UM::Autonomy's SharkByte

Cyrus Anderson, Travis Bowers, Lance Cummings, David Hendershot, Jakob Hollerbauer, Michelle Howard, Parker Howard, Evan Lee, Jeremy Lipshaw, Kunjan Sing, Ankit Singh, Daniel Snyder, Tanner Trombley, Steve Ward, Marisa Witcpalek

University of Michigan
Ann Arbor, MI 48109

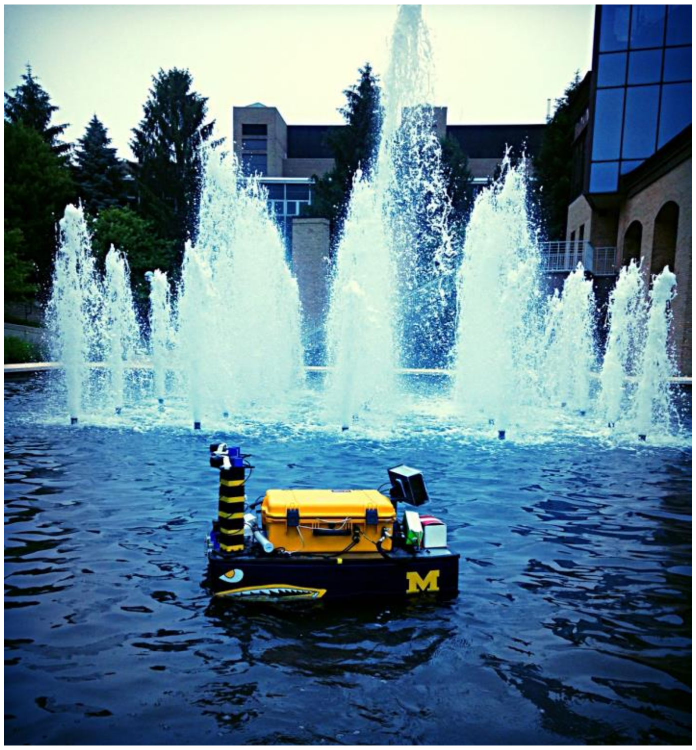

Figure 1: SharkByte in the Lurie Fountain.

ABSTRACT

SharkByte is a fully autonomous surface vehicle with custom pontoon hulls designed for maximum stability and maneuverability. This boat was designed to compete in the Ninth Annual AUVSI Foundation Autonomous Surface Vehicle Competition. As a part of this competition, SharkByte will navigate an obstacle field, dock at a designated sign, and locate an underwater pinger. Our main focus this year was increased stability in the hulls and producing a fully waterproof electrical system. This paper details the improvements made this year based on lessons learned in past competitions.

1. INTRODUCTION

SharkByte is UM::Autonomy's 2015 submission to the AUVSI Foundation Autonomous Surface Vehicle Competition. Like his predecessors SharkByte is designed to improve on lessons learned from past years and to successfully complete the challenges set forth by the competition. As it would be impractical to discuss every aspect of our design in this paper, the focus will be on improvements made over past years.

A few of the notable changes to SharkByte include: a deck that is lower to the water, a new hull shape, as well as a fully waterproof electrical box. The details of these and other improvements will be discussed in the remainder of the paper.

2. HULLS AND DECK

2.1 Hull Design

The hull design of SharkByte was focused primarily on increasing the stability of the boat in comparison to last year's vessel, Hugh Jackman. Additionally, secondary design themes included ease of outfitting and degree of maneuverability.

In regards to past UM::Autonomy boats, SharkByte represents a peak in the

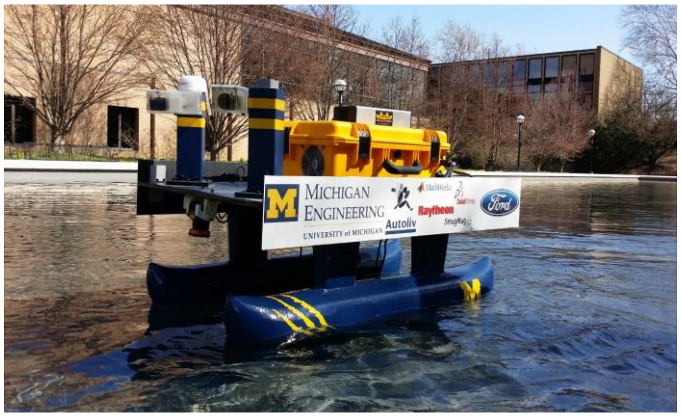

Figure 2: Hugh Jackman underway. Note the great freeboard height of 16 inches.

application of practical naval architecture principles. The overriding design trend among previous boats featured two cylindrical pontoons arranged in a catamaran style with tall struts extending upwards to the deck; this configuration placed the deck high above the water and significantly raised the center of gravity of the boat, making it unstable. Oftentimes, Hugh Jackman would enter into a standard turn and heel as much as 30 degrees from level, jeopardizing the safety of several thousands of dollars of electronics on the surface of the deck. To alleviate this issue, the sponsons of SharkByte were redesigned to be wider, shorter, and more akin to the hulls of a catamaran than the pontoons of a party barge.

The end product of this redesign was a far more stable SharkByte with a freeboard height 10 inches less than Hugh Jackman. Compromises were made; the increased waterplane area of SharkByte induces more hydrodynamic drag and constructive interference than Hugh Jackman. However, the electronic components of the boat were made safe from capsizing or accidental immersion thanks to this lowered center of gravity.

An additional benefit of the new hull design was the increased ease of outfitting (the ease with which thrusters and other devices could be attached to the hull). The large and long surfaces on the bottom of the hulls permitted aluminum U-strut to be fashioned to the keel of each hull; this developed into a rail-mounting system that made thrusters and other submerged equipment easily attachable and easily removable.

Lastly, the changes to the hull design created significant improvements in SharkByte's maneuverability. In addition to the width increase of each hull, the overall beam of the boat was increased as well. This lead to an improvement in the beam-to-length ratio of the boat and allowed the boat to turn more easily than any previous designs.

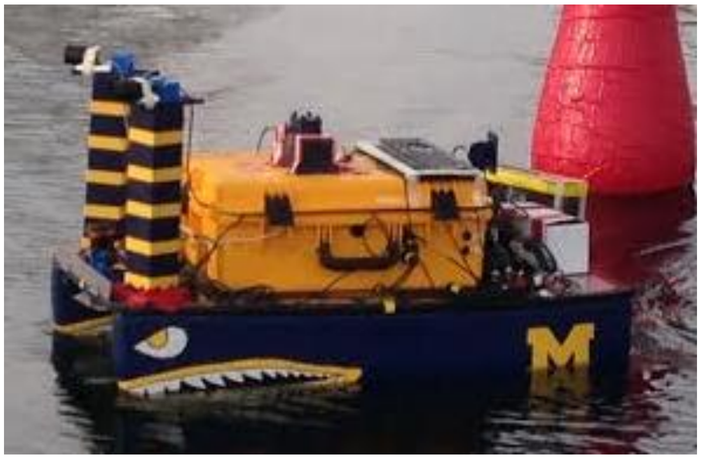

Figure 3: SharkByte underway. The freeboard height is substantially lower than Hugh Jackman at full load condition.

2.2 Hull Construction

Initially, the hulls for SharkByte were drawn in a Solidworks[N] computer-aided drafting (CAD) program. Several designs were debated upon that emphasized different features for the hulls; hard chines and soft chines, sloped bows and straight bows, and wall sides and flared sides were taken into consideration at this stage.

The final design, seen on the boat today, features wall sides, hard chines, and a gently curved bow. Our team imported this model to a CATIA[N] CAD program and generated a toolpath that could cut the shape of the hulls from closed-cell insulation foam on a CNC Routing machine. In order to accommodate the working height of the router, the model was cut into 2-inch cross sections from the keel to the deck in planes parallel to the waterline. When the routing of the foam finished, these cross-sections were glued together into a foam hull with the same molded dimensions as the original CAD model.

Next, woven 80-20 fiberglass (fiberglass with 80 percent of the fibers running in one direction and 20 percent of the fibers running in the perpendicular direction) cloth was cut into long strips running down the port side of the foam hulls from deck level, underneath the keel, and up the starboard side of the hulls back to deck level. Each strip corresponded to one of five stations extending from the bow to the stern of the boat and extra strips were cut to cover exceedingly curved spaces on the bow and stern of the foam hull. For each hull, a second layer of strips was cut in the same fashion as the first and all the strips for both hulls were soaked with epoxy resin and laid onto the foam hulls in the correct spaces. Once all the strips were laid down on both hulls, the strips were permitted to dry and harden into a solid, continuous, fiberglass hull.

At this point, the tops of each hull were still uncovered, so pre-machined aluminum plates were laid into the open foam top and sealed in with epoxy resin; these plates ultimately served as the mounting surfaces for the deck. Lastly, more fiberglass strips were laid on top of the foam hulls to form a completely enclosed hull space; these were sanded profusely to achieve a smooth surface finish and promptly painted. The foam was never removed from the inside of the fiberglass so that in the event of internal flooding (owing to damage to the hulls during operation or transit) less buoyancy would be lost and the boat could safely return to shore.

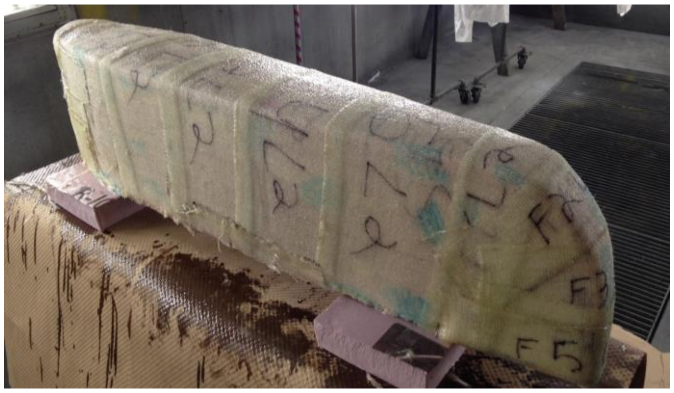
Figure 2: the port sponson during the lay-up stage of fiberglass construction. The thick ridges on the hull are evidence of the overlapping between fiberglass strips.

2.3 Sensor Mounts

2.3.1 Camera Mounts

Unlike last year's camera mounts, this design encapsulates a waterproof camera bottle. The camera bottle consists of three main parts: the top case, the bottom case, and two lenses. The top case provides half of the camera mount, two frames for the lenses' integration, and a slit for the O-ring. Moreover, the bottom case provides the second half of the camera mount, integration to the boat, and ridges to accommodate the O-ring and top case. Both the top case and the bottom case were 3D printed in the University of Michigan's 3D lab. Respectively, the front and back Acrylic lenses, fabricated by a laser cutter, allow the camera to see clearly through the case and

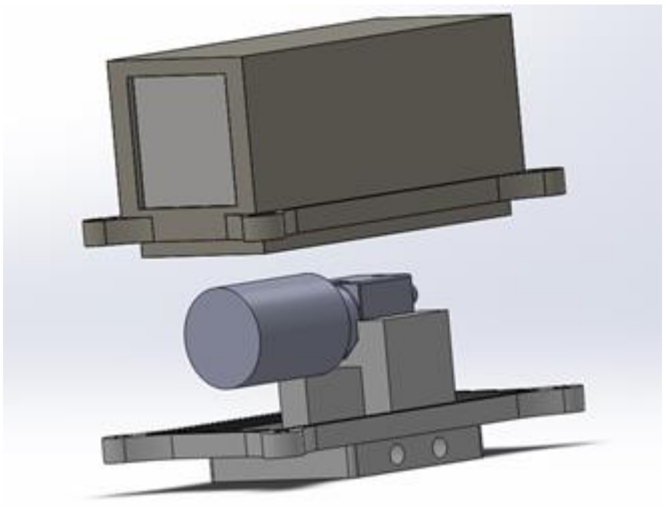

Figure 3: Solidworks Model of the Camera Bottle.

allow us to know if the camera is on via the light on the back of the camera. The front of each lens is covered in a polarized film to reduce glare in the cameras. The bottom case is attached to the boat through the sensor tower and the top case slides into the bottom case. By bolting the cases together, the O-ring seal will be complete and the camera will be successfully waterproof. (Figure 5)

On our right camera bottle, the compass box was integrated directly into the camera bottle's top case. Adding an O-ring to the box, we have retained the camera bottle's (and therefore the compass box's) ability to

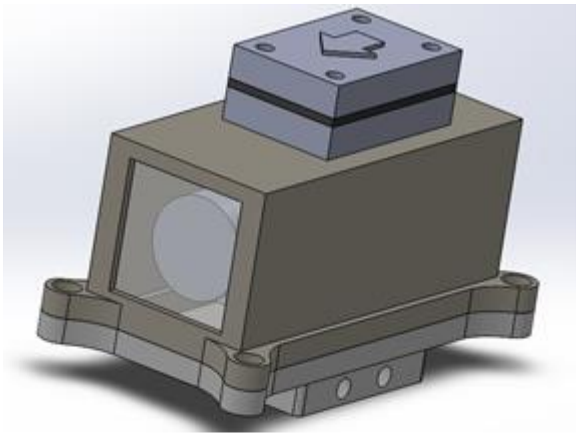

Figure 6: Solidworks Model of Camera Bottle and Compass Box.

be waterproof. (Figure 6) Similar to last year, the camera mounts are placed as far apart on the front deck and elevated as far above the surface of the water as the competition height limits allow to obtain the widest field of view without creating any blind spots.

2.3.2 LIDAR

SharkByte's LIDAR collects data from a single 2D plane. In order to create a 3D point cloud, the LIDAR is mounted on a pivot so the LIDAR can be rocked up and down. This custom rocking LIDAR mount was fabricated using a 3D printer (Figure 7). This year because SharkByte's deck is lower to the water than his predecessors the LIDAR is mounted on the top of the deck to avoid splashes.

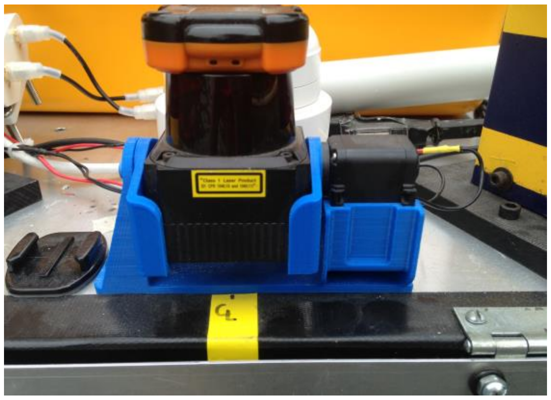

Figure 7: Custom 3D printed LIDAR mount.

2.3.3 GPS/Compass

SharkByte is equipped with a GPS and a compass. The GPS is mounted on the deck behind the LIDAR mount. The compass mount is integrated into the top of the 3D-printed camera bottle as described in earlier sections.

2.3.4 Hydrophone

We are using an Aquarian Audio Products H2a high impedance hydrophone to detect the location of the acoustic beacon. This hydrophone is normally omnidirectional but we created a special housing that allows us to use it as a directional hydrophone. It is mounted to the underside of the deck.

3. ELECTRICAL SYSTEM

3.1 Waterproof Electrical Box

Much of SharkByte's electrical system is similar to those of our past boats but there are a few notable changes to this year's setup. One main goal for our electrical system was that it be fully waterproof. To achieve this goal we had to use water-cooling to keep the motherboard cool instead of the fans used to create airflow in previous years. We used a radiator on the outside of the box to cool the water running through SharkByte's system. In addition to water-cooling the motherboard, we also added a shelf off the bottom of the box. This shelf allows us to rigidly mount our electrical components while not drilling holes in the base of the box that might allow water in.

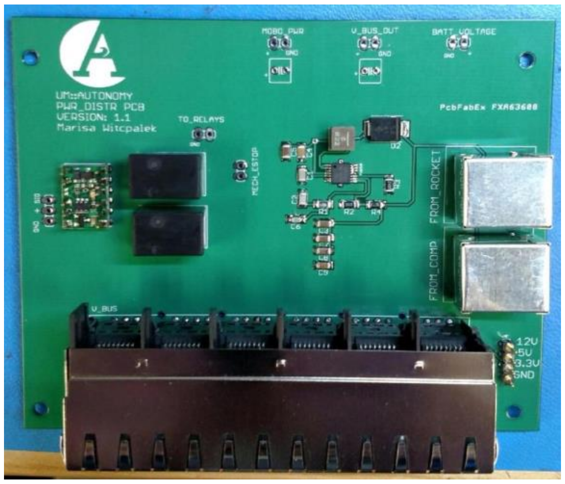

Figure 8: Custom PCB designed for power distribution and e-stop functionality.

3.2 Custom PCB

This year SharkByte has a custom printed circuit board (PCB) used for power distribution seen in Figure 8. We assigned the pins in an Ethernet hub to be 12V, 5V, 3.3V, and ground. This method has an advantage over past years because it is smaller and the connectors lock in more securely. In addition to power distribution, our PCB also contains our e-stop functionality, both remote e-stop and the e-stop attached to the boat.

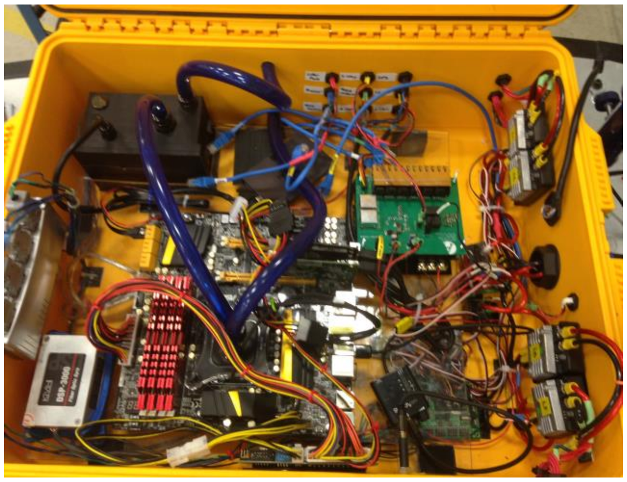

Figure 9: SharkByte's electrical box.

3.3 Screen

This year to allow easy access to the computer on the boat during testing as well as the ability to easily see SharkByte's status, we added a screen to the rear of the boat. We chose a PixelQi Screen, similar to a tablet screen, which would be easy to read outdoors and could be powered by our boat. This addition has proved invaluable during testing time, allowing us to test without carrying a monitor around to attach to the boat.

3.4 Motherboard and Sensors

As in past years SharkByte's electrical system is housed within a Pelican Storm iM2700 case (22" x 17"x 8") as seen in Figure 9. This case was chosen because it is rugged and machinable. Computationally, SharkByte has an Intel Core i7-4771 Quad Core processor mounted on an Asus ASRock motherboard and has 16 GBs of high-speed RAM. This formidable set-up allows for multiple processor intensive programs such as image processing to run at a high rate at the same time without any problems.

SharkByte is equipped with a collection of sophisticated sensors that allow him to function autonomously. He has a GPS, a fiber optic gyroscope, and a digital compass that relay relative position and spatial orientation. Two Point Grey Flea 2, 1.3MP cameras and a Hokuyo UTM-30LX LIDAR are used in conjunction to create a map of the world. By using these sensors SharkByte is able to understand and react to his surrounding environment. He uses the information from the Garmin 19x HVS GPS unit to determine location and speed. The OceanServer OS5000 USB digital compass is used to determine SharkByte's initial heading. After this initial calculation the KVH DSP-3000 fiber optic gyro is used to determine how far SharkByte has rotated from this initial heading. Each of these sensors communicates with SharkByte's computer via a serial RS 232 signal. SharkByte's cameras provide images of the surroundings so obstructions and features can be identified. Combined, the two cameras have a slightly overlapping field of vision that spans approximately 180 degrees. A Dynamixel AX-12 servo oscillates a LIDAR over a 0.2 radian arc to obtain a 3D point cloud from the planar LIDAR sensor.

Additionally SharkByte has an Aquarian Audio Products H2a high impedance hydrophone to locate the pinger.

3.5 Networking

To address the new networking challenge this year we are using Ubiquiti Rocket M5's to establish communication between the boat and our base station on shore. The Rocket on shore is attached to a laptop via Ethernet. On the boat we use a power-over-Ethernet adapter to power the Rocket from our battery. The Rockets operate in the 5470MHz-5825MHz frequency range and there is an antenna both on the boat and on the base station on shore to allow communication.

4. VISION AND PERCEPTION

4.1 Buoy Detection

SharkByte detects buoys using a late fusion approach that combines the classifications from the camera and LIDAR systems. This allows the buoy detection system to utilize both the color and pattern information in the camera images while at the same time making use of the real world locations of the objects detected with the LIDAR.

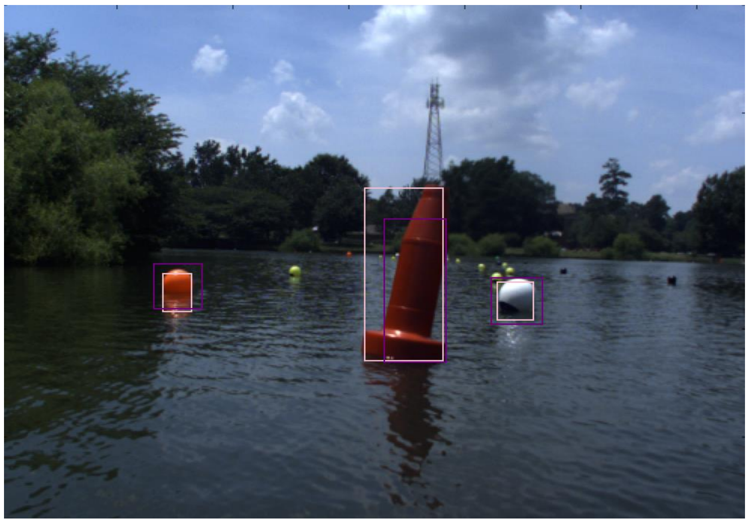

Figure 10: Successful Buoy Detections.

The camera system employs an ensemble learning method to identify regions containing buoys using a block variant of local binary patterns (LBP) [1]. The Gentle AdaBoost algorithm constructs an attentional cascade of weak learners that processes the camera images in real time [2]. Additionally the block-LBP provide a robust feature on which to train the learners by mitigating the effects of local noise.

On the LIDAR side, SharkByte uses a union-find algorithm to cluster sufficiently close points. Several heuristics are then applied to filter for the clusters representing buoys. First, clusters with too many or too few points are discarded. Next the computed radius for each cluster is checked to lie within a given interval. The remaining clusters are brought to image coordinates by applying a transform to the cluster centers.

The fusion consists of matching cluster centers to the nearest buoy found in the camera image. Pairs of camera buoys and LIDAR clusters that are sufficiently close to SharkByte are then considered buoys.

4.2 Sign Detection

Signs are detected first solely with the LIDAR. They are then matched to detected shapes (if any) in the image from the camera. Similar to the buoy detection, sufficiently close together LIDAR points are grouped together into segments using Union-Find. Then, using the x and y dimensions of those points, a line of best fit is found using Deming (orthogonal) regression. If the linear fit is good enough, the average z value (height) of the points in the segment is sufficiently high, and the largest distance between any two points (size of the sign) is sufficiently low, then the segment is considered a sign. The slope of the linear fit is also used to calculate the angle of the sign relative to SharkByte, for use in the route planning.

The location of the sign is transformed into pixel coordinates. Then, for each shape detection, the closest unmatched sign is found. If they are sufficiently close, we consider the sign to have that shape. The shape detections that don’t match up are not considered.

SharkByte accomplishes shape detection using the OpenCV library. To detect shapes, we use OpenCV image processing filters such as blur and noise reduction (bilateral filter) to improve the edge detection from the canny filter. Once the canny filter is applied, we use the OpenCV contour detection algorithm to detect contours from the edges. The contour, which is basically a set of points where the shape is located, is used with the approxPolyDP function to get the corners to form a polygon. Those corners are then used to form the edges of the shape, which is used to determine if the contour fits the parameters of a triangle, cross, or circle. The contour images are then analyzed to determine which color group each symbol matches.

4.3 Gate Detection

To detect the gates, we consider all of the buoys that SharkByte has seen so far in his run. First, all of the buoys whose radii are too small are discarded. Then, every subset consisting of one red buoy, two white buoys, and one green buoy is considered. Our goal is to find the subset(s) we are most confident that are red, white, white, green buoy gates. We represent this confidence as a real number between 0 and 1, with 1 being very confident, and 0 being not at all confident.

For each subset, we first find the maximum distance between any pair of buoys. If the maximum distance is too large, the subset is discarded. Otherwise, a line of best fit is found on the x and y coordinates of the buoys using Deming (orthogonal) regression. The confidence for this subset is set to the "R squared" value for the linear fit. Then, we check if the buoys are in the correct color order along the line. If they aren't, they are discarded. Finally, to check how even the spacing is between the buoys, we find the distances between each pair of adjacent buoys. We multiply the confidence by the minimum pair distance divided by the maximum pair distance.

If a subset makes it to the end of that without being discarded, it is considered a gate with the corresponding confidence.

5. SLAM

SharkByte employs an Extended-Kalman-Filter SLAM feature based mapping system [3]. Since there are few things to localize on in the competition pond, we use a feature-based mapper and fall back on dead-reckoning when feature detections are not available for localization purposes. Due to the inclusion of the highly accurate Fiber Optic Gyro (FOG), we use the FOG returns exclusively for heading information.

On initialization, we collect a set of compass and fog observations and compute a "globalization" constant for the FOG measurements. This allows us to use both GPS and FOG data in a global frame for the purpose of dead reckoning. For building data correspondences between buoy detections, we use the recursive Joint-Compatibility Branch and Bound algorithm. Some challenge station features are unique; in that case, there is a known correspondence and data association is trivial. We also have some capabilities for map correction. These include detection and elimination of duplicate features and detection of map corruption. By maintaining a good map of the competition environment, we are afforded many advantages that are not possible with a simple short-term mapping technique, namely, adaptive obstacle course navigation. This will be discussed at length in the following section.

6. ROUTE PLANNER

6.1 Speed Gates

SharkByte listens to the buoy detections to determine which buoys are detected as speed gates. Filtering only the speed gate buoy detections, he proceeds forward based on what was detected. If both a red and a green were detected to form a pair/gate, SharkByte heads straight between the two. If only a red or green buoy was detected, SharkByte heads slightly to the right or left of the single buoy respectively. In the worst case, if no buoys were detected, SharkByte heads straight in the direction it was deployed.

6.2 Challenge Tasks

6.2.1 Docking

To begin attempting this challenge, SharkByte first head towards the GPS location of the docking challenge. He uses LIDAR data to identify possible signs on the dock and camera data to detect the symbol and color on each sign. SharkByte then heads towards whichever of the two assigned signs he detects first. The priority settings rank the signs based on the order we want to dock in, so if any sign of higher priority is detected, the SharkByte changes directions accordingly. For each detected sign, he heads straight for a certain amount of time to ensure the dock has been reached.

6.2.3 Acoustic Beacons

The boat uses a modified breadth-first search algorithm to find the acoustic beacon. It visits each buoy and uses the hydrophone to take a reading of the sound intensity. From there the boat will navigate to the next unmeasured buoy (backtracking if necessary) and continue to take readings until all buoys are found. Once all are located, the boat picks the buoy associated with the highest intensity reading and travels in a circle around it to signal the beacon has been found. While this isn't the most time efficient solution that we worked on, it had the highest success rate during simulation runs.

6.2.4 Obstacle Field

For this task SharkByte listens to detections of buoys in the Red-White-White-Green (RWWG) pattern and from this heads toward the gate specified in the JSON message.
While avoiding obstacles, SharkByte maximizes his available range of motion by keeping near to the calculated center of the channel, and at the same time examining sets of buoys matching the RWWG pattern. The channel is exited just as it is entered.

7. UTILITIES

7.1 Bot-Procman

Due to the design, our process manager can be used both for test/debug sessions and competition runs. The process manager is invoked with a configuration file as an argument. The configuration file names all the processes that can be run. Because the configuration is not hard-coded, we can use a different configuration when testing as opposed to a trial run. In addition, the process manager sports an interactive graphical user interface for managing the processes. This allows the developer to start and stop a process or view the process's output without searching for the terminal it was started from. The process manager also attempts to ensure that processes are running and working as intended. For this, the process manager will restart a process if it crashes for whatever reason. Also, the process manager listens to the LCM channels of the managed processes. Using the frequency of publishes the process manager can determine if a process has become stuck even if it has not terminated. These features aim to improve robustness by preventing a total system failure due to a minor bug.

7.2 Vis

There are too many LCM messages at one time to quickly understand as a human. To quickly gain knowledge of what the boat is doing at a certain time, we have developed a visualization system. In this environment, we can see what is within SLAM. This includes a 3D environment, where we can pan and zoom onto locations of the boat, buoys, and other challenge station elements. This aids the debugging process, since we can quickly see what led the boat to performing unwanted behavior.

8. Conclusion

SharkByte's design has many changes compared to the designs of previous years based on lessons learned. A few of the major changes include a new hull design that is more stable and a fully waterproof electrical system. These changes combined make SharkByte the most capable UM::Autonomy submission to date.

9. REFERENCES

[2] Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). The annals of statistics, 28(2), 337-407.

[1] Liao, S., Zhu, X., Lei, Z., Zhang, L., & Li, S. Z. (2007). Learning multi-scale block local binary patterns for face recognition. In Advances in Biometrics (pp. 828-837). Springer Berlin Heidelberg.

[3] Davison, A. J., Reid, I. D., Molton, N. D., & Stasse, O. (2007). MonoSLAM: Real-time single camera SLAM. *Pattern Analysis and*

Machine Intelligence, IEEE Transactions on,
29(6), 1052-1067.