Wave Adaptive Modular Vessel (WAM-V) - Unmanned Surface Vehicle (USV) Simulator

Submitted To:

MSC Summer Research Institute
At Stevens Institute of Technology

By:

Asif Uddin and Alice Huston
Stevens Institute of Technology

Date of Submission:
25 July 2019

Written and presented with the support of the Maritime Security Center, A Department of Homeland Security Science and Technology Center of Excellence.

ABSTRACT

When intercepting a foreign hostile on open seas near coasts, humans are put into the line of danger resulting in setbacks and injuries. Humans should not needlessly endanger themselves when robotic solutions can automate these processes. In open waters, a swarm of Unmanned Surface Vessels (USV) can be deployed to intercept and deflect the trajectory of incoming vessels. In particular, the Wave Adaptive Modular Vessel (WAM-V) is a viable and scalable solution to create a security perimeter around an area. To develop the autonomy of the WAM-V, the research group has focused on mapping the Virtual RobotX simulated competition environment using the Wave Adaptive Modular Vessel (WAM-V) with a combination of LiDAR and rectified image data. These two datasets were independently processed to detect objects and then cross-referenced to determine if the predicted object locations were valid. Once verified to be the significant objects, or buoys in the context of Virtual RobotX, their locations were plotted in 3-space using a package called OctoMap and then projected into a 2D occupancy map. The WAM-V's ability to autonomously classify objects and find their position with respect to the map shows that the current mapping algorithm can be combined with pathfinding algorithms such as A* to plot and execute a series of maneuvers to traverse navigation channels. The methods used in this paper could also be abstracted to more refined object classification and probabilistic pathfinding in extreme marine conditions.

This material is based upon work supported by the U.S. Department of Homeland Security under Grant Award Number 2014-ST-061-ML0001.

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

I. INTRODUCTION

The goal of the research team was to sense and map the navigation channel task in the Marine Virtual RobotX (VRX) competition. Achieving this goal would allow the Wave Adaptive Modular Vessel (WAM-V) to develop autonomy in the context of the challenge- navigating between a green and red buoy. Fully automating the WAM-V, not just in the context of the challenge, serves different purposes. First, the WAM-V can serve as a first line of defense and intercept threats. As a first line of defense, multi-vehicle WAM-V Security systems can be deployed to intercept a foreign hostile. Although the WAM-V currently does not have any weapons capability, the WAM-V can surround unknown vessels and deflect their trajectory to avoid the incoming danger. Second, the WAM-V is a platform to inspect different environments in different domains. This unmanned surface vessel (USV) has the potential to be outfitted as a multi-domain autonomous marine system with other autonomous robots such as unmanned aerial vehicles (UAVs) and remotely operated vehicles (ROVs). Third, the WAM-V can be a valuable data acquisition tool for marine surveys such as navigation surveys, marine infrastructure, berth clearance, and dredging surveys or for mapping parts of the coast line.

The RobotX Competition is a biennial competition that gathers marine autonomy talent from all parts of the world. In this challenge, teams outfit a WAM-V with various onboard sensors and a deployable vehicle to autonomously navigate through different tasks on the waters. Some tasks include Entrance and Exit Gates, Avoid Obstacles, Find Totems, and more. (For further tasks and details on each task, please check the RobotX website[1] under the header Rules & Requirements > Tasks.) The focus is on a Virtual RobotX Competition which uses a simulation environment Virtual RobotX (VRX) designed to support the RobotX Challenge[2]. Students are asked to complete similar tasks to the RobotX Challenge. During the limited time over the summer, the research team focused on the main requirement to compete in the simulated challenge- autonomous control of a robot through a navigation channel.

To implement the research goal and our main task of navigating between the buoys in the simulated world, our group broke the main task into several subtasks. Our first task was to use camera data to locate the bearing relative to the bow of the ship and type of the buoy in the image. The second task was to use LiDAR data to create a 2D occupancy map of the relevant features in the image. The final task was to fuse the two types of sensor data to create an updated 2D occupancy map of the buoys in the image.

This report is intended to consolidate the research team's findings on how to proceed with the data gathered and tools developed during the team's eight weeks of research. This information can be used to further progress the autonomy of the WAM-V, not just in the context of the Virtual RobotX Competition, and provide a foundation for the work already established by the research team.

II. METHODOLOGY

The WAM-V research group - part of the Maritime Security Center's 2018 Summer Research Institute - focused on the first mandatory challenge in the Virtual RobotX (VRX) Competition- navigating the main task of sensing and mapping the navigation channel into three subtasks in a mapping pipeline. The VRX competition takes place in a simulated world and requires the use of the software environment ROS and visualization packages RViz and Gazebo. The three software tools are described briefly below.

A. ROS

ROS is a set of software libraries and tools to help you develop robot applications. ROS is a useful environment because it supports code "reuse" in robotics development. It is a distributed framework of processes (processes are called nodes) that redistribute the flow of information. Information is passed into nodes and out of nodes through multiple methods. One method to pass information through nodes is topics. Topics are one way data streams which nodes can subscribe ("receive") or publish ("send out"). Topics contain messages. Topics are where the nodes should look for information and messages are the information the node is looking for.[3]

B. RViz and Gazebo

In order to display the virtual environment, the group used Gazebo[4] which is a robot simulator that collaborates with ROS. In this environment, the actions of the robot at a specific time-step can be easily displayed. This can be seen in Figure 1 below. However, Gazebo does not paint a full picture of

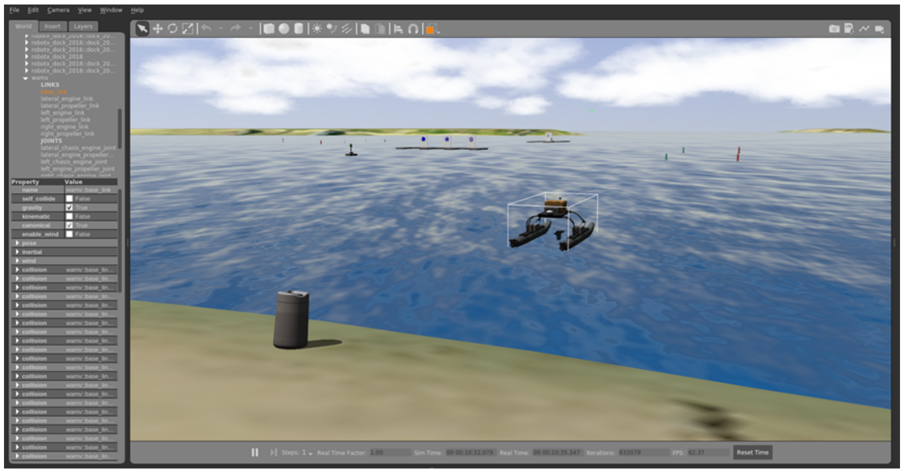

Figure 1: Gazebo World of VRX

the information at each section. Figure 1 depicts a third person perspective of the world but does not offer visualisation in other information such as sensor data. RViz is a **R**obot **Viz**ualization software that allows the group to display information about certain topics.[5] This software is useful for seeing what the robot does behind the scenes. In Figure 2, the Display panel in the top left hand corner displays the information, the image feed in the bottom left hand corner displays the camera data, and the large frame in the middle displays the lidar point cloud, the 2D Occupancy Map generated from the point cloud, and the robot

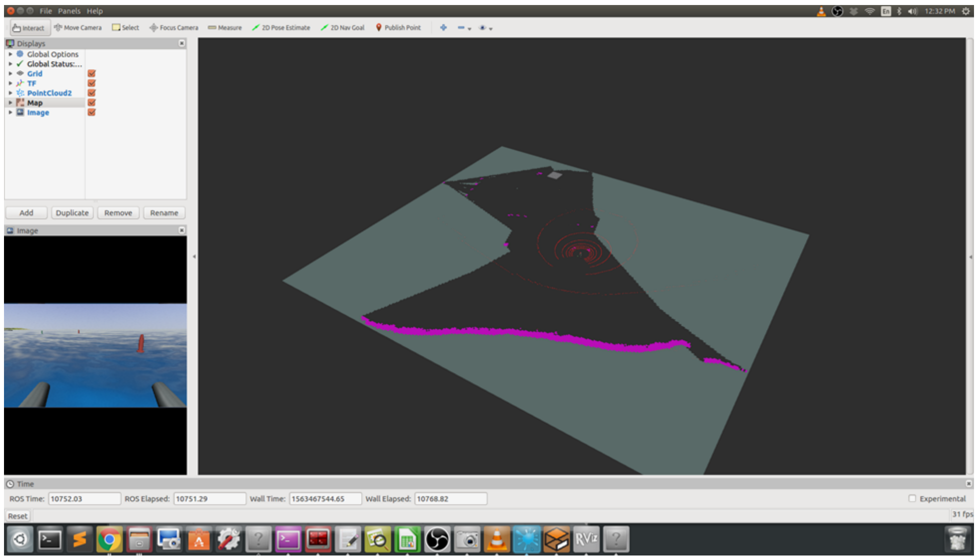

Figure 2: RViz of Lidar Data of VRX World

C. Mapping Pipeline

The mapping pipeline is the structure of our package. Figure 3 below shows the flow of data in the package and the three different subtasks.

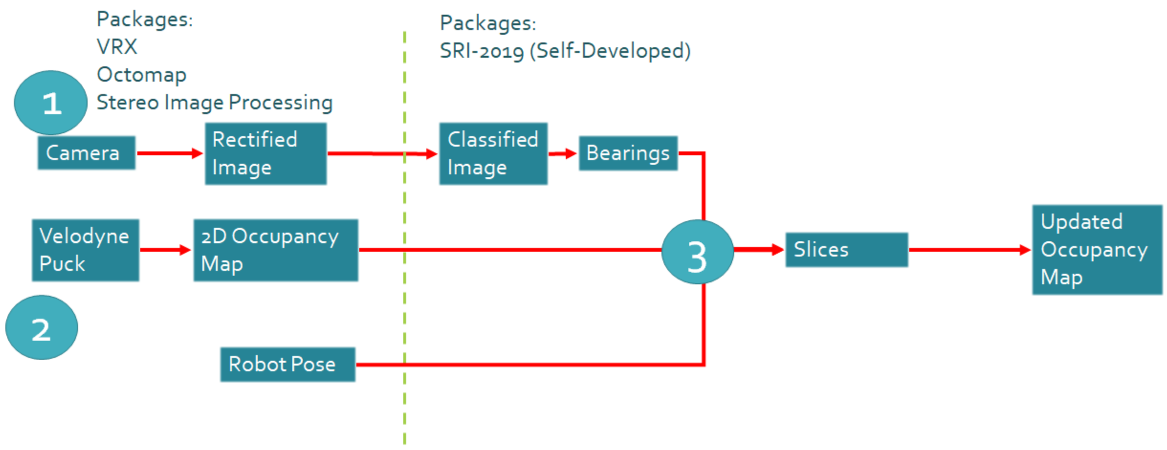

Figure 3: Mapping Pipeline

Description of the three tasks are listed below:

1. Take camera data and extract the bearing relative to the robot and type of buoy in the image
2. Take LiDAR data and create a 2D occupancy map of the relevant features in the image
3. Fuse the sensor data from camera, LiDAR, and robot pose to create an updated 2D occupancy map labeling the navigation channels

Together, the research group focused on sensing and mapping the navigation channel using a simulated WAM-V in the Virtual RobotX (VRX) environment software.

D. Classify Rectified Image

The objectives of the first subtask is to classify a rectified image. The location of the camera is displayed in the image below.

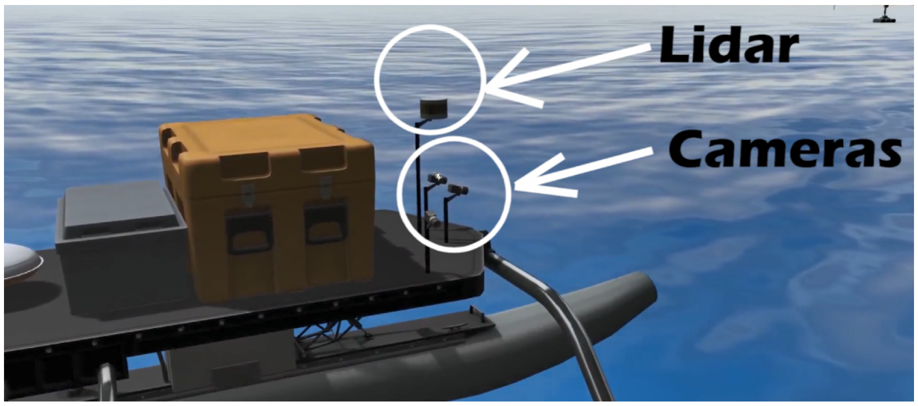

Figure 4: Location of LiDAR and Camera

A rectified image takes an image from a stereo camera and transforms an original raw image to a new image where the angle between each pixel is uniform. This means that the distance between pixels is directly proportional to the distance between objects in the image. This task can be broken into the following subtasks:

1. Take a rectified image of a front facing camera
2. Identify the buoys from the image
3. Classify the buoy as either red or green
4. Calculate the bearing of the buoy relative to the robot's orientation

E. Occupancy Map from LiDAR Data

LiDAR is a remote sensing method that uses light in the form of a pulsed laser to measure ranges.[6] The location of the LiDAR on the WAMV is depicted in Figure 4 above. This point cloud data can be used to generate distance data of objects in the environment. The objectives of the second task is to generate a 2D occupancy map of relevant features

1. Take the point cloud generated from the LiDAR
2. Filter points on the point cloud to obtain relevant features
3. Generate a 2D occupancy map.

F. Sensor Fusion of Camera, LiDAR, and Robot Pose

The objectives of the third task is to generate an updated 2D occupancy map with labels on the navigation channel. This task can be broken into the following subtasks.

1. Take the robot orientation from the transform listener
2. Generate slices of an image to check for occupancy
3. Change occupied pixels to the correct buoy value

III. RESULTS

The research group focused on sensing and mapping the navigation channel using a WAM-V in a virtual environment. The following sections are addressed to present the results of this investigation.

A. Classifying Rectified Images

This section is intended to discuss the progress that was made in classifying buoys in the space around the WAM-V. Specifically, the following subsections will discuss the use of the stereo image processing package for object classification and algorithms to classify images.

A.1 Stereo Image Processing

To detect buoys independently inside of an image, the group needed rectified images. As mentioned earlier, a rectified image is an image where the angle between each pixel is uniform. The group used Stereo Image Processing package to generate rectified images. This package is a vital part of the ROS Image Pipeline library that outputs rectified images, disparity maps, depth clouds, and compressed images among other things.[7]

A.2. Pre-Processing in HSV Colorspace

The first step of pulling out useful information from the rectified image was to implement color-filtering.

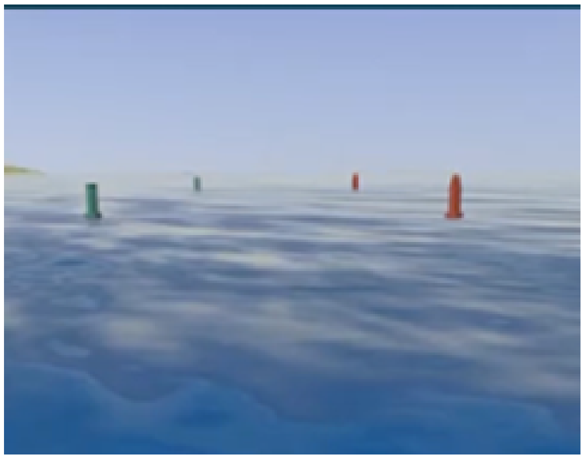

Figure 5: Unprocessed Rectified Image

From Figure 5, the group can make the assumption that the buoys can be identified within a range of color values. Considering these two statements require specificity on the group's part in the RGB colorspace, the color filtering was instead implemented in the HSV colorspace. HSV is a range of colors that means hue, saturation, value. By using the HSV colorspace, filters can be created based on a pixel's hue (perceived color), intensity, and brightness. The filter implemented for buoy detection excluded all blue pixels and any pixel below a value (brightness) of 2%. This initial filter removed the sky and most of the ocean, save for a few blue-green waves. The result is shown below in Figure 6.

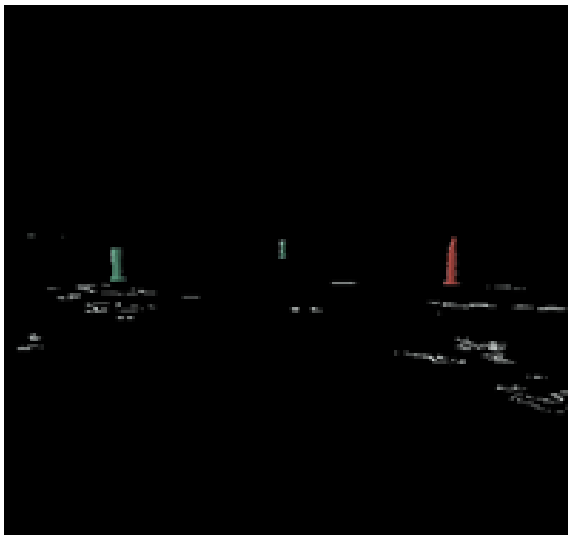

Figure 6: HSV Filtered Image

Once this filter was implemented, there was room for improvement for pulling out objects with the colors the WAM-V is looking for. However, enough detail was lost when implementing these additional color filters that other filters were determined to be more realistic.

A.3. Clustering with DBSCAN

The initial versions of the color-filtered image had a significant amount of noise that would still interfere with classification and there was no way to pull out buoys based on their shape. A proposed solution to both of these problems was to implement a clustering algorithm known as DBSCAN.

DBSCAN is an algorithm that searches for a specified number of neighboring points in a specified area of a pixel. Any pixels that pass this test and have enough similar pixels around them are considered to be clusters.[8] Figure 7 applies the DBSCAN algorithm to the filtered image above.

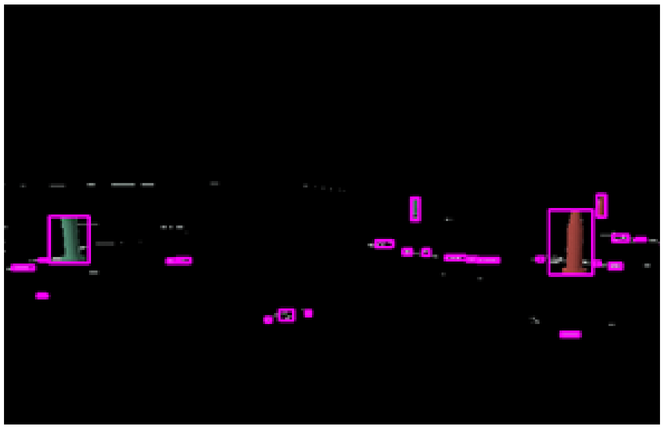

Figure 7: Clustered Image Using DBSCAN

A.4. Classification

As shown in figure 7, a large number of clusters appear in the image even after using the color-filtering to remove everything that does not match the approximate colors of the buoys. A solution to this was implemented by finding the color and proportionality of each pixel. It was determined that if a cluster was not in the same color spectrum as a buoy and the height to width ratio was off then the cluster would be discarded as irrelevant data. A third class of objects, besides red and green buoys, was made for buoys that were definitely the correct proportion but the color was wrong.

The combination of these filters still showed some false positives, but these rare false positives were mitigated by the implementation of lidar. Figure 8 shows the result of the classification algorithm.

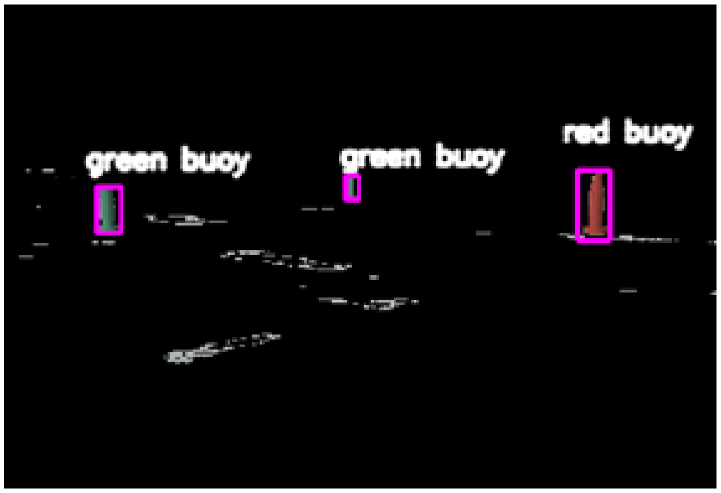

Figure 8: Classified Image of Buoys

A.5. Bearing

With the buoys isolated inside the image, the bearing of each object had to be found in order to cross reference the image with the LiDAR data. A minimum area rectangle was applied to each cluster that defined the centroid, length, width, and angle of the rectangle. The centroid of each minimum area rectangle should be the same as the centroid of each cluster and once the centroid had been identified, the bearing of each buoy could be found. Since the occupancy data was being mapped on a 2D map, the azimuth of each cluster was ignored in favor of only processing the bearing data. The bearing data was saved with the image coordinates for each pixel and then fed directly into the mapping pipeline for further analysis. The bearing measurements are shown below in Figure 9.

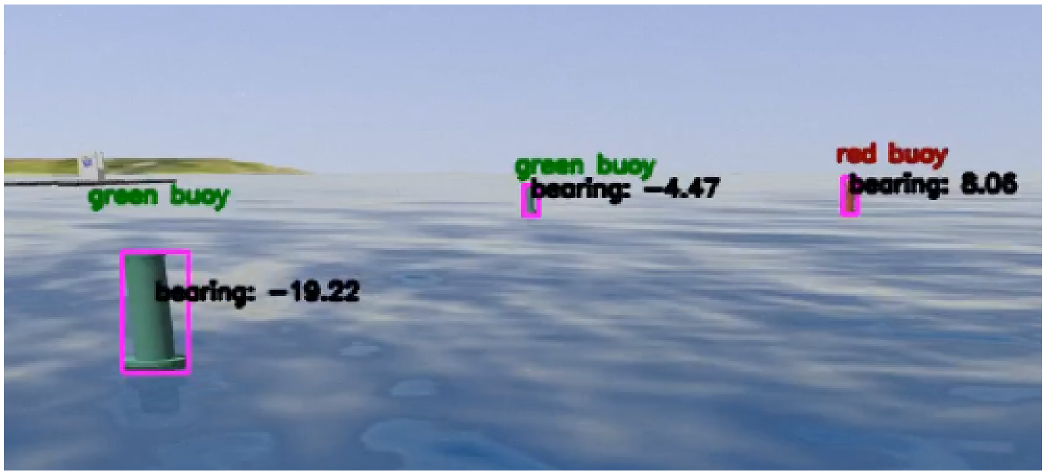

Figure 9: Bearing Measurements for Buoys

A.6. Abandoned Techniques

The following two subsections will discuss techniques that were attempted and abandoned for various reasons.

A.6.I. Feature Extraction

The original effort for isolating the buoys in the image was focused on two feature detection algorithms, SIFT and SURF. Both are algorithms that attempt to describe features by marking key points on an object and checking if those patterns match a pattern in another image. In theory, this would have worked great for pulling out where the buoys were provided we fed the algorithm an example of a buoy. This did not work out so well in practice however. While testing both SIFT and SURF, the sample buoys did not generate a lot of key points and as a result there were a lot of false positives and false negatives. Overall, neither algorithm proved particularly effective and then efforts were shifted to the color/cluster filtering.

A.6.II. Pre-Processing: RGB Colorspace

Digital color is often taught in the RGB colorspace, meaning that people are taught to understand color as combinations of the primary colors red, blue, and green. This is great for having a fundamental understanding of the colors, but for filtering the images it became much more difficult to describe certain situations, like dark colors or encapsulating every shade of green in one inequality. This difficulty made using the RGB colorspace impractical for the needs of the WAM-V, and instead the HSV colorspace was implemented in all of the filters to allow for a more robust interpretation of color.

B. Occupancy Map from LiDAR Data

To find the location of the buoys, our pipeline had to find some way of getting range values relative to the robot from the image. The group chose the lidar sensor because this sensor provided the most accurate range values in the image over a previous alternative: range images from stereo cameras.

B.1 Octomap

To convert lidar point cloud data into a 2D occupancy map, the team used a 3D Mapping Framework to generate and filter information. Lidar Point Cloud data is visualized in Figure 10. The team chose octomap because of the convenience- octomap accepted Point Cloud Data and gave the team a 2D Occupancy Grid topic to subscribe to. Although 3D Occupancy mapping was not our final intended result, the 3D Occupancy Map allowed the team to easily understand the effect of filtering which made troubleshooting and progress in the research quicker.[9]

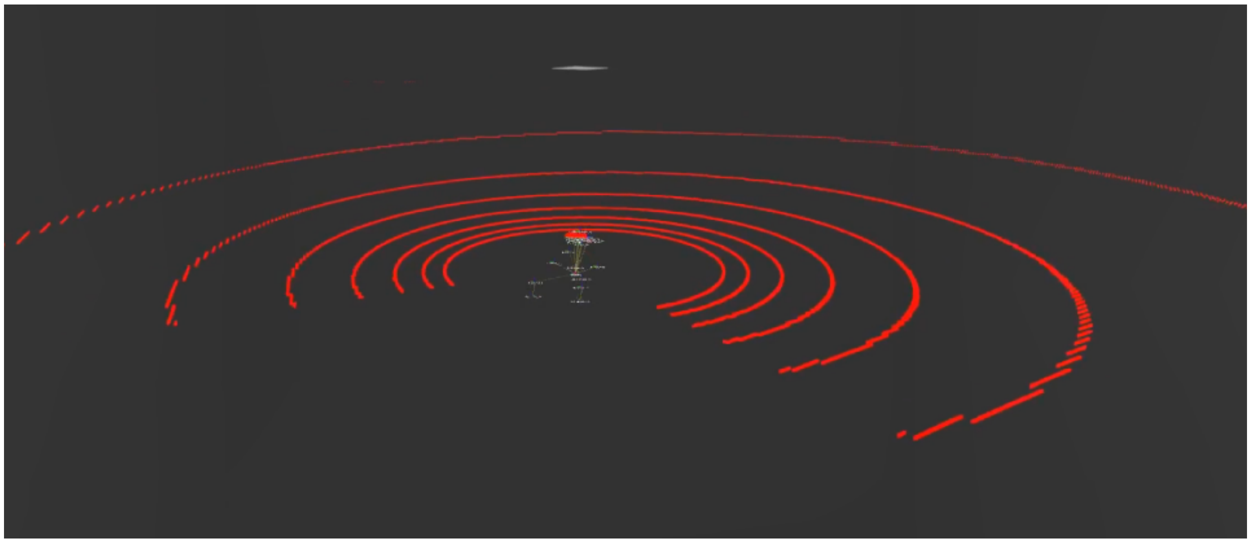
Figure 10: RViz Point Cloud Visualization

B.2 Filtering Point Cloud Data

Lidar data provides a point cloud of range values. In the ROS environment, lidar data is represented in the topic /lidar_wamv/points. The raw data is then fed into Octomap. When OctoMap began processing LiDAR data, the original view was very noisy as everything from the surface of the water to the back of the WAM-V appeared as occupied space. Figure 11 depicts the noisy occupancy grid and Figure 12 portrays the occupancy map. These two images gave the team no useful information about the location of the buoy.

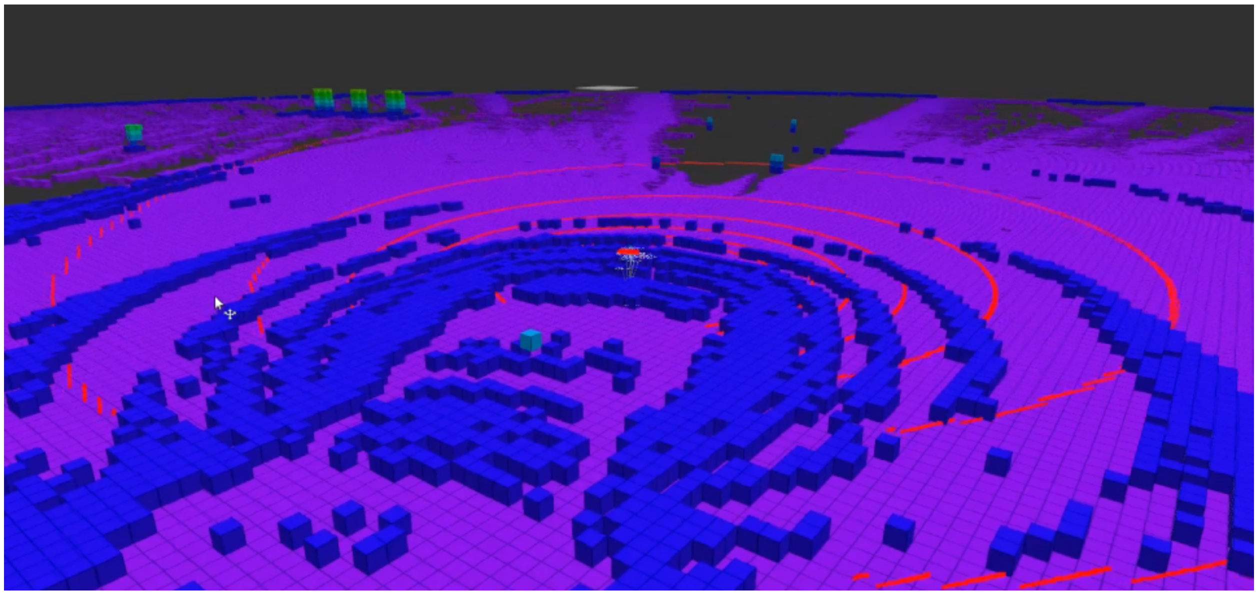
Figure 11: Noisy Occupancy Grid

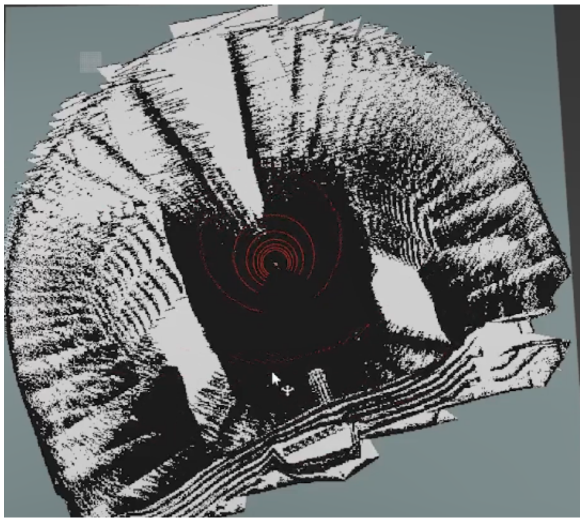

Figure 12: Noisy Occupancy Map

To remedy this, a height filter was implemented so anything at water level or above the lidar was filtered out. This allowed for most of the noise to be removed, only leaving pieces of the VRX tasks and the four buoys of interest. The filtered occupancy grid in Figure 13 is quite different than the occupancy grid in Figure 11.

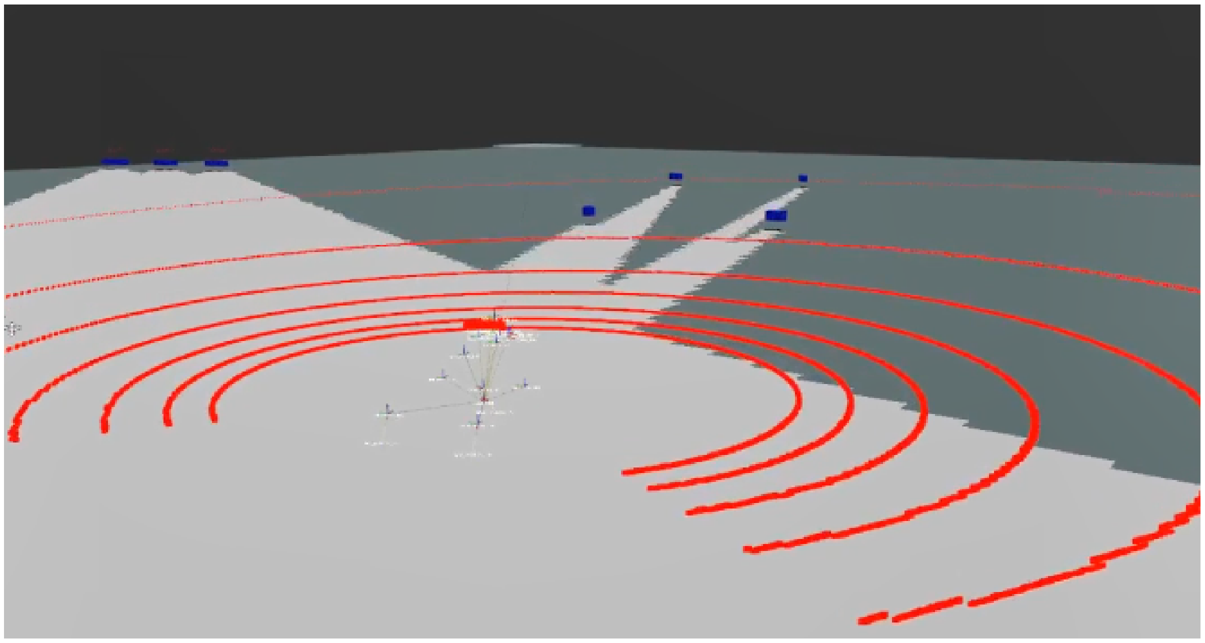

Figure 13: Filtered Occupancy Grid

Once this filter was established, the 3D map was converted to a 2D projected map as shown in Figure 14. and then implemented with the camera data.

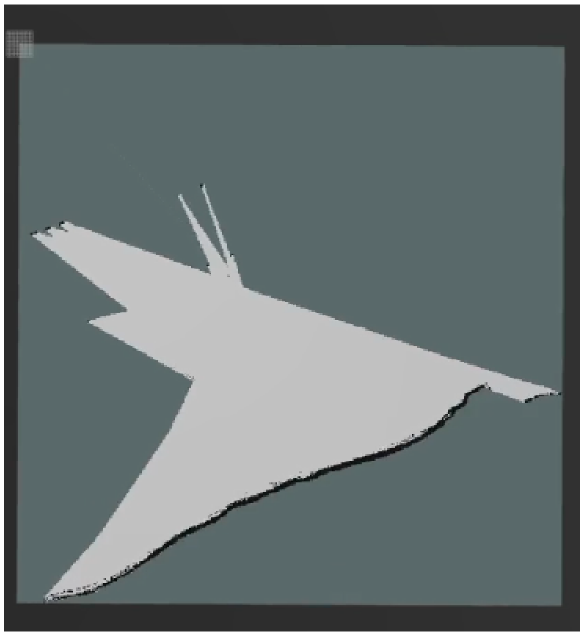

Figure 14: Filtered Occupancy Map

C. Sensor Fusion

With the 2D occupancy map and the bearings with tags from the classified rectified image, the group combined the sensor data to create an updated 2D occupancy map properly labeling the red and green buoys.

C.1. Transform Listener

In order to combine a map of the robot's environment with continuous live feed image data of a robot, the group needs to know the position and orientation of the robot. This is accomplished by using the /tf topic which is where the transform listener receives its information. The transform listener is an important feature of a robotics environment as it contains information about the robot's state or 6 degrees (x translation, y translation, z translation, x rotation or pitch, y rotation or roll, z axis or yaw). In the ROS environment, robots are modeled according to URDF format. In this format, robots are defined as a combination of joints and links. Figure 15 visualizes the links on the WAM-V.

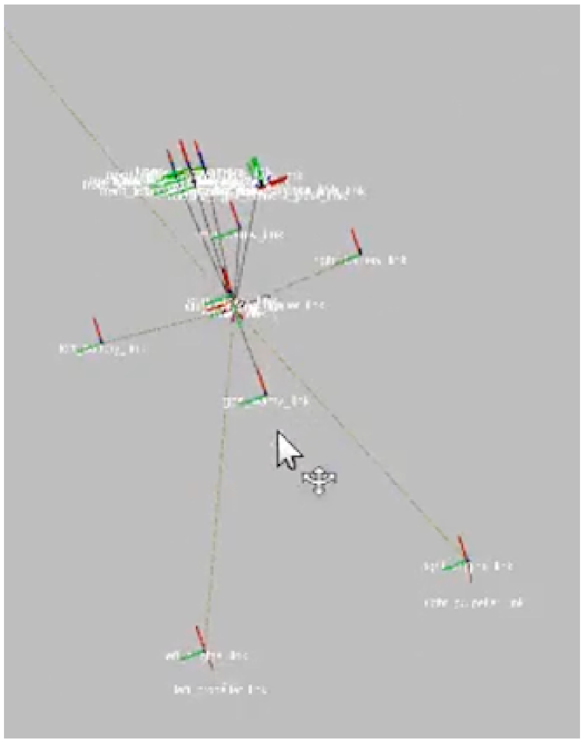

Figure 13: Links of WAM-V

By assigning an arbitrary point in space as the origin which is referred as odom in the report, the group can calculate the position and orientation of any link from odom.[10]

C.2. Origin, Robot Position, and Buoy Yaw

In the project, the joints and links have already been defined through the VRX repository developed by Open Robotics and Naval Postgraduate School. To get the tf transformations, it is important to define the two links for the transform. The group's two links are /odom and /front_left_camera_link. The first link is stationary in the environment and the map. It is important to have a position known in the map to base off all the transform calculations. Without it, the group would never be able to find the location of the camera. The front_left_camera_link is traveling with the robot and connected with the robot's camera. This link is the nearest to the camera. The transform will be able to get the position and orientation of the camera on the map. The tf transformation outputs a translational transform array and a rotational transform array. From the three values of the translational transform array (x translation, y translation, and z translation), the group was interested in the x translation and y translation. The z translation was negligible because the WAM-V is on an ocean surface. As shown in Figure 16, the robot's position with respect to the origin or odom can be calculated.

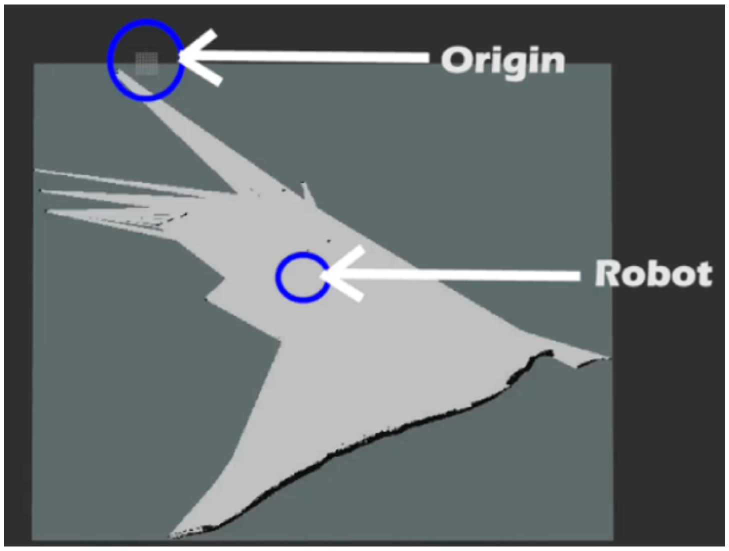

Figure 16: Position of Origin and Robot on Map

The rotational transform array outputs a quaternion array – the group transforms the array into an euler array to make the information more intuitive. This gives an array with three values- roll, pitch, and yaw. From these three values, the group was only interested in yaw. The robot's buoy yaw is shown in green in Figure 17. It is important that the bobbing motion of the robot creates error in the yaw. Roll and pitch are negligible because the robot is moving along the ocean surface. With these three pieces of information from the transform, the group can determine origin, robot position, and buoy yaw.

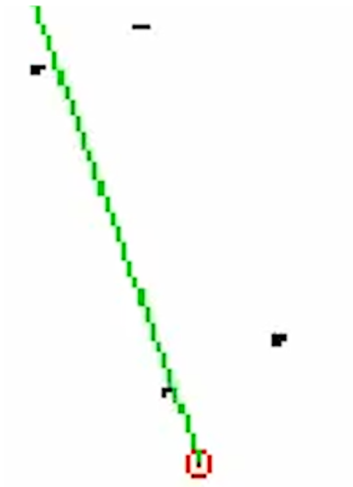

Figure 17: Depiction of Robot's Yaw

C.3. Area to Approximate Distance

In the classified rectified image, the group creates a minimum bounding rectangle over the relevant clusters in the image. This allows the group to calculate the area of the minimum bounding rectangle.

While troubleshooting the process, the group saw a pattern. As the robot approached a buoy, the size of the buoy increases proportionally. The group noticed there was an inverse relationship with area and the distance away from the buoy. After collecting various data points, a power regression model was developed for the relationship between area and distance. The excel is included with the source code in the Github. The approximation is a weak estimate and can be improved with more data and more accurate models. This machine learning problem can be a source of future work.

C.4. Triangle Rasterization

In the third step of the mapping pipeline in Figure 1, the group combined robot and origin pose (position and orientation), classified image feed, and occupancy data to generate slices of high probability regions of buoys. In order to generate slices, the group needed to define the boundaries of the slices. The group chose to make the slices triangles because the inaccuracy of the robot's yaw increases the further away the buoy is from the robot. (See figure for information) The side of the triangle farthest away from the robot is constant and a global variable in the program that can be changed based on the user's discretion. In the future, the triangle side should become a variable and be dependent on the conditions in the field. To generate the three points in the triangle, the group chose the first point as the robot's current position. A temporary point was assigned along the robot's yaw at an approximate distance location calculated from the section above. From two points, slope was calculated. From slope, the perpendicular slope was calculated. After solving the system of 2 equations (point slope equation and distance formula), the other two points in the triangle were calculated. With the x and y coordinates of the three points and the image, the triangle algorithm can calculate all the points inside the image. This is done by first finding the center of the triangle. Once found, the algorithm finds all the points that lie on the "inside"- side of all lines inside the triangle. Afterwards, it eliminates all the points below the lower approximate distance. Figure 18 demonstrates the shape and size of the slices.

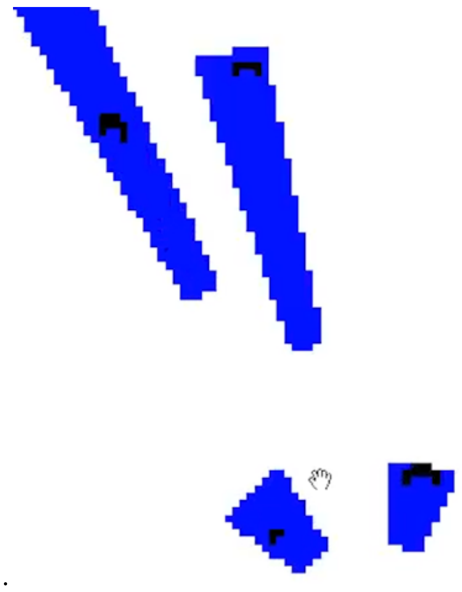

Figure 18: Slices of Potential Buoy Areas.

C.5. Updated Occupancy Map

With the slices generated fusing the sensor data from the robot and origin pose, classified rectified images, and 2D occupancy map, the group can check the map values in the slices to determine whether a specific area is occupied. If occupied, the tag is compared to the area, and can be classified as either a green buoy or a red buoy. Figure 19 below demonstrates this ability.

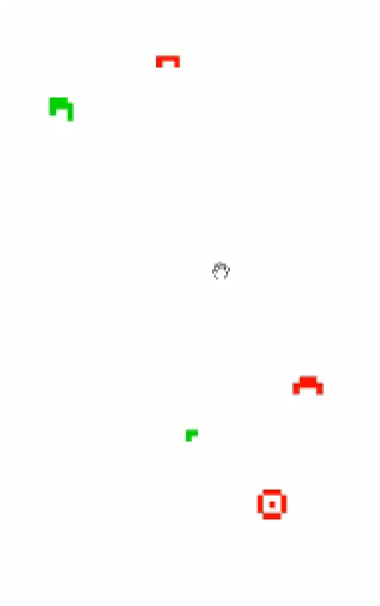

Figure 19: Updated 2D Occupancy Map

C.6. Abandoned Techniques: Disparity Image

During the development stage, the group's original intent was to use a range image generated from a disparity image. The two monocular cameras in the front of the vehicle in Figure 4 were configured as stereo cameras and a disparity image was generated from the stereo image processing library. To convert a disparity image to a range image, the group used a simple equation. However, the range image generated inconsistent results because of the limitations of the cameras. In Figure 20, the disparity image believes the buoys are closer than they actually are. To fix the inconsistencies, the stereo cameras needed a higher resolution and/or a larger baseline (distance between the cameras). Since these two solutions were beyond the group's control, the group decided to transition to lidar which proved more accurate.

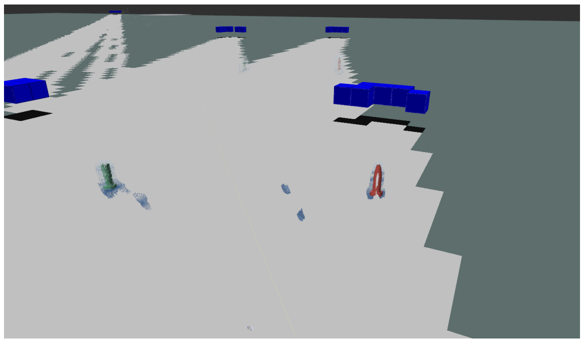

Figure 20: Discrepancy with Disparity Image and Lidar Data

IV. CONCLUSION

Over the course of this project, data from cameras and LiDAR were combined to create a complete map of its environment. Our current approach to image filtering almost always finds and classifies buoys in an image based on color. These buoys are then paired with their bearing and applied to a map in order to find each buoy's position in reference to the rest of the environment. While the process used to filter the data successfully isolated the buoys, the approach taken to filtering could be improved in another iteration of the project. A way to create a more robust solution in future works with the WAM-V would be to implement a deep learning solution for image classification and distance approximation. This would allow a similar or greater amount of accuracy for processing images while decreasing processing time. As it is though, the current implementation works for mapping the environment and correctly determining the location of buoys in a region accurately and efficiently. All code is publicly available on Github[11].

V. RECOMMENDATIONS FOR FUTURE WORK

After completing our research goal of sensing and mapping the buoys in the navigation channel, the next steps for completing the first task in the VRX environment is a pathfinding algorithm that can create a path for the robot to follow between the navigation channel. In addition, throughout this paper, there are recommendations for future work. However, there are two recommendations of chief importance to improving the functionality and efficiency of the program: Deep learning method to classify an image and Machine Learning Model to Approximate Distance. These short-term recommendations will allow the robot to complete the first task autonomously. In the long term, future work needs to be done to complete the other complicated tasks. Unfortunately, the package the group developed will not be tested in a VRX competition- the deadline for registration has already passed. Next year, the Robust Autonomy Field Laboratory is interested in registering for the VRX competition. This work will serve as a foundational work to develop a competition entry in the future.

VI. REFERENCES

1. www.robotx.org
2. https://bitbucket.org/osrf/vrx/src/default/
3. http://wiki.ros.org/
4. http://gazebosim.org/
5. http://wiki.ros.org/rviz
6. https://en.wikipedia.org/wiki/Lidar
7. http://wiki.ros.org/stereo_image_proc
8. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html
9. https://octomap.github.io/
10. http://wiki.ros.org/tf/Tutorials
11. https://github.com/ahuston-0/SRI-2019/