UM::Autonomy's Flying Sloth

Nathan Brown, Edward Fenwick, Thomas Huang, Benjamin Johnson, Elliot Mueller, Christopher Shu, Anthony Uytingco

***Abstract* – Flying sloth is a fully autonomous surface vehicle with a custom trimaran hull designed for maximum maneuverability and stability while maximizing deck area. This boat was designed to compete in the tenth annual AUVSI Foundation RoboBoat competition. As a part of this competition, Flying Sloth will navigate an obstacle course, identify acoustic beacons, locate a docking bay around a moving carousel, and launch an autonomous drone. Our main focuses this year were the application of new sensors and documentation of our entire code base. This paper details the efforts we made to meet these goals and each individual challenge for the 2017 competition.**

Fig. 1. UM::Autonomy's 2017 boat, Flying Sloth.

I. INTRODUCTION

Flying Sloth is UM::Autonomy's 2017 entry for the AUVSI Roboboat competition. Like our previous entries, Flying Sloth is designed to improve on lessons learned from previous designs as well as complete all challenges set forth by the competition. As it would be impractical to describe the design in full, this paper focuses on the improvements made over previous designs.

A few of the notable design changes include the transition to a trimaran design instead of a monohull or catamaran as used in previous years and the re-writing of our simultaneous localization and mapping implementation (SLAM). The details of these changes will be enumerated in the following sections.

II. HULL AND DECK

We have radically changed the design of the hull and deck for this year's competition. This section will discuss our design decisions, construction process, hull attachments, and sensor mounts for the hull and deck.

A. Hull Design

Our main goals this year were to have more room on our deck for the drone challenge and address some of the issues identified from our previous monohull design. Specifically, this included increasing stability, reducing weight, and repositioning thruster placement. As such, our design process focused on deck area and stability. Before deciding on the trimaran design, an alternate consideration was a SWATH design suggested by our advisor. However, due to time constraints and lack of experience, the team decided to proceed with the trimaran design.

The idea behind a trimaran was that it would be a good diversion from the typical catamaran design that had been attempted in previous years and it seemed to be an improvement on the monohull design used in last year's competition.

A few requirements for the design were to maximize deck space for the drone, maximize interior space for a large electrical box (which will be discussed more thoroughly in Section III, A), keep water displacement low to allow for a lower center of gravity and reduce freeboard, increase beam for greater stability while still enabling passage through doorways, and allowing fixed thruster placements between the hulls for ease of deployment and better entanglement prevention.

In order to achieve all the desired characteristics, a few compromises had to be made. For example, to fit a large electrical box within the boat the center hull had to have a wide base which increased water displacement. This meant more weight is required to reduce freeboard and keep the center of gravity low enough to guarantee high stability. The large center hull also reduced the space allowed for the thrusters between the main hull and the side hulls, such that the boat cannot be laid on a flat surface without proper supports. However, the fixed thrusters have a reduced possibility of entanglement with ropes and weeds and eliminate the hassle of attaching and detaching outriggers, as was the case with the previous monohull design.

B. Hull and Deck Construction

Initially, the hull design was ideated through hand drawn sketches and multiple iterations based on engine driven trimarans. The design was then digitally drawn using the computer-aided drafting (CAD) software, SOLIDWORKS. Various modifications were made to the hull dimensions to guarantee a large enough interior while maintaining limited outer dimensions. Once the design was finalized, three closed-cell insulation foam cores, one for each hull, were cut on a CNC routing machine. To accommodate the working height of the router, the model was cut into multiple slices, using 2" and 3" thick cross sections parallel to the waterline and eventually glued together to create the representative foam core (see Fig. 2).

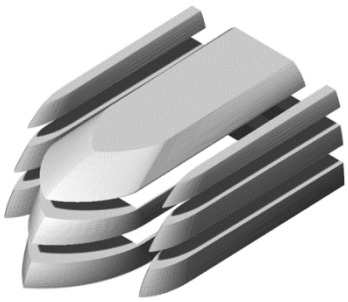

Fig. 2. The CAD model of the hull used to generate the foam core.

Next, with the foam cores upside down, stitched triaxial fiberglass, available from previous years, was used along with epoxy resin and hardener to permanently cover the hulls independently. The sides hulls' foam cores were completely enclosed in fiberglass to ensure waterproofness. Once two layers were applied, the side hulls were individually attached to the main hull by applying fiberglass over two connecting joints (see Fig. 3).

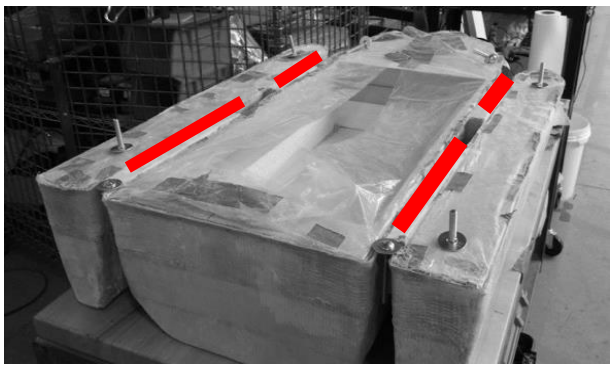

Fig. 3. Fiberglass hull connections

To reduce overall weight and keep the center of gravity as low as possible, 0.5 inch honeycomb aluminum was used for the deck. This material provided a rigid and thick enough surface to securely screw components. The deck profile was determined by cutting a paper template to shape while placing it over the hulls. Using this template, the honeycomb aluminum was cut using a reciprocating jigsaw. In order to secure the deck to the hulls, two screws were fiberglassed upside down to each side hull and holes were drilled on the deck to prevent it from sliding off. In addition, to prevent the deck from lifting off of the hulls wing nuts and washers were used to clamp the deck on the electrical box edge, which was further waterproofed using foam seal typically used in weatherproofing window gaps. Finally, to aid in transportability and to provide attachment points for the crane at competition, four stainless steel handles were attached via fiberglass symmetrically around the main hull of the boat to prevent strain on the side hull and main hull connections (see Fig. 4).

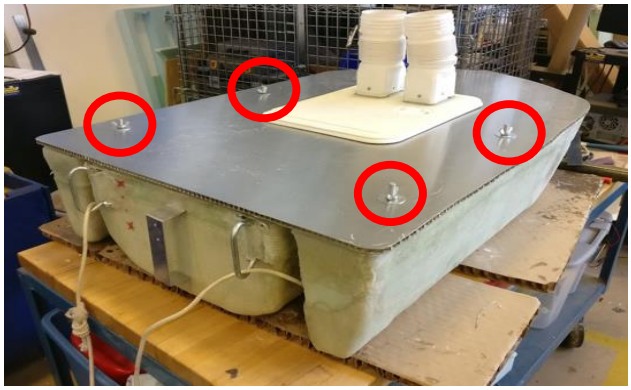

Fig. 4. Deck, deck attachments, and rear handles.

C. Sensor Mounts

This sub-section will present the types of sensors and the mounts used in this iteration of UM::Autonomy's autonomous boat.

We used the same general design for the waterproof camera mount as we did the previous year. 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 cases 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 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 to the cameras. The bottom case is attached to the boat through the sensor tower and the top case slides directly into the bottom case (see Fig. 5). By bolting the cases together, the O-ring seal will be complete and the camera will be successfully waterproof. The camera is elevated as far above the surface of the water as the competition height limits allow to obtain the widest field of view.

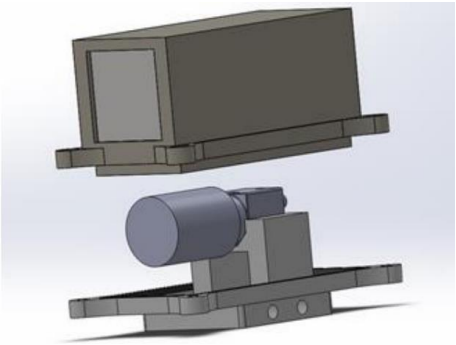

Fig. 5. A CAD model of the camera bottle used this year.

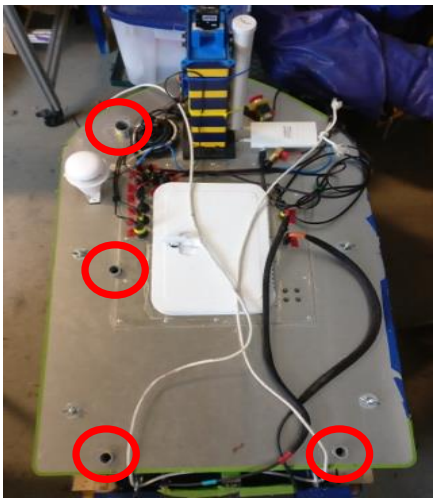

Fig. 6. Hydrophone placements.

We can use up-to four Aquarian Audio Products H2a high impedance hydrophone to detect the locations of the acoustic beacon. These hydrophones were mounted using PVC pipes through which the cables would be fed. Holes were drilled on the deck at four positions and bulkhead fittings were used to mount the pipes. Three were aligned along the port side of the boat between the main and side hull and one was aligned with the rear hydrophone, on the starboard side (see Fig. 6).

III. Electrical System

The following section will describe the changes we have made to the electrical system of Flying Sloth compared to previous boats. The primary changes we made include the implementation of several new sensors and controls necessary to pilot an autonomous drone taking off from the boat's deck.

A. Electrical System

Many of the components making up Flying Sloth's electrical system are the same or equivalent to previous boat electrical systems. Similar to the previous year, we created a custom box (28" x 15.5" x 7") out of sheet aluminum to house our electrical system and recessed it within the hull of the boat. The top of the electrical box is flush with the deck and is watertight from a seal around its top edge with the underside of the deck. This has allowed for greater deck space for drone landing and takeoff while also lowering our center of gravity.

B. Motherboard and Sensors

This year, Flying Sloth's electrical system was housed within a aluminum box designed to be recessed within the hull of the boat. It was slightly larger than previous years so as to allow greater maneuverability and access to components. Within this box is housed the computer along with several sensors and controllers necessary for the boat.

The electrical box is a closed system for which connections with sensors and other hardware is made through waterproof connectors imbedded within the deck. The two exceptions to this are the deck lid which opens allowing access to within the box and two water tubes through the back of the hull to a radiator used for water-cooling the motherboard's CPU.

Computationally, Flying Sloth has a water-cooled Intel Core i7 processor mounted on an ASUS Z97 Extereme6 motherboard with 8 GBs of high-speed RAM. This powerful 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.

Flying Sloth is equipped with a collection of sophisticated sensors that allow her to function autonomously. Specifically, she uses: a Hokuyo UTM-30LX Lidar, a Point Grey Firefly 1.3PM USB camera, a Garmin 19x HVS GPS, a KVH DSP-3000 fiber optic gyroscope (FOG), Aquarian Audio Products H2a high impedance hydrophones, a PNI Prime 3-Axis digital compass module, and a Sparton AHRS-8 inertial measurement unit (IMU).

Flying Sloth uses a GPS, FOG, and compass in conjunction to identify her relative position and orientation in space. Additionally, she uses a single camera and LIDAR concurrently to allow her to visually and spatially perceive her surroundings. By using each of these sensors, Flying Sloth is able to understand and react to her surrounding environment effectively and efficiently.

She uses the information from the GPS to determine her location and speed with data from the compass to accurately orient the data with regards to magnetic north. In addition, the FOG allows her to detect instantaneous change in her orientation on the horizontal plane. Each of these sensors communicate with Flying Sloth's computer via a RS-232 serial communication protocol.

The AHRS-8 IMU provides additional velocity and acceleration data for error calculation with regards to previous sensors. It communicates via USB to Flying Sloth's computer.

The LIDAR is rotated over a 0.2 radian arc by a Dynamixel AX-12 servo to obtain a 3D point cloud from the planar LIDAR sensor. It and the fixed camera provide the necessary visual data for buoy detection and mapping over USB communication.

One or two hydrophones are used to identify the location of an underwater pinger. They communicate with the system's computer over 3.5 mm audio cables through corresponding sound cards.

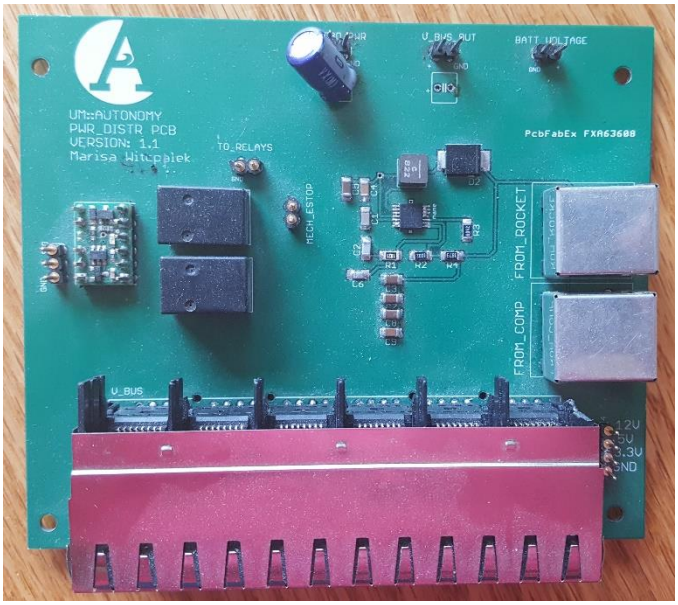

Fig. 7. A preliminary model of our custom PCB

C. Custom PCB

We continued to use this year a custom pcb that had been designed in previous years. This pcb acts as our main voltage bus, controls the boats two estop capabilities, and offers some protection circuitry for our battery. The greatest changes were the addition of larger lines to accomadate greater currents and the addition of several capacitors to attenuate sudden changes to the input voltage.

The battery protection circuitry used is designed to prevent the boat from draining its 22.2 V lithium polymer 6S battery too low. This consists of a tri-state buffer and a voltage divider to provide the current battery voltage as the input to an Arduino. Once the Arduino detects that the battery has

dropped below a safe operating voltage threshold, it triggers the e-stop, causing the thrusters to lose power. Since the thrusters provide our largest power draw, by turning them off once a threshold voltage is reached, we are able to prevent the battery from draining below the 3 V per cell minimum voltage.

D. Networking

Much like previous years, we are using a pair of Ubiquiti Rocket M5's to establish communication between the boat and our base station on shore. The on-shore-Rocket is attached to a laptop via Ethernet. On the bat, we use a power-over Ethernet adapter to power and communicate with the Rocket directly. The Rockets operate in the 5470MHz – 5825 MHz frequency range and there is an antenna connected to boat rockets to allow this communication.

IV. Software Support Structure

The following section will describe and elaborate on the structure used to support Flying Sloth's autonomous operation.

A. Operating System

Our boat runs the LTS version of Ubuntu 14.04 off of a Samsung SSD 850 EVO. We choose this operating system for compatibility reasons with existing code and external libraries.

B. Lightweight Communication and Marshalling

We use Lightweight Communication and Marshalling, heareafter referred to as LCM, to perform inter-process communication on the boat. LCM is a low latency, language independent, message transfer protocol designed by Olsen, Huang, and Moore [1]. LCM allows us to define simple messages that can be published to an open channel. An example of an LCM message can be seen in figure 8.

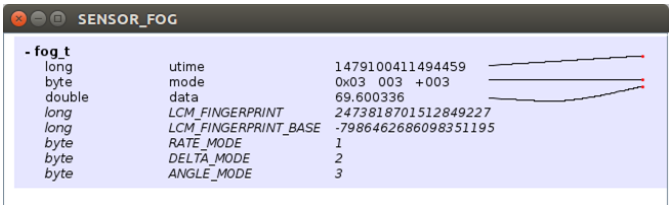

Fig. 8. An example LCM message for the FOG sensor

Each LCM message is buffered before being published to a shared memory address. Any other process on the boat can subscribe to the channel and extract information through the LCM interface in a format specific to the current language.

V. General Code Structure

Our code structure is broken into three layers: driver, detection, and decision. Each layer communicates through LCM and all of the layers and the general form of the code can be seen in figure 9.

A. Driver

The driver level is the lowest layer of boat code and the layer that communicates directly with the boat hardware. Each process communicates with a single sensor directly through a serial interface and publishes the data after some light formatting to LCM. We attempt to minimize preprocessing as much as reasonably possible in this layer both in order to allow for faster polling rate and in order to allow for modifications to the processing algorithms without affecting this layer.

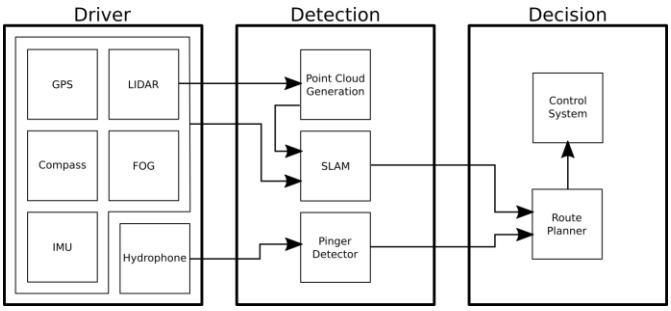

Fig. 9. The general structure of Flying Sloth's code base. Each square is a separate process that communicates via LCM. Each arrow represents an LCM channel or groups of channels.

B. Detection

The detection layer runs in tandem to and is dependent upon the driver level. The detection layer identifies objects, updates positional estimates, and runs our SLAM system. A map of the currently observed course and current state of the boat is published over LCM along with an estimate of the pinger's position.

C. Decision

The decision layer is the final layer on the boat and controls the direction and position of the boat in the real world. The decision layer runs a route planner which implements a list of tasks that the boat must perform. The tasks are abstracted and can be as simple as navigating to a GPS position and as complex as navigating an entire obstacle course. The decision layer publishes waypoints that the boat should travel to. The actual movement of the boat such as PID and motor control is accomplished in the control system module.

VI. Software Utilities

In this section, we will describe and discuss the utilities we use in testing and managing Flying Sloth's computer.

A. Simulation

Due to the winters in Michigan, it is impossible to test the boat for a large portion of the year. A large benefit of using LCM is the ability to collect and playback logs. Our logging program can be seen in figure 10.

The use of the logging program allows us to perfectly simulate the driver level of our boat and test the entire detection layer accurately. The simulation program does have some drawbacks however. It is unable to react to the outcomes of the decision layer and requires the boat to have physically been run on any course that needs to be simulated.

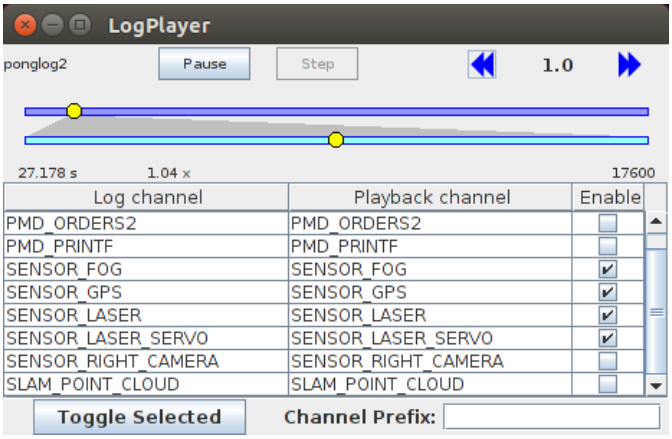

Fig. 10. Our logplayer can simulate all or a subset of LCM channels that run on the boat. The LCM messages are posted in the relative order and relative timing that they were received when the log was actually taken.

B. 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 process 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, affording us great flexibility during both runtime and testing sessions. 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. This aids us in debugging any problems Flying Sloth may encounter because we can see the sensor data as well as any LCM messages being sent.

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 any 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.

VII. Hydrophone Implementation

Flying Sloth has been modified to support the use of four hydrophones. In our implementation however, she will only make use of one or two when being used in conjunction with the docking challenge. This section will discuss Flying Sloth's method for determining the pinger's location.

If only one hydrophone is connected, Flying Sloth will approach each gate and record a ping. Then, she will dock at whichever gate has the most prominent ping in the frequency range of 25 KHz to 40 KHz as determined by using a high pass filter and Fast Fourier Transform (FFT).

If two hydrophones are connected, then Flying Sloth will record a ping while in-between two gates. Using this recording, a cross-correlation is performed to determine the time difference of arrival (TDOA) between the two hydrophones [2]. Based on the positioning of the hydrophones, the TDOA will tell us if the ping lies to port or starboard. Using this determination, Flying Sloth will either dock at the single dock to a side or perform a second recording between the remaining two gates.

VIII. SLAM Implementation

Flying Sloth has a redesigned simultaneous localization and mapping (SLAM) system which is vital in many of the challenges this year. While the boat had an Extended Kalman Filter based SLAM that was built and utilized in previous years, this system was poorly maintained and eventually broke down entirely. To avoid repeating this problem, a new SLAM system was designed with a focus on ease of understanding and maintainability.

The ultimate goal was for the new SLAM system to be extensible and easy to pass onto new team members so that it would be usable even after the original programmers graduated or left the team. To achieve this goal, the new design was based on a fairly simple occupancy based grid map. While this design is less state of the art, it is much easier to explain to new team members that lack the theoretical background for a more complex system.

A. Localization

The first aspect of Flying Sloth's SLAM is Monte Carlo localization. In each iteration of the system, both Generational and Non-Generational particles are created. Generational particles are created by taking particles from the previous iteration and predicting their location forward using velocity data from the boat's IMU and GPS. Non-Generational particles are simply generated in a Gaussian distribution around the boat's most recent pose estimate.

Each particle is then assigned an angle using a Gaussian distribution around the FOG angle. Each of the generated particles represents one possible pose of the boat, which is made up of an X position, a Y position, and a current angle relative to north.

Once the particles have been generated, they are put through a weighting process. By comparing to the boat's current angle and position (given by the GPS and the FOG), the system assigns a probability that a given particle is the true pose of the boat to all generated particles. A similar weighting process is run using the boat's LIDAR. By comparing the most recent map to the current point cloud created by the LIDAR, the boat weighs each particle. A high weight in this case means that the current point cloud agrees closely with the map that has been generated in previous iterations of the system. Finally, these weights are combined and the particle

with the greatest weight is selected as the current pose of the boat.

B. Mapping

The second aspect of SLAM is mapping. The map is represented as a grid of values from 0 to 255, with 0 being definitely empty and 255 being definitely full. After the boat uses localization to find its pose, it rotates the current point cloud from local coordinates (relative to the boat) into the coordinates of the SLAM map.

Flying Sloth's LIDAR pans up and down to get a 3D view, but the SLAM map is 2D. To simplify this view down to a 2D map, the system records a hit at a certain position if there was a hit there at any point in the LIDAR pan. This ensures that no position that contains an object at any height will be seen as empty. For every LIDAR hit, the value of the hit position gets incremented. Additionally, the value of all points between the hit and the boat get decremented since there could not be an object there if the LIDAR passed through its location. If no hit was recorded at a certain angle from the boat's front, then all squares from the boat to the minimum distance at this angle are decremented.

Over many iterations this system will converge on a map that the boat can utilize with reasonable certainty. It is important to note that any area not covered by the LIDAR minimum range is not modified in any way, ensuring that the map is not changed unless the boat has new information about the area in question.

C. Evaluation

The new SLAM system was reasonably successful in its goals of ease of understanding and ability to maintain. The original developer of our new SLAM system graduated in the Fall semester, requiring the team to transition the project to a new manager immediately. In the following semester, the new manger has successfully taken over the code base, added compass and IMU support, and improved the documentation for future management. The code that depended on the prior SLAM was fairly easy to adapt to the new system; All vital functions are now working with the new system.

IX. Challenges

Our approaches to address the dramatic changes to each challenge as well as the new challenges will be described in the following section.

A. Autonomous Navigation

Flying Sloth will go forward from the starting position until two gates are detected. Then, Flying Sloth will head along a straight-line path perpendicular to the two gates until two more gates are detected and exit through them.

B. Speed Challenge

Flying Sloth will first identify the starting gates. Then, Flying Sloth will head along a straight-line path perpendicular to the two gates until the blue mark buoy is detected. Flying Sloth will circle that buoy counterclockwise and head back to the starting gates. Using SLAM, Flying Sloth will know approximately where these gates are. These gates will be verified using LIDAR and camera data and then exited through.

C. Automated Docking

The first bay will be located using the hydrophone method described above and Flying Sloth will dock at that bay. Each bay will be identified using camera and a buoy detection algorithm. After docking, Flying Sloth will back up and return to the staring position. Since we do not have a UAV that meets the required safety guidelines, we are unable to completely verify the location of the second bay. Thus, we will randomly select one of the bays.

D. Find the Path

Flying Sloth will first circle the course once or twice to build a comprehensive SLAM map. Then, openings wide enough for the boat will be labeled and visited one by one until the middle of the obstacles can be reached. If an opening turns out to be a dead end, Flying Sloth will back up and continue to the next opening. While going through the opening, buoys will be avoided using an obstacle avoidance algorithm that steers Flying Sloth away from buoys directly in front. Once inside, Flying Sloth will detect the can buoy in the middle, circle it, and then exit through the same opening.

E. Follow the Leader

Flying Sloth will first wait for the flag with the correct number to appear. The correct number will be identified using camera data passed through a number detection algorithm. Then, Flying Sloth will enter the carousel. She will move in a circular path and avoid the flags using LIDAR. If the flag in front gets too far away, she will go faster. If the flag in front gest too close, she will go slower. Using SLAM, Flying Sloth will know when a full rotation has been completed and will exit the carousel.

F. Return to the Dock

To return to the dock, Flying Sloth will head towards the GPS coordinates of the dock while using her obstacle avoidance algorithm to avoid any objects in the way.

X. Conclusion

Flying Sloth underwent several major changes this year to better meet and adapt to old and new challenges. A new hull design was used for greater stability and maneuverability and we adapted and rewrote our SLAM code for clearer and easier implementation. Each of these changes and others together help make Flying Sloth an effective and versatile vehicle for navigating and completing each challenge.

XI. References

[1] Huang, A. S., Olson, E., and Moore, D. C., 2010, "LCM: Lightweight Communications and Marshalling," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, pp. 4057-4062.

[2] Awaludin, I., Prihatmanto, A. S., Hidayat, E. M. I., and Machbub, C., 2015, "Hyperbola tracing algorithm based on particle filter approach within a half-quadrant space for signal source localization," *IEEE International Conference on System Engineering and Technology,* Shah Alam, pp. 17-22.