11-2017

Maritime Object Detection, Tracking, and Classification Using Lidar and Vision-Based Sensor Fusion

David John Thompson

MARITIME OBJECT DETECTION, TRACKING, AND CLASSIFICATION USING LIDAR AND VISION-BASED SENSOR FUSION

by

David John Thompson

A Thesis Submitted to the College of Engineering Department of Mechanical Engineering in Partial Fulfillment of the Requirements for the Degree of Master of Science in Mechanical Engineering

Embry-Riddle Aeronautical University
Daytona Beach, Florida
December 2017

MARITIME OBJECT DETECTION, TRACKING, AND CLASSIFICATION USING LIDAR AND VISION-BASED SENSOR FUSION

by

David John Thompson

This thesis was prepared under the direction of the candidate's Thesis Committee Chair, Dr. Eric Coyle, Associate Professor, Daytona Beach Campus, and Thesis Committee Members Dr Patrick Currier, Associate Professor, Daytona Beach Campus, and Dr. Brian Butka, Professor, Daytona Beach Campus, and has been approved by the Thesis Committee. It was submitted to the Department of Mechanical Engineering in partial fulfillment of the requirements for the degree of Master of Science in Mechanical Engineering

Thesis Review Committee:

[illegible]
Eric Coyle, Ph.D.
Committee Chair

[illegible]
Brian Butka, Ph.D.
Committee Member

[illegible]
Patrick Currier, Ph.D.
Committee Member

[illegible]
Jean-Michel Dhainaut, Ph.D.
Graduate Program Chair,
Mechanical Engineering

[illegible]
Eduardo Divo, Ph.D.
Department Chair,
Mechanical Engineering

[illegible]
Maj Mirmirani, Ph.D.
Dean, College of Engineering

[illegible]
Christopher Grant, Ph.D.
Associate Vice President of Academics

12/7/17
Date

Acknowledgements

I would like to thank everyone that helped me on my path to complete my thesis. First, I would like to recognize my advisor, Dr. Eric Coyle, who brought me into the Mechanical Engineering Department. His advice and guidance proved invaluable for my graduate education.

I want to acknowledge the growing team of engineers I work with on the RobotX project. Specifically, I want to mention Tim Zuercher, who provided countless hours of support debugging code with me. Marco Schoener and Jefferson Romney, who can get RobotX in the water and running at a moment's notice. Also, to Abby Butka, for working as my camera calibration assistant. Most importantly, Stephen Cronin, for being the team morale officer.

To my committee members, Dr. Brian Butka, and Dr. Patrick Currier, along with Dr. Charles Reinholtz, who continue to support my work, the RobotX team, and the robotics association. Their support continues to provide all sorts of opportunities for me and the hardworking members of the robotics association.

To Jeremy Brown, who helped Dr. Coyle and myself build the perception module of RobotX to what it is today.

Lastly, I want to acknowledge my friends and family that have supported my education from the beginning.

Abstract

Researcher: David J Thompson

Title: Maritime Object Detection, Tracking and Classification using Lidar and Vision-based Sensor Fusion

Institution: Embry-Riddle Aeronautical University

Degree: Master of Science in Mechanical Engineering

Year: 2017

Autonomous Surface Vehicles have the capability of replacing dull, dirty, and dangerous jobs in the maritime field. However, few successful ASV systems exist today, as there is a need for greater sensing capabilities. Furthermore, a successful ASV system requires object detection and recognition capabilities to enable autonomous navigation and situational awareness. This thesis demonstrates an application of LiDAR sensors in maritime environments for object detection, classification, and camera sensor fusion. This is accomplished through the integration of a high-fidelity GPS/INS system, 3D LiDAR sensors, and a pair of cameras. After rotating LiDAR returns into a global reference frame, they are reduced to a 3D occupancy grid. Objects are then extracted and classified with a Support Vector Machine (SVM) classifier. The LiDAR returns, when converted from a global frame to a camera frame, then allow the cameras to process a region of their imaging frame to assist in the classification of objects using color-based features. The SVM implementation results in an overall accuracy 98.7% for 6 classes. The transformation into pixel coordinates is shown here to be successful, with an angular error of 2 degrees, attributed to measurement error propagated through rotations.

Table of Contents

List of Tables

List of Figures

Chapter I
Introduction

Many vehicles today are equipped with a sensor package that aids a driver or operator in completing the tasks performed by the vehicle. For instance, aircraft may be equipped with downward facing radar to measure precise altitude for assisted landings, automobiles may be equipped with rear-mounted ultrasonic or camera sensors to assist the driver in backing up, and boats may be equipped with sonar for surveying the seafloor or thermal cameras for nighttime bay security. In both manned and unmanned operation, this sensor data is often intuitively displayed to a human operator to interpret the data and complete vehicle tasks more efficiently.

To move toward autonomous operation and away from reliance on human operators, advanced algorithms must be developed to interpret this sensor data and act upon it. For an autonomous surface vessel (ASV), base level autonomy begins with interpreting GPS data to guide the vessel to a desired location. This operation is sufficient for open water, but has issues in port operations or other crowded environments. Prime among these is that the ASV is not capable of detecting obstacles and avoiding them. Additionally, when traversing occupied waters, it is necessary to follow The International Regulations for Preventing Collisions at Sea (COLREGs), the United States navigation rules for surface vessels [1]. These navigation rules require the detection of other vessels and yielding to less maneuverable traffic, which cannot be completed solely through GPS navigation. Therefore, advanced tasks for ASVs require increased sensing capabilities beyond single GPS sensors.

Many Unmanned Surface Vehicles (USVs) in operation today are used for the completion of dull, dirty, or dangerous tasks. A USV may be deployed for surveying infrastructure, patrolling a harbor, or even destroying mines [2]. Autonomous vehicles have significant potential to expand the capabilities in these sectors by removing the human operators and improving efficiencies and safety. ASVs could more easily be deployed in large groups and monitored from a single location and operator, with no human on board [3]. Converting a manned ship to an ASV has the potential for even larger efficiency gains by removing several of the crewmembers. These ships could remain at sea longer than its manned counterpart, improving the chances of a search and rescue mission succeeding. In the shipping sector, much of the cost of transport goes to paying and accommodating the crew while at sea [4]. Furthermore, safety is inherently improved by removing human operators from the vehicle. The reasons listed above demonstrate a need for more advanced ASVs in the field.

This thesis focuses on the integration of sensors to increase the capability of ASVs, by detecting and classifying objects. This is accomplished through the integration of a high-fidelity GPS/INS system, 3D LiDAR sensors, and a pair of cameras. Object detection and the initial classification is performed by the LiDAR sensors based on spatially distinct features. The LiDAR returns, when converted from a global frame to a camera frame, then allow the cameras to process a region of their imaging frame to assist in the classification of objects using color-based features.

1.1 Platform and Competition

To accomplish this thesis the Minion ASV platform is used. Minion is Embry-Riddle Aeronautical University's (ERAU) entry into the Association for Unmanned

Vehicle Systems International (AUVSI) Maritime RobotX Challenge (MRC) [5].

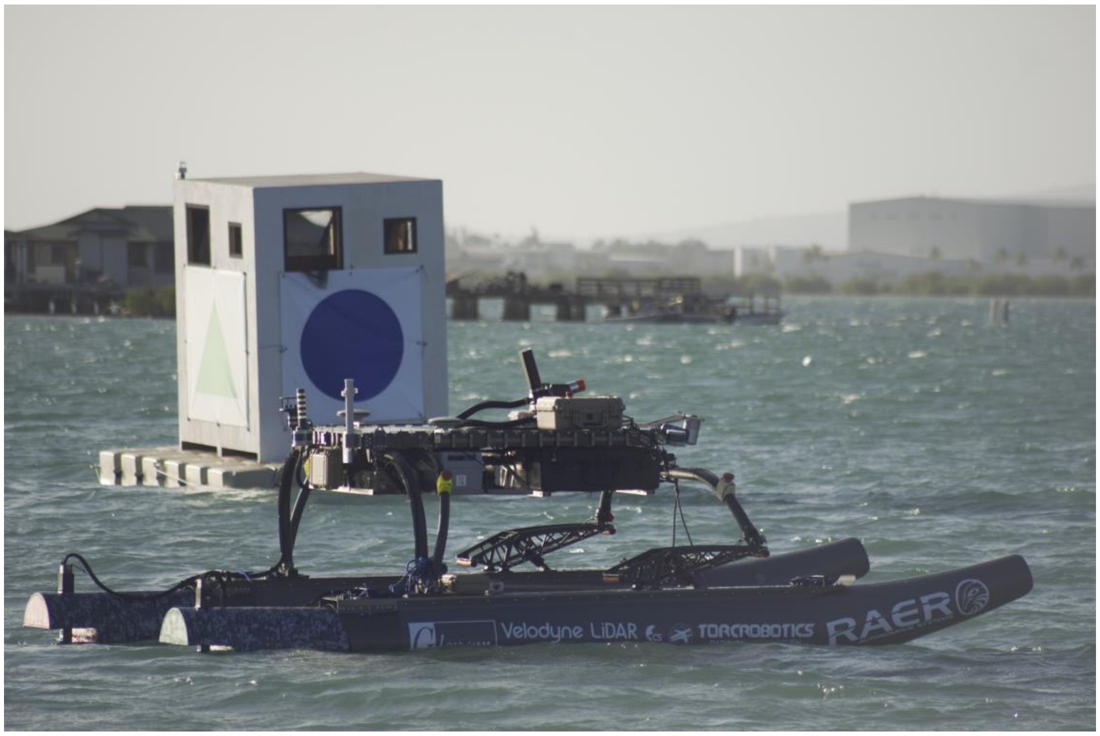

Figure 1: The Minion ASV on the competition field in Hawaii. Object in background is the detect and deliver target.

The 2016 MRC consisted of 8 tasks, which require the use of multiple sensing modalities. The 100m by 150m course contained 7 of the tasks to be completed in nearly any order by the platform. Some of these tasks include traversing a buoy channel, locating an underwater acoustic pinger below a row of buoys, and automated docking. Figure 2 shows an example course layout containing all the challenges. This figure is for illustrative purposes as the final course layout was unknown prior to competition.

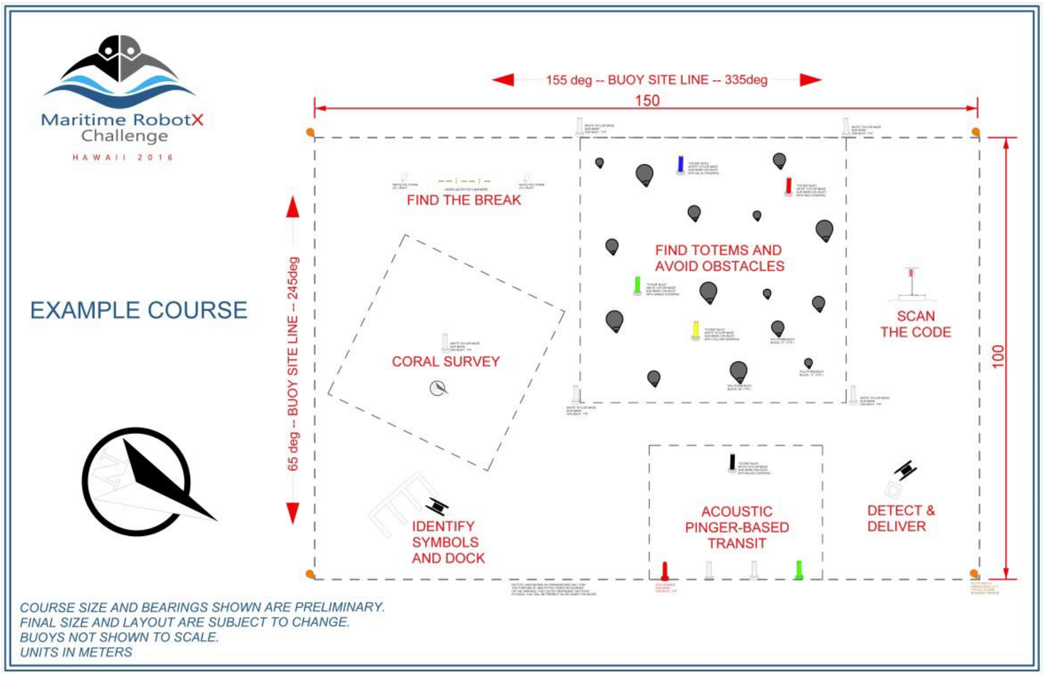
Maritime RobotX Challenge
HAWAII 2016
155 deg -- BUOY SITE LINE -- 335deg
150
100
65 deg -- BUOY SITE LINE -- 245deg
EXAMPLE COURSE
FIND THE BREAK
FIND TOTEMS AND AVOID OBSTACLES
SCAN THE CODE
CORAL SURVEY
ACOUSTIC PINGER-BASED TRANSIT
DETECT & DELIVER
IDENTIFY SYMBOLS AND DOCK
COURSE SIZE AND BEARINGS SHOWN ARE PRELIMINARY.
FINAL SIZE AND LAYOUT ARE SUBJECT TO CHANGE.
BUOYS NOT SHOWN TO SCALE.
UNITS IN METERS

Figure 2: Example course layout. Final course layout is not known prior to competition. [7]

To gain access to the course, each team was required to complete the first task aptly named "demonstrate navigation and control." This task pictured below in Figure 3 shows the layout of the task. Using the LiDAR and camera sensors, Minion must first recognize two Taylor Made Sur-Mark Can Buoys [6] roughly 10m apart and 1.5m tall drive through the start gate, and then traverse through the end gate. Upon completion, teams were permitted to enter the course and begin any other tasks.

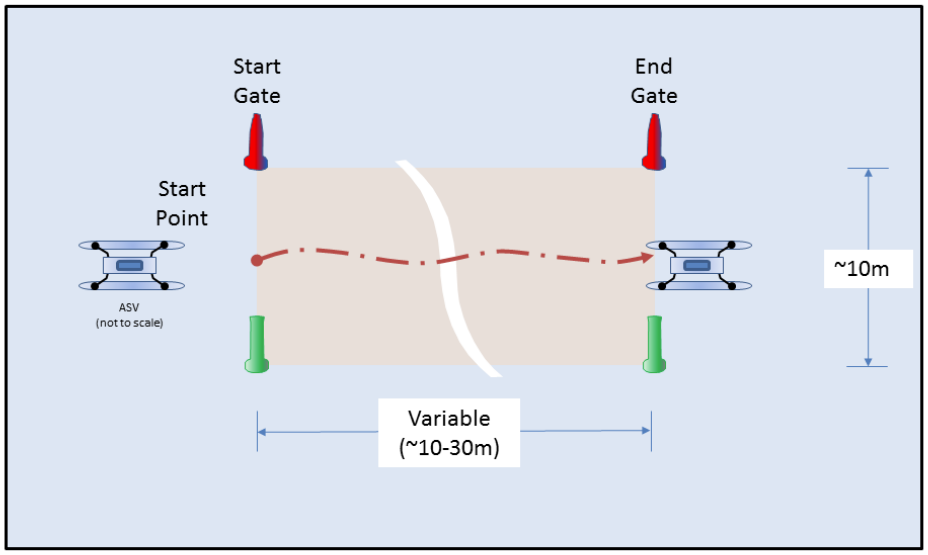

Figure 3: Demonstrate Navigation and Control Task [7]

While all the tasks require object detection and classification, the three tasks most dependent on this ability are scan the code, find totems and avoid obstacles, and identify symbols and dock tasks. Each of these three tasks required Minion to identify these objects, map their location and correctly identify color and/or shape of the object. For the scan the code task, Minion must identify a light tower roughly 3m tall and accurately identify a 3-color code displayed on the tower face. For the find totems and avoid obstacles task, minion must identify buoys of varying shapes and sizes. However, some of these buoys are totems of varying color that Minion must circle to indicate the totem has been properly recognized. Lastly, the identify symbols and dock task requires minion to identify a 3-bay dock and the shape of color of each bay's corresponding sign. Minion is required to dock in the correct bay corresponding to a known dock sign color and shape.

This competition requires the successful integration of object detection techniques to complete all the competition challenges. Utilizing the LiDAR sensors, objects may be detected and then classified with a SVM classifier. The LiDAR returns may also be used for extracting a camera region of interest for additional color processing capabilities.

Chapter II
Review of the Relevant Literature

Much maritime research has been devoted to the detection of objects, particularly in unmanned and autonomous operations. Object detection permits greater environmental awareness while also allowing for capabilities such as obstacle avoidance, threat detection, and scene reconstruction. In many applications, multiple sensors are used for object classification, often fusing a camera with GPS. Other common solutions fuse a combination of LiDAR, camera, RADAR, and/or SONAR together with GPS/INS state information. However, while many solutions involving LiDAR sensor fusion have been demonstrated in the ground domain [8] [9], few solutions appropriately apply this type of perception in maritime environments.

In maritime environments, it is important to detect waterway markings to maintain compliance with COLREGs and avoid underwater hazards. COLREG compliance has been explored using a single-beam LiDAR and a USB webcam [10] . However, the object classification was performed using a background subtraction method. Through some tuning [10] could remove the false positives generated by the dynamic motion of the waves. However, [10] only considered a stationary vehicle scenario, and even with tuning a camera in motion is generally not capable of performing background subtraction and thus, is inadequate for most practical use cases. The LiDAR is also only used in this case for single-point range measurements and is not used for classification, leading to robustness concerns.

Utilizing a multi-beam LiDAR, [11] developed maps for surveying partially-submerged vessels. Through the fusion of LiDAR and SONAR, [11] could map the entirety of a partially-submerged vessel by combining a surface map from the LiDAR and underwater map from a SONAR. Scan data was post-processed using an Iterative Closest Point (ICP) method, which can be a slow process, so that multiple scans could be compiled without the need for state data. This was accomplished by first simplifying raw point cloud locations to a 3-dimensional occupancy grid with a 30cm cell size. For this method to be more useful in real-time applications, the incorporation of a GPS is needed to remove the need for an intensive ICP algorithm, which the author acknowledges. In this application, the classification of the object being scanned is already known, and although it may be used for obstacle avoidance, may not be applied to an ASV with no apriori knowledge of the environment.

One of the most practical use cases for an ASV is automated docking, allowing for autonomous deployment and recovery. In one application, [12] approached this problem using a 32-beam LiDAR. In this research, the LiDAR was not mounted on a vessel at all but instead on a cart to simulate the mounting location of LiDAR on a vessel. This research also compiled multiple LiDAR scans, but unlike [11], [12] incorporated preprocessing of the points using a GPS and compass to make a more accurate initial guess of ICP. The preprocessing method was used to reduce the false positive rate of the ICP method that stems from the repeating structure of a dock. However, it should be noted that the cart could not simulate the rocking motion typically seen in a USV. In this approach, dock classification was performed using image morphological operations and the RANSAC algorithm. Points were flattened to a 2-dimensional occupancy grid and treated as a binary image. This approach permits binary morphological operations to be performed so that the shore may be filtered out of the dataset. Then using a RANSAC method, the flat walkway and round pilings of the dock are extracted. This method demonstrated an ability to accurately detect several docks, however the authors note that this method only applies to long straight docks, and not docks with bends or T-shapes, as well as partially occupied docks.

Other research, such as [13], uses a method for flattening a 3D point cloud into a 2D image. However, where [12] uses binary images for morphology, [13] used grayscale images where the color channel represents laser intensity. This research also performed classification on multiple objects through the implementation of a back-propagation neural network and Support Vector Machine (SVM) classifier. However, the neural network proved to be computationally intensive, and despite a 95% accuracy, may not be practical use for real-time applications.

The work that preceded this thesis on the Minion platform, [14], implemented a Multi-Variate Gaussian (MVG) classifier for detection of multiple objects using a 32-beam LiDAR and state information from a GPS/INS solution. Like other work, [14] flattened 3D returns into a 2D occupancy grid. The MVG classifier was only used to differentiate 2 classes of buoys from an unknown class. As a result, only three features were needed to efficiently classify the objects. Using distance to object, object radius and number of rings, [14] achieved an overall classification accuracy of 93.84%. However, the methods of [14] treated all objects as being circular and was completely based on the current LiDAR scan, electing to throw away previous scan data.

Most prior research on autonomous surface vehicles has been primarily focused on the use of camera-based classification. The works that incorporate LiDAR sensors into a sensor fusion scheme most often use them as a distance measuring aid rather than detection and classification. Those applications that did perform detection and classification of objects primarily with LiDAR used only a few classes. To accurately classify multiple maritime objects, a different approach must be used that allows fusing features from both vision and LiDAR sensors. A SVM classifier may be used to do this, so here the efficacy of a LiDAR-based SVM classifier is investigated as well as a method to use objects detected by a LiDAR to improve feature extraction from imagery. A support Vector Machine classifier works by mapping features to a higher dimension so that a linear separating hyperplane may be found between two classes [15]. As there may be many possible separating hyperplanes, SVM selects the hyperplane that maximizes the margin between the two data sets, effectively minimizing overtraining. This can be shown below in Figure 4.

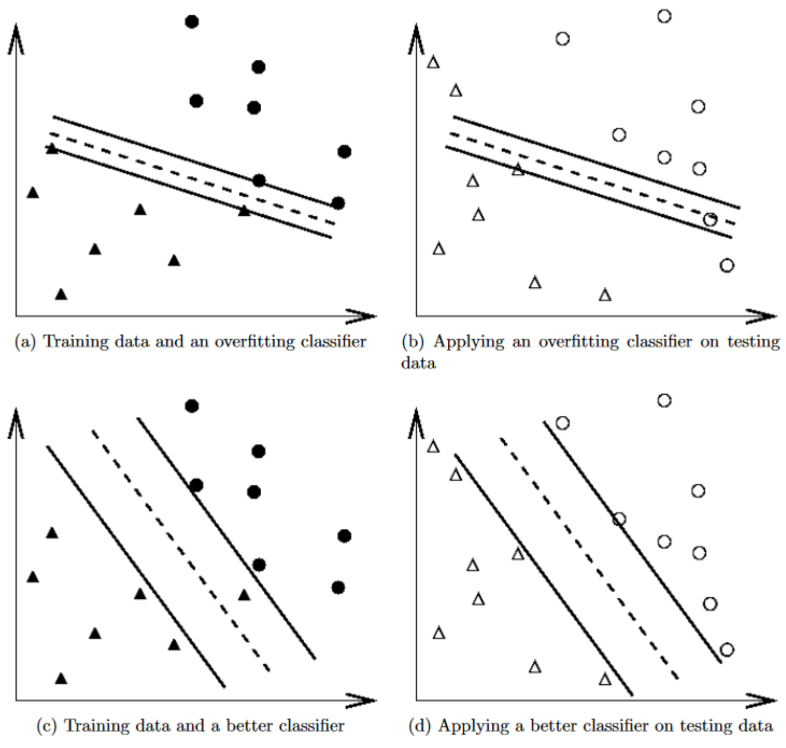
(a) Training data and an overfitting classifier
(b) Applying an overfitting classifier on testing data
(c) Training data and a better classifier
(d) Applying a better classifier on testing data

Figure 4: comparison of two classifiers, where SVM maximizes the margin to create (c) and (d)

Because SVM finds a linear hyperplane between two datasets, it is only suitable in its base form for solving 2-class problems. To solve multi-class problems, SVM must be implemented with a “1 vs. 1” or “1 vs. all” approach. In a “1 vs. 1” approach each class is tested against each other class and the class with the highest number of vote is estimated as the correct class. In “1 vs. all” one class is tested against all other classes simultaneously. This process is repeated for every class. For a “1 vs. all” approach, the classification is chosen using a decision function outlined in [17].

Chapter III
Methodology

3.1 Sensor Suite

The Minion platform utilizes 4 LiDAR sensors, 2 visible light cameras, and a GPS/INS system for sensing vehicle state and environment. The placement of the sensor suite can be seen in Figure 5 and Figure 6.

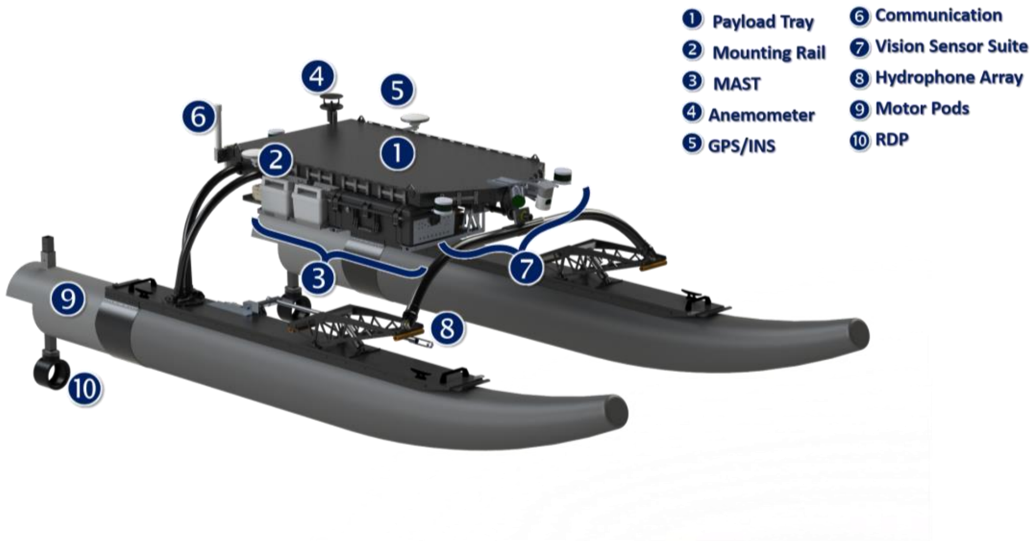

Figure 5: 2016 Minion Platform

The onboard GPS/INS system is a TORC Pinpoint which provides inertial NED frame outputs of the vehicle state. The cameras used are two Point Grey Blackfly cameras with Theia ML410M lenses to provide ~90 degree field of view (FoV), with the cameras rotated ~45 degrees apart. The camera housings are labeled “3” in Figure 6 shown below. The cameras are positioned to increase the overall field-of-view (FOV) of the vision suite and are not capable of stereovision depth estimation.

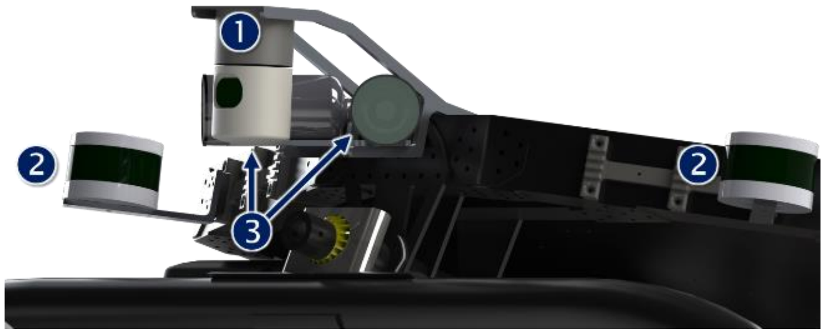

Figure 6: 1: HDL-32E mounted upside down 2: Front VLP-16 HD Sensors 3: Point Grey Blackfly Cameras

The LiDAR labeled "1" in Figure 6 is a Velodyne HDL-32E, while the LiDAR sensors labeled "2" are Velodyne VLP-16 HD's. The platform also has a Velodyne VLP-16 LiDAR mounted at the stern of the platform. The HDL-32E is a 32-beam laser rangefinder with an equal beam spread ranging from 10 degrees up and 30 degrees down while the VLP-16 is a 16-beam laser rangefinder with equal beam spread ranging from 15 degrees up to 15 degrees down, measured from the center of the sensor. The VLP-16 HD is a narrow version of the regular VLP-16, reducing the angular spread from ±15 degrees to ±10 degrees. The HDL-32E is mounted on the bow of the platform, along the port and starboard centerline, upside down, and pitched 10 degrees down. The VLP-16HDs are mounted on the port bow and starboard bow of the platform and pitched down ~5 degrees. The rear VLP-16 is pitched down ~2 degrees. The mounting configuration of the front-mounted Velodyne sensors can be seen in Figure 6. The LiDAR mounting positions were determined through simulation to maximize the return density on the water surface.

For this laser configuration, a simulation of expected LiDAR returns on a flat water surface is seen below in Figure 7. Each color corresponds to a different LiDAR sensor, providing qualitative feedback on sensor position. Given the platform spends most of its time driving forward, a higher forward density of returns was prioritized over aft returns. A configuration was deemed sufficient when a Polyform A3 buoy could not fit between laser returns within the black region in Figure 7. The resulting region provides sufficient coverage 28m in front of the vessel, 12m aft of the vessel, and 28m to the port and starboard sides of the vessel.

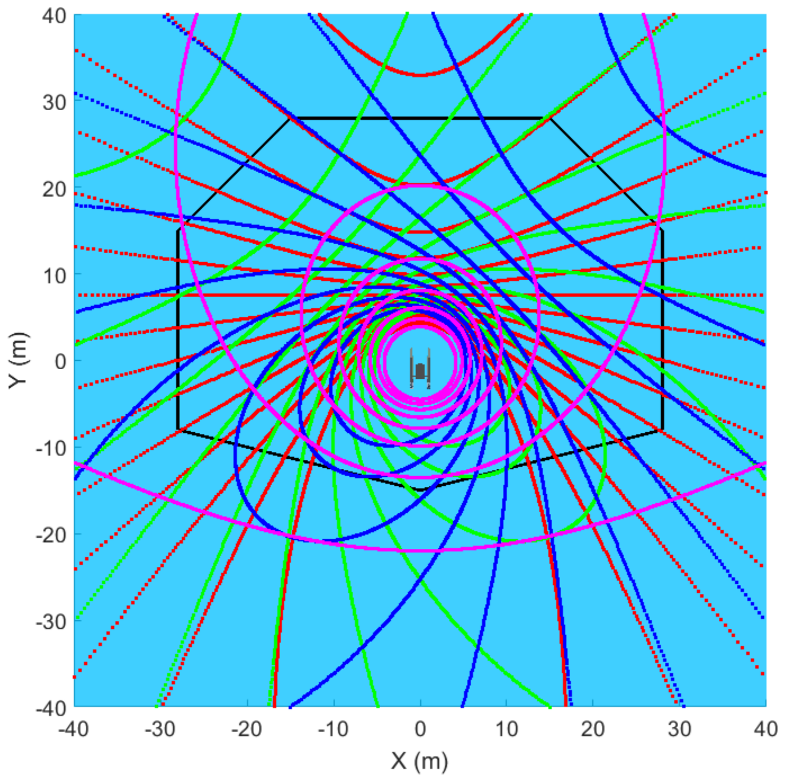

Figure 7: Simulated LiDAR return pattern on water surface. The black polygon represents the classification region, red lines represent returns from the HDL-32E, purple lines represent the aft VLP-16, and blue and green lines represent the starboard and port side VLP-16HD's respectively.

To accomplish all the tasks for the MRC, an advanced software suite was developed, shown below in Figure 8. The software suite is designed to interpret state and sensing data for completion of the competition task. This is accomplished with modules for each set of sensors, and a tracker for managing the boat's actions. Figure 9 below shows the process steps within the perception and vision module that is covered by this thesis. The perception steps utilize the robot state to transform LiDAR returns into a global north-east-down (NED) coordinate frame. Object segmentation is performed by fitting LiDAR returns to an occupancy grid and extracting objects from a flattened 2D grid. Extracted objects are then classified and this data is provided to vision for region of interest extraction. The vision module in red uses robot state and the perception objects to generate pixel coordinates containing objects of interest. This is accomplished by rotating the LiDAR points into pixel coordinates and then distorting the coordinates to conform to the camera lens. The region of interest is then padded to compensate for measurement errors and the resultant ROI may be extracted for processing. While thesis focuses entirely on increasing the capabilities of the perception and vision modules, other modules are utilized to retrieve vehicle state and the object list.

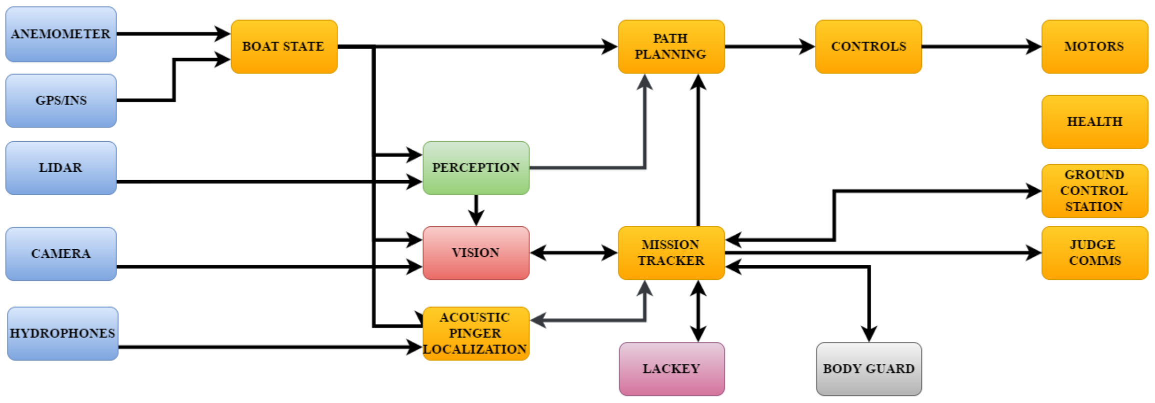

Figure 8: Software diagram of the Minion boat platform

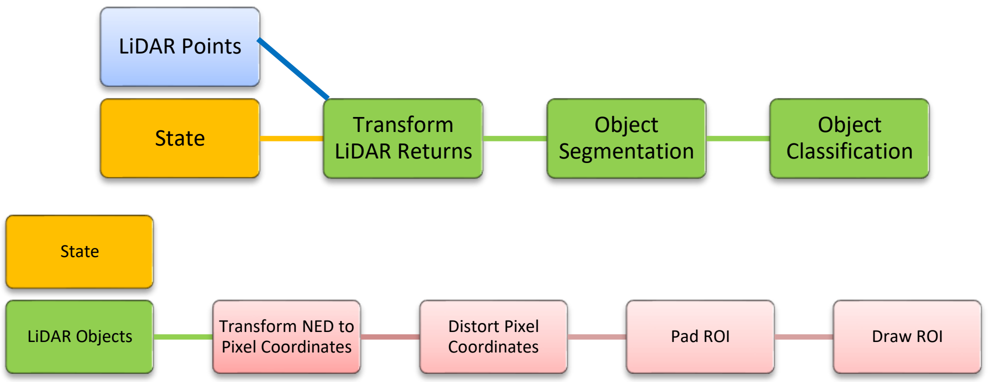

Figure 9: Above - Perception process. Below - Vision ROI extraction process

3.1 Coordinate Transformation

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

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

$$\boldsymbol{p_{VEL}} = \begin{bmatrix} \boldsymbol{R\sin\omega\cos\alpha} \\ \boldsymbol{R\sin\omega\sin\alpha} \\ \boldsymbol{R\cos\omega} \\ \boldsymbol{1} \end{bmatrix}, \tag{1}$$

where P_{VEL} is a single LiDAR return in the Velodyne's reference frame, R is the distance measurement reported by the sensor, α is the rotational azimuth angle reported by the sensor, and ω is the elevation angle of the laser. This transformation is illustrated below in Figure 10. It is important to note, however, that this general form does not account how the vehicle state changes between sensor updates. For example, a vehicle with 50Hz state updates moving at 2m/s with no angular velocity could accumulate up to 0.04m between updates of vehicle state. The solution used for this implementation interpolated vehicle state to apply a more accurate position with every transformation.

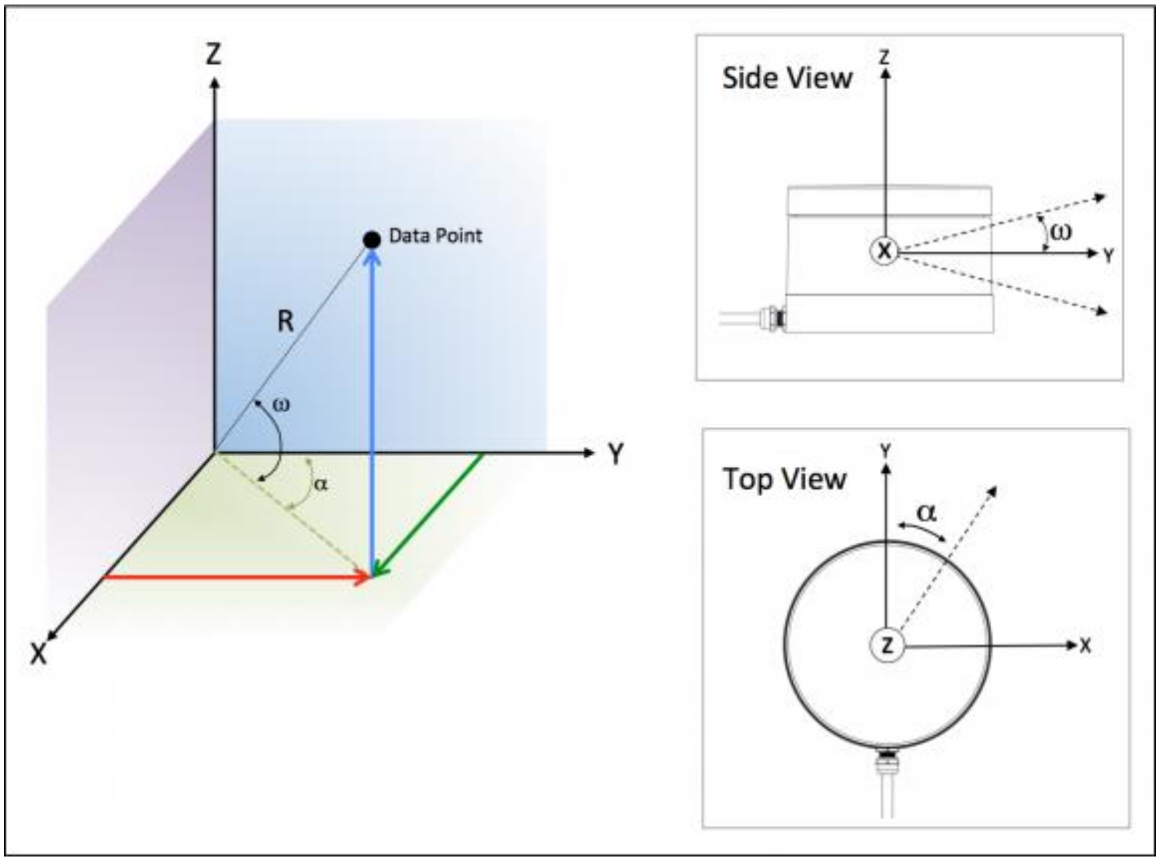

Figure 10: Coordinate frame of the Velodyne sensors [18].

Then, using the known mounting location and orientation of each LiDAR, the points are moved into the FRD reference frame using:

$$p_{FRD} = T_{FRD}^{VEL} p_{VEL}, \tag{2}$$

where T_{FRD}^{VEL} is the homogeneous transform from the Velodyne LiDAR's local frame into the local FRD reference frame of the vessel. Similarly, the LiDAR returns are then moved into the NED global reference frame using the Torc PinPoint GPS/INS timestamp reported state of the vessel, i.e. the NED location and Euler angles. This is given by:

$$p_{NED} = T_{NED}^{FRD} p_{FRD}, \tag{3}$$

where T_{NED}^{FRD} is the homogeneous transform from the local FRD frame to the global NED frame. T represents a homogenous transform of the form shown below in equation 4, where R_{zyx} is the 3x3 rotation matrix of order ZYX and t is the 1x3 translation vector.

$$T = \begin{bmatrix} R_{zyx} & t \\ 0 & 1 \end{bmatrix} \tag{4}$$

3.1 Occupancy Grid

When dealing with large point clouds it is necessary to use an efficient means of processing this data and segmenting the point cloud into objects. An occupancy grid has the advantage of removing redundant data points which also simplifies object segmentation. This is accomplished by fitting discrete points of LiDAR data to grid cells. Multiple points within the same cell will result in only a single filled cell. Operations are then performed on the grid, such as object extraction, for aiding the tracking and classification of objects. The process will be detailed below, with figures illustrating the process are shown in the results section.

Object extraction is performed by first compressing LiDAR returns into a 3D occupancy grid. The occupancy grid is referenced to the global frame, but the grid range is limited to tunable area around the vessel. Here, the $MxMxN$ grid matrix has a size defined by:

$$M = 1 + (d/\delta), \tag{5}$$

and

$$N = 1 + (h/\delta), \tag{6}$$

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

$$p_{idx} = round\left(\frac{\acute{p}_{NED} - q_{NED}}{\delta}\right) + \frac{[M \quad M \quad N]^T}{2}, \tag{7}$$

where $\acute{p}_{NED}$ is a 3x1 vector comprised of the first three elements of the 4x1 p_{NED} vector, $round(x)$ rounds all elements of x, and q_{NED} is the current NED location of the USV. This equation can easily be inverted to give NED location for any indices in the grid.

The current discussion has been limited to using LiDAR returns from the most recent LiDAR scan. However, a single scan is not sufficient to detect geometry of a waterborne object due to gaps between the sensors' lasers. Instead, a temporal decay of grid cells is used to both fill in the detail of objects over time and allow the object's location to change. To maintain efficiency, the temporal information is stored within each cell of the grid with a single data byte. A byte value of zero indicates an unoccupied cell, but 1-255 represents an occupied cell and its age η. The age η of a cell is set to η_{max}, a tuned parameter representing the maximum number of scans iterations that the cell will remain filled, for any cell with a current index p_{idx}, and decremented otherwise.

Realize that while object heights are useful as classification features, the height is rarely needed for path planning. This is because unlike a ground environment where there are plentiful overhead obstacles such as foliage, signs, lights, and overpasses, a maritime environment generally only has bridges that create overhead obstacles. Thus, the grid can be flattened to 2D for navigation purposes. To this end, the 3D occupancy grid is first flattened to 2D, resulting in a binary matrix. While object segmentation could then be performed by clustering algorithms, these algorithms can be computationally intensive and may require the number of objects to be pre-determined.

3.2 Object Extraction

Here object extraction is performed using the pixel following method. This method is described fully in [19]. Any object with a point inside of another object is ignored, so that only the outer object boundaries are retained. The coordinates of these contours are moved from the image frame and into the NED frame using the previously discussed operations in (7), resulting in a list of objects $A = \{a_1, \quad a_2, \quad \cdots \quad a_n\}$. Where each object $a_1 = \{x_1, x_2, \ldots\ x_n, y_1, y_2, \ldots\ y_n\}$ is defined by a set of NED vertices, x is a northing coordinate, and y is an easting coordinate. It should be noted that any 2D or 3D occupancy grid cell within the bounds of a_1 and $\eta > 0$ can be used to compute spatial characteristics of the object such as size and surface area.

While the list of objects A has been created, it is beneficial to reduce the number of points that represent the polygon a_1 to improve path planning computations. To accomplish this, the Ramer-Douglas-Peucker point decimation algorithm is used [20]. This algorithm uses an iterative method to reduce the number of points on a curve or polygon to find a similar polygon subject to a perpendicular distance constraint. This distance is treated as a tunable value, but in general should be at least as large as the grid resolution δ. Setting the value too high will result in a loss of object resolution to the point of distortion.

3.3 Classification

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

$$F = [F_1, F_2, F_3, F_4, F_5, F_6, F_7, F_8, F_9, F_{10}]$$

Where:

$F_1 - Max\ Intensity\ of\ Object\ Returns$

$F_2 - Min\ Intensity\ of\ Object\ Returns$

$F_3 - Average\ Intensity\ of\ Object\ Returns$

$F_4 - Standard\ Deviation\ of\ Intensity\ of\ Object\ Returns$

$F_5 - Max\ Height\ of\ Object$

$F_6 - Total\ Number\ of\ Filled\ Cells\ in\ the\ Object$

$F_7 - Polygon\ Perimeter$

$F_8 - Polygon\ Area$

$F_9 - 2D\ Minor\ Axis\ Length$

$F_{10} - 2D\ Major\ Axis\ Length$

Training on these features was performed using the libSVM [15] library. Using the libSVM library allowed for use of the SVM classifier in multiple languages, including LabVIEW and MATLAB. This library also supports linear and non-linear applications. In many cases a linear SVM should be sufficient for classification, and given its quick application, was used as a first pass for validating the training sets. However, to provide a more accurate model, non-linear SVM was used for the final model. A Radial Basis

Function (RBF) kernel [16] was selected for training as it provides the most versatility when compared to a polynomial function or linear models.

All training data was scaled using Equation 8. Where F is the resulting feature, f_i is the input data, and f_{min} and f_{max} are the highest and lowest values for that feature in the training set. The equation maps all data points in training to a minimum value of 0 and a maximum value of 1. When handling test data, the data must still be scaled, but f_{min} and f_{max} retain their values from training.

$$F_n = (f_i - f_{min})/(f_{max} - f_{min}) \tag{8}$$

Data sets were collected by labeling known objects in real-time by manually driving the vessel. This was accomplished by assigning a unique ID to each discovered object, labeling the class of the object manually, and storing the object's feature vector for post-processing. The use of object IDs improves data collection accuracy by ensuring only the feature vectors of interest are saved to the correct object. The data was formatted as a text file in the LibSVM standard with an example shown below. The first value before the feature vector indicates class, and each feature was assigned a number corresponding to its order in the vector such that the features may be in any order and the training process would still be correct.

$$1\ 1{:}F_1\ 2{:}F_2\ 3{:}F_3\ 4{:}F_4\ 5{:}F_5\ 6{:}F_6\ 7{:}F_7\ 8{:}F_8\ 9{:}F_9\ 10{:}F_{10}$$

A C-SVC model uses tuned values C and gamma paired with an RBF kernel. C is a parameter describing the error margin when training. A low value for C is well suited for problems with high noise, as outliers are more easily rejected. A high value for C is used when there is low noise, and all training values may be considered for support vectors. Gamma is used to define the inverse of the standard deviation of the kernel. A small value for gamma corresponds to a Gaussian function with high variance, while a large gamma value corresponds to a Gaussian function with small variance. To quickly find values for C and gamma that return high model accuracies, a grid search approach may be used. This approach iterates through values C and Gamma, training the model several times. The results from this approach yield a confusion matrix from each iteration of C and Gamma. The grid with the highest accuracy is then used for cross-validation training.

3.4 Region of Interest Extraction

Using the location of the object detected from the LiDAR, and the known position of the boat, it is then possible to compute the object bounds in the pixel frame used by the onboard cameras. After this operation, a region of interest may be extracted from a camera image containing only the object for which color information may be found. This method reduces computation time by only processing a small portion of the image and provides a simple process for fusing information between LiDAR and camera objects.

Any object expressed in an inertial NED frame may be translated into pixel coordinates through a known geometry of the camera and calibration procedure. This translation to pixel coordinates is valid for any camera represented with the pinhole model using the method developed by Zhang [21]. With the pinhole model, a ray can be drawn from the target through the pinhole and projected onto the image plane. The intersection point between the ray and the image plane provides the pixel coordinates for an undistorted image. This is shown below in Figure 11. The tree on the right side is the actual object. Where the rays intersect is the pinhole, with the second tree being the image projected onto the camera. The tree on the far left represents a second valid solution, despite not being the actual object.

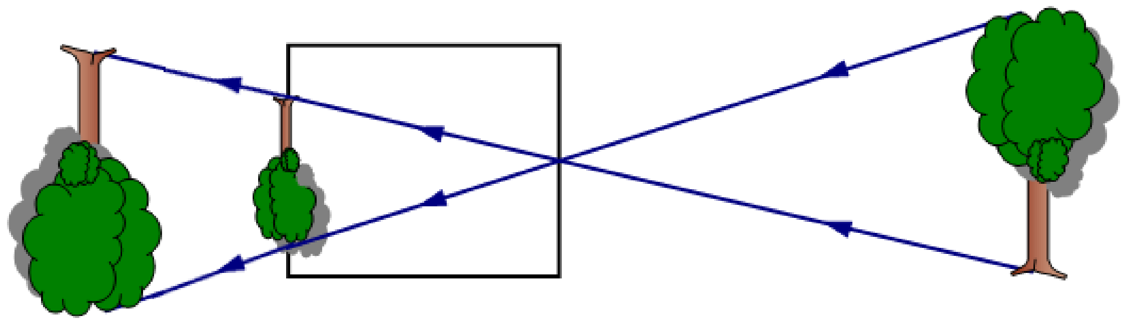

Figure 11: Pinhole model with two possible solutions

Extra steps must be taken to verify the object is in front of the imaging sensor or there will be erroneous results. Furthermore, the pinhole method does not compensate for any distortion of the lens. The undistorted points may be converted for distorted images by adjusting for the radial distortion of the image using the method created by Heikkila [22]. The calibration procedure yields a set of three distortion coefficients used to describe the radial distortion around the camera frame. The distortion coefficients and the method for distorting image points is shown in equations 12 and 13.

The general form for converting world coordinates to pinhole pixel coordinates is expressed in equation 8. The general form pinhole equation may also be illustrated using Figure 12.

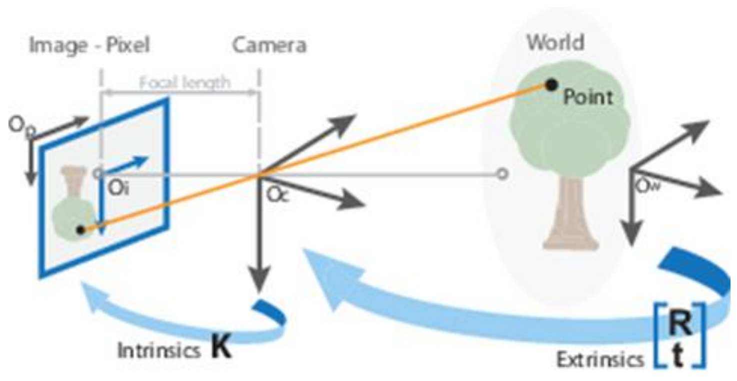

Figure 12: Rotation from World Coordinates to Pinhole Model [23]

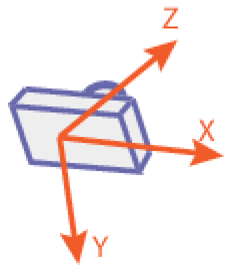

Figure 13: Right-Handed Z-Out Camera Frame [23]

$$w[\,u\ v\ 1\,] \;=\; [\,X\ Y\ Z\ 1]\begin{bmatrix} R_{Pixel}^{NED} \\ t \end{bmatrix} K \tag{9}$$

Where:

$$w = scale\ factor$$

$$u = horizontal\ image\ coordinate$$

$$v = vertical\ image\ coordinate$$

$$X, Y, Z = World\ Coordinates$$

$$R_{Pixel}^{NED} = 3x3\ Rotation\ Matrix$$

$$t = 1x3\ Translation\ Vector$$

The world coordinates $X, Y, and\ Z$ are in the camera frame shown in Figure 13. Image coordinates $u\ and\ v$ have an origin at the top left corner of the image, and therefore may only be between zero and the horizontal width and height of the image respectively. The pinhole coordinates u and v must be adjusted for distortion to be valid for most cameras. This may be accomplished by first normalizing the image coordinates, applying the distortion correction, and converting back to regular image coordinates. The normalizing equations are shown in equations 10 and 11. Using the normalized image coordinates the distortion correction can be calculated with equations 12 and 13. The resulting values x_{dis} and y_{dis} are then converted back to image coordinates using equations 10 and 11.

$$x = 2\frac{u}{w} - 1 \tag{10}$$

$$y = 2\frac{v}{h} - 1 \tag{11}$$

$$x_{dis} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{12}$$

$$y_{dis} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{13}$$

Where:

$$x, y = normalized\ coordinates$$

$$w, h = camera\ resolution\ width\ and\ height, respectively$$

$$x_{dis}, y_{dis} = distorted\ image\ coordinates$$

$$k_{1,2,3} = distortion\ coefficients$$

$$r = magnitude\ of\ x\ and\ y$$

This procedure is valid for any camera with known geometry and calibration. The calibration procedure can be accomplished with a printed checkerboard and with a software package such as OpenCV or MATLAB. The outputs from these software packages will be the distortion coefficients k and the 3x3 camera intrinsic matrix K. An

Equation 13 shows the translation and rotation to convert the target NED coordinates into target FRD coordinates with respect to the vehicle FRD frame. Equation 14 expands on equation 13 by translating the resultant by the FRD position of the camera, P_{FRD}, and rotating it into a Z-out camera-centered frame. The resultant, P_{cam}, can then be used as the world coordinates in Equation 9.

$$P_{FRD} = R_{FRD}^{NED}(T_{NED} - P_{NED}) \tag{14}$$

$$P_{CAM} = R_{CAM}^{FRD}(T_{FRD} - P_{FRD}) \tag{15}$$

Where:

$T = 1x3\ translation\ vector\ in\ respective\ frame$

$P = 1x3\ coordinate\ vector\ in\ respective\ frame$

The algorithm was also developed to be robust to cropped imaging sensors. Cropped sensors offset the region of interest as the camera intrinsic matrix is found using the full frame of the camera. This requires the region of interest coordinates to be translated from the full frame origin to the cropped frame origin. This can be performed by simply adding an offset value equal to the difference between origins in pixel coordinates. Some cameras may be set up in a way to crop the actual sensor in the camera firmware before streaming the images. This method is widely used as it allows for a higher framerate while still potentially providing the same areas of interest for processing. For cameras that do not use cropped frames, the offset values may be set to 0.

This process shows the general form rotation from NED coordinates to camera pixel coordinates. To generate an ROI, this process must be repeated 4 times for each corner position of the ROI. The bottom two coordinates of the ROI are extracted from the two outermost NED points from the LiDAR returns. The top two ROI coordinates are generated in the NED frame by projecting the two lower points upward by the known height of the classified object. This requires the object to be accurately classified, and the object height to be known beforehand. Lastly, to retain a rectangle shape, the ROI points are modified so that the largest extent of each side is taken. This correction is shown below in Figure 14.

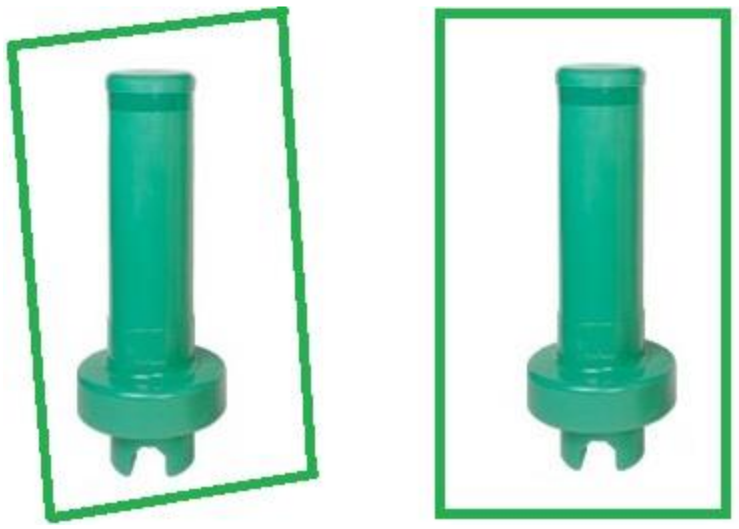

Figure 14: Left: Distorted ROI Right: ROI with using extent coordinates

Chapter IV
Results

4.1 Simulation Results

A simulation environment was initially used to validate the expected behavior of the occupancy grid implementation [5]. The simulation environment is built in MATLAB and uses CAD models of buoys and other waterborne objects to build an operating environment. Ray Tracing is then used to compute LiDAR returns from the faces of these objects. The simulation also allows for inserting the error characteristics of the platform's GPS/INS system and LiDAR sensors. In this way, LiDAR returns are accurately simulated as the vehicle navigates the simulated environment. Figure 15 shows the 3D environment consisting of obstacles and the ASV.

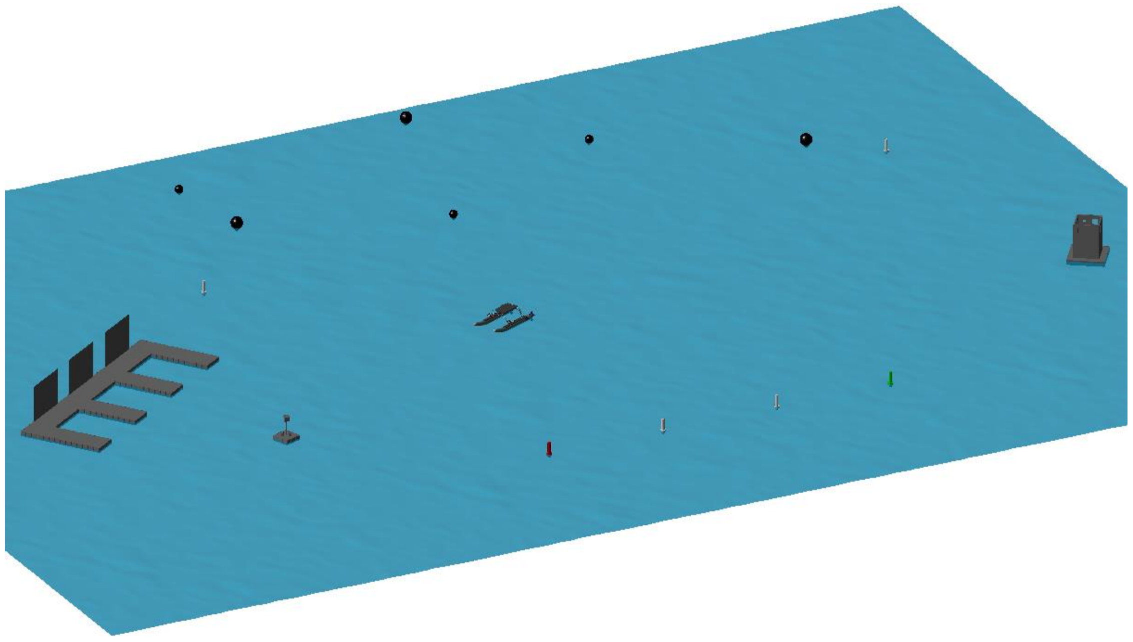

Figure 15: Simulation environment, showing buoys, a dock, and other large waterborne objects.

For this discussion, an arrangement of buoys, as well as some larger and more complex objects have been placed in the simulation. It is important to note that the image above corresponds to a breakpoint in the simulation after the vehicle has already mapped part of the environment. Thus, the current 2D occupancy grid, G, shown below in Figure 16, corresponds to LiDAR returns received over the past 20 scans (4 seconds with 5Hz scan rate) of simulated movement with a size of $d = 80m$, $h = 8m$, and $\delta = 0.1m$ in the presence of typical sensor error from a Velodyne LiDAR and the Torc PinPoint GPS/INS.

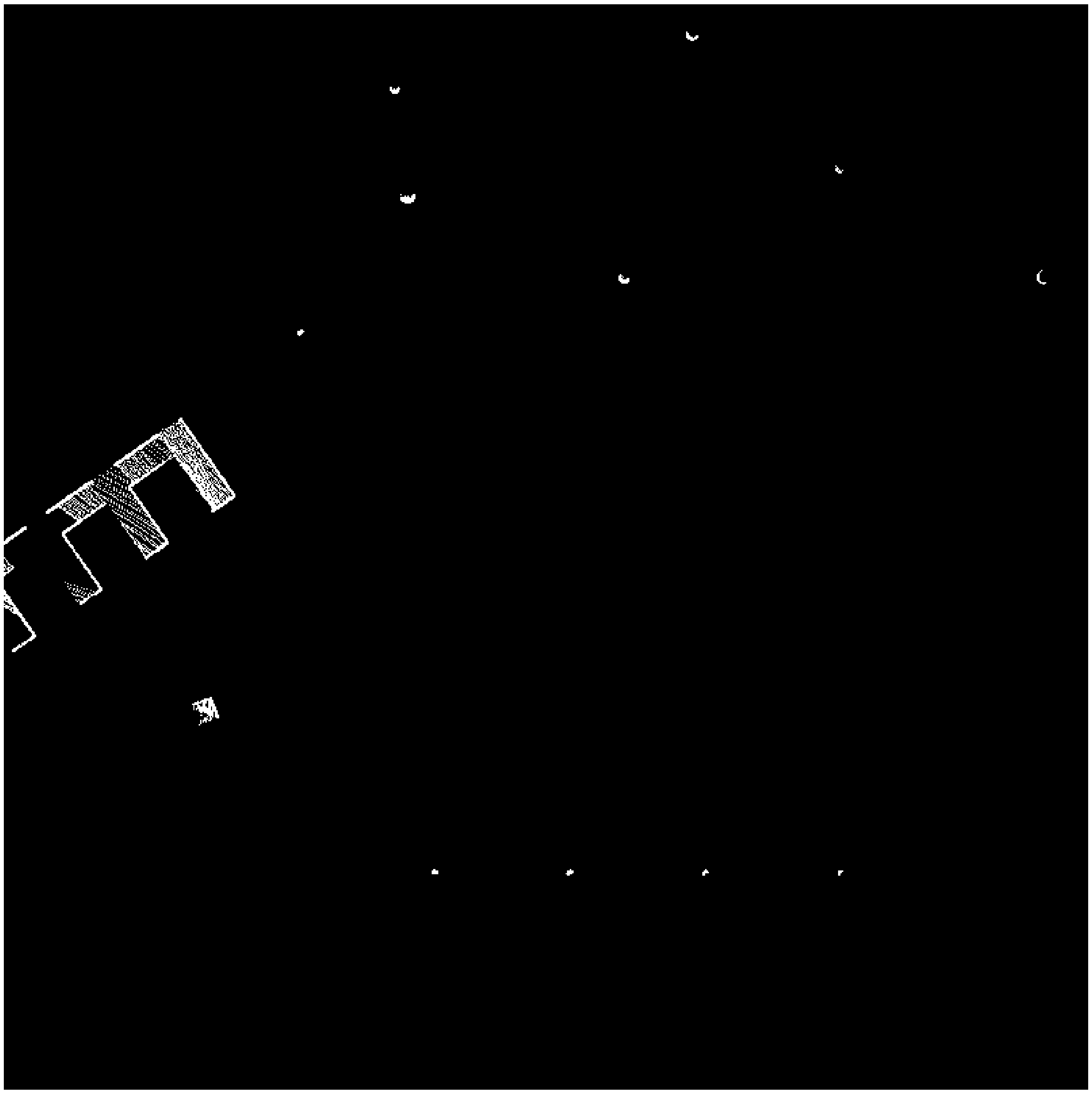

Figure 16: Occupancy Grid, G. This grid contains cells that have been filled over the previous 20 scans (4 secs with a 5Hz scan rate).

Figure 16 shows that part of all objects within range of the vehicle have been detected, but not all of the object has been filled in G. The grid G is then used to extract a set of polygon objects A, using [19], and the Douglas-Peucker point reduction method is applied subject to a 0.25m decimation tolerance. The resulting object list A is shown in Figure 17.

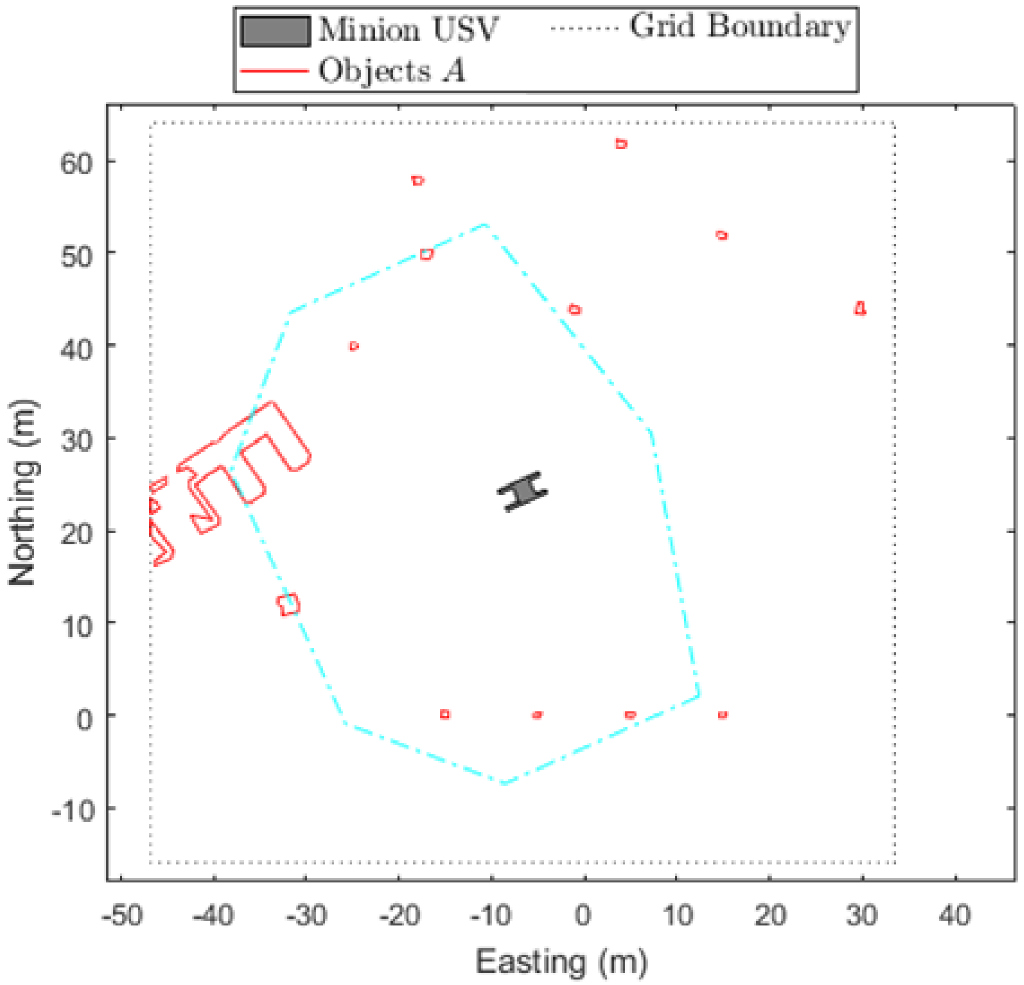

Figure 17: Objects A, after performing point reduction. The unlabeled cyan boundary indicates the region where there is enough detail to classify the objects as discussed in [24]. This is the same region explained with Figure 7.

These simulation results demonstrates the capability to effectively detect objects within the effective sensor range shown in the cyan boundary. This process was then validated with real world testing. Shown below in
Figure 18 is a representation result with labeled classification using colorization. Polygon representation is shown, and mapping accuracy is further explained in [24].

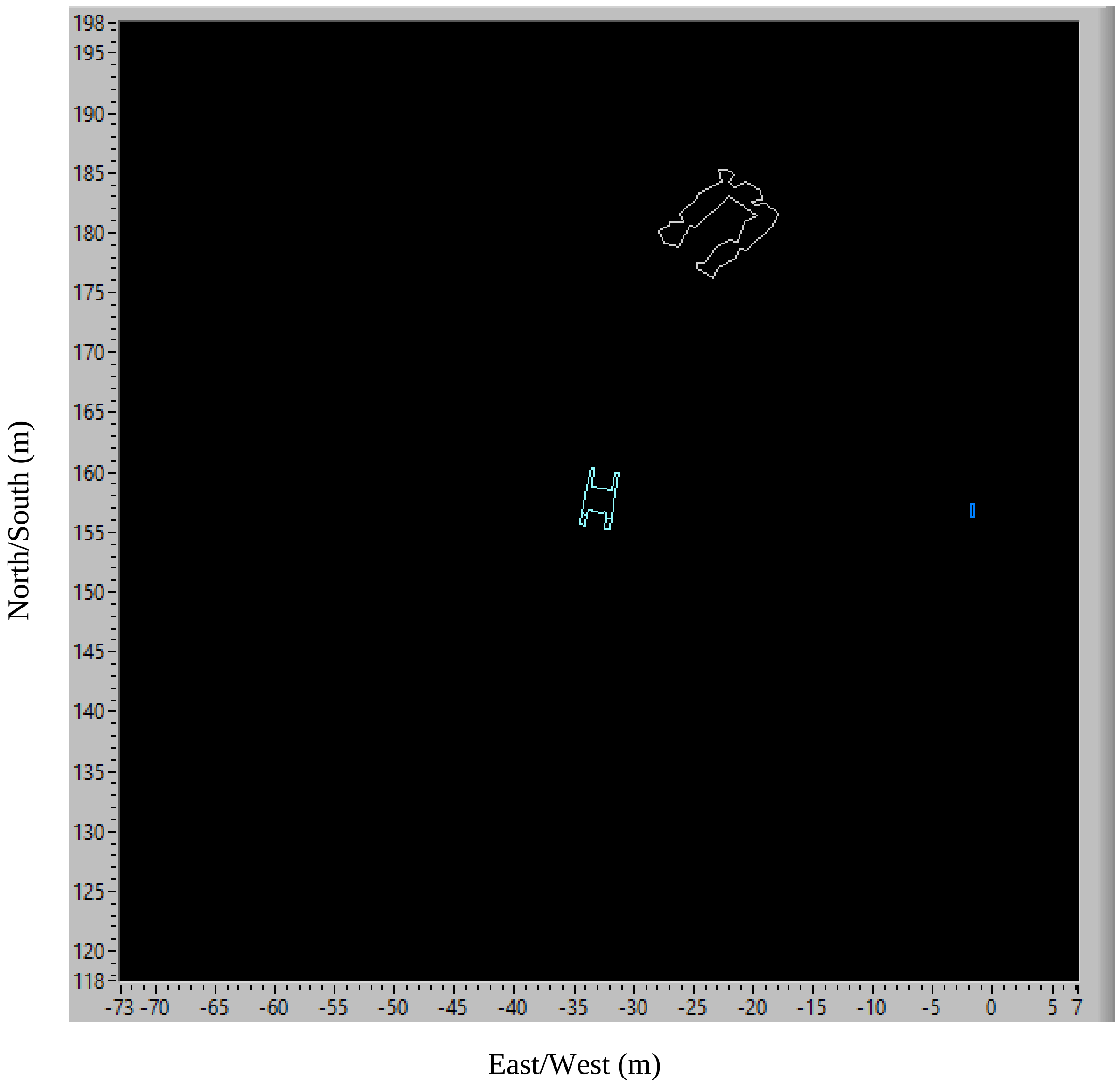

Figure 18: plot of the objects currently in the occupancy grid. The dock is shown as a gray polygon while a small A-3 buoy is shown as the blue polygon.

4.2 Classification

The SVM classifier was trained on 6 object classes, where each corresponds to an object from the RobotX competition. The object classes are Taylor Made Sur-Mark green, red and white can buoys, Polyform A-3 black buoys, Polyform A-7 black buoys [25], a dock, the competition light tower element, and the competition detect and deliver element. The objects are all seen pictured below in Figure 19, Figure 20, and Figure 21.

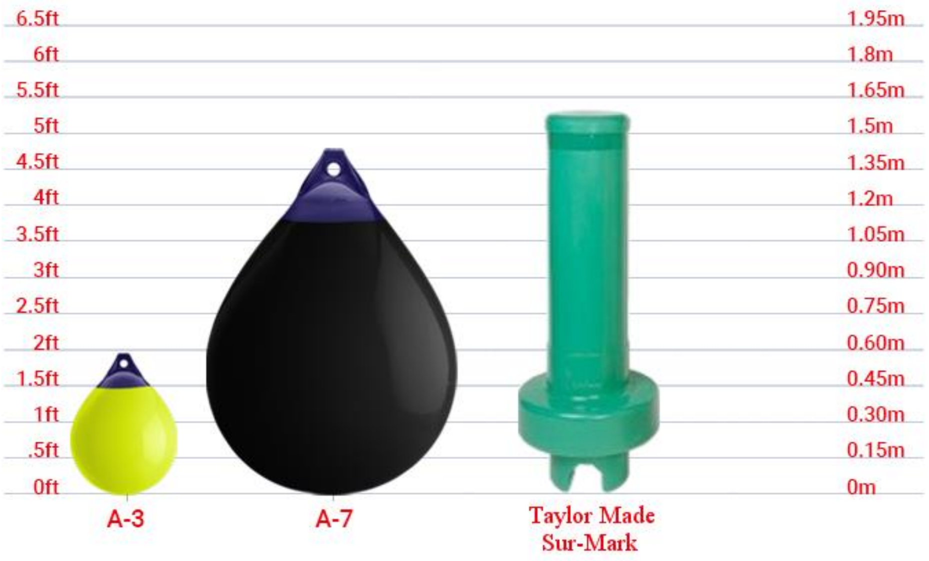

Figure 19: Left - Two Taylor Made Sur-Mark Can Buoys. Right - Polyform A-3 and A-7 Buoy [25]

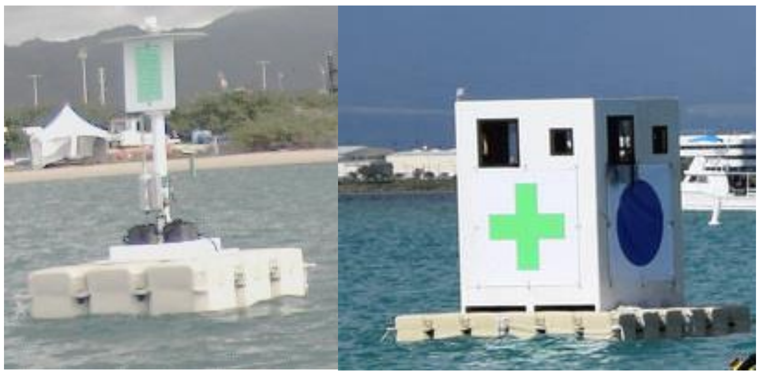

Figure 20: Left - Competition Light Tower. Right - Competition Detect and Deliver Target

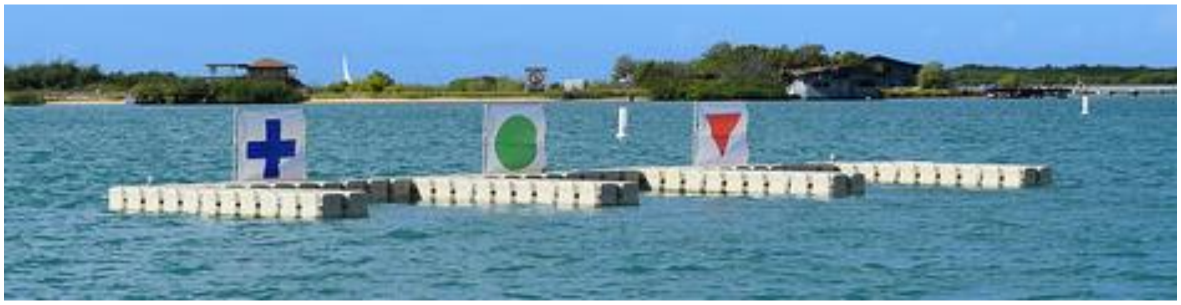

Figure 21: Competition Dock

The data was collected using the real-time supervised labeling approach described in Section 3.2. Utilizing the unique IDs of each object, the platform was able to perform complex maneuvers around the objects without losing the association between each object and its features. Unique IDs are created by tracking the extent of the object. Objects that retain their position within 0.5m retain their ID. This method permitted the capture of test data in as many operating conditions as possible. This includes driving towards and away from the objects, sitting stationary, strafing the object, or circling the object. The data was then sorted and saved to a text file so that it could be trained immediately. An example of this data is shown in Appendix A.

Given the C-SVC method of SVM has two parameters to tune, C and Gamma, a grid search method was applied to optimize the values for training. Using the full data set and 10-fold cross validation, the grid search method determined values for C and Gamma to be 32768 and 2, respectively. The data was then split into a test and train set by randomly sampling 20% of the training data from each class.

Prior work on this platform implemented a Multi Variate Gaussian (MVG) classifier for classifying two different objects, the Taylor Made Sur-Mark Can buoys and the Polyform A-3 buoys [25]. For comparison with SVM, a MVG classifier was also trained on the data set. Table 1 below shows the confusion matrix for a data

test set containing the 6 classes for the MRC competition. Table 2 below shows the confusion matrix for the MVG classifier on the same data set.

Table 3 below shows a comparison in classification accuracy between both classifiers as well as the mean classification percentage and misclassification percentage.

Table 1: Confusion matrix for SVM test set

		Actual Class					
		Taylor Made Can Buoy	Light Tower	Dock	Detect and Deliver Target	Polyform A-3	Polyform A-7
Predicted Class	Taylor Made Can Buoy	3082	0	0	0	27	0
	Light Tower	1	508	0	0	0	0
	Dock	0	0	58	0	0	0
	Detect and Deliver Target	0	0	0	83	0	0
	Polyform A-3	5	0	0	1	798	7
	Polyform A-7	0	0	0	0	5	87

Table 2: Confusion matrix for MVG test set

		Actual Class					
		Taylor Made Can Buoy	Light Tower	Dock	Detect and Deliver Target	Polyform A-3	Polyform A-7
Predicted Class	Taylor Made Can Buoy	2982	1	0	0	123	3
	Light Tower	2	507	0	0	0	0
	Dock	0	0	58	0	0	0
	Detect and Deliver Target	0	0	0	83	0	0
	Polyform A-3	30	0	3	0	733	45
	Polyform A-7	0	0	0	0	15	77

Table 3: Classification accuracies by class, mean classification, and misclassification rate

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

Except for the perfectly predicted classes: the dock and detect and deliver target, the SVM implementation has higher accuracy on each class than the MVG method. The largest improvement is seen with the Taylor Made A3 and A7 black buoys. This can most likely be attributed to the distribution of values for the intensity features, $F_{1 \ldots 4}$, being so similar between classes, but distinct from the other 4 classes. For example, the average intensity feature for A3 and A7 buoys are 12 and 14 respectively. However, the average intensity for an object such as the light tower is generally 27. Furthermore, these features exploit physical properties of the objects. Properties such as color and material that vary among other classes are identical for the A3 and A7 classes, adding to the confusion. Lastly, the confusion is increased for MVG because these features have a non-Gaussian distribution, inherently reducing the accuracy of a Gaussian classifier such as MVG. For example, shown below in Figure 22 is the minimum intensity attribute for all classes.

This distribution is biased heavily towards low intensity values and creates a non-gaussian distribution.

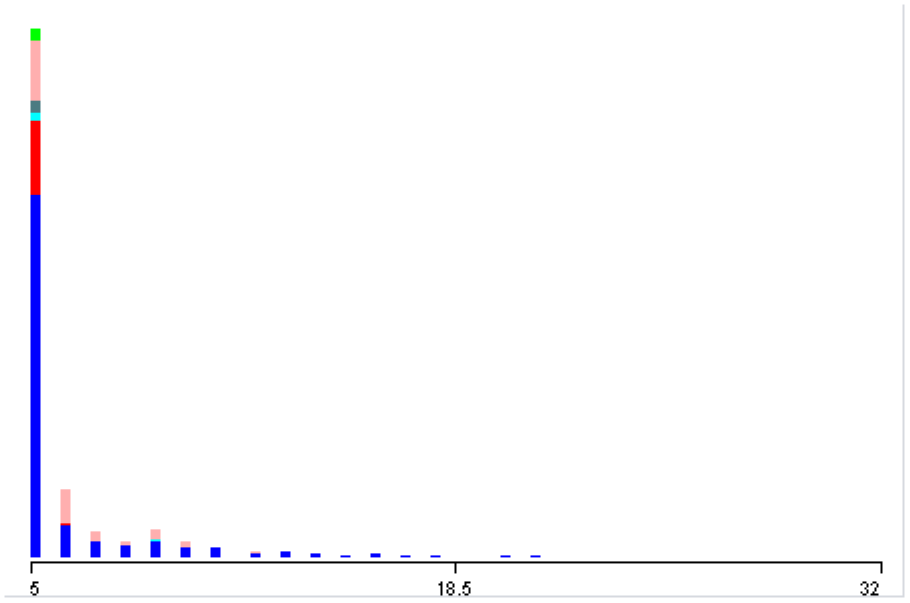

Figure 22: Minimum intensity for all classes. Blue - Can Buoy. Red - Light Tower. Teal - Dock. Gray - Raquetball Tower. Pink - A3 Black Buoy. Green - A7 Black Buoy

The most important value for comparison is the misclassification rate of both classifiers. Switching from MVG to SVM yields a 3.8% drop, or 3.9x improvement, in misclassifications. Since SVM have no Gaussian assumptions regarding the data, it's feasible to believe the difference in misclassification between classifiers will grow with an increase in the number of classes tested, especially if the classes have similar feature values. However, the SVM implementation developed does not account for unknown classes. As a result, all objects detected will result in a classification, even if the predicted class does not belong to any of the trained classes. The SVM classifier, however, does allow for an unknown class to be trained. To address this issue, a simple voting scheme was used. The voting scheme requires a certain number of the same classification before considering the object classified. While the object doesn't have enough votes, its class remains unknown. Future work should also account for unknown objects such as moving objects or the shoreline.

For visualization, a 2D plot was used, which displays polygons for all currently detected or visible objects. These polygons are then color coded according to predicted class. Figure 23 shows the 2D plot populated with some classified and unclassified objects. Polygons shown in red represent Taylor Made Sur-Mark Can buoys. The dock is shown as the yellow object, the light tower is shown in the image as a pink square, and the smaller Polyform A-3 buoys are shown in blue.

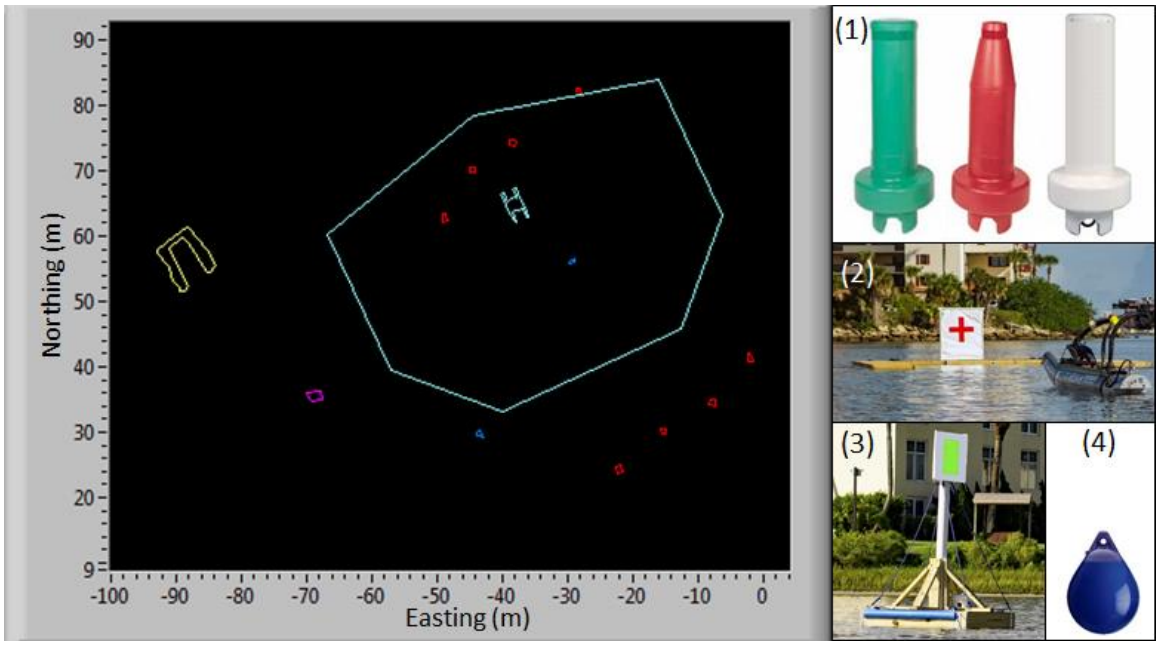

Figure 23: 2D plot showing all objects detected by the vehicle. 1) Taylor Made Sur-Mark Can buoys - red 2) dock – yellow 3) light tower - pink 4) Polyform A-3 buoys - blue

4.3 Camera Sensor Fusion

The camera sensor fusion application performed sufficiently for extracting regions of interest. An example region of interest generated for the dock is shown below Figure 24 and an example ROI for a tall buoy is shown in Figure 25. This generated region is padded by 100 pixels in each direction to account for measurement error. Unfortunately, the accuracy of this method is entirely dependent on the geometric measurements of the LiDAR and camera sensors.

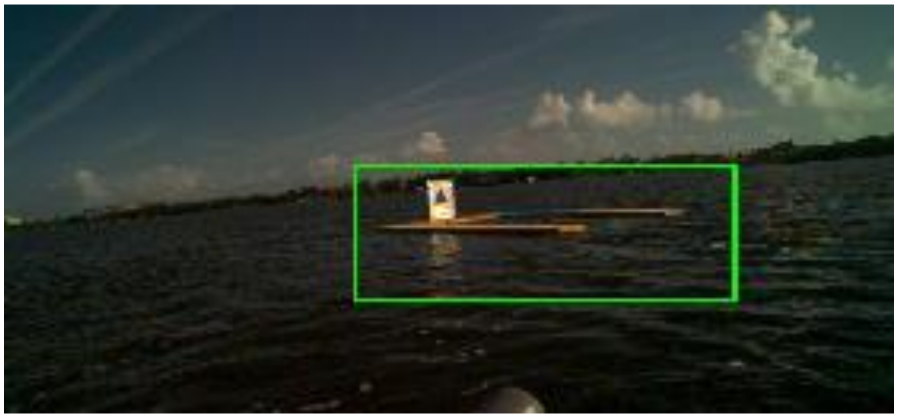

Figure 24: ROI of dock object from Minions port camera

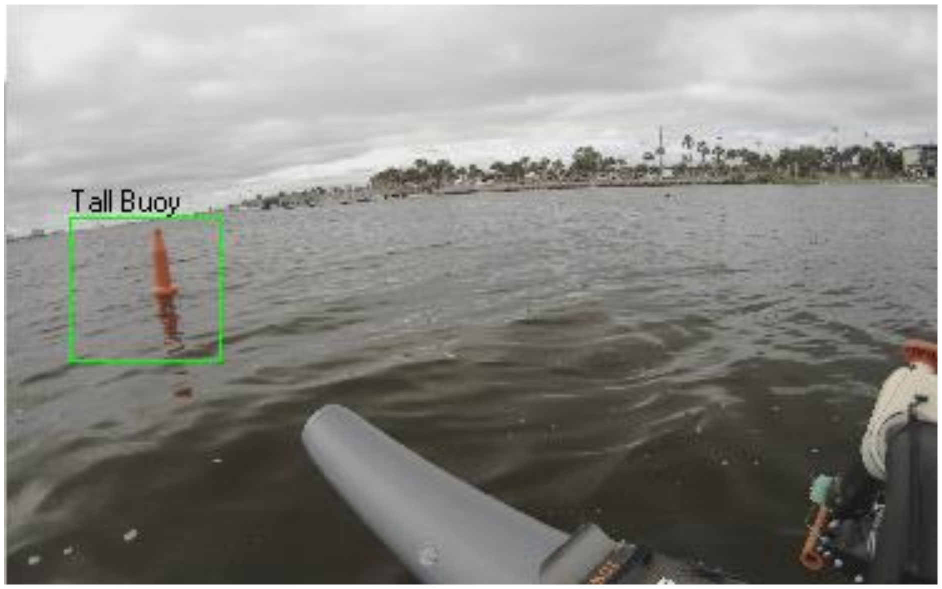

Figure 25: Tall Buoy classification and ROI from Minion's starboard camera

This error may be demonstrated by tracking an object in the camera at varying distances and angles. In both frames there is 0 padding applied to the image. The buoy is measured to be 1.5m tall. When the buoy is moved to a distance 20m from the boat, the

ROI has drifted halfway up the buoy, resulting in 1.7 degree. This error can be attributed to a measurement error in the camera and LiDAR geometry. Moving the buoy to the side of the frame was then used to measure the left-right error of the approach. Estimating the frame to being 0.7m from the center of the buoy results in a 1.4 degree error. Further work can be done to automatically adjust the ROI padding based on the distance from the object. As shown in Figure 26 padding is not necessary for objects located at the center of the frame and minimal distance. Though as the distance increases, the <2 degree error in measurement begins to accrue and padding becomes more necessary.

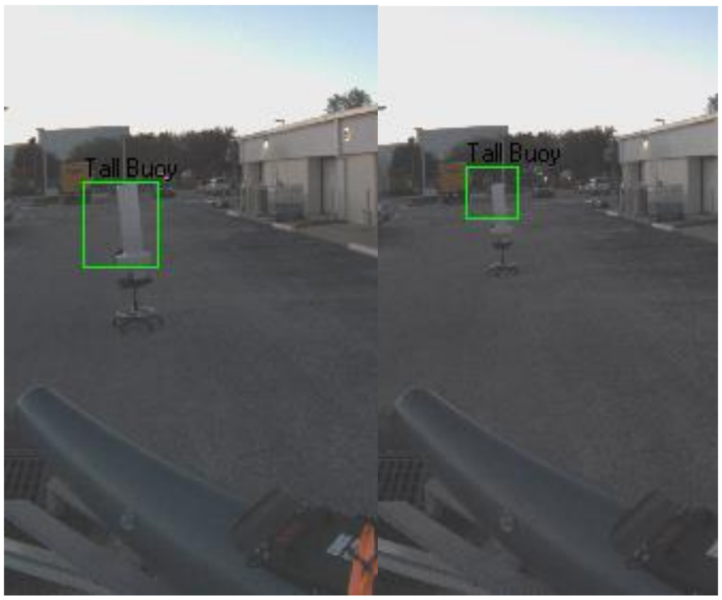

Figure 26: Left: Close object in center of frame. Right: Object further away

Chapter V
Discussion, Conclusions, and Recommendations

The methods shown above provide a highly effective application of LiDAR sensors in maritime environments for object detection, classification, and camera sensor fusion for vision-based object extraction using regions of interest. Utilizing a GPS/INS, four LiDAR sensors, and a pair of cameras, the minion platform could accurately detect and classify objects. It is shown that LiDAR returns may successfully be fit to a 2D occupancy grid with 20cm cells, allowing for object extraction through image operations. A 3D grid is retained and stores spatial information for the extracted objects for use in classification.

Object classification with a Support Vector Machine (SVM) classifier yielded a 98.7% mean accuracy over 6 object classes. This method improved on the 94.9% accuracy of Multi-variate Gaussian (MVG) as several features such as object intensity had non-Gaussian distributions. This is shown more strongly with objects such as the Polyform A-3 and A-7 buoy where SVM had a 98.4% and 94.6% accuracy while MVG had a 90.4 and 83.7% accuracy, respectively. Despite the high accuracy of the SVM classifier, an unknown class should be trained alongside the object classes to lower the false-positive classifications of objects not belonging to the 6 trained classes. Furthermore, moving objects cannot be classified at all due to the restrictions of the current occupancy grid application.

Using the geometric properties of the LiDAR and camera sensors, a region of interest was extracted from the camera using LiDAR returns. This method proved accurate with a <2 degree angular error on the 3D rotations. However, when reaching distances of 20m, this error results in regions that have 0.6m uncertainty. As a result, region bounds were expanded by 100 pixels for all frames to correspond with 2 degrees of error at 20m. While successful for this application, this approach may be improved by dynamically adjusted the padding based on object distance.

This combination of methods greatly improves the perception capability of an ASV like Minion. By accurately detecting and classifying objects, an ASV may make more intelligent decisions based on its improved environmental awareness. Future work may further improve these capabilities by retraining the SVM classifier for an unknown class, as well as moving objects, dynamically adjusting the camera region of interest based on distance, or decreasing the measurement error of the sensor positions.

References

[1] U.S. Department of Homeland Security, *Navigation Rules, International – Inland* [online] Available: https://www.navcen.uscg.gov/pdf/navrules/navrules.pdf

[2] U.S. Navy, The Navy Unmanned Surface Vehicle (USV) Master Plan, 2007 [online] available: http://www.navy.mil/navydata/technology/usvmppr.pdf

[3] Hsu, Jeremy, *IEEE Spectrum,* "U.S. Navy's Drone Boat Swarm Practices Harbor Defense", 2016 [online] available: http://spectrum.ieee.org/automaton/robotics/military-robots/navy-drone-boat-swarm-practices-harbor-defense

[4] Rødseth, Ørnulf Jan & Burmeister, Hans-Christoph, Developments toward the unmanned ship [online] available: http://www.unmanned-ship.org/munin/wp-content/uploads/2012/08/R%C3%B8dseth-Burmeister-2012-Developments-toward-the-unmanned-ship.pdf

[5] Hockley, Christopher J, & Zuercher, Timothy A., Design of the Minion Research Platform for the 2016 Maritime RobotX Challenge, unpublished, 2016

[6] "Taylor Made Products Sur-Mark Can Buoy (Green)," Amazon.com [online] available: http://www.amazon.com/Taylor-Made-Products-Sur-Mark-Green/dp/B000MUD4TW

[7] *2016 Maritime RobotX Challenge Task Descriptions (rev 1.0),* 2016 [online] available: http://robotx.org/images/files/2016-MRC-Tasks-2016-11-28.pdf

[8] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, et. al., "Autonomous Driving in Urban Environments: Boss and the Urban Challenge," Journal of Field Robotics, vol. 25, Issue 8, pp. 425-466, August 2008.

[9] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, et. al., "Junior: The Stanford Entry in the Urban Challenge," Journal of Field Robotics, vol. 25, Issue 9, pp. 569-597, September 2008.

[10] An automatic COLREGs-compliant obstacle avoidance system for an unmanned surface vehicle

[11] G. Papadopoulos, H. Kurniawati, A. Shariff, L.J. Wong, and N.M. Patrikalakis "3D-Surface Reconstruction for Partially Submerged Marine Structures Using an Autonomous Surface Vehicle," 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, September 2011.

[12] J.M. Esposito, and M. Graves, "An Algorithm to Identify Docking Locations for Autonomous Surface Vessels from 3-D LiDAR Scans," 2014 IEEE International

Conference on Technologies for Practical Robot Applications (TePRA), April 2014.

[13] Z. Shen, “New approach of imagery generation and target recognition based on 3D LIDAR data,” in *International Conf. on Electronic Measurements & Instruments*, 2009, pp. 612-616

[14] C.L. Kennedy, “Development of an Exteroceptive Sensor Suite on Unmanned Surface Vessels for Real-Time Classification of Navigational Markers,” M.S. thesis, Dept. Mech. Eng., Embry-Riddle Aero. Univ., Daytona Beach, FL, 2014

[15] C.C. Chang, and C.J. Lin, LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2011.

[16] C.W Hsu, C.C Change, and C.J. Lin, CA Practical Guide to Support Vector Classification, 2003 [online] available: https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

[17] C.W. Hsu, C.J. Lin, A Comparison of Methods for Multi-class Support Vector Machines, [online] available: https://www.csie.ntu.edu.tw/~cjlin/papers/multisvm.pdf

[18] VLP-16 User’s Manual and Programming Guide, Velodyne Acoustics, Inc. Rev A, August 2015. [online] available: http://velodynelidar.com/docs/manuals/VLP-16%20User%20Manual%20and%20Programming%20Guide%2063-9243%20Rev%20A.pdf

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

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

[21] Z. Zhang, A Flexible New Technique for Camera Calibration, in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, issue: 11, November 2000

[22] J. Heikkila, and O. Silven, A Four-step Camera Calibration Procedure with Implicit Image Correction, in *Computer Vision and Pattern Recognition*, 1997

[23] What is Camera Calibration?, Mathworks, [online] available: https://www.mathworks.com/help/vision/ug/camera-calibration.html

[24] D. Thompson, E. Coyle, and J. Brown, “Efficient LiDAR Based Object Segmentation and Mapping for Maritime Environments”, unpublished, 2017.

[25] “Polyform US A-Series Buoys” [online] available: https://www.polyformus.com/buoys/a-series

Appendix A
Training Data Format

```
1 1:0.136842 2:0.000000 3:0.193578 4:0.184768 5:0.532258 6:0.005639 7:0.018220
8:0.008790 9:0.001596 10:0.001225
1 1:0.252632 2:0.000000 3:0.256307 4:0.210891 5:0.500000 6:0.006031 7:0.017081
8:0.008581 9:0.002713 10:0.000629
1 1:0.226316 2:0.000000 3:0.248471 4:0.152998 5:0.387097 6:0.008224 7:0.014569
8:0.007116 9:0.002138 10:0.000344
1 1:0.594737 2:0.000000 3:0.182358 4:0.453672 5:0.209677 6:0.014098 7:0.021466
8:0.012871 9:0.002188 10:0.001176
1 1:0.621053 2:0.000000 3:0.305616 4:0.653677 5:0.177419 6:0.012061 7:0.018349
8:0.009209 9:0.002118 10:0.000789
1 1:0.231579 2:0.000000 3:0.320813 4:0.222136 5:0.225806 6:0.018484 7:0.018141
8:0.009209 9:0.004112 10:0.000591
1 1:0.263158 2:0.000000 3:0.187368 4:0.174438 5:0.225806 6:0.015038 7:0.030173
8:0.016011 9:0.002112 10:0.002512
1 1:0.052632 2:0.333333 3:0.214067 4:0.011232 5:0.064516 6:0.000705 7:0.011960
8:0.005546 9:0.001111 10:0.000530
1 1:0.242105 2:0.148148 3:0.327313 4:0.207630 5:0.161290 6:0.003524 7:0.009213
8:0.004395 9:0.001172 10:0.000292
2 1:0.500000 2:0.000000 3:0.316719 4:0.315281 5:0.451613 6:0.057566 7:0.057390
8:0.049184 9:0.018514 10:0.004243
2 1:0.505263 2:0.000000 3:0.275936 4:0.361927 5:0.403226 6:0.093828 7:0.068623
8:0.061741 9:0.026145 10:0.004846
2 1:0.221053 2:0.000000 3:0.180142 4:0.188836 5:0.467742 6:0.049185 7:0.071061
8:0.070427 9:0.027718 10:0.005399
2 1:0.500000 2:0.000000 3:0.294698 4:0.311716 5:0.354839 6:0.071272 7:0.066605
8:0.064881 9:0.024655 10:0.004343
3 1:0.978947 2:0.000000 3:0.220477 4:0.255523 5:0.580645 6:0.883929 7:0.973427
8:0.967141 9:0.981354 10:0.939817
3 1:0.010526 2:0.000000 3:0.022936 4:0.022812 5:0.483871 6:0.000000 7:0.016254
8:0.006907 9:0.002025 10:0.003849
3 1:0.031579 2:0.074074 3:0.076453 4:0.052683 5:0.387097 6:0.000000 7:0.028474
8:0.011720 9:0.000000 10:0.011267
4 1:0.463158 2:0.000000 3:0.387729 4:0.342916 5:0.548387 6:0.444784 7:0.159835
8:0.218606 9:0.104441 10:0.027274
4 1:0.500000 2:0.000000 3:0.411881 4:0.326719 5:0.596774 6:0.474232 7:0.157161
8:0.255128 9:0.108393 10:0.028666
4 1:0.536842 2:0.000000 3:0.587477 4:0.523042 5:0.564516 6:0.464364 7:0.162012
8:0.225408 9:0.087787 10:0.02697
```

Appendix B
Project Dependencies

Project Dependencies:

- Labview 2016 64-bit
    - http://www.ni.com/download/labview-development-system-2016/6055/en/
- Labview Vision Add-ons 2016
    - http://www.ni.com/download/vision-development-module-2016/6304/en/
    - http://www.ni.com/download/ni-vision-acquisition-software-september-2016/6422/en/
- Labview Machine Learning Toolbox (MLT)
    - https://forums.ni.com/t5/NI-Labs-Toolkits/LabVIEW-Machine-Learning-Toolkit/ta-p/3514074
- LibSVM for Labview v1.1
    - https://github.com/oysstu/LabVIEW-libsvm