# Technical Design Paper for the 2018 RobotX Competition

Joe Lemanksi, Andrea H. Robey, Javaud Ahangari, Ntiana Sakioti, Chris J. Lovin, and Michael Nilsen

***Abstract*—This paper is a description of hardware and software design strategies implemented by Old Dominion University's Team ODUSSea for the 2018 Maritime RobotX Challenge. The goals of this paper are to clearly outline the team's approach to converting their Unmanned Surface Vehicle into an Autonomous Maritime System (AMS) intended to perform the RobotX challenge tasks and to explain the results of various testing for future teams.**

## I. Introduction

The purpose of this document is to explain the overall design of Old Dominion University's Autonomous Maritime System (AMS). Design strategy is briefly examined, followed by an in-depth discussion of hardware execution and software system implementation. Approaches for both Simulation and on water testing are also reviewed with analysis and response to testing results.

## II. Design Strategy

The ODUs-Sea team took a requirements style approach when determining the overall design of the USV system. Each of the competition tasks were utilized to determine design requirements and overall functionalities for each subsystem component. There are seven subsystems; AI, LiDAR, Guidance, Vision, Hydrophone, Payload Delivery, and Hardware. For each RobotX Challenge Task a subsystem task was created mapping to capabilities in each of the subsystems. This produces specific features for the design of clearly defined objectives for each subsystem. The appendix shows the break-down of the Challenge Tasks mapped to the Subsystem tasks, also containing a description of each.

The AI subsystem is the brain of the USV system and is comprised of nested state machines. The base layer of the state machine controls which challenge task the USV is attempting to accomplish. For each of the features a separate state machine is used to implement desired functionality. This approach enables the reuse of sub-states in the different Challenge Tasks. The drawback to using state machines for implementing desired behavior to complete competition tasks is the possibility of misconstrued environment parameters leading to incorrect iterations through the states. To mitigate these concerns intensive simulation tests under multiple situations are being conducted to identify most of the edge cases that could be encountered.

The LiDAR subsystem is the primary component relied upon for detecting environment objects. The sensor used is a Velodyne VLP-16 which is mounted towards the front of the boat. The subsystem uses the LiDAR points along with data from the GPS and INS system to create an occupancy grid. This grid is employed to identify different obstacles according to certain size criteria. This approach is simple which facilitates implementation and processing requirements but could potentially lead to misidentification of objects. In order to minimize this, great care is being taken to filter out unwanted data and keep the criteria for classification as specific as possible.

The Guidance subsystem translates AI motion commands to thrust and rotation values for each pontoon's motor and servo. The subsystem has three core modes: heading-speed, line-following, and station-keeping. Though limited, this approach satisfies USV motion desired to complete each competition task.

The Vision subsystem is the secondary source for detecting obstacles, mainly focusing on color detection and pattern recognition. The subsystem uses two GoPro Hero 6 Black cameras with two video encoders that stream video to an independent computer for processing. The approach uses the OpenCV library to find colors and patterns in the images and perform depth mapping for object detection.

The Hydrophone subsystem is designed to detect pings from an underwater Pinger. There are several hardware components that are utilized to filter and analyze the signals before they are communicated to the AI subsystem. The subsystem only reports values from the ping and is not intended to perform decisions about signal source location or desired direction. This approach allows system complexity to remain in the AI, reducing potential logic issues.

The Payload Delivery subsystem consists of a launcher device with a software interface. This subsystem was designed to propel a regulation racquet ball through a hole. The approach is as follows, the interface takes an [x, y, z] point from the perspective of the launcher which is used to direct where the system is aiming. When commanded the system would then fire. This once again allows the complexity of determining the target location to remain in the AI subsystem, where most of the processing and decisions should occur.

There are other supporting systems that provide GPS and IMU data, fusing of the obstacles form vision and LiDAR, and sending data to the control station. These will be discussed in more detail in the next section.

## III. Vehicle Design

This section outlines the hardware and software components of the AMS. The hardware design encompasses the WAM-V vessel, propulsion mechanisms, electronics, launcher and all sensors included. The software sections

involve deeper discussion of the AI System, control station, GPS IMU, guidance system, all sensor software implementation and simulation.

### *A. Hardware Design*

The Hardware Design section details purpose and installation of the main hardware components.

#### *1) WAM-V Overview*

Unmanned Surface Vehicles (USVs) have been developed to perform a variety of missions such as payload delivery, remote sensing, and surveillance in marine environments. One such vessel is the Wave Adaptive Modular Vessel (WAM-V) manufactured by Marine Advanced Research Inc. Old Dominion University's WAM-V was donated by the Office of Naval Research (ONR) to promote advancement in research of USV technology. The WAM-V is a modular vessel that utilizes springs, shocks and ball joints to allow the watercraft to adapt and conform to the surface of the water. It has two inflatable pontoons that help absorb external forces and improve transportability when deflated. Attached to the pontoons by hinges, the engine pods are designed to keep the propellers fully submerged always [1]. Combined, the suspension, pontoons and hinged engine pods allow for a stable platform, thus improving sensor data. The WAM-V structure is illustrated in Figure 1 below.

*Figure 1: Image of ODUSSea WAM-V on starboard side with view of motor shafts.*

#### *2) Propulsion System*

ODUSSea's thrust is produced by two Minn Kota trolling motors placed at the ship's stern. The craft has two main steering capabilities: differential thrust and servo control. Differential thrust utilizes the motors factory ability to go in reverse, one motor applying forward thrust and the other reverse thrust, similar to how tanks drive. This mode of movement reduces the turning radius and allows the craft to essentially turn in place. Servo control is utilized at higher speeds to make turns over a greater distance. The two Minn Kota motors are rated for 80lbs of thrust each which gives the craft an approximate top speed of 5 knots. When at these higher velocities steering with servo control proves effective but adds additional stress on the servos turning the motors. According to a document created by one of ODU's senior design teams, at 5 knots and 90 degrees of steering angle, the motors and motor shafts are submitted to 11.828 Nm of torque. To combat the added torque, Volz DH 30 servos are used. According to the Volz DA 30 technical specifications, at a rated 24V supply each servo produces 16 Nm of torque, enough to fight the drag due to water resistance [2]. DA 30 servos also have a max travel angle of ±85° = 170° total travel. To get the full range of motion ±90°, a 40 to 48 sprocket ratio is used. The motor shaft mount is aluminum and is attached to the motor pods through six nuts, bolts, rubber washers, and metal washers. The motor shaft itself is attached to the motor shaft mount with shaft collars. The chains are rated for 100lbs working strength to avoid breakage during use. To attach the drive sprockets to the servos, a Volz servo horn is used in addition to a 3D printed carbon fiber adapter meant to attach the horn to the drive sprockets hole pattern. The servos are mounted to the motor pods with adjustable mounting holes for chain tensioning. Driven sprockets have two clamping collars per, sandwiching it with four screws and tightened to the motor shaft with set screws. Current issues include chain derailing's and loss of horn and drive sprocket assemblies.

#### *3) Electronics*

The computer processing units (CPUs) of the ODUSSea WAM-V are secured to a sheet aluminum in a medium sized pelican case. Main computing is done through a VIPER board and Vision computing done through two LION boards. The pelican case is a hard plastic and waterproof case that is closed with clips. To keep the hardware cool and maintain positive case pressure, the box has an air duct system that utilized PVC piping and a blower fan. Both the inlet and exit of the ducts are PVC 90s pointed downward to avoid the entrance if water. Besides the blower and duct, there is a 120mm computer fan blowing directly over the hardware. All case hardware as well as mechanization board hardware is powered through a single 12V smart battery with a battery protection circuit. The Vision computers have a regulator to maintain proper voltage. An Ethernet injector is utilized to supply power to our dual band router and is secured with a 3D printed case. Two HDMI to Ethernet h.264 encoders are placed beside the computer fan on the cases top for convert the HDMI video to stream. There are waterproof connectors that feed any cables into holes in the case. Two motor control boards control low level input to the servos and motor controllers. An Ethernet switch below the encoders is used to manage the different Ethernet wires from various systems.

The mechanization board is placed at the front of the crafts platform. The mechanization board handles movement of all small servos as well as power for the light stack, launcher motor, and vision cameras. To protect from rain or water splash, the board is encased in a 3D printed box.

#### *4) LiDAR Mount*

To increase the field of view at close range a pivoting mount is used for the Velodyne VLP-16 LiDAR. The mount consists of a U-shaped bracket attached to a rod with bearings. The rod has a stationary gear that the servo can move the platform on. The system is setup to move the lidar from a horizontal position to a 13.6-degrees angle downward towards the front of the boat. This angle can be specified at

0.1-degree increments. The main purpose of this is to view the docking bay as the boat travels in. It is also being used to increase the visibility at farther distances by moving the LiDAR at 1-degree increments. The 16 lasers are 2 degrees apart and moving the LiDAR by 1 degree can decreases the gaps at farther ranges.

*Figure 2: LiDAR Mount bracket shown in blue. It allows the system to pivot from level to 13.6 degrees down.*

*5) Racquetball Launcher*

The racquetball launcher is designed with the intent to complete the Detect and Deliver task. This design incorporates servo motors, and aluminum brackets that are connected to achieve a ball launcher. The focus of this design is to incorporate two wheels that spin to force the racquetball out of the launcher. The design sits atop the front of the boat and is attached to a pivoting arm and rotating base. The rotating base gives the launcher rotation about its vertical axis, to achieve left and right aiming. The pivoting arm allows the launcher to tilt up or down depending on the angle and distance the ball needs to travel. This design is a simple, yet effective method of launching the blue racquetball different distances. The launcher receives power and control from the mechanization board. The AI subsystem is in control of aiming the launcher through an algorithm that will calculate the distance and height of the target. Also, the vision cameras are incorporated to give the AI subsystem guidance to where the target is. This system can recognize the different sized targets and gives a depth perspective as to how far the launcher is away.

*6) Hydrophone*

The USV is equipped with a hydrophone system that is used to convert underwater acoustic pings into digital data. The design employs three RESON TC 4013 Hydrophones, three Digilent Wi-FIRE Microcontroller Boards, and an anti-aliasing and filtering board. Together, components are connected to create a system that will allow the USV to identify entrance and exit gates. The hydrophone, Microcontroller, and anti-alias board have the following specifications that are important to the design of the hydrophone detection system:

*TABLE 1: HYDROPHONE TECHNICAL SPECIFICATIONS [3]*

| | |
|---|---|
| Frequency Range | 1 Hz to 170 kHz |
| Receiving Sensitivity | -211 dB ± 3dB re 1V/µPa |
| Nominal Capacitance | 3.4 nF |
| Operating Depth | 700 m |
| Impedance | Varied with Frequency:<br>2 kΩ @ 25kHz<br>1 kΩ @ 50 kHz |

*TABLE 2: MICROCONTROLLER TECHNICAL SPECIFICATIONS [4]*

| | |
|---|---|
| Processor | PIC32MZ processor |
| Memory | 2MB Flash – 512 kB RAM |
| Operating Speed | 200 MHz |
| I/O Available | 43 Pins |
| ADC Module | 10-Bit |
| Networking Chip | Microchip MRF24WG0MA WiFi module |

*TABLE 3: FILTERING BOARD TECHNICAL SPECIFICATIONS*

| | |
|---|---|
| Sampling Rate | 100 KHz to 200 KHz |
| Filter Knee Frequency Start | 40 KHz (-3dB) |
| Attenuation @ 60 KHz | 20 dB or more |
| Amplification | 20,30 dB (switchable) |

The hydrophones connect to the anti-aliasing board through standard Bayonet Neill-Concelman (BNC) connectors. These three hydrophones are mounted to an arm that will allow them to be submerged under water during the gate detection tasks. The filtering board has three wire outputs that connect to the Analog-to-Digital converters (ADC) of the microcontrollers. The microcontrollers are connected to the USV's on-boat WiFi that allows for communication between them and the main host CPU. The anti-aliasing, filtering, and amplification board is responsible for buffering and anti-aliasing of the captured data that feeds into the microcontroller. This connection is important, because the raw data that feeds into the boards is needs to be processed before the connection to the microcontroller to ensure safe operating voltage of 3.3V. Next, the three outputs of the anti-aliasing filter board will directly connect to each of the three microcontrollers ADCs. The microcontrollers are the capturing and detection portion of the system and will be collecting and saving data on a continuous buffer. Since the beacon ping is 4 ms in length, sampling rates of 1000-1200 KHz will take less than 1024 samples to capture the entire signal.

The design is enclosed in an aluminum box and sits atop the USV platform. This enclose houses the microcontrollers and anti-aliasing board. The three hydrophones are routed out of the box and connected to the accompanying swing arm on the bottom of the boat.

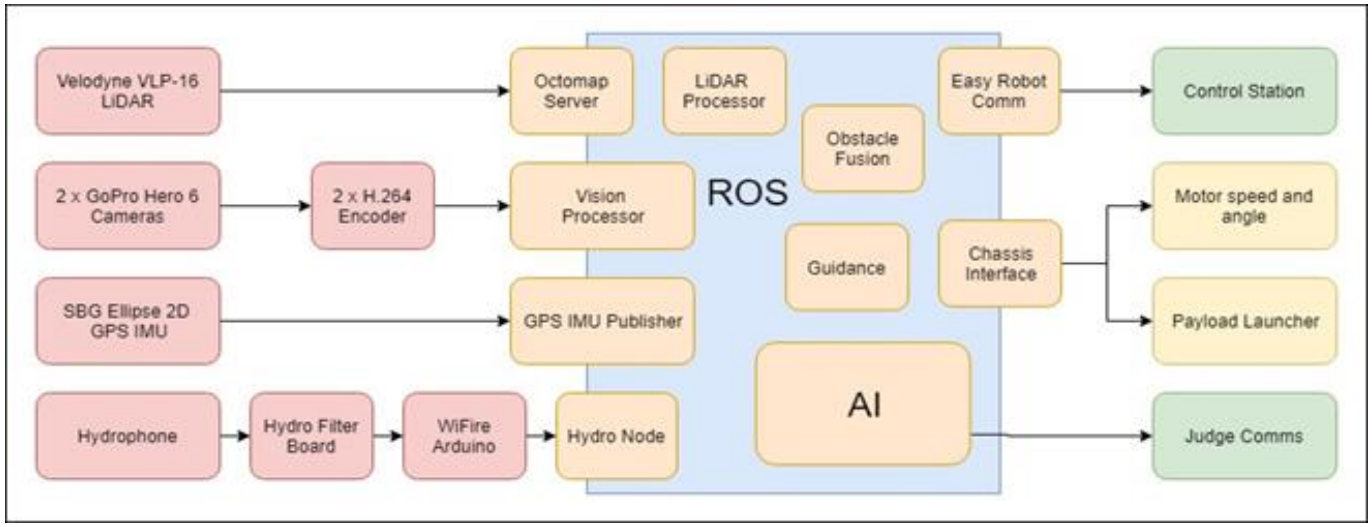


*Figure 3: System Architecture and Software Architecture Design*

*7) Camera System*

The two GoPro cameras are mounted on a single tube that goes through the front pivot point of the WAM-V top platform. The cameras are mounted 22 inches apart from lens to lens and are powered by a USB 3.0 power cable that is converted from the 12-volt power supply from the electronics case. The video is sent via HDMI cable from the cameras into h.264 encoders. The encoders use RJ45 cables to stream the video over Ethernet to the Vision Computer. The Vision computer is a Versa Logic Lion (VL-EPMe-42) mounted within the computer case.

*B. Software Design*

The base of the software design was centered on the open source Robot Operating System (ROS) Lunar pub-sub framework [5]. Using this framework provided a common architecture to facilitate communications between C++ and Python development. ROS also provides many open source modules that were used to speed up the development, testing, and capability of the overall system. Based on the design strategy of mapping Challenge Tasks to subsystem tasks a set of core applications was identified. These subsystems had specific functions within the overall framework. The general functionalities included sensors that received data (LiDAR point clouds, Video feeds, and GPS and IMU data), and processed that data into useful messages, make decisions based off task objectives, and control the vehicle to achieve the task objectives. Figure 3 shows an overview of the system architecture and the software architecture design.

*1) AI*

The AI/Strategy Planning subsystem is implemented using the Python client library for ROS, SMACH, a Python library to build hierarchical state machines, and SymPy, a Python library for symbolic mathematics. A task file that lists the desired order of competition task completion is utilized to build the top-level state machine (SM), which controls the order desired tasks are attempted. Each state of the top-level SM is in turn another SM pertaining to each competition task to be attempted. To increase modularity and allow component reuse, core functionalities utilized in numerous tasks are implemented using separate state machines, leading to some task SMs to be nested state machines.

Functionalities reused in numerous tasks, for which distinct state machines are created, include going through a gate - required for both the "Demonstrate Navigation Control" and "Entrance and Exit Gates" tasks, circling a buoy/totem - required for both "Entrance and Exit Gates" and "Identify Totems" tasks, driving to a waypoint -utilized in all tasks, as well as performing obstacle avoidance which is necessary throughout all operating areas. All task and core functionality state machines and states follow a common format, utilizing inputs from all peripheral modules - Vision, LiDAR, Hydrophone, and GPS, and outputting necessary information to the Guidance module to drive the boat through the sub-tasks.

An important benefit of utilizing SMACH to build system state machines is the graphical presentation tool smach viewer. This tool can be and is used to validate and debug state machines as it shows running state and sub-state machines with all possible transitions, as well as current active state and user data. It allows for real-time observation of what state the USV is trying to accomplish. This functionality is shown in Figure 5 below, which depicts two separate state machines, one for the "Demonstrate Navigation Control" task, and another testing the Guidance module, the current state being "TestStationKeep". These two state machines make up the top-level state machine.

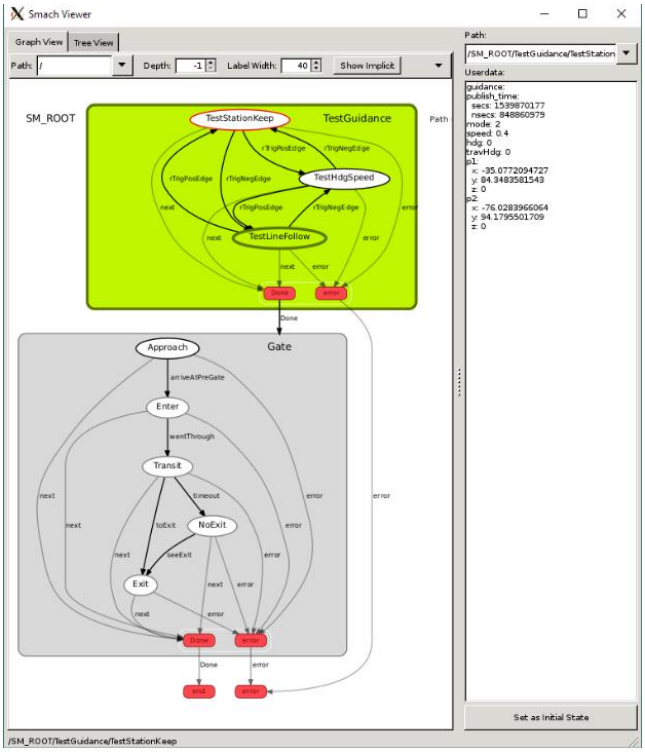


*Figure 5: Smach viewer showing two connected state machines*

*2) Control Station*

The USV control station is a QT, C++ based GUI that provides an interface of the main functions and systems of the boat. The control station was created with implementation from the ROS network to allow both to work concurrently. The main plugins of the control station allow operators to see critical information from each system of the USV in one screen. The categories include: position data, IP address, networking information, competition heartbeat connection, status modes of the boat, GPS lock display, remote control data, battery levels and temperatures, propulsion system data, and even detected objects from the AI system. The control station is used as a feedback system to the main controller of the boat.

The main window uses an RVIZ plugin that maps the boat in a 3D grid using real-time position data from the various on-board sensors. This function allows the operator to see the boat in a virtual 3D world with actual information from real-world.

The control station has a left pane with all the USVs vitals and critical information. This pane holds all the on-board system's feedback in one place.

The first section holds the vessel's current position data pulled from the GPS IMU. This section includes a display of GPS Lock, which shows if the boats on-board GPS have locked to satellites or is still searching.

The vitals pane also shows feedback of the three system batteries and their relative temperatures. This key information allows the operator to alert when battery levels are reaching low, or if any of the batteries are approaching unsafe operating conditions. The next section includes feedback from the remote control. The section shows the left and right joysticks X, Y, and Z values as mapped from the RC. Also, this section includes information on the mode the USV is in including ready local, autonomous, line follow, station keeping, etc.

The propulsion systems pane allows for display of a few of the key components of the thrusting motors. This can show steering angles, thrust, and humidity of the motor servos. The humidity information is alike the battery temperature, allowing a visual of unsafe conditions that could harm our thrusting servos.

The last section of the pane shows display visuals from the AI system. This section allows for three visual modes: boat camera, overhead camera, and scene camera, that all show the boat and virtual environment in different angles. This section also includes detected objects from vision, fusion, and lidar data. The operator can choose between the different sets of objects and then see them in the RVIZ plugin with the 3D boat. This section is critical for showing all the detected course objects that allow for different systems that use the data to achieve competition tasks.

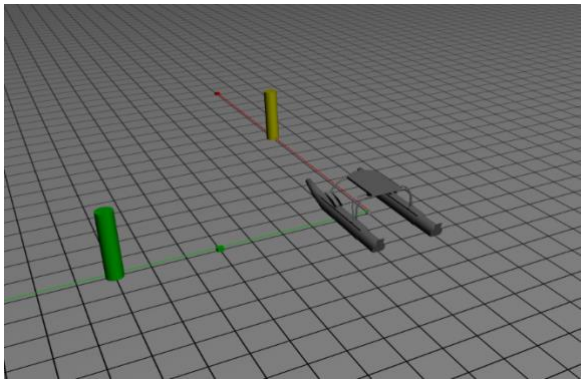

*Figure 6: USV Virtual Environment*

The control station also includes two graphing modes that show motor feedback from systems that use the AI autonomous tasks. The first graph is the velocity controller responsible for graphing the USV's actual speed, desired speed, and thrust from the left and right motors. This graph is key for showing the outputs of the autonomous mode that controls the USV's speed in all the AI tasks. The second graph is the heading controller that maps the USV's desired heading, actual heading, desired heading rate, and actual heading rate. This graph is responsible for showing the outputs of the AI controllers when put in modes that require the USV to lock to a certain heading, whether it be north, south, …, or a combination of headings.

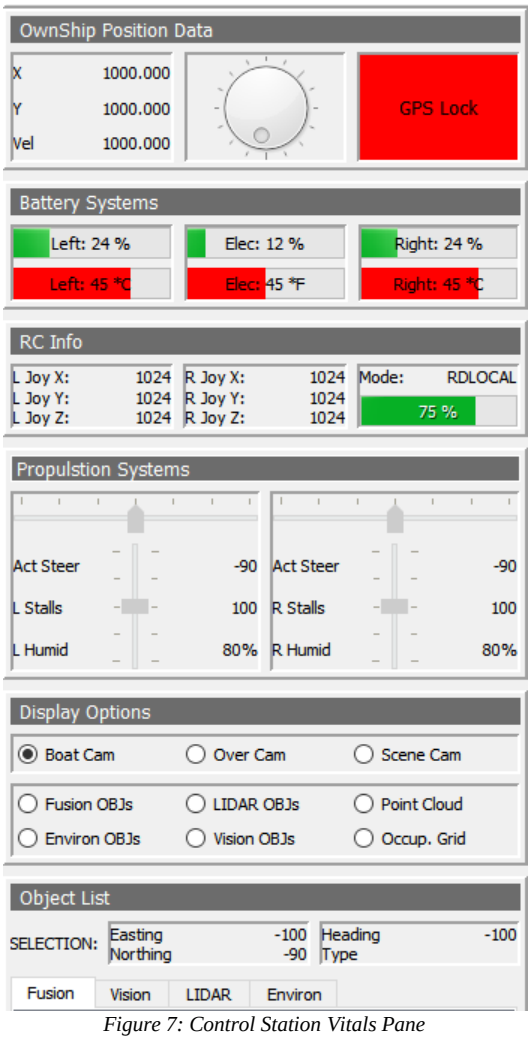


*Figure 7: Control Station Vitals Pane*

### *3) Fusion*

This subsystem receives sets of defined obstacle lists from both the LiDAR and Vision subsystems. Based on some simple correlation style rules it combines the two obstacle lists into a single list. These types of rules include only allowing color values to be specified by the Vision system. Weigh averaging the location of close obstacles letting the LiDAR's location data to take precedence over the Visions specified location.

### *4) GPS / IMU*

The SBG Ellipse2-D duel antenna GNSS inertial sensor provides the location and orientation information for the USV system. This critical piece of the software design is central and feeds almost all other subsystems to some degree with information. The SBG Ellipse2-D provides 0.1° Roll and pitch and 0.2° heading using the duel antenna GNSS system. A ROS driver for the SBG Ellipse2-D is available for the lunar ROS variant and was used to publish the data from the RS 232 serial port on the main computer [6].

A GPS IMU node was created in ROS that subscribes to the various messages from the SBG driver and publishes various messages. The main message is our own version of the vessel's position, "ownship_pose" that provides Latitude and Longitude, UTM Easting/Northing, speed over ground, course over ground, yaw and velocity of yaw, roll, pitch, yaw, magnitude velocity and x/y components, and finally some information about the SBG ellipse state that includes accuracy, number of space vehicles, and fix type. This node also provides a ROS style transform (TF2) from the World to region (which is static), and then a dynamic transform within the local region. Most of the is processing occurs in this regional space. This region is set the first time the system reaches mode 4 meaning the Kalman filter inside the SBG device is accurately calculating the position of the vehicle.

During simulation an SBG device is not present, and a simulated system is used that was built in MATLAB Simulink. This GPS SIM node interfaces with the Simulink model to publish out the same information as the operational node.

### *5) Guidance*

The Guidance of ODUSSea is a translation between two ROS Message types, High-Level and Low-Level Guidance. The High-Level Guidance message specifies which guidance mode the vessel will be operating in as well as defines parameters specific to each mode. This information is processed through a respective control system which publishes the Low-Level Guidance in the form of percent effort of the motors and motor steering angle. The High-Level Guidance modes are used to define which control system is actively publishing to the Low-Level Guidance, the first of these control systems, is the Heading-Speed Controller.

The Heading-Speed Controller, as its name suggests, allows the vessel to navigate to and maintain a specified heading at a specified velocity. This was achieved by two separate feedback control loops. The velocity is controlled by a single PID with static gains. The output of the velocity controller defines the percent effort of both port and starboard motors published to Low-Level Guidance. The Heading controller uses a cascading feedback loop applied to track, both the heading and the angular velocity of the vessel with the use of PIDs. The cascading controller uses variable gains for both heading and angular velocity PIDs these gains are defined for medium and high speed; the values are then interpolated based on the controllers desired speed. The result of the Heading Controller is published as steering angle for both the port and starboard motors. This method allows for smooth correction of the vessel's heading with little to no oscillation or over shoot [7].

The Line Following control system calculates a desired heading to guide the boat to a line defined by two waypoints. The calculation is accomplished using methods like the Vector Field Construction Algorithm described in Nelson, Barber, McLain, and Beard 2006 [8]. To summarize, the method resolves the waypoints and the position of the boat into two vectors vector 1 being from waypoint 1 to the position of the boat and vector 2 being from waypoint 1 to waypoint 2. Then to calculate if the boat is behind waypoint 1 or past waypoint 2 it solves for the dot product of the two

vectors and divides by the square of the geometric distance from waypoint 1 to 2. If this calculation is negative the boat is behind waypoint 1, if greater than one the boat has passed waypoint 2, and if between 0 and 1 the boat is in the area adjacent to the travel line. Next the algorithm must find the boat's location projected onto the line from waypoint 1 to waypoint 2. The error is found by calculating the boat's distance from the desired path which is the geometric distance from the boat's position to the projected position on the line. The side of the path the boat is travelling is determined by calculating the cross-product of vector 1 and vector 2, depending upon if the product is positive or negative, we will know that the boat is on the right or left respectively. Then using the calculated distance from the path and a threshold variable distance that is set, pending needed performance the equation (5) is employed.

$$\theta = \theta_0 - \rho\theta_\alpha \left(\frac{d}{\tau}\right)^k \qquad (1)$$

Where $\theta$ is the desired heading, $\theta_0$ is the angle of the line from waypoint 1 to waypoint 2, $\rho$ is the sign of the dot product described above, $\theta_a$ is the angle the boat follows when the distance d is larger than the threshold distance $\tau$, k is a transition gain greater than 1.

Then using this heading, the control system calls on the Heading Speed controller to provide the Low-Level Guidance.

The Station-Keeping Control system uses a table of effort and angle configurations that allow the vessel to travel in a desired direction without affecting yaw. The configurations are obtained by a brute force iterative calculation script that cycles through every possible angle and effort to find a setting that, using simple statics, provides a resulting force in the desired direction and a moment of near zero. This table is used when the vessel is a specified distance from the station, However, when the boat is within the distance, the controller then switches to heading control only, which is accomplished with differential thrust with proportional feedback control applied. A total of twelve configurations were needed to cover the entire unit circle divided into 30-degree sections. The calculations were simplified by the assumption of symmetry, as well as the knowledge that forward or reverse movement requires only forward and reverse thrust reducing the number of needed configurations to five. To allow for these configurations to control vehicle heading, proportional feedback control is applied to the effort of a single motor for each configuration, adjusting the commanded effort a small amount.

*6) Hydrophone*

The hydrophone detection system includes three microcontrollers that have on-board WiFi. Each of the three boards is connected to the USV's WiFi. This connection allows for processed data from the microcontrollers to be sent to the main host CPU and to the ROS network. The data captured and processed from the hydrophones is sent over UDP to the ROS network on a single node. This node is responsible for sending the USV to all three entrance/exit gates to determine the location of an active course beacon.

The hydrophones capture data on a continuous loop and if no ping is detected, the end of the buffer is overwritten by new incoming data. After a successful ping has occurred, data is pulled from the buffer memory and sent over UDP through the on-boat WiFi to the main host CPU. This CPU will take care of the Fast Fourier Transform and processing through the ROS network to give the USV information on the amplitude of the signal ping captured.

*7) LiDAR*

The LiDAR subsystem is broken down into three separate nodes. The first node is a ROS driver for the Velodyne VLP-16 sensor to the PCL data type publisher [9]. The main consumer of the LiDAR data is another node from ROS called the OctoMap Server [10]. This node creates an occupancy grid that is published and consumed by the LiDAR obstacle Processor. This node produces a list of defined obstacles for other subsystems to use in order to make decisions depending on the current challenge task.

The Velodyne driver node is configured to filter out any returns that are closer than 2 meters. This reduces the need to create a complex filter for angles that have blockages. Such as the computer case, tray table, and antenna mount poles.

The OctoMap ROS is configured to filter out the ground and any point that is 0.25 meters above the water level of the WAM-V. The original ROS implementation was intended to map a large area and keep the history for later use. Our application has a requirement to deal with mobile objects. This means that the original ROS node was modified to incorporate decay into the occupied cells. The OctoMap server uses a Log(odds) approach to calculate probabilities, but only stores this value in each of the cells. There is a built-in clamping function that is supposed to allow for a dynamic environment letting empty voxels be removed by "miss" hits on the previously occupied cells. In our case we get less "miss" hits. This is because our environment has less return than a typical land base system. To account for this issue a manual decay function was added that subtracted some odds value throughout each sensor scan. This value can still use some tuning, but as of now is set to -0.25 per second of occupied cells.

Once an occupancy grid is determined the LiDAR Obstacle Processing node uses both the 2D map and 3D occupied cells list to identify specific types of obstacles. The 2D map is used in an OpenCV function to find circles and return the specific radii values. This is then put through a filter function that classifies the radii into different buoy types. It also uses the 3D occupied cells list to filter for cells in those areas and look for height characteristics. This is enough information to classify the buoy type obstacles. A similar type of approach is being employed to detect the square obstacle types. Figure 8 shows an example of how the system processes the raw LiDAR returns into an occupancy grid and then classifies an obstacle.

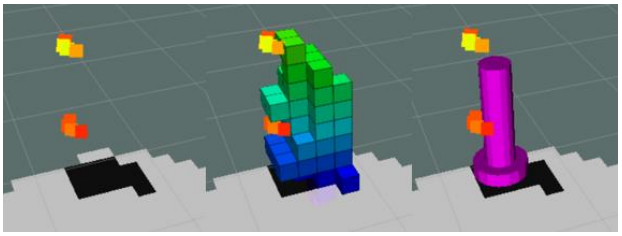

*Figure 8: [left] raw lidar points, [middle] the occupancy grid built over time from the raw points. [right] obstacle verified based on 2D map below and height of occupied cells.*

*8) Vision*

The Vision subsystem has three main functions; finding totem colors and buoy locations, identifying the light pattern for the scan the code task, and find the symbols for delivery and dock tasks. OpenCV is used to accomplish each of these functions.

To find the totem colors a color finding method is used. This is a color search that looks for a group or blob of pixels with a matching color. A simple UI with sliders is used to tune the color value to be exact for each application required by tasks. This is an indication of a buoy or totem. It then uses a matching function that tags the item to a totem color. This should account for different lighting conditions. Once a location in the frame is found for both left and right cameras a transformation function is called that performs the depth estimation calculations. This method uses known camera parameters (focal length, distance apart, and calibration values) to calculate a distance of the object from the cameras. Then, using the ROS TF function the objects are mapped into the obstacle grid space. The scan the code operation uses a function to look for changing colors in the same location over multiple observations. Once a pattern is recorded going through three different color states the system sends out a message.

To identify symbols, templates were made of the different shapes required; circle, triangle, and cruciform. These templates are used to find matching patterns in the current frames of video. The location in the image is used similar to the totems and buoys function to convert the location into the obstacle space. Known issues with the system include finding objects behind other objects and finding too many objects. To limit the chance of detecting unnecessary objects, the resolution is halved.

*9) Simulation*

The goal of the simulator was to allow for the AI and other sub systems to test without the need of having the physical boat. This required the visualization of environment objects such as the boat and totems in the simulator. The simulation framework is based in ROS and comprised of four main components: the environment simulator node, the GPS simulator node, the MATLAB Simulink physics model of the USV, and the LiDAR simulator.

The environment simulation node uses a file containing obstacles, their positions, wind data, and initial boat position information. The node in turn publishes the information to the applications which visualize the environment and objects.

The GPS simulation node was reviewed earlier, and it connects to the MATLAB Simulink physics model to publish the current USV position and transform information to act the same as the GPS IMU node. It is important to note that that roll and pitch are not currently simulated, but could be added in the future.

The MATLAB Simulink physics model uses a physics tool box to create a realistic model of the boat. It receives as input parameters that are similar to how the Low-Level-Guidance values would be sent to the motors for direction and speed. The model then produces an updated x, y, yaw value that is published to the GPS simulation node to update the position of the boat. The equations and drag calculations were taken from live testing of the WAM-V on the water to get a realistic boat movement.

The LiDAR simulator has two different modes. One mode provides a processor intensive simulation that uses GAZEBO to simulate the physical world and has a simulated Velodyne VLP 16 LiDAR that produces points similar to how the real-world device would. This is very intensive and not needed for most other subsystems. A second simulation was created that utilizes the environment simulation values directly and performs simple manipulations to ensure behavior similar to the real processors. This includes filtering for distance and changing of the type ever so often.

## IV. Experimental Testing and Results

The Experimental Testing and Results section describes the approaches to testing of subsystems both in water and in the simulated environment. Results of the tests are discussed and analyzed and any failures and responses therein are also mentioned.

*1) Simulation Testing and Results Analysis*

Most of the simulation work has been using the simple version of the LiDAR simulation node. This testing was first centered around getting the MATLAB Simulink physics model to properly move the boat. Then the Guidance subsystem went to work to test the High-Level Guidance modes of head and speed control, line following, and station keeping. Once that work was completed the AI simulations started and that has been the bulk of the simulations.

Simulation testing was consistently conducted to test the functionality of different subsystems before on-water testing was attempted. In Simulation, the Guidance subsystem's separate controllers were tested until usability was ensured, at which point on-water testing was used to perfect the response of each controller, this was useful in correcting Simulink model inaccuracies as well. The Station Keeping control system in simulation shows the ability to maintain within 5 meters of the station keeping point in sustained and changing winds of up to 200 mph – an extreme case used to test controller response.

The processor intensive LiDAR simulation (VLP 16 Sim) was heavily used to get an initial implementation of the obstacle processing algorithm. This included creating a realistic obstacle field using the environment simulation. The VLP 16 Sim used the output of the environment simulation to populate the GAZEBO world with realistically sized buoys and totems. The VLP 16 Sim would output the reflection of the simulated LiDAR off of the realistic obstacles and return a cloud of points. These points are realistic but the VLP 16 Sim was unable to exactly represent the same point cloud that would be in the real world. The VLP 16 driver node that covets the LiDAR point cloud from the LiDAR frame to the region frame relies on the output of the GPS IMU subsystem. In the VLP 16 Sim this did not include the heave of the boat or any roll or pitch. The result is a point cloud that was very stable and in the real world this cloud should move over top of the obstacles more. This insight was learned much later from the development of the VLP 16 Sim, and in the future should be updated to add this type of movement. Once the basic processing algorithm was completed, recordings of live data from on water testing was used.

During on water testing recordings of the LiDAR and GPS IMU output were conducted. These recordings played an important role in the completion of the LiDAR obstacle processing algorithm as they were played back in real time and slower to allow the fine tuning of the algorithm. These playbacks showed how the movement of the boat over the waves caused the LiDAR points to fill in the objects making it easier to classify them.

This information gathered during simulation and on water recordings for the LiDAR subsystem allowed for the fine tuning of the simple simulation as well. The simple simulation was tweaked so that obstacles wouldn't appear in the system until the boat was a certain distance away from them. This allowed for the AI subsystem to get a better simulation of what would happen in the real world.

The simple simulation has also been utilized to conduct competition task implementation testing. So far, implementation testing has only been conducted for the "Demonstrate Navigation Control" task, as well as circling a totem functionality needed for both the "Entrance and Exit Gates" as well as the "Find Totems" task. Simulation allowed for several different gate configurations as well as starting USV positions to be tested, yielding successful navigation results through both gates at all configurations tested. Testing for the functionality of circling a totem is still in progress. The gate totem configuration is utilized, with USV attempting to circle the closest totem detected. So far, functionality has not been validated, possibly due to logic issues concerning the order of points the USV is trying to navigate to circle the totem.

*2) On Water Testing and Results Analysis*

Approximately 6 on-water tests were conducted to test different subsystem functionalities as well as competition task implementation. Testing showed consistent LiDAR object detection up to 30 meters. Using temporary RC commands tailored to test the guidance subsystem independently, the heading and speed, line-following, and station-keeping control implementation was fine-tuned and validated. The USV successfully navigated at numerous set speed and heading values, as well various line-following configurations. Station-keeping control also showed satisfactory results at winds up to 11 mph. Lateral movement configurations proved to be consistent with simulated results.

Testing the LiDAR obstacle processing subsystem on water allowed for tuning of the classification algorithm. It also showed that there is still some tuning that can be done to the GPS IMU subsystem. The Z value from the GPS is mostly removed with the assumption that the heave of the boat is minimal. It was found that at times this can vary greatly and causes problems. This has been mostly filtered out by increasing the ground plain filter to 0.25 meters above the water. It was also observed that on windy days the waves start to make returns and in calm waters the LiDAR can start to pick up active fish and the anchor chain for the buoys. This on water data also provided the recorded files needed to playback and improve the classification algorithm as mentioned in the simulation section.

On-water testing was also conducted for the "Demonstrate Navigation Control" qualifying task. Numerous runs issues concerning LiDAR detection of the second gate were revealed, requiring a time parameter increase while waiting on totem detection. Course over-correction while going through the second gate was also observed necessitating intermediate course calculation. After the changes were implemented, the USV effectively navigated through the set of gates using numerous different starting positions and gate orientations.

## V. ACKNOWLEDGEMENTS

Team ODUSSea acknowledges the Old Dominion University senior design teams in the Electrical and Mechanical Engineering departments. The team also acknowledges faculty and professional advisors Dr. Yannis Papelis and Thomas Langhorne, respectively. Additionally, the team recognizes Old Dominion University MSVE, SimIS Inc., VMASC, Volz, SRC, and SIS for their generous contributions.

## VI. REFERENCES

[1] Marine Advanced Research. (2018). Marine Advanced Research - 16' WAM-V USV. Available at: http://www.wam-v.com/16-wam-v-usv

[2] "DA 30 Technical Specification." Volz Servos.

[3] Hydrophone TC4013 Miniature Reference Hydrophone. Teledyne Marine. USA. http://www.teledynemarine.com/reson-tc4013?ProductLineID=48

[4] chipKIT Wi-FIRE Reference Manual. Diligent. USA. https://reference.digilentinc.com/reference/microprocessor/wi-fire/reference-manual

[5] Quigley, M. C. (2009). *ROS: an open-source Robot*

*Operating System*. Retrieved from wiki.ros.org: http://www.willowgarage.com/papers/ros-open-source-robot-operating-system

[6] Mézo, T. L. (n.d.). *sbg_driver*. Retrieved fromwiki.ros.org: http://wiki.ros.org/sbg_driver

[7] Papelis, Y., Weate, M., 2013. Operations Architecture and Vector Field Guidance for the Riverscout Subscale Unmanned Surface Vehicle, Proceedings of the International Defense and Homeland Security Simulation Workshop 2013, Jan 13, Norfolk, VA, USA.

[8] Nelson D.R., B. D. (2016, June 14-16). Vector Field Path Following for small unmanned air vehichles.

[9] Jack O'Quin, P. B. (n.d.). *velodyne_driver*. Retrieved from http://wiki.ros.org: http://wiki.ros.org/velodyne_driver

[10] Cyrill, A. H. (2013). OctoMap: An Efficient Probabilistic {3D} Mapping Framework Based. *Autonomous Robots*.

## VII. Appendix – Challenge Task Breakdown

This appendix shows the mapping that was done from the Challenge Tasks to the subsystem requirements.

| | AI | LIDAR | Guidance | Vision | Hydro | Payload |
|---|---|---|---|---|---|---|
| **Demonstrate Navigation and Control** | 1 | 1 | 1, 2, 3 | 1, 4 | | |
| **Entrance and Exit Gates** | 2, 5 | 1 | 1, 2, 3 | 1, 2, 4 | 1 | |
| **Avoid Obstacles** | 6 | 2 | 1, 2, 3 | 5 | | |
| **Find Totems** | 3, 6 | 1 | 1, 2, 3 | 1, 4 | | |
| **Station Keeping** | | | 1 | | | |
| **Scan the Code** | 5 | 3 | 1, 2, 3 | 2 | | |
| **Identify Symbols and Dock** | 4 | 4 | 1, 2, 3 | 3 | | |
| **Detect and Deliver** | 7 | 5 | 1 | 6 | | 1 |

| Features | Description |
|---|---|
| **AI-1** | Process Demonstrate Navigation and Control task |
| **AI-2** | Process the Entrance and Exit Gates task |
| **AI-3** | Process the Find Totems task |
| **AI-4** | process the Identify Symbols and Dock task |
| **AI-5** | Process Scan the code and reporting |
| **AI-6** | Avoid obstacles (totems, buoys) |
| **AI-7** | Process detect and payload delivery task |
| **LIDAR-1** | Identify totems |
| **LIDAR-2** | Identify buoys A-3, A-5, A-7 |
| **LIDAR-3** | Identify the Scan the code obstacle |
| **LIDAR-4** | Identify the dock obstacle and the 2 dock locations |
| **LIDAR-5** | Location of the large and/or small square to give a location for the payload delivery device |
| **Guidance-1** | Boat station keeping |
| **Guidance-2** | Boat navigation by line following |
| **Guidance-3** | Boat navigation by heading speed |
| **VISION-1** | Identify colors of totems |
| **VISION-2** | Determine if the "Scan the Code" is "actively displays a light pattern" and what the pattern is. |
| **VISION-3** | Identify cruciform, a circle, or a triangle in the colors red, green, or blue at the end of each dock location. |
| **VISION-4** | Identify location of totem |
| **VISION-5** | Identify buoys A-3, A-5, and A-7 location |
| **VISION-6** | Identify the location of the large and small square for the payload delivery device |
| **Hydro-1** | detect the active underwater beacon |
| **Payload-1** | Propel a ball or object through a large or small square in the face of the dock location |