into two fronts. First, the team wanted to develop a strong dynamic model for the WAM-V platform that could be used to develop and test new controls methods, and to improve mission simulations. Second, the team wanted to develop a new control system that would provide a much greater degree of reliability and robustness.

Previously, Minion was a differentially-steered vehicle due primarily to the available propulsion system. The new azimuthing capabilities brought a need for an actual solver to determine the thrust/angle solutions. A nonlinear equation optimizer system was developed to use the available motor configurations to find the closest solution to the thrust configuration settings.

This appendix is organized into several sections. Section II discusses the approaches used for modeling and estimating vehicle parameters. Section III describes the approach to vehicle controls. Section IV presents results from using the control system *insitu*. Lastly, Section V provides a summary of the Appendix.

II. BOAT ESTIMATION

The first goal of the 2018 controls overhaul plan was to develop a robust model of the WAM-V's dynamic performance. The generic model for the dynamics of a maritime platform is described in [1]. The team choose to implement a model using the dynamic equations from several sources [1] [2] [3] [4], combining the generic model from [1] with models for disturbances and actuators. A grey box estimation technique would then be applied using the model to estimate the model parameters [5]. After the parameters are determined, the model could be used for simulation, control, and planning.

Table I

Measurable Boat Parameters

Parameters	Value
Length, m	4.5
Length on Water Line, m	4
C.G. (x,y), m	(-0.01, 0.05)
Mass, kg	254.50
Moment of Inertia, kgm^2	303.50
Beam Length, m	2.00

A. Frames

For modeling and control there are two coordinates frames that are important. The first coordinate frame is the body-fixed frame. This right-handed cartesian frame in $\mathbb{R}^3$ is aligned with the principal axes of inertia, generally resulting in the x-axis positive forward (bow), the y-axis positive to the right (starboard), and the z-axis positive downward (towards the water). This is also known as the Front-Right-Down (FRD) frame. Velocities, forces, and moments are usually represented using this body-fixed frame.

The second frame is the inertial frame, which uses the NED frame representation. The NED frame is a transformation of the Earth Centered Earth Fixed (ECEF) frame, so the origin is on the surface of the earth and located nearby. NED is also a cartesian frame in $\mathbb{R}^3$ with the x-axis positive toward north, the y-axis positive towards the east and the z-axis positive down. NED only approximates an inertial frame. The approximation gets worse the further from the origin you travel. The relatively small operating area of the vessel makes this approximation unnoticeable. The NED frame is used to represent the positions and velocities of a system.

In a full six degree of freedom (6-DOF) model, a complete rotation matrix or quaternion would need to be used to transform between the FRD and NED frames. However, because the model will be simplified to a three degree of freedom (3-DOF) planar representation, only a singular rotation needs to be applied, Eqn. *(1(4*.

$$\vec{\boldsymbol{v}}^b = (u, v) \tag{1}$$

$$\vec{\boldsymbol{v}}^n = (v_N, v_E) \tag{2}$$

$$R_b^n = \begin{bmatrix} \cos(\varphi) & -\sin(\varphi) & 0 \\ \sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$\vec{\boldsymbol{v}}^n = R_b^n \vec{\boldsymbol{v}}^b \tag{4}$$

In this rotation, u and v are surge (forward) and sway (sideways) velocities respectively; v_N and v_E are velocity in the northing and easting directions respectively; and ψ is the yaw (heading) angle of the vehicle, measured off true north. An example diagram showing these frames is in Fig 1.

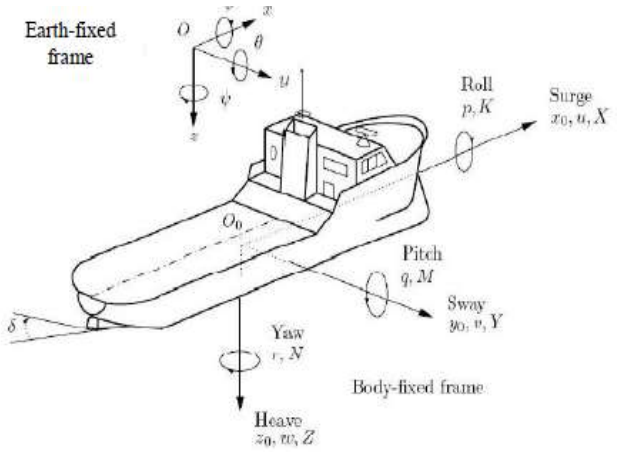

Fig 1. Diagram of FRD frame on a marine vessel. Retrieved from [1]

B. Equations of Motion

The general equations of motion are given in [1]. Using this general form, the following subsections break the model into four components: rigid-body motion, hydrodynamic motion, disturbances, and actuator forces. Each section describes that aspect of the model's construction in detail and can be combined into a complete parameterized model of the vehicle.

1) Rigid-Body Motion

A 6 DOF model as given by [1] is shown in Eqn. *(5(10.*

$$\begin{aligned} X = {} & m\big(\dot{u} - vr + wq - x_{CG}(q^2 + r^2) \\ & + y_{CG}(pq - \dot{r}) \\ & + z_{CG}(pr + \dot{q})\big) \end{aligned} \tag{5}$$

$$\begin{aligned} Y = {} & m\big(\dot{v} - wp + ur - y_{CG}(r^2 + p^2) \\ & + z_{CG}(qr - \dot{p}) \\ & + x_{CG}(qp + \dot{r})\big) \end{aligned} \tag{6}$$

$$\begin{aligned} W = {} & m\big(\dot{w} - uq + vp - z_{CG}(p^2 + q^2) \\ & + x_{CG}(rp - \dot{q}) \\ & + y_{CG}(rq + \dot{p})\big) \end{aligned} \tag{7}$$

$$\begin{aligned} K = {} & I_{xx}\dot{p} + \big(I_{zz} - I_{yy}\big)qr + (\dot{r} + pq)I_{xz} \\ & + (r^2 - q^2)I_{yz} \\ & + (pr - \dot{q})I_{xy} \\ & + m\big(y_{CG}(\dot{w} - uq \\ & + vp) \\ & - z_{CG}(\dot{v} - wp + ur)\big) \end{aligned} \tag{8}$$

$$\begin{aligned} M = {} & I_{yy}\dot{q} + (I_{xx} - I_{zz})rp - (p + qr)I_{xy} \\ & + (p^2 - r^2)I_{xz} \\ & + (qp - \dot{r})I_{yz} \\ & + m\big(z_{CG}(\dot{u} - vr \\ & + wq) \\ & - x_{CG}(\dot{u} - uq + vp)\big) \end{aligned} \tag{9}$$

$$\begin{aligned} N = {} & I_{zz}\dot{r} + \big(I_{yy} - I_{xx}\big)pq - (\dot{q} + rp)I_{yz} \\ & - (q^2 - p^2)I_{xy} \\ & + (rq - \dot{p})I_{xz} \\ & + m\big(x_{CG}(\dot{v} - wp \\ & + ur) \\ & - y_{CG}(\dot{u} - vr + wq)\big) \end{aligned} \tag{10}$$

Where X, Y, and W, are surge sway, and heave (vertical) forces respectively; K, M, and N are roll, pitch, and yaw moments respectively; w is the vertical speed (heave) and p, q, and r are the roll, pitch, and yaw rates respectively; m is the mass in kg and I is the mass moment of inertia.

As in [1], this can be represented in a vector form as

$$M_{RB}\dot{\vec{v}} + C_{RB}(\vec{v})\vec{v} = \tau \tag{11}$$

where M is the mass matrix, C is the Coriolis matrix, τ is the actuator forces and moments, and v is the state vector in the FRD frame.

To simplify this initial set of equations three assumptions are applied:

- The roll, pitch, and heave states are neglected. This results in a horizontal plane model. Wave motion is not properly captured with this assumption.
- The x and y center of gravity do not necessarily sit at the boat's center of origin (C.O.)
- A coupling exists between the surge and yaw states

Applying these assumptions leads to the following form of the model:

$$\vec{v} := (u, v, r) \tag{12}$$

$$\vec{\tau} = (X, Y, N) \tag{13}$$

$$M_{RB} = \begin{bmatrix} m & 0 & -my_{CG} \\ 0 & m & mx_{CG} \\ -my_{CG} & mx_{CG} & Izz \end{bmatrix} \tag{14}$$

$$C_{RB} = \begin{bmatrix} 0 & 0 & -m(x_{CG}r + v) \\ 0 & 0 & -m(y_{CG}r - u) \\ m(x_{CG}r + v) & m(y_{CG}r - u) & 0 \end{bmatrix} \tag{15}$$

2) Hydrodynamic Motion

The addition of hydrodynamic terms into the equations allows the description of the rigid-body motion to include the effects of interacting with the water. These effects are described as added mass terms in both the mass and Coriolis matrices as well as a drag matrix. Eqn. *(16* is the new 3-DOF vectorized equations of motion. The additional parameters are the added mass coefficients and the drag coefficients. These coefficients are nonlinear and can vary based on velocity. This model assumes that they are constants.

$$M\dot{\vec{v}} + C(\vec{v})\vec{v} + D(\vec{v})\vec{v} = \vec{\tau} \tag{16}$$

M, the mass matrix, is an inertia tensor that includes both the rigid-body, M_{RB}, and added mass matrix, M_{AM}.

$$M = M_{RB} + M_{AM} \tag{17}$$

$$M_{AM} = \begin{bmatrix} X_{\dot{u}} & 0 & 0 \\ 0 & Y_{\dot{v}} & Y_{\dot{r}} \\ 0 & N_{\dot{v}} & N_{\dot{r}} \end{bmatrix} \tag{18}$$

C, the Coriolis matrix, is a tensor that describes the Coriolis effects in the EOM.

$$C(\vec{v}) = C_{RB} + C_{AM} \quad (19)$$

$$C_{AM} = \begin{bmatrix} 0 & 0 & Y_{\dot{v}}v + r\frac{(Y_{\dot{r}} + N_{\dot{v}})}{2} \\ 0 & 0 & -X_{\dot{u}} \\ -Y_{\dot{v}}v - r\frac{(Y_{\dot{r}} + N_{\dot{v}})}{2} & X_{\dot{u}} & 0 \end{bmatrix} \quad (20)$$

The drag matrix D approximates the hydrodynamic drag the vessel experiences. The drag model is constructed of a set of linear terms, D_L, and nonlinear terms, D_NL. This results in a second order drag model cross-coupling surge to yaw and sway to yaw. This drag model assumes that sway motion does not affect surge motion.

$$D(v) = D_L + D_{NL} \quad (21)$$

$$D_L = \begin{bmatrix} X_u & 0 & X_r \\ 0 & Y_v & Y_r \\ X_r & N_v & N_r \end{bmatrix} \quad (22)$$

$$D_{NL} = \begin{bmatrix} X_{uu}|u| & 0 & X_{ru}|u| \\ 0 & Y_{vv}|v| + Y_{vr}|r| & Y_{rr}|r| + Y_{rv}|v| \\ X_{rr}|u| & N_{vv}|v| + N_{vr}|r| & N_{rr}|r| + N_{rv}|v| \end{bmatrix} \quad (23)$$

3) Disturbances

Disturbances affect the vessels motion through water and ultimately its controllability. There are three disturbances that are of potential concern: currents, wind, and waves. The full 3-DOF equation of motion, including disturbances, is given by

$$M\dot{\vec{v}} + C(\vec{v})\vec{v} + D(\vec{v})\vec{v} = \vec{\tau} + \vec{\tau}_d \quad (24)$$

a) Current

Current can create extra drag on the vessel that either hinders its motion (against the current) or boosts it (with the current). Both situations can be detrimental to controllability. This can be accounted for in the model by considering current to be uniform in the water. This assumption means that there is a relative velocity between the NED frame and a frame aligned with the NED frame but traveling with the current. This relative velocity effects the hydrodynamic terms of Eqn. *(24*. And can be modeled as seen in Eqn. *(25-(26* .

$$\vec{v}_r = \vec{v} - \vec{v}_{current} \quad (25)$$

$$M_{RB}\dot{\vec{v}} + C_{RB}(\vec{v})\vec{v} + M_{AM}\dot{\vec{v}}_r + C_{RB}(\vec{v}_r)\vec{v}_r + D(\vec{v}_r)\vec{v}_r = \vec{\tau} + \vec{\tau}_d \quad (26)$$

b) Wind

Wind disturbances affect vessels with larger surface areas by adding a drag term. This can be estimated by calculating the relative velocity of the wind with respect the vehicle and using the resulting velocity to calculate aerodynamic drag. This drag can be added as a term in τ_d.

c) Waves

Waves were not considered as a disturbance for this model. For a 3-DOF model, waves can be accounted for using several different methods. Commonly an adaptive notch filter is used to remove the wave disturbance from the sensor signals which allows the system to operate as if waves are nonexistent.

4) Actuator Forces

The output of thrusters is often characterized as either a linear relation to the throttle percentage or a general curve based on power input to the system. These methods give a base estimates for where the vessel should be but do not take into account the flow or physical propeller properties. Several alternative methods were explored to address this weakness.

a) Linear Regression

The simplest method of motor modelling is to have a known relationship of input to output. A commonly used method is a linear regression between the input throttle and the output forces/speeds.

The force is either based on datasheets from the thruster's specifications or related to command versus force from different throttles. The relationship can be determined by surge drag tests where the throttle command is related to the force which then relates to the vehicle's speed. This assumption is valid under the thrust equals drag assumption and sets a basis for modelling under controlled, simple cases.

The issues from this model is that it does not handle the dynamics due to flow effects from the propellers. The cross-correlation between states of motion are not directly gathered accurately, but the model can approximate gross characteristics of the propulsion system.

b) Open-Water Motor Model

A commonly used model for general marine vessel is the open-water motor model. This model generates the thrust and torque of a thruster using a coefficient known as the advance ratio (J_0). This model is used for keeping a minimum speed to the water with respect to control surfaces [2] and encompasses the regions of motion where the thrust is related to its direction of travel as shown in Eqn. *(27(29*.

$$J_0 = \frac{v_a}{nd} \quad (27)$$

$$T = \rho d^4 k_T(J_0) n^2 \quad (28)$$

$$Q = \rho d^5 k_Q(J_0) n^2 \quad (29)$$

where: v_a (vessel's advance speed), n (propeller rotational speed), d (propeller diameter), k (coefficient in relation to thrust/torque), ρ (density of water), T (Thrust), and Q (Torque).

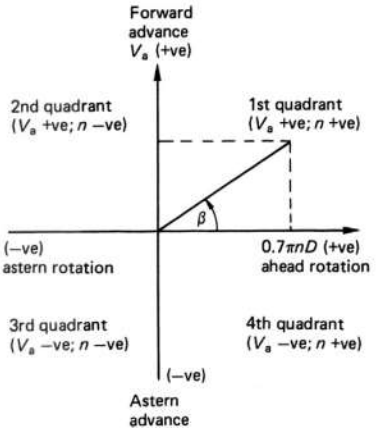

Fig 3. Four-quadrant regions of motion for a thruster setup [4]

The issue with this model is that it is not suited for operations around maneuvers that require fast turns or sudden changes of motion. To gather that sort of data, more regions (or quadrants) of operation need to be available (Figure Fig 2).

c) Four-Quadrant Motor Model

This motor model has been around since at least the 1950s to 1960s. This model takes into consideration the four regions (or quadrants) of motion for thruster or motor.

The quadrants operate similarly to a DC motor. Quadrant 1 defines forward speed (advance) versus positive torque (rotation). The motor is attempting to accelerate through this configuration. Quadrant 3 performs the same operation, but with both operations moving negatively, thus accelerating backwards while remaining in reverse. The other two regions handle braking and changes in direction. Quadrant 2 is moving at a forward speed while resisting that motion using the motor, similar to a car braking. Quadrant 4 is then reversing with the intention of moving to a stop or forward. These extra modes define what the thrusters are actually doing in cases of quick turn, station-keeping, and any other sudden motion.

The equations of motion for the four-quadrant model still defines the thruster thrust and torque based on coefficients. The added information that separates these models is the advance angle (β) value that defines which quadrant is being operated in. The two speeds considered are the advance velocity and the propeller velocity.

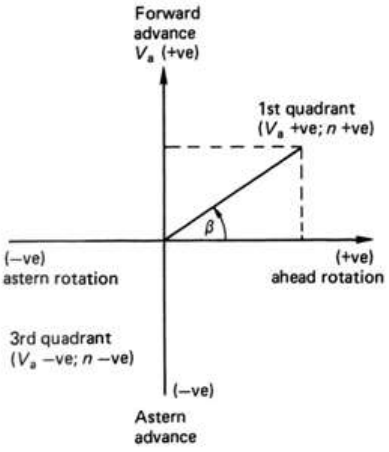

Fig 2. Open-water operation regions of motion [4].

$$T = \left(\frac{1}{2}\right)\rho c_T(\beta)\left(v_a^2 + v_p^2\right)\pi\left(\frac{D}{2}\right)^2 \quad (30)$$

$$Q = -\left(\frac{1}{2}\right)\rho c_Q(\beta)\left(v_a^2 + v_p^2\right)\pi\left(\frac{D}{2}\right)^2 D \quad (31)$$

$$\beta = atan2(v_a, v_p) \quad (32)$$

$$v_p = 0.7\pi nd \quad (33)$$

$$v_a = 0.5\left(u \pm \left(\frac{B}{2}\right)r\right) \quad (34)$$

The model was approximated to represent more idealized curves for more ideal cases in use of optimization of motor coefficient solutions. Healey [2] shifted the frames from the thrust and torque models to the lift (L) and drag (D) frame of the propeller. Since the torque is tangential to the thrust, the same can be said about the lift and drag and it transforms the equation in the following Eqns. *(35)*-*(36)*.

$$L = \left(\frac{1}{2}\right)\rho\left(v_a^2 + v_p^2\right)\left(\frac{d}{2}\right)^2\left(c_T(\beta)\cos(\beta) + \left(\frac{d}{0.7\left(\frac{d}{2}\right)}\right)c_Q(\beta)\sin(\beta)\right) \quad (35)$$

$$D = \left(\frac{1}{2}\right)\rho\left(v_a^2 + v_p^2\right)\left(\frac{d}{2}\right)^2\left(-c_T(\beta)\sin(\beta) + \left(\frac{d}{0.7\left(\frac{d}{2}\right)}\right)c_Q(\beta)\sin(\beta)\right) \quad (36)$$

The lift and drag coefficients are transformed into the angle of attack (α) frame to determine maximum coefficients (Eqns. *(37)*-*(39)*). Finding the maxima of the thrust—torque coefficient in relation to the lift-drag coefficients (Eqn. *(40)*-*(*41) defines the original coefficients in terms of the lift—drag frame.

$$c_L^H(\alpha) = c_L^{max}\sin(2\alpha) \quad (37)$$

$$c_D^H(\alpha) = \frac{c_D^{max}(1 - \cos(2\alpha))}{2} \quad (38)$$

$$\alpha = \varphi - \beta \quad (39)$$

$$c_T^H(\beta) = c_L^H\cos(\beta) - c_D^H\sin(\beta) \quad (40)$$

$$c_Q^H(\beta) = -\frac{0.7}{2}\left(c_L^H(\beta) - c_D^H\cos(\beta)\right) \quad (41)$$

where: α (angle of attack) and φ (propeller pitch angle).

This model gives simple, yet continuous sinusoidal functions to gather torque and thrust values. Given that the motor is instrumented, the torque can be solved for and ultimately the thrust can be solved for as well based on sensor readings and not on throttle commands.

This model takes in more complexity of propeller physics, but takes some assumptions as ignoring induced vorticities and exact transformation on the induced angles rather than solely

the physical propeller. The results give an answer for any maneuver based on propeller and water flow.

5) System Identification

To determine the parameters and desired motions of the USV, a standard set of maneuvers are performed. A set of marine standards [3] define the maneuvers and what physical quantities come of these tests. The tests are as follows: turning circle, zig-zag, and stopping.

The turning circle test is used to determine the vessel's turning radius at constant speed and steering angle.

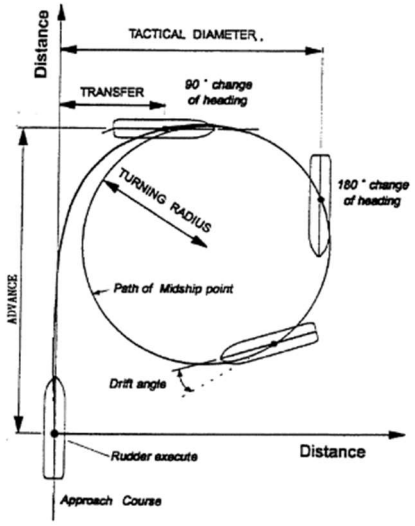

Fig 4. Turning circle test example [3].

The zigzag test is designed to test vessel course-keeping and heading overshoot. The vessel starts moving forward at a constant heading, then the rudder is diverted to an angle and remains there until the vessel crosses that angle difference. Once the vessel changes heading, the rudder is commanded to the same heading change on the opposite side of the initial heading. Repeating this maneuver catches the hydrodynamics between the surge and yaw motions.

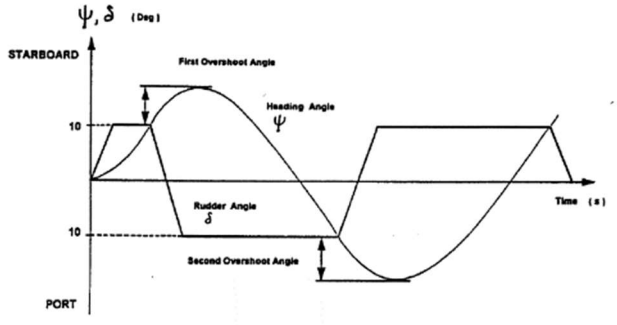

Fig 5. Zig-zag maneuvering test [3].

The stopping test is the total time it takes the vessel to stop dead in its track. The vessel moves at a constant speed forward and then suddenly commands a stopping speed to halt all motion. This is helpful with a known motor model, but similar results can be retrieved by moving at a constant forward speed and letting the vessel drift to a stop. This can give the 2nd-order drag curves for the vessel and different speeds tests can determine the added mass coefficient for surge.

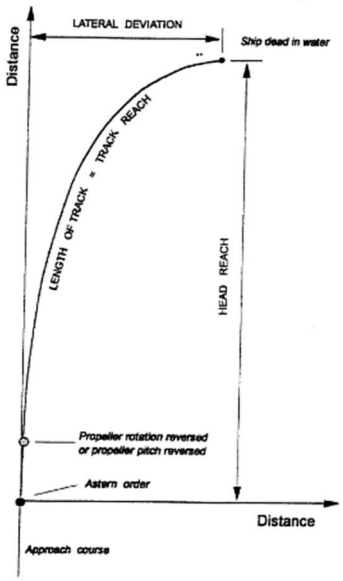

Fig 6. Stopping maneuver test example [3].

Required tests for getting good estimates on drag and motor forces are bollard-pull and measured drag tests. The bollard-pull tests tie the vessel stationary to a dock or fixed object with a force gauge in series with the holding line. The vessel begins thrusting at different commands and the force gauge returns the force exerted. Depending on levels of instrumentation can lead to more complex models. A measured drag test is letting the boat attached to a chase boat and letting the chase boat drag the vessel around. If the vessel can keep steady in the surge and sway directions, the forces read will be accurate to generate a drag model at steady speeds.

III. Path Planning

A. Architecture

The path planner's architecture (Fig. 7) spans between 3 modules: MinionTask, Path Planner, and Controls. MinionTask begins by generating a desired target for the Path Planner to follow. The target definition consists of an entire set of modes that span to more than just waypoint travelling. Current input modes are Stop, Waypoint, Path with Heading Hold, Station-keep, Circle, Dock, Heading Hold, Point Hold, and Direct.

All the input modes fit into 3 categories to describe overall autonomy control types: Path, Direct, and Stop. Path modes describe Minion driving a path with specific qualifiers and include Waypoint, Path with Heading Hold, Circle, and Dock. Direct modes describe a more direct command to Minion to perform specific tasks and include Station-keep, Heading Hold, Point-Hold and Direct. Stop mode demands that Minion stops all motor motion completely.

The target is then received by the Path Planner and triggers the algorithms to find the best path to the target that avoids obstacles. The Path Planner has five states in order to ensure the target calculation finishes. The states are Stopped, Calculate, Recalulate, Update Path, and Done.

Stop is the default state that tells Controls to stop all motion. The Calculate state attempts to find a viable path using a

constrained A* search before the user-defined timeout. If the path is found and valid, it is sent to Controls. If the Calculate fails, the Recalculate state occurs and reattempts the path given the previous path nodes and cost and attempts to change some path parameters to find the path more effectively.

After either the Calculate or Recalculate states are found to be valid, the Update Path state queues up the path to send to Controls. After the Update Path leads the planner to be in the Done state which waits for a new target or any path error reported back to the planner from Controls.

While a valid path exists, a separate thread is searching the path for any new obstacles that have intersected the path and triggers a replan accordingly. Each target mode has an inherent end mode case to distinguish the small differences in the modes. The end cases are Stop, Heading-Hold, and Station-keep and define the behavior when the end of the path is reached.

After the path nodes and modes are sent to Controls, a path error message is sent back to the Path Planner. The path error defines if Minion is following the path as intended. Minion's path-following algorithm in Path mode is defined as Minion not keeping up with the time-based path for a certain amount of time or exceeded a cross-track error threshold. If Minion is off path for a specified time, this message triggers Path Planner to Stop and trigger a replan of the last target available.

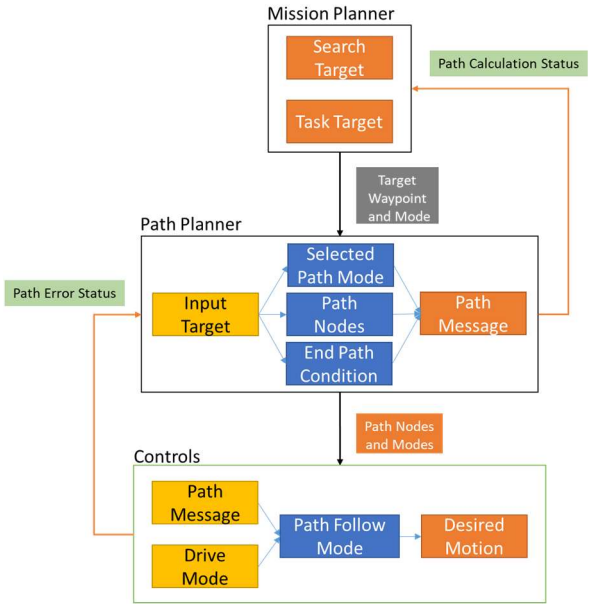

Fig 7. Control and path planning stack. Objectives are transformed and communicated between MinionTask Path Planner, and Controls.

IV. Control System

The second phase of development for Minion's controls team was the design and implementation of a new control system software stack. This involved redesigning both the Controls software module and the algorithms that are used inside the module. There were also a series of hardware modifications that modified the Minion's capabilities and methods for control.

In the 2016 configuration, Minion was a differentially driven vehicle [6]. The 2018 configuration expanded Minion's capabilities by azimuthing the thrusters (see Appendix D). This enabled control over sway motion. In addition the thrusters were upgraded and the electrical system was migrated to 48V (see Appendix K). This greatly increased the achievable control forces and moments.

The control system design for Minion consists of three levels: Trajectory, PID, and Control Allocation. The first level, trajectory control, is a leader follower type trajectory control system that operates in Minion's NED frame. It takes inputs from the Path Planning modules and outputs targets for the level 2 PID (Proportional-Integral-Derivative) system. Level 2, PID, is a parallel set of PID controllers that control surge, sway, yaw, and yaw rate and produce force and moment targets for the control allocator in Level 3. Level 3, Control Allocation, optimizes a cost function to distribute forces and moments to the various actuators. The optimizer is a nonlinear optimizer based on the sequential quadratic programming method. Level 3 outputs desired thrust level and azimuth angles.

A. Level 1: Trajectory Control

The trajectory controller has been revamped from 2016 to use a time-based leader-follower control scheme. The control algorithm utilizes the NED coordinate frame allowing for global disturbance rejection.

Paths are generated by the Path Planning module. These paths consist of a control type, a list of nodes, and an end type. There are three possible control types: Stop, Direct, Path. Stop and Direct control types cause the path controller to be skipped and the commands to be forwarded to the PID controller.

In Path mode, the nodes are a list of position, heading, and speed commands at a path time t. This allows a NED frame trajectory consisting of

$$S(t) = [N(t), E(t), \psi(t)]' \tag{42}$$

and their derivatives to be constructed

$$\dot{S}(t) = \frac{d}{dt}S \tag{43}$$

$$T_r(t) = \left[S(t), \dot{S}(t)\right] \tag{44}$$

.

This trajectory is fed into the trajectory controller along with the current path time t_c. The current path time tacks how long we have been following the current trajectory, and subsequently where we should be on the trajectory. The controller is segmented into four sections: Setpoint generator, Feedforward, Proportional-Integral, Output. A diagram of the path control system can be seen in Figure 8

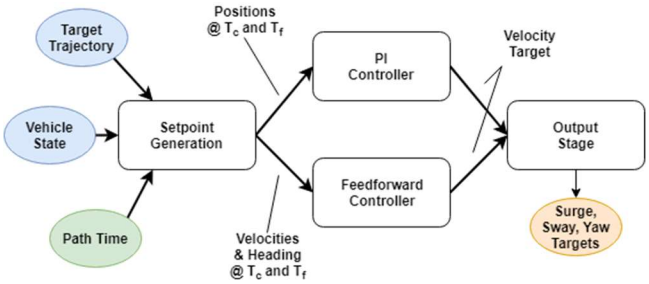

Fig 8. Depiction of the processing done by the trajectory controller. The target trajectory and vehicle state are used to produce surge, sway and yaw targets by applying a PI controller and a feedforward controller.

The setpoint generator takes in the vehicles current state, the trajectory data, and the current path time to generate a set of targets and projected states. The targets and states are projected by a lead time t_l.

$$t_f = t_c + t_l \tag{45}$$

$$Target_{@t_c} = T_r(t_c) \tag{46}$$

$$Target_{@t_f} = T_r(t_f) \tag{47}$$

The current state is projected forward by the lead time using forward Euler.

$$S(t_f) = S(t_c) + \dot{S}t_l \tag{48}$$

Other methods of projection are possible including using the dynamic model to predict the future state. The forward Euler method is suitable for this case because the time projection is usually small (1 – 2 seconds) in comparison to the timescales of vehicle motion (5 – 10 seconds). An example of these setpoints and states can be seen in Figure 9.

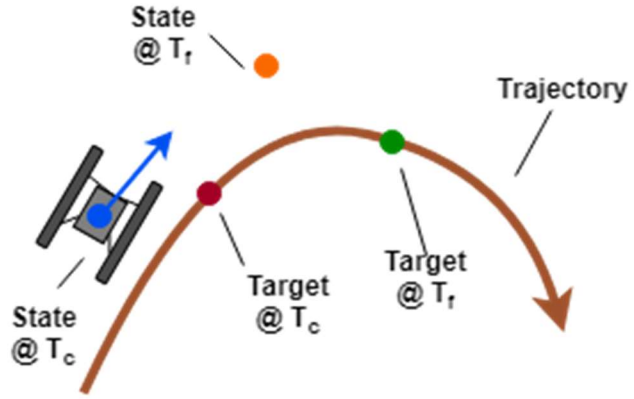

Fig 9. Current and future and targets. Future states and targets are generating by projecting forward by the lead time, t_l.

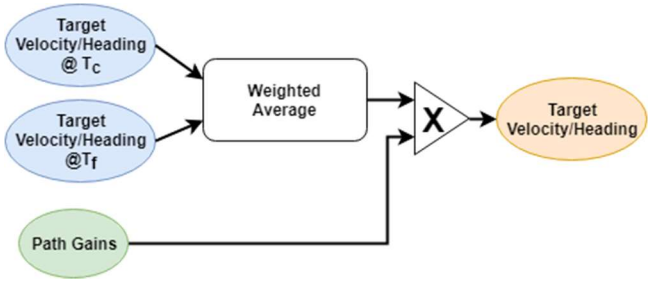

Fig 10. Diagram of feed forward system. A weighted average between the current and future targets is performed.

The feedforward section in Figure 10 performs a weighted average of the velocities and heading of the current and projected targets. The velocities and heading are fed forward in the controller as these are the targets that the PID's in Level 2 are trying to achieve. This reduces reliance on the integral term in the PI controller and produces an initial level for the output. The PI term can then modify this output appropriately, so the trajectory is tracked,

$$Target_{ff} = Target_{@t_f} + (1 - w)Target_{@t_c} \tag{49}$$

The PI section of the trajectory controller, Figure 11 implements a PI controller on the error of the current trajectory target.

$$e_{@t_c} = Target_{@t_c} - S(t_c) \tag{50}$$

$$Target_{PI} = K_i \frac{e_{@t_c} + e_{@t_c-1}}{2} \Delta t + K_2 e_{@t_c} \tag{51}$$

The projected target error is including as an additional proportional term.

$$Target_{PROJ} = K_2 \left(e_{@t_f}\right) \tag{52}$$

The resulting targets are then

$$Target_{SUM} = Target_{PI} + Target_{PROJ} + Target_{ff} \tag{53}$$

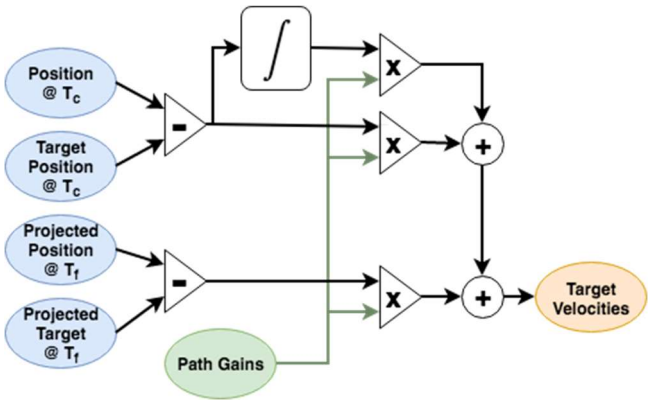

Fig 11. The PI control system for the trajectory controller. The error is calculated for both the current and projected targets. The integral is only calculated on the current target.

This form allows for the controller to look forward to future control actions and start reacting to upcoming changes. The integral action allows the controller to reject disturbances such as wind and current. When combined with the output stage this often results in a tacking motion during high winds.

The output section, Figure 12, converts the output of the controller into the desired targets for the PID level. Different conversions are necessary depending on end of path type and drive mode. There are three drive modes of concern: full (velocity), differential forward, differential reverse. The difference between modes is whether heading is coupled in the modes.

In full mode, surge, sway, and yaw can be controlled pseudo independently. In differential mode yaw is not independent. Forward and reverse reflect yaw to follow the path in the correct direction.

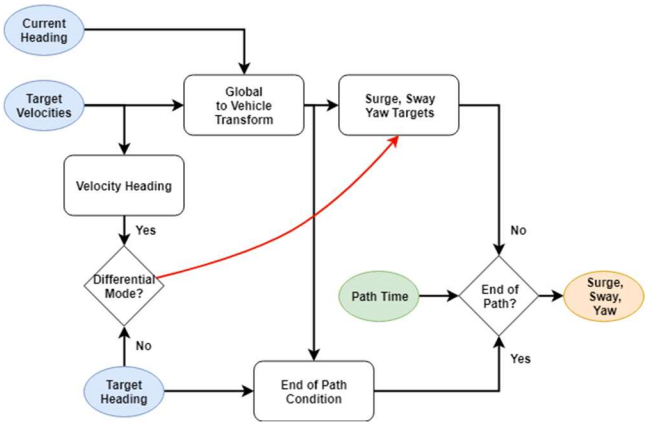

Fig 12. The output section of the trajectory controller converts the generated target velocities into the body frame. The output also detects different drive modes and the end of path condition and modifies the targets using this information.

B. Level 2: PID Control

The controllers are set in parallel for the surge, sway, and yaw rate states with the yaw controller feeding into the yaw rate setpoint. The surge and sway controllers produce the boat's forces and the yaw rate controller produces the boat's turning moment about the center of mass.

The controllers have been modified to ensure safe and proper control during the tuning phase (Fig. 13). All of the controllers are input and output rate limited to maintain the boat's acceleration rate and avoid jerky behaviors. The PID gains for each controller is gain-scheduled based on known states of operation to ensure the controllers are properly limited to handle the given states. And the controller's integrals have anti-windup when a fault occurs such as switching to RC mode or not reaching a valid solution for the particular state.

The outputs of the surge, sway, and yaw rate controllers are directly fed into the optimizer to allocate control forces to the actuators.

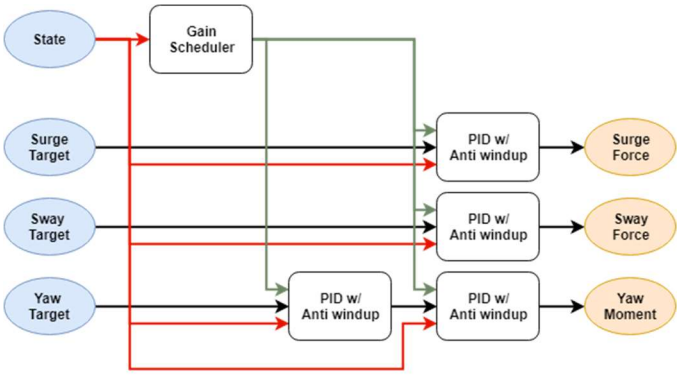

Fig 13. The level 2 PID system consists of a set of parallel PID controllers that control surge, sway, yaw, and yaw rate.

C. Level 3: Control Allocation

The optimizer takes the target forces and a moment and attempts to find a setpoint for each of the four actuators that satisfies these forces and moments. The generic nature of the optimizer allows for arbitrary constraints to be met, such as maximum or minimum thrust values, maximum or minimum azimuthing angles, or actuator failure. This allows for a well-defined actuator envelope (Fig. 14).

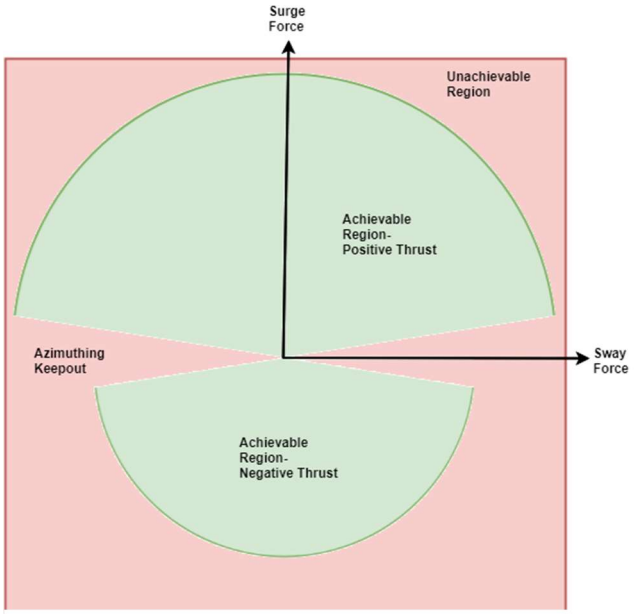

Fig 14. The optimizer produces a best fit of the control allocation while remaining constrained to the achievable region.

This allows us to have different drive modes for different situations. There are currently a set of eight configured drive modes (Table II).

Table II

LIMP MODES AVAILABLE TO MINION. THESE ARE ALTERNATE CONFIGURATION OF ACTUATORS IN THE CASE OF AN ACTUATOR FAILURE.

Limp Mode	Actuators	Forces/Moments
Full	Port Thr./Azi, Star. Thr./Azi.	Fx, Fy, Mz
Differential	Port Thr., Star. Thr.	Fx, Mz
Left Limp	Port Thr./Azi.	Fx, Mz
Right Limp	Star. Thr./Azi.	Fx, Mz
Left Crutch	Port Thr./Azi, Star Thr.	Fx, Fy, Mz
Right Crutch	Star Thr./Azi., Port Thr.	Fx, Fy, Mz
Left Twerk	Port Thr.	Mz
Right Twerk	Star. Thr.	Mz

If the optimizer fails to find an exact solution it puts out a best fit solution based on weighting. Each of the objectives is weighted (surge, sway, yaw, change in value) the objectives with higher weights are given precedence in finding a best fit solution. If a best fit solution is found instead of an exact solution, then the integrators affected by that best fit are unwound.

V. SUMMARY

Minion has a completely redesigned control system. The desired result of parameter estimation was not accomplished, but he new control system is multi-tiered and has shown robustness in actual testing. In-water testing has shown that trajectories can be followed within 0.5 – 1.5m and station-keeping is accurate to 1m.

The result of in water testing shows that the problems experienced in 2016 were overcome and that the new system is capable of being successful in 2018.

REFERENCES

[1] T. I. Fossen, Handbook of Marine Craft Hydrodynamics and Motion Control, John Wiley & Sons Ltd., 2011.

[2] A. S. J. H. Andreas J. Häusler, "Four-Quadrant Propeller Modeling: A Low-Order Harmonic Approximation," in *IFAC Conference on Control Applications in Marine Systems*, Osaka, 2013.

[3] I. M. ORGANIZATION, "EXPLANATORY NOTES TO THE STANDARDS FOR SHIP MANOEUVRABILITY," NTERNATIONAL MARITIME ORGANIZATION, 2002.

[4] J. Carlton, Marine Propellers and Propulsion, Waltham: Elsevier Ltd., 2010.

[5] J. L. Mask, "SYSTEM IDENTIFICATION METHODOLOGY FOR A WAVE ADAPTIVE MODULAR UNMANNED SURFACE VEHICLE," Florida Atlantic University, Boca Raton, 2011.

[6] C. Hockley, H. Patel and T. Zuercher, "Development of the Minion ASV for the Maritime RobotX Challenge," Embry-Riddle Aeronautical University, Daytona Beach, 2016.

[7] E. I. Sarda, "Development of a USV Station-Keeping Controller," *IEEE.*

[8] W. B. Klinger, "Adaptive Controller Design for an Autonomous Twin-Hulled Surface Vessel with Uncertain Displacement and Drag," Florida Atlantic University, Boca Raton, 2014.