École Polytechnique Fédérale de Lausanne

HydroContest CVLab

Semester Project - Fall 2016

Real Time Obstacle and Competitor Detection in a Maritime Scene

Author:

Roger Fong
January 13, 2017

Abstract

This report provides a new computer vision based method that uses a combination of existing background modeling and salient object detection techniques to locate and place bounding boxes around objects of interest in a maritime scene. Detection is to take place during the annual HydroContest Competition where objects of interest consist of obstacles in the water such as buoys and competitor boats. The detection algorithm is intended to aid in future efforts to provide automatic collision avoidance and path planning functionality to the platform upon which the main camera is mounted.

1 Introduction

The goal of this project is to work with the EPFL HydroContest team to provide an on-board camera based vision system for their competition boats that will automatically detect obstacles and other competitors and assist in collision avoidance. In order to save weight, we would only like to use existing sensing hardware on the boat to provide detection, namely the existing on-board front facing camera. The project will in total span two semesters. This first part of the project will focus on my exploration of image based recognition of obstacles. In particular strategies relating to edge detection, dynamic textures, object salience and probabilistic models were tried. We highlight a number of these methods in this paper. In the end a combination of two different models were chosen, *Real-Time Salient Object Detection with a Minimum Spanning Tree*[18] and *Fast Image-Based Obstacle Detection From Unmanned Surface Vehicles*[10]. It will be shown that while both of these methods separately offer comparable detection results, a combination of the two can offer significant improvements. Another goal of the solution is to have a relatively high frame rate, as it would ideally be run fairly often (though not necessarily every frame) on the boat.

2 Related Work

There is a wide breadth of possibilities and angles from which we can approach this problem of determining presence of boats and obstacles in the water. In investigating relevant research however there are a number of con-

straints on our instance of the problem that make many existing methods not viable.

2.1 Problem Constraints/Difficulties

To begin with we go over some of the difficulties of the problem at hand. Many papers assume that we are using overhead imagery [[2], [4], [7], [9], [12], [16]]. This assumption greatly reduces the difficulty in determining the presence of boats. This is because all visible points are at roughly the same depth from overhead image acquisition platform (say from a satellite). Thus boats of equal sizes appear as the same size in all points of the image and water does not change in visual quality across the image. This allows methods to develop very accurate models for boats, obstacles and water due to their more consistent nature. This is not the case with an on-board camera in which ships farther away of course look smaller and water very close to the camera will have to deal with issues of reflections in the water. Indeed dynamic texture qualities of water change in the scene as well as we approach the horizon line.

Many proposals assume a stationary camera [[2], [3], [9], [15]]. This also greatly simplifies the problem as typically the boundaries of the water area do not have to be determined on the fly. It is much easier to establish a background model if you know that that the background does not change too much. A stable scene also allows easier use of techniques that use optical flow. In a non-fixed camera scene, one would have to compensate for the global motion, which is doable but not quite as simple on dynamic texture such as water which is also moving randomly and chaotically. This is especially for optical flow methods that track feature points across frames as feature points such as points near the crest of a wave, can disappear. Below I survey some of the more promising approaches, a number of which were implemented and tested.

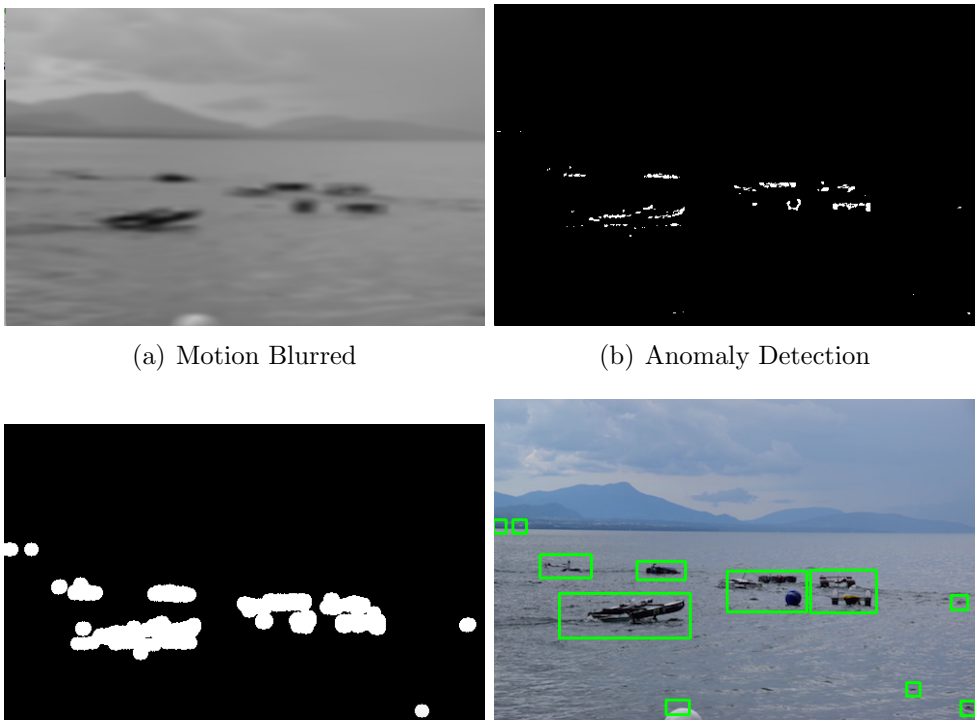
2.2 Edge based methods

One idea is that the edges of a non-water obstacles are much more distinct from the noisy edges of the water and therefore with careful tuning a method can be found that outputs edges found of just these obstacles. Knowing locations of edges in a binary map over the image, we can cluster edge pixels together or just use morphology operations to dilate the edges to find regions

where an obstacle is likely to exist. Another advantage of these methods is that they tend to have very fast performance as they rely primarily on simple global filtering options. However we will also find that it is this very simple global nature that also results in methods that are not very adaptable and are too sensitive to parameter tuning.

2.2.1 Motion blur based anomaly detection

The method in *Anomaly based vessel detection in visible and infrared images* [8] relies on the assumption that points on the edges of obstacles will be much more affected by an artificial linear motion blur than edges found in background or water areas. A fairly successful application of the method can be seen in figure 1, excusing a number of small bounding boxes where the method did not manage to completely filter out noise from the water:



(a) Motion Blurred

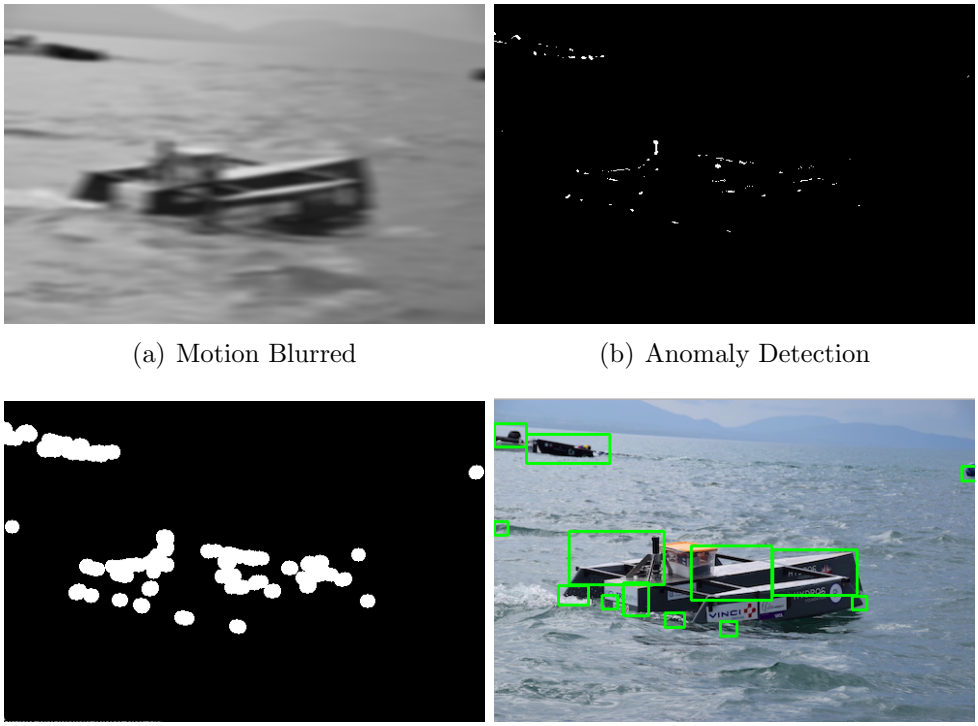
(b) Anomaly Detection

(c) Dilating Anomaly Detection Results

(d) Finding bounding boxes

Figure 1: Decent results for the motion blur anomaly detection method

The weaknesses of this method and indeed all purely edge detection baesd methods is that when a boat is too close to the camera and only edges are detected, and the surface of a non textured boat will not appear in the detection. In this method in particular, only certain pixels on the edge that are considered as anomalies are detected in order to minimize the number of water edges detected, which means a boat that is passing close to the camera would likely be split into multiple boat detections as can be seen in figure 2 if the anomalous points are too sparse. This is not ideal in our application as the boat will be involved in a race, so obstacles and competitors will likely be fairly close to our boat. This is not a problem when a boat is farther away since the anomalous edge pixels will be closer together and with some amount of dilation/clustering, blobs representing whole obstacles in the water can be found.



(a) Motion Blurred

(b) Anomaly Detection

(c) Dilating Anomaly Detection Results

(d) Finding bounding boxes: sparse anomaly map results in too many boxes

Figure 2: Poor results for the motion blur anomaly detection method

2.2.2 Frei Chen edge filter

This Frei-chen[5] edge detecion method is actually a very old edge detection method (from 1976) that relies on running a series of different filter kernels of the image. The method tries to ignore edges that are part of surfaces that appear more textured. The general idea is that the method is more robust to noisy textured areas such as water. In our findings we find that in combination with some amount of blurring as pre-processing, the filter does indeed do a much better job at filtering out edges caused by water while retaining most of the edges from the boats and obstacles as compared to a canny edge detection, as can be seen below3.

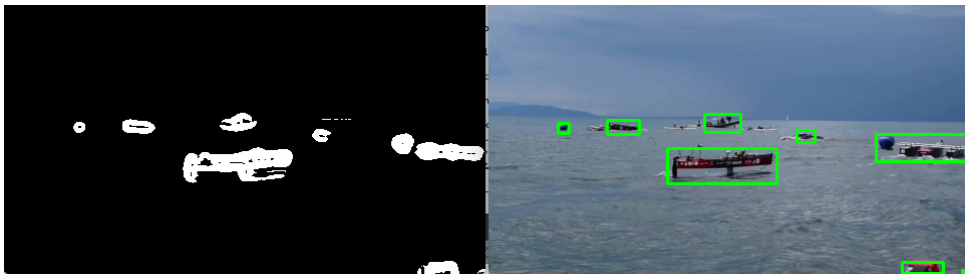


Figure 3: Water is successfully removed by the Frei-Chen filter while obstacles remain

However, it still does not manage to eliminate enough of the edges from the water and the results are very dependent on the parameters of the initial blur. The method especially has difficulty when dealing with water that is closer to our view point, and therefore may appear less like a texture and more like distinct objects, see figure 4. The number of water edges are certainly reduced but not to the desired level and in many cases any further pre-processing with blurring would also eliminate the edges of obstacles such as the buoys and boats. The problem with these methods is that the structural elements of objects in the scene are not taken into account. For example, edges from the water could be ignored as the often times do not form closed contours. These edge detection methods merely perform global image processing and therefore cannot hope to take advantage of underlying structural information in the image.

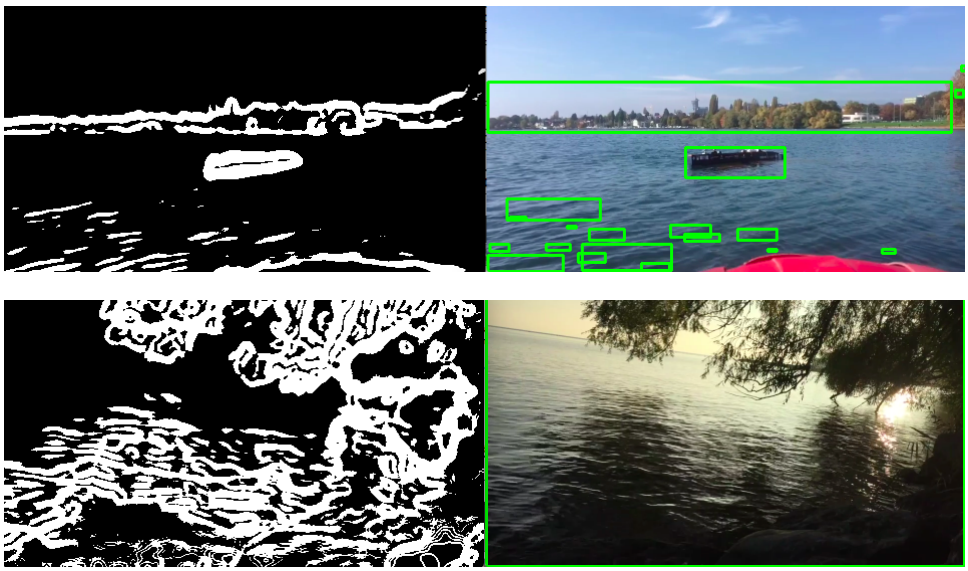


Figure 4: Frei-Chen having problems filtering out highly reflective areas of water and areas closer to the boat

2.3 Dynamic Textures

2.3.1 Motivation

Dynamic textures are areas in your scene that can be considered to be a single texture that changes over time. Examples of dynamic textures can include things such as a field of grass, smoke or fire and of course, water. For our purposes the idea is that if we can detect the water as a dynamic texture, then anything contained within that does not belong to said texture will be a boat or obstacle. Capturing the dynamic quality of water is an appealing solution as doing so what allow us to ignore issues of reflections in the water or specular highlights from the glare of the sun, which in many color or contrast based models would be detected as an intruding obstacle, as can be seen in figure 35.

2.3.2 Original Approach

Among the various methods proposed for dynamic texture segmentation, many involve using optical flow. A variant of the method from *Water detection with segmentation guided dynamic texture recognition*[15] was imple-

mented and from it can be observed that such techniques may not be so applicable to our situation. The original paper uses the Lucas Kanade optical flow method and tracks the path of feature points in the water. By appending the optical flow vectors for said feature points across multiple frames, a containing circle for the optical flow path is found, the diameter of which can be used in the measure of entropy for that point. With enough feature points, the water area can be determined under the assumption that areas of the image containing water have a much higher density of pixels marked as water than those that do not. However, tracking the path of certain feature points does not make sense in a rapidly changing scene such as the one viewed from the boat. Our feature points can quickly disappear from our view which would result a lot of uncertainty near the edges of the scene.

2.3.3 Modified Approach

We modify the algorithm by simply measuring the entropy at a pixel as the sum of the angle changes of the optical flow vector across multiple frames. We use dense optical flow so that a flow vector exists at each pixel. While the pixels identified as water are roughly in the correct region of the image (figure 5) the density of correct identifications is not nearly high enough. The results could be blurred or dilated in a post processing step to fill in the gaps, but the density is so low that that un-marked obstacle areas would also be obscured by such steps (figure 6). A segmentation scheme is provided such that the image is segmented into regions based on visual quality and an entire region is labeled as water if more than half of the pixels in said region have been detected as water[15] though it was not implemented due to other weaknesses of the approach. Namely, it does not work so well with water that is closer to the horizon line as it does not appear to be in motion at such a distance. Additionally the method is computationally expensive even at lower resolutions since it relies on a dense optical flow.

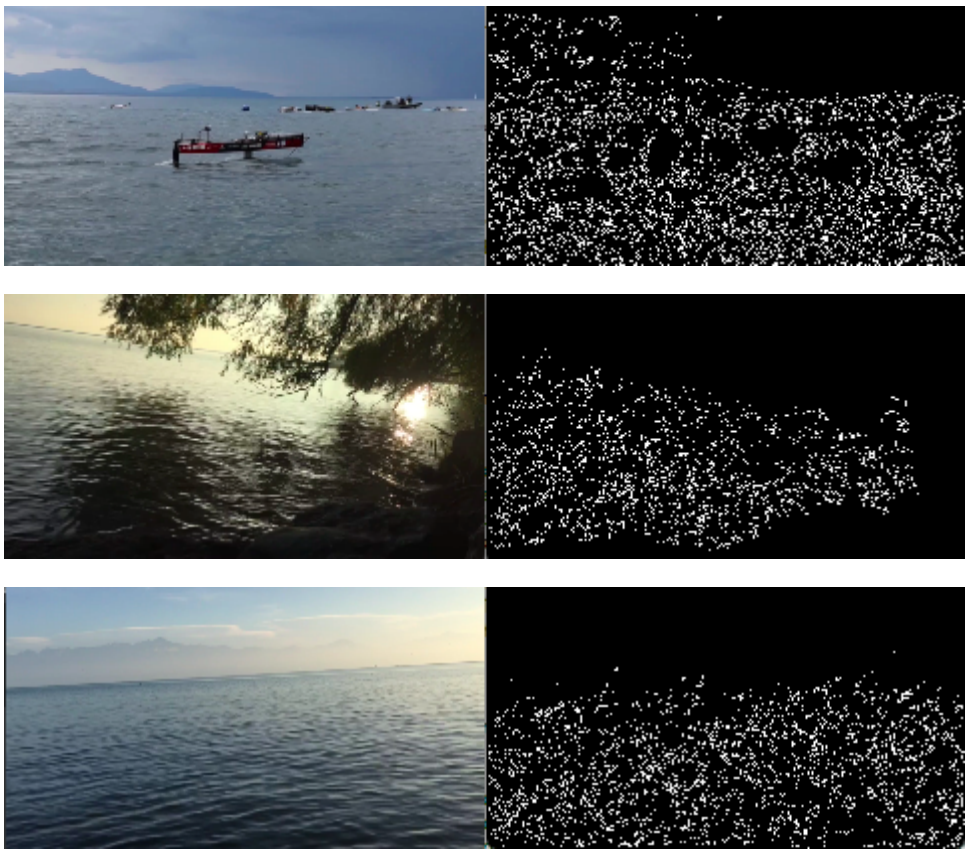


Figure 5: Map of pixels marked as water by dynamic texture method

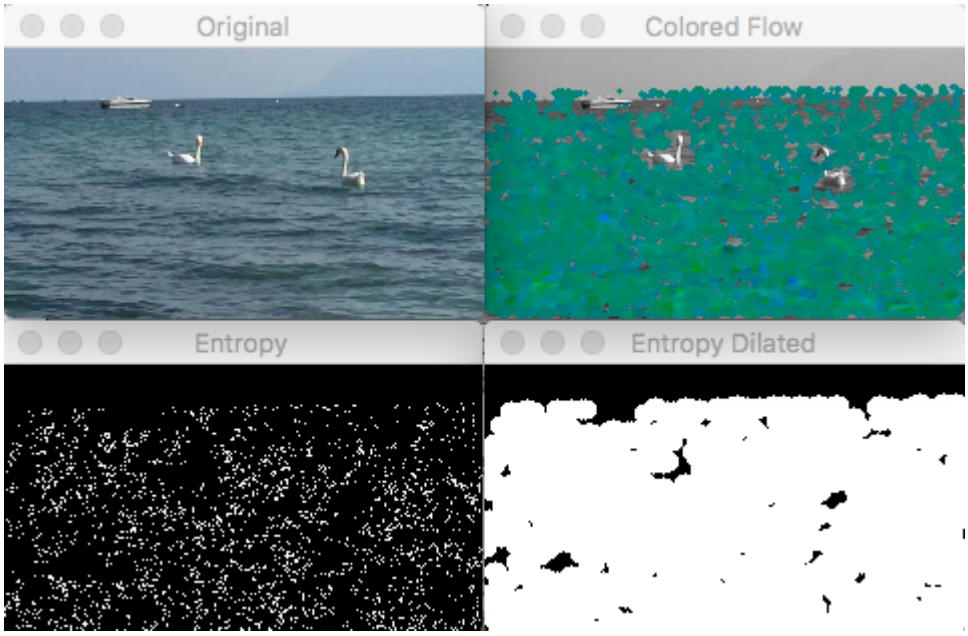


Figure 6: Density of water pixels is too low, swans are obscured in the process of dilation, the segmentation scheme in [15] may have better results

2.4 Machine Learning methods

Many machine learning methods have also been proposed for this task [[2], [7], [12], [14], [16]]. They generally involve either learning the appearance of boats (if the goal is boat detection), of water (if the goal is obstacle in water detection), or of both.

2.4.1 Learning Boat Models

Learning the appearance of boats is a tricky issue for us since the boats in the competition hardly look like ordinary boats 7, which means our training data is limited to whatever pictures were taken during competition. As it turns out this was not many. There are methods such as domain adaptation as proposed in *Simultaneous Deep Transfer Across Domains and Tasks*[19] though they were not explored in this paper in favor of less data driven methods.

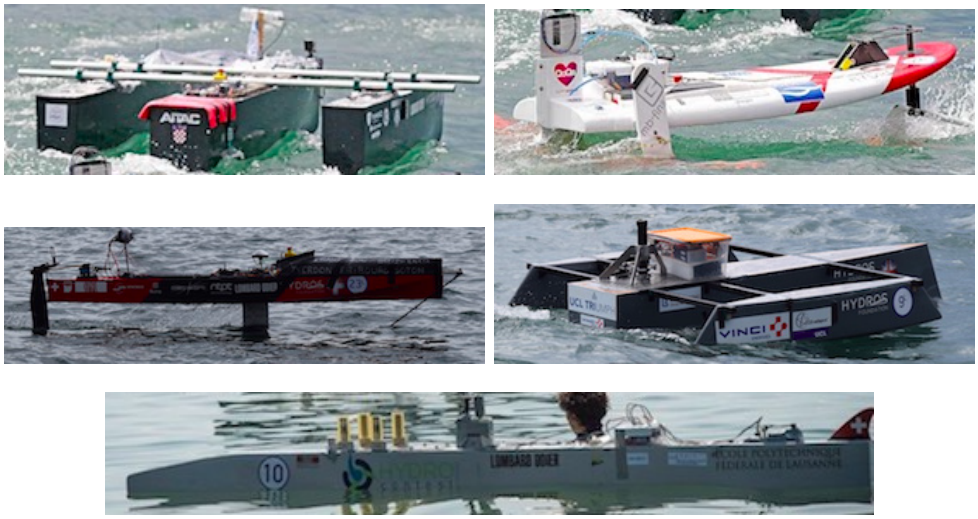


Figure 7: Examples of "boats" in the race

2.4.2 Learning water models

The method in *Water Region Detection Supporting Ship Identification in Port Surveillance*[2] proposes an SVM based model that learns whether or not each pixel should be classified as water based on a pixel's neighborhood. The method seems like it could be promising though it is applied only on overhead imagery, which tends to be easier than on-board as there is no problem with the water changing appearance closer to the horizon (since there is no horizon). Nevertheless, this method may have warrant further investigation.

2.4.3 YOLO

You Only Look Once[14] or in short YOLO, is a very recent method that divides the scene into a grid and uses deep learning to determine what the contents of each grid contains. The method is extremely fast, running at frame rates that are easily real time if not faster, and seems to perform quite well. In brief, the way it works is that it splits the image into a grid and runs a neural network over each grid space to determine the probability of that space belonging to background or an object. There are a number of weaknesses with this method though. There are spatial constraints that depending on the resolution of the grid may decrease the accuracy of our bounding boxes.

In particular this results in difficulties in accurately finding bounding boxes for clusters of smaller objects, which we may be a common occurrence in our application. We also may run into problems with lack of training data since the method does rely on learning the appearance of objects in the scene, and as stated in the paper, will have difficulty with objects in unusual configurations/aspect ratios. Simply given the superior computational speed of the method however, it may still warrant further investigation, perhaps if used in tandem with a domain adaptation technique.

2.5 Object Salience

Detecting object salience is equivalent to identifying potential objects of interest in the scene. As will be seen in the next section, an object salience method was indeed chosen as the core of our final model. We will first highlight some of the other object salience methods tested and give brief insight as to why they were ruled out.

2.5.1 BING

BING or *BInarized Normed Gradients*[6] falls under the category of an objectness method. This class of methods tries to find bounding boxes around everything that may be considered an object, which sounds like exactly what we want. In particular the BING method runs at 300 fps which certainly meets our performance requirements. However, the issue with many of these methods is that they have high recall but low precision. They generate a large quantity of proposal boxes and out of those there is a high probability that the desired bounding boxes have been found, but there are many false positives, which is not what we want. Indeed a machine learning method could be overlaid on top to further classify which boxes contain boats but this is also not trivial as many boxes could contain the same boat yet cover very different regions and additionally there can be hundreds if not thousands of proposals for a single image. For some images, the ideal bounding boxes can be found relatively quickly as in figure 8, but in other cases it may take much more, as in figure 9. We have no way of knowing how many bounding boxes proposals are needed for each image ahead of time though. We are looking for a method that provides us with both high accuracy and high precision bounding boxes that have a good percentage of overlap with the ground truth annotations.

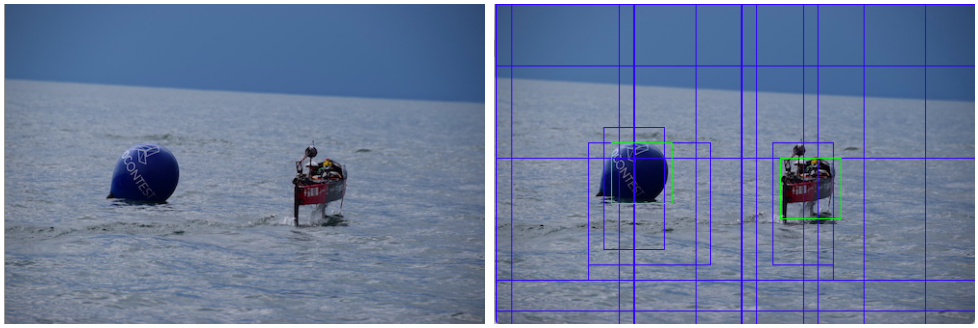


Figure 8: 10 bounding box proposals

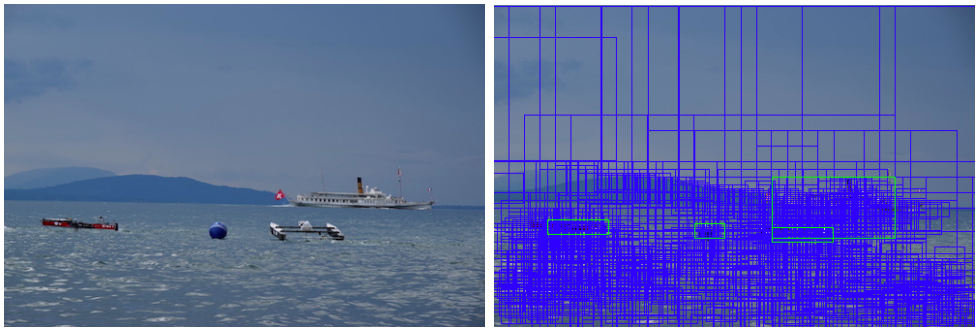


Figure 9: $>$ 1000 bounding box proposals

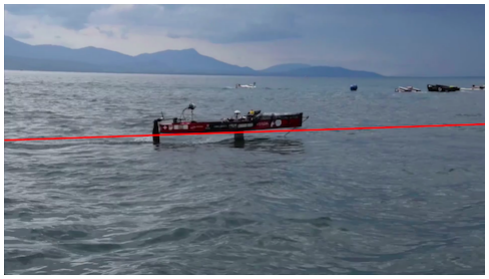
2.5.2 FASA

FASA, short for *Fast, Accurate, and Size-Aware Salient Object Detection*[21], developed by the IVRL lab at EPFL is another real time method for determining object salience. The method takes advantage of information about variance of colors in the image to locate and size potential salient objects. However, qualitatively just looking at the examples below it does not work very well in a water environment where there is some amount of contrast variance in just the water areas themselves, which will often times be the case, see figure 10. This approach was ruled out based on qualitative observations on the produced salience probability maps.

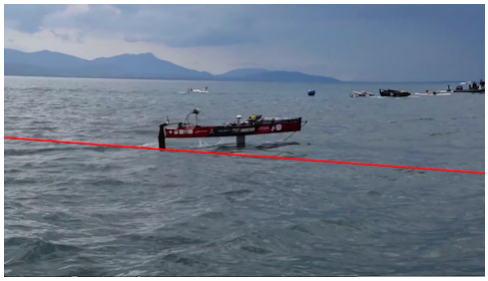
Figure 10: FASA does not do well at ignoring turbulence in the water

2.6 Horizon Line Detection

Horizon line detection for our algorithm is very important. We only want to detect obstacles of interest that are in the water. However, as we will see, in our final approach we will use an object salience based approach that does not distinguish land and water, simply finding all objects of interest. Having a robust horizon line estimation would allow us to simply remove any detected obstacles above the horizon line. In *Comparison of Methods for Horizon Line Detection in Sea Images*[11], various approaches to image based horizon detection are described and the one that offers the best trade off between performance and accuracy is done by Canny Edge Detection followed by a Hough line transform scheme. This method performs well when the edges generated by the water are not too prominent or when the horizon in the image is easily distinguishable as can be seen in figure 11. However since we cannot guarantee either of these, our results are very inconsistent. As will be described later, this may be mitigated with the help of readings from a gyroscope and temporal constraints when considering video.

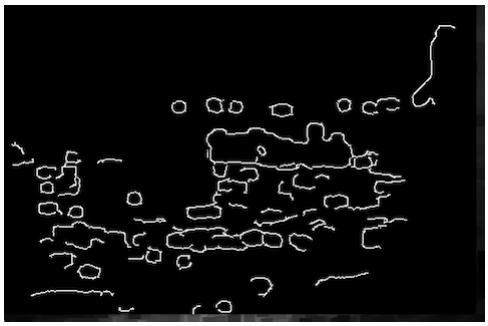


(a) The long edge of the boat is detected as the horizon.



(b) Edges generated by turbulence in the water result in false horizon line.

(c) No horizon line is detected



(d) Canny Edge image of 11(c)

Figure 11: Horizon Line Detection failure cases

3 Proposed Model

Our final algorithm is composed primarily of work from two major papers. We will discuss each of these separately and then the strategy used to combine the two.

3.1 Fast Image-Based Obstacle Detection From Unmanned Surface Vehicles

This method is described in the paper *Fast Image-Based Obstacle Detection From Unmanned Surface Vehicles (USV)*[10]. The core of this method relies on a constrained gaussian mixture model (GMM) consisting of three gaussian models. The strength of the method comes from the fact that it makes assumptions on the content of the scene, assuming that there is a land, sky

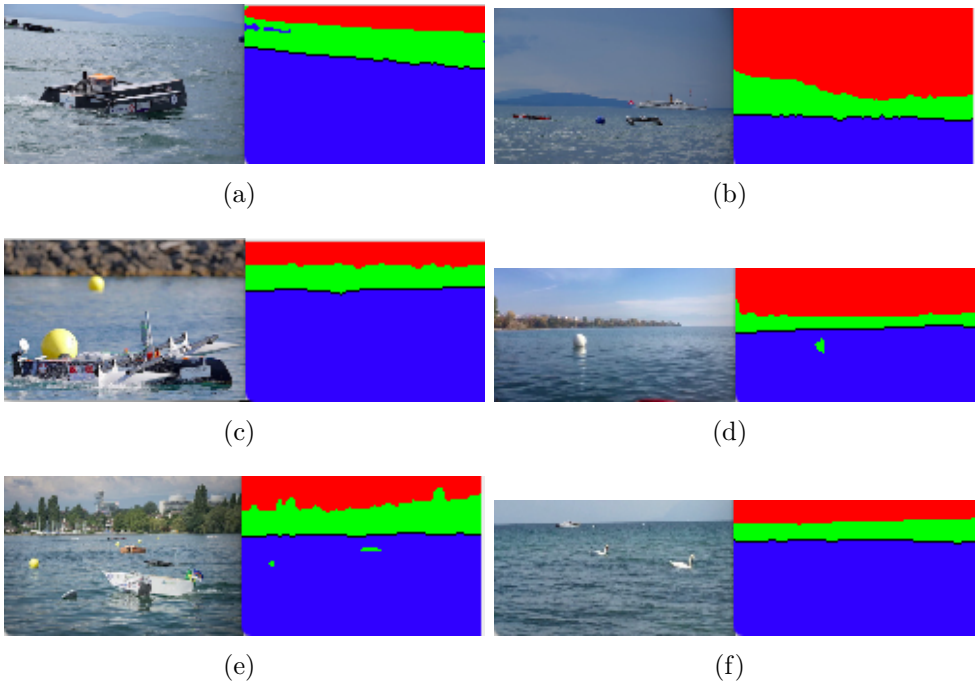
and water zone, which are each represented by a 5-D gaussian distribution where the feature vector consists of the colors and positions of each pixel.

3.1.1 Gaussian Mixture Model and Uniform Component

As mentioned the model consists of 3 Gaussians modeling the background and sky, land and water. A 5 dimensional feature vector is used: **(Column, Row, H, S, V)**. An MRF constraint that penalizes changes in values in neighbouring pixels is placed over the posterior and prior of the model. A Markov random field is placed over both the priors and posteriors to guide the connectivity of the final region labels. This can be done using a Kullback-Leibler divergence term in our problem's cost function which penalizes the differences between the prior and posterior distributions over the neighboring pixels. There is also a uniform component which in the author's implementation is just set to a constant amount for all pixels, the value of which depends on the color space used in modeling the background (HSV in our case).

3.1.2 Expectation Maximization

Expectation maximization is run in which in the expectation step we update priors and posteriors based on the gaussian parameters, and in the maximization step we update the gaussian parameters based on the priors and posteriors. Convergence is reached once the change in the prior gets below a certain threshold, as defined once again by the author's implementation. Typically convergence take about 3 iterations though in our implementation we set a max number of iterations at 5. To perform the expectation maximization we refer to equations 1 and 12 to update the posterior and prior respectively, and equations 10 and 11 to update the means and co-variance matrices respectively. Details of the our cost function and the resulting derivations of these equations are not given here as they are explained in great detail in [10]. Examples of the learned background assignments are shown below12. Note that not all scenes contain land, sky or both and we can end up with more than one model representing different areas water as will be discussed further in section 3.1.6.



(a) (b) (c) (d) (e) (f)

Figure 12: Assignments of pixels to different background models.

3.1.3 Shore line extraction

Next we extract the shoreline. We do this by looking at the largest connected component marked as water and then for each column we scan up the rows of the image, setting the last row that is marked as water as part of the shoreline.

3.1.4 Obstacle extraction

This shore line can be used as the boundary to determine whether or not an obstacle is in the water. In our model we have four probabilities, one for each gaussian and one for the uniform component. Taking the set of all connected components where the uniform component probability is greatest, a connected component is marked as an obstacle if the lowest point of the component is beneath the shoreline. Alternatively, instead of the shoreline we can also use a predetermined horizon line which in most of our images is a sufficient approximation. A binary map of the obstacles is created, upon

which bounding box selection can be applied (see 3.4). A visual outline of the pipeline is provided below13:

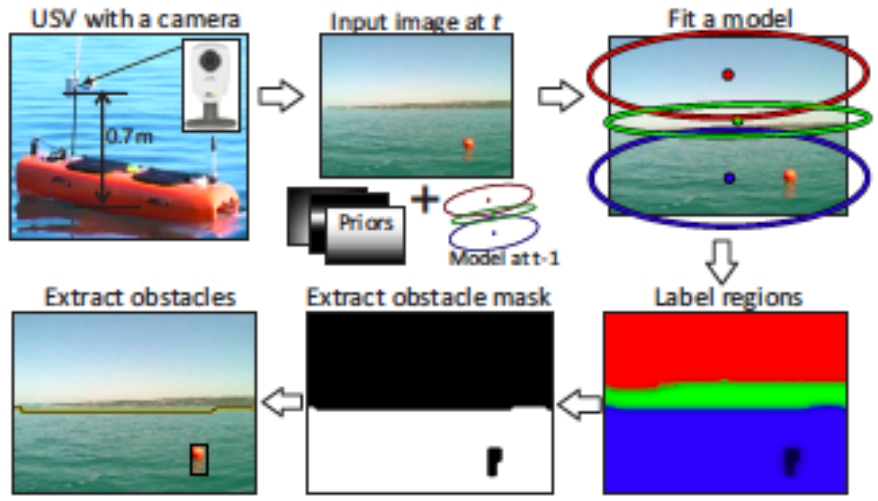


Figure 13: An outline of the GMM based method provided by [10]

3.1.5 Importance of the horizon line

This method is sensitive to the initial gaussian parameters that we feed to it. In particular the image must be initially separated into three regions based on the horizon line from which we can use the feature vector values to learn each of the gaussian parameters. Doing so allows us to initialize the model with values that roughly separate the image into three distinct regions, which firstly decreases the likelihood of converging to a bad local minimum and secondly helps with convergence speed.

3.1.6 Modifications

The paper also places a weak priors on the gaussian parameters which can be initialized via training data. However, we firstly do not having enough training data for this, and using the prior would make the method a lot less flexible as we may have to re-train the prior when the environment being operated in changes. Finally since it is a "weak" prior it's not even strictly necessary.

Indeed preliminary testing showed inclusion of the weak prior had little to no effect on our results and it is not included in the final implementation.

Another distinction made was in determination of the obstacles. In our final implementation of the method we treat the horizon line as the boundary between land and water. The distinction of what zone corresponds to water, which is necessary in the original paper to determine where obstacles in the water are. Indeed in some of our images the horizon line is near to the top of the image or may not exist in the image at all (we just see the water area). In such cases, the three primary gaussian models no longer learn the appearance of the sky, land, and water, but of simply three vertically aligned background regions in the image. We can see how this can be useful as the appearance of water can change depending on its height in the image. For example, water near the bottom will be closer to the boat and be more affected by reflections and turbulence, while water higher up will appear more uniform due to the water's distance from the camera. As an example, note how in figures 12(a), 12(c) and 12(f), there are really only two distinct background regions so we end up with two models for water instead of one, one of which models water closer to the horizon and the other which models all the rest.

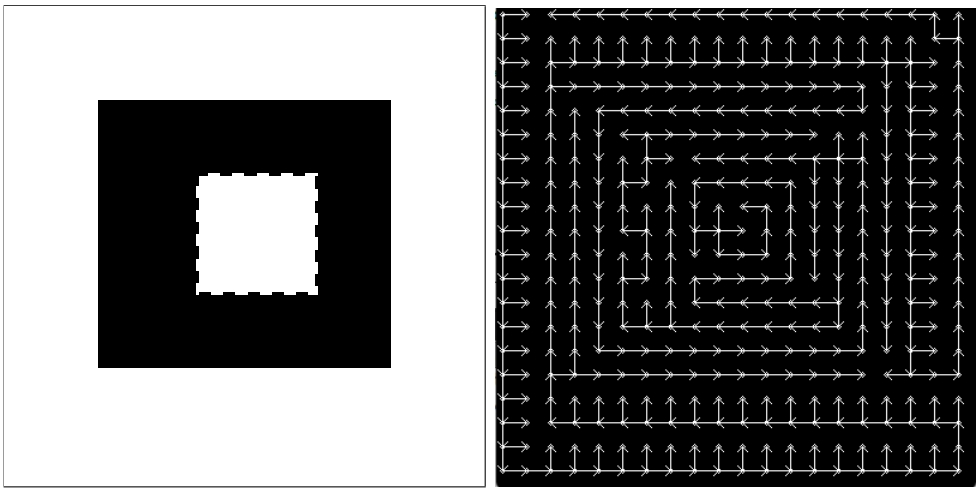
3.2 Object Saliency from a Minimum Spanning Tree

This next method is described in the paper *Object Saliency from a Minimum Spanning Tree*[18], which I will provide an outline of. The core of this method comes from the idea of representing the image in the form of a minimum spanning tree which takes advantage the structural information of the image. Each pixel can be represented via a node in said tree and the value at each node represents a measure of object salience or how likely that pixel is to belong to an object of interest. We outline the steps of the algorithm below.

Figure 14: The initial scene from which we will run the MST Object Saliency detection pipeline.

3.2.1 Creating the minimum spanning tree

In the first step we create the minimum spanning tree on the gaussian blurred, gray scale version of the original image. The image is represented as a grid of nodes where each node is connected to each of it's direct neighbors (left, down, up, right). The weight of the edge between two pixels is simply the difference in intensity. Given this a minimum spanning tree is created using Prim's algorithm though it is done so in an optimized way using a priority queue. Details can be found in the paper. We include figure 15 to show how the minimum spanning tree is guided by the structure of the image. Another important aspect here is that all nodes that are at the boundary of the image are considered as background and are marked as background seed nodes.



(a) Example image for creating the MST (b) Resulting MST

Figure 15: The image on the left is a scaled up version of the original 20x20 image from which the Minimum Spanning Tree on the right is created. Note how there is only one edge in the tree going from either white region to the black region (and vice versa). Thus the path to go from a white boundary pixel to any other other outer white pixel is shorter than to any of the black pixels.

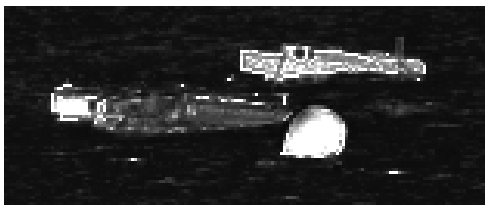
3.2.2 Calculating the distance transform

The distance value at each node represents the difference in intensity of the current pixel with the closest background seed node in the tree. To do this we must do two passes through the tree, one up to the root and one down starting at the root. A higher value at a node indicates a higher difference with the background seed nodes and thus a higher likelihood of being a foreground object. The resulting image is the MST distance transform image. The name of this distance transform used is Minimum Boundary Distance (MBD) transform and implementation details of the up and down passes through the tree are given in [18].

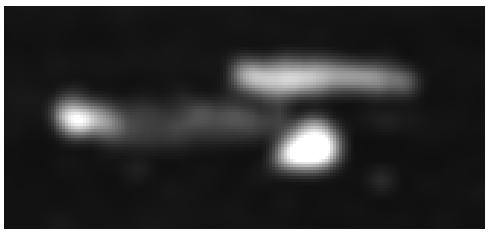
Figure 16: The MBD transform image on our image in 14

3.2.3 Creating the boundary dissimilarity map

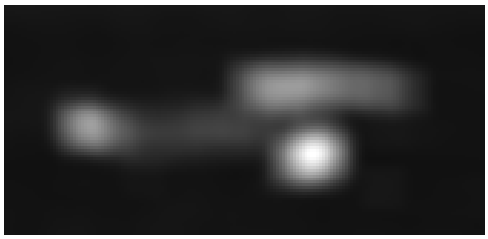
The boundary dissimilarity can be calculated very simply by collecting all pixels from a pre-specified border width (we used 20), clustering them into its dominant colors using K-Means and then performing a simple weighted distance calculation between each pixel of the image and the representative background colors, the weights for each cluster color being directly proportional to the number of boundary pixels in each of the clusters. The boundary dissimilarity is further blurred by a tree based filter which applies a Gaussian blur over the image which maintains main edges defined by the tree structure. In brief, a neighbouring pixel contributes less to the blur if difference in the tree distance with the central pixel is larger, figure 17.



(a) The boundary dissimilarity map before any blurring



(b) Blurring the boundary dissimilarity map in 17(a) with a gaussian blur



(c) Blurring the boundary dissimilarity map from 17(a) using the tree filter with the same gaussian parameters as in 17(b)

Figure 17: Comparison of Gaussian Blur and Tree Filter on boundary dissimilarity map. While both 17(b) and 17(c) are blurred, note that in 17(b) the area taken up by the boats and buoy have actually shrunk whereas in 17(c) they take up about the same area as in 17(a). This is because the tree filter respects the boundaries of the objects and uses very low weights for the darker background areas outside said boundaries. Thus using the tree filter allows us to remove high frequency noise from the image while retaining the shape of important objects.

Figure 18: The tree filtered boundary dissimilarity map of 14

3.2.4 Combining and thresholding the map

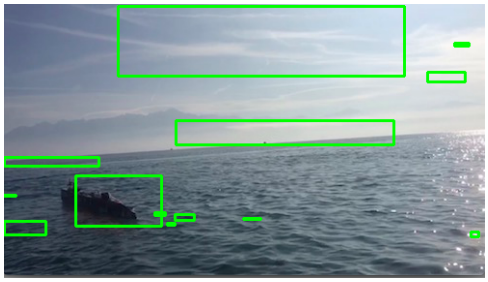
The two maps are then simply added together, normalized and contrast is increased using the post processing operation specified in the paper.

Thresholding is done using a Otsu Thresholding[13] continuously on subsets of the image until less than 25% of the image remains detected as obstacles. By this we mean that the second threshold operation is done only on the subset of pixels that are selected as a result of the first thresholding operation, and so on. We do this because we can generally assume that the majority of the scene consists of background. If the initial thresholding results in more than 25% of the image being detected as obstacle, then we likely have false positives. Since water will generally have a lower value in our post-processed salience map than actual objects, performing additional thresholding operations may help reduce these detections. The result of this thresholding scheme can be seen in 19. Bounding box selection can then be done on the resulting binary salience map22 (see 3.4).

(a) Result of running otsu thresholding just once

(b) Bounding boxes obtained from 19(a)

(c) Result of running otsu thresholding until at most 25% of the image is marked as obstacle



(d) Bounding boxes obtained from 19(d)

Figure 19: Comparison of running otsu thresholding just once vs using multiple runs. The example here is a bit extreme but illustrates the effect of the thresholding scheme well. With a single run, the whole image ends up being considered an obstacle. With multiple runs, while the result is still far from perfect it is certainly much improved.

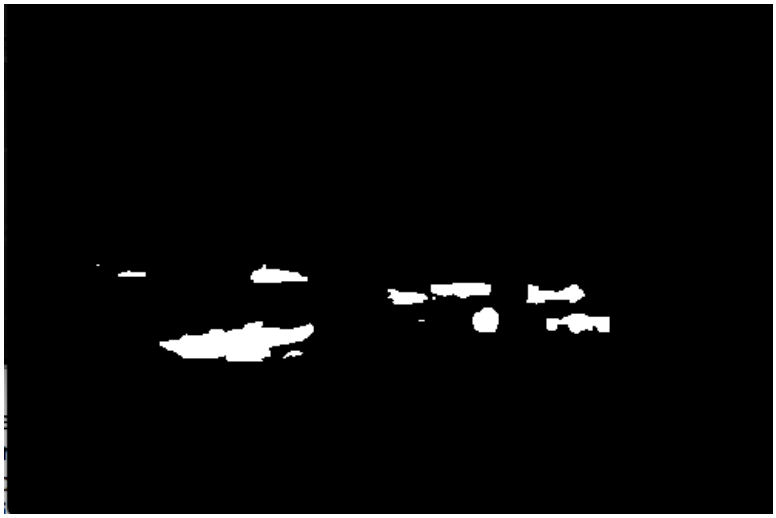


Figure 20: Combining 16 and 18, post processing, and thresholding

A visual outline of the full pipeline from the paper can be seen below 21.

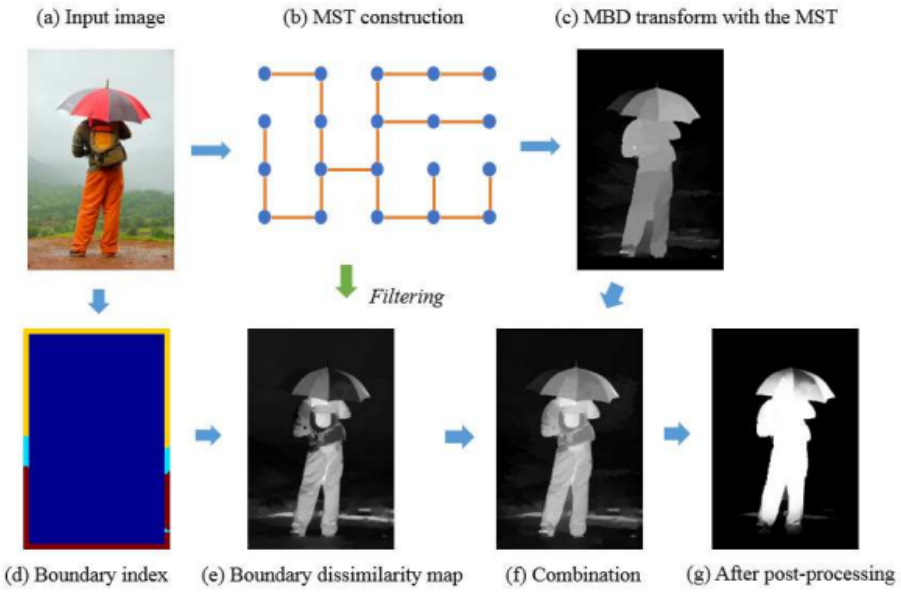


Figure 21: An outline of the MST based method provided by [18]

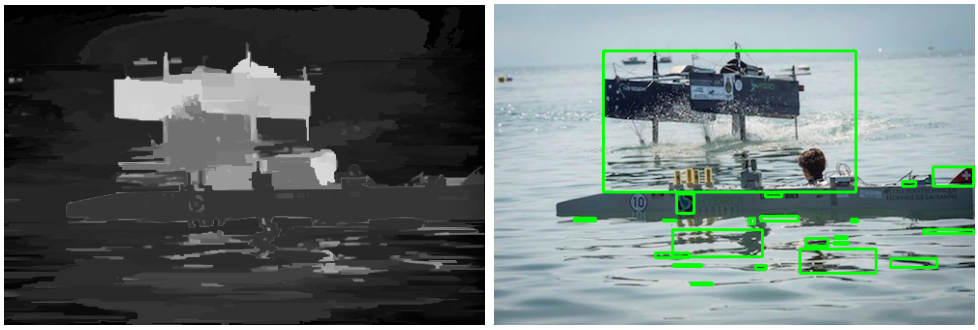
3.2.5 Modifications

The paper also suggests performing a post processing operation on the combined salience map that places more weight on more centrally located objects. The intent of this operation was to account for the bias that photographers have to put objects of interests in the center of their frame but clearly this would be detrimental for our purposes so we left this step out.

Figure 22: Bounding boxes drawn over 14 using 20

3.3 Final Combined Model

The main issue with using using the Minimum Spanning Tree model alone is that the assumption that all boundary pixels are background is a pretty weak one. In the most obvious failure case a boat just entering from the edge of the scene would end up being treated as background. An example is provided in figure 23 Seeing as the method will be used in a boat race, this will likely happen fairly often. In addition, choosing only the boundary pixels as being representative of the entire background is just a poor assumption that is sure to miss many background pixel values. We can modify the MST model and combine information gained by the GMM model in order to choose a better set of background seed nodes.



(a) MBD transform of a problem scene (b) Resulting bounding boxes

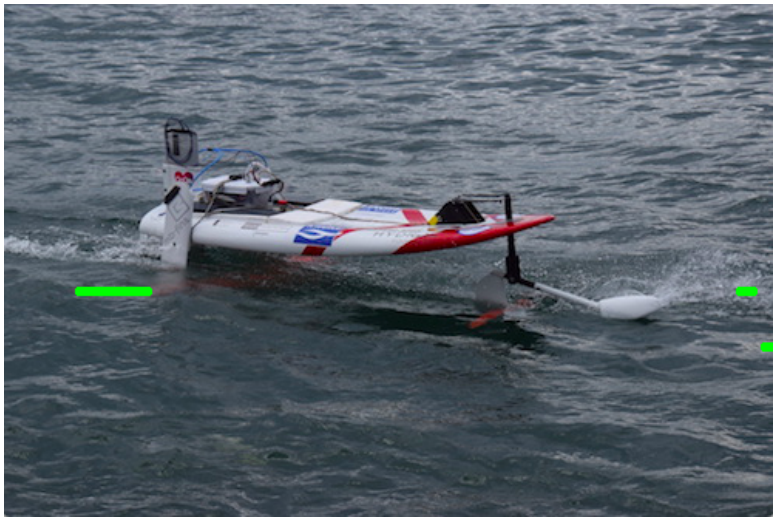
Figure 23: The boat coming in from the right of the scene is not detected as an obstacle since some of its pixels are marked as background seed nodes.

Initially a straightforward strategy was attempted in which pixels that have a high probability of being a background (land, sky or water) are set as background seed nodes, after which the minimum spanning tree is applied as normal. These probabilities are found simply using the posterior distribution from the Gaussian Mixture Model method. This solution however was not very good primarily because, for example, by choosing only the water pixels with the highest probability you end up choosing pixels that look very similar to each other. In reality the background is composed of many different values so it doesn't make sense to choose just a few of them to represent the background as a whole.

A different approach was taken in which we first mark the entire image as background, then find pixels where there is more likely to be an obstacle by taking the bitwise OR of the maps found in the GMM method and a thresholded boundary dissimilarity map used in the MST method. The idea here is that both these methods provide different results for what pixels should be considered obstacles. In particular one of the weaknesses of the GMM Model is that if an obstacle takes up a large portion of the screen the obstacle will be learned as the background model (figure 24), which is undesirable. However if the object does not overlap too much with the edges of the frame, then the boundary dissimilarity map will find the obstacle. Conversely if an object is too near the boundary, and hence won't be caught by the boundary dissimilarity map, the GMM Model is more likely to pick it up. However, we may still not pick up on large obstacles that both take up a large amount of the image and overlap a significant amount of the border

area. At the distance that an obstacle would have to be at in order to satisfy these requirements though, it could probably also be easily detected using cheap, light weight range finders. Ideally, the collision avoidance algorithm would also prevent the boat from getting into such a close quarters situation in the first place.

(a) Zone assignments of a scene where an obstacle takes up a large portion of the image



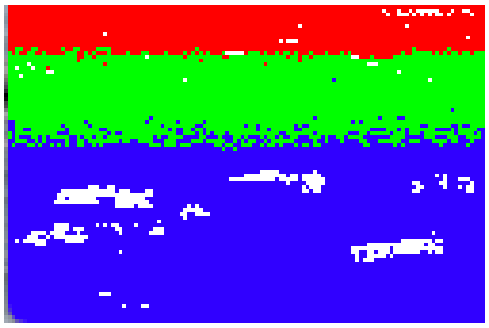
(b) Resulting bounding boxes

Figure 24: The boat it its entirety is learned as a completely separate background model due to its dominance in the image. As a result the boat cannot be detected as foreground.

At this point we can say that all pixels that are not determined to

be foreground are set to white on the binary seed node map. In order for the MST method to work properly, every seed node must be located outside of the objects of interest, which may not always be the case with the way our current seed node map is defined. There may be a very small number of pixels set as background seed nodes within the bounds of an object and just one is enough to result in the MST model's failure to detect the object. Thus we erode the seed node map until the percentage of pixels marked as background seeds in the map is beneath a certain threshold, in our case an experimentally determined 75%. Finally the MST model is run using the new set of seed background nodes and bounding boxes are found. In this next section we describe how bounding boxes are chosen given the binary foreground/background map. We show the full process for an example scene below.

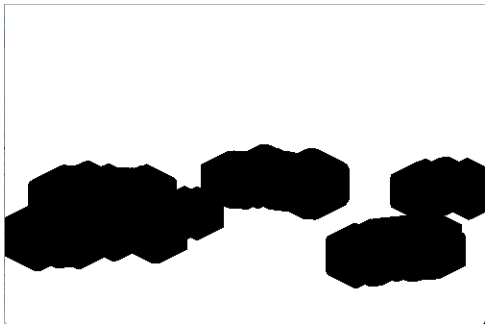
Figure 25: Example scene we will use to demonstrate our final model pipeline



(a) Zone mapping result from GMM model. Pixels marked white that are below the predetermined horizon line are considered potential obstacles.

(b) Boundary dissimilarity map. This map will be thresholded to create another potential obstacle map.

(c) Merge the obstacle maps obtained from 26(a) and 26(b). In the seed node map, set foreground to black and background to white.



(d) Erode 26(c) until less than 75% of the image remains as background seed nodes

Figure 26

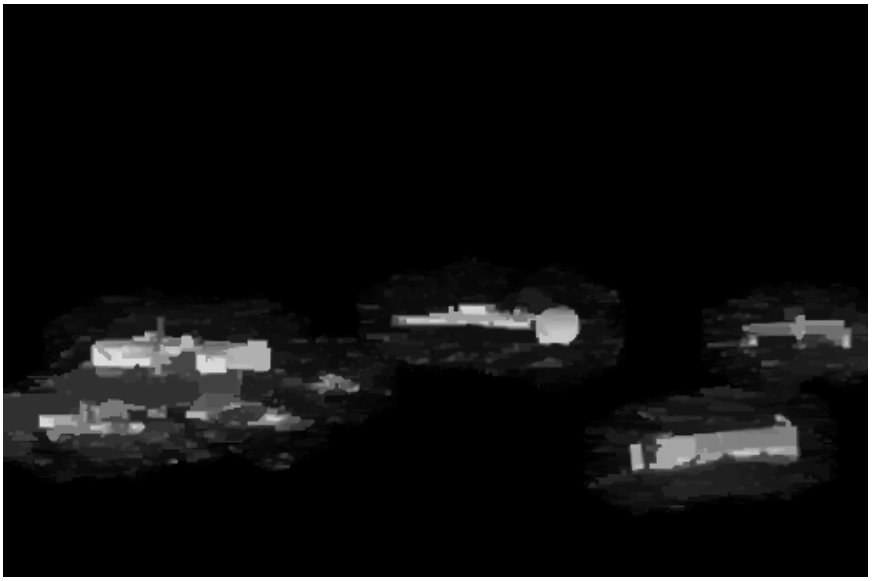


Figure 27: The resulting MBD transform given the seed nodes specified from 26(d)

Figure 28: Adding together the boundary dissimilarity map from 26(b) and the MBD transform image from 27, we normalize, post-process and threshold to find the resulting binary salient object map.

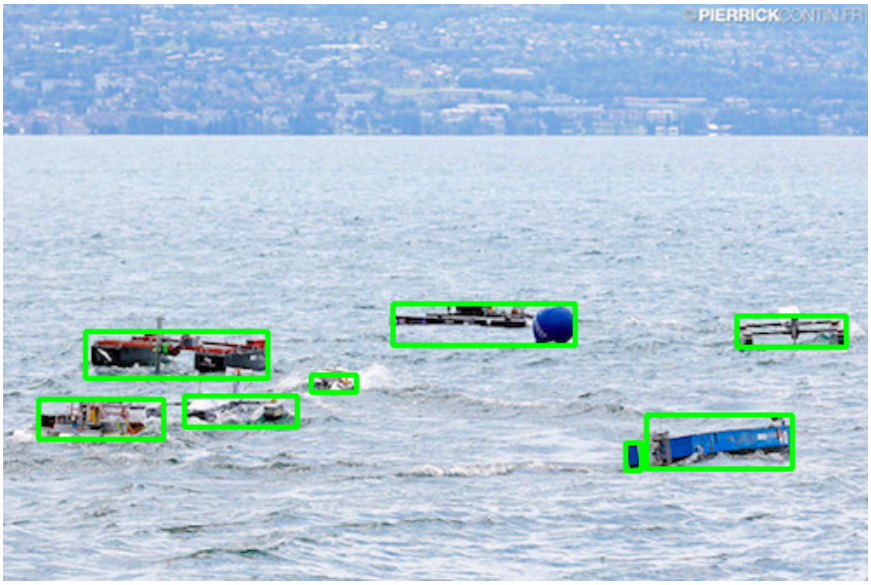


Figure 29: Use the binary map from 28 to obtain or final bounding boxes.

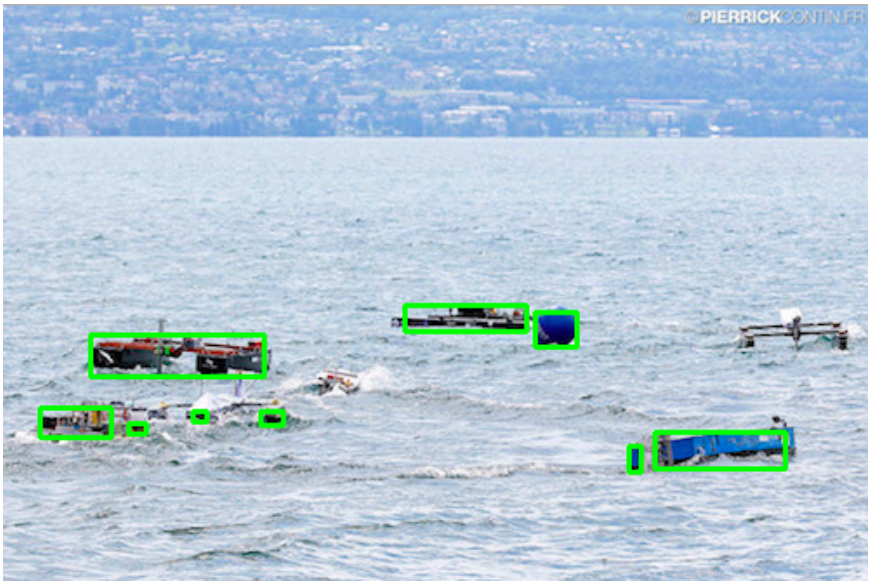


Figure 30: For reference, this is what our bounding boxes look like if we skip the eroding step and just use the seed node map from 26(c). Two entire boats are not detected and two more boats are only partially detected.

3.4 Bounding box selection

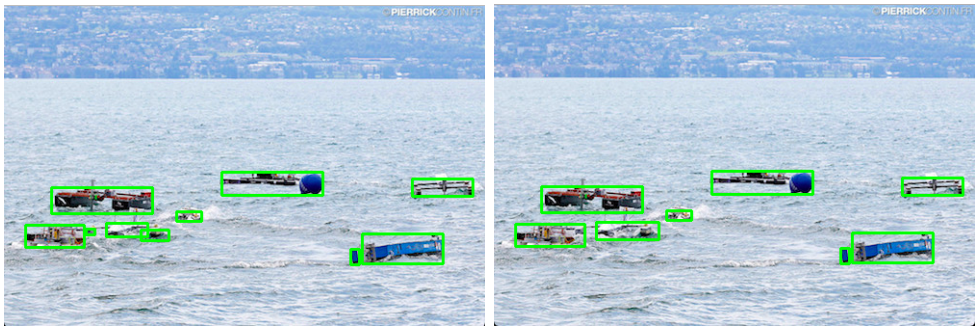
For all models bounding box selection is done in the same manner. For each resulting binary image, we use OpenCV's implementation of the Suzuki's contour following method in *Topological structural analysis of digitized binary images by border following*[17]. For each resulting contour the smallest bounding box that encloses the contour is found. As output, the bounding boxes are drawn over the original images and the coordinates and dimensions of each bounding box is given as (Column, Row, Width, Height).

3.4.1 Bounding box merging

After this, boxes that are completely contained in other boxes are merged with the larger boxes. Any boxes that are partially overlapping or touching are merged if either a size similarity or color similarity measure are under above a pre-defined threshold. These measures are defined by *Segmentation as Selective Search*[20] as $s_size(r_i, r_j) and$s_colour(r_i, r_j), where r_i, r_j are the subsets of pixels within each bounding box that are marked as obsta-

cles in the binary salience map (we don't want to include water pixels in the comparison). The paper itself describes an object salience method though has the same issue as BING in that it has very high recall but very low precision. It also takes about a full minute to process a single image which makes it unfeasible for real time applications such as ours.

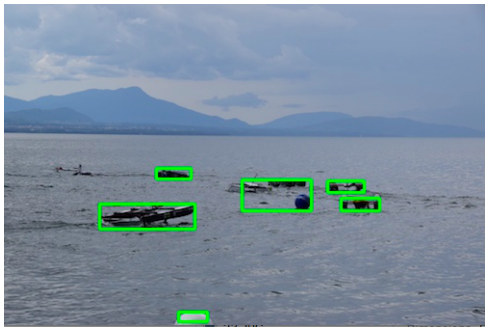
In practice the effects of the bounding box merging are hard to accurately measure since there aren't too many cases in which boxes are partially overlapping or just touching. However, in the final model, it does seem to help very slightly in training, raising our final training score by 0.88%. Qualitatively, the effects of the merging can be seen in 31.



(a) Bounding box result before merging.

(b) An instance where merging is beneficial.

(c) Bounding box result before merging.



(d) An instance where merging is detrimental.

Figure 31: Examples of the effects of our bounding box merging strategy.

4 Analysis

4.1 Training and Test Data

As mentioned previously we unfortunately do not have access to a wealth of data. In fact, even including images of the HydroContest taken by other teams found online we still have only 100 images, most of which are also unfortunately from the shore. In the future we hope to be able to acquire much better data. Indeed the methods will likely have to be re-tuned based on the new data but for now we will have to make do with what we have. Each image is annotated use a MatLab script which outputs the position and size of each bounding box in the image as well as the resulting annotated image32(a). Each image also comes with horizon line data (also via a MatLab script32(b)), thus we assume that we have acquired an accurate horizon line for each image.

(a) annotated bounding boxes (b) annotated horizon line

Figure 32

Additionally only a few actual videos were provided and there was also no frame by frame gyroscope data needed to make an accurate horizon line estimate. As a result testing was constrained to static images only. The hope is to gather more video data with full side by side sensor readings next semester.

4.2 Performance measure

As previously described, these methods have various parameters that need to be tuned. In order to determine the best parameters for our data we use

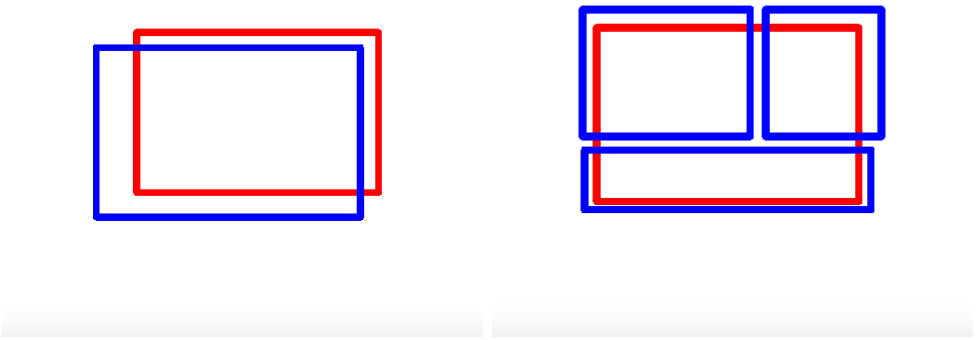
the average of two different scores to determine the accuracy of our method.

The first is the **FScore**. To calculate precision, for each bounding box, we calculate the percent overlap the set of all ground truths has with the box, and then take the average percentage over all proposal boxes in the scene. To calculate recall we do just the opposite, taking the average of the percent overlap that the set of all proposals has with each of the ground truth boxes.

The second is the **Mean Average Best Overlap (MABO)** metric. This metric is defined in *Segmentation as Selective Search for Object Recognition*[20]. For each ground truth annotation it finds the proposal with the highest percentage of mutual overlap (so a bounding box that just covers the whole image does not produce a high score). It then takes the average across all ground truths in an image and then the mean average score over all images to produce the final measure. The formula the mutual overlap is given by the area of the intersection of the two regions over the union:

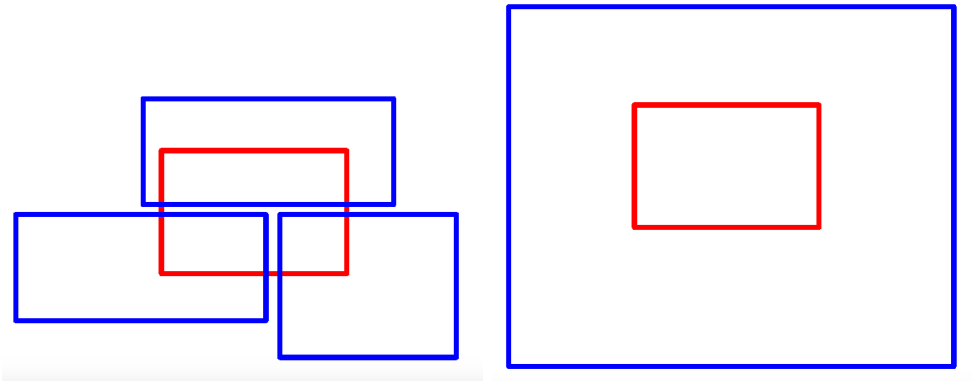
$$Overlap(r_i, r_j) = \frac{r_i \cap r_j}{r_i \cup r_j}.$$

For both measures, if the amount of overlap with a bounding box detected is less than 50%, then the score for that box is counted as a 0. We feel that both of these scores measure different types of accuracy that are important to us. Thus the final performance measure is the average of these two measures, a perfect score being a 1.0. We first tune our parameters using 50 of our images and then generate our final scores with confidence intervals using the remaining 50.



(a) Both the MABO and F-Score score well, regardless of whether red is ground truth and blue is proposal or vice versa.

(b) The F-Score scores well here but MABO does poorly. The result is still okay because we are properly detecting the location of potential threats in the water.



(c) MABO performs poorly here. If the red is ground truth and blue is proposal recall does well, but precision does poorly. The opposite is true when blue is ground truth and red is proposal.

(d) MABO performs poorly here. If the red is ground truth and blue is proposal recall does well, but precision does poorly. The opposite is true when blue is ground truth and red is proposal.

Figure 33: Bounding box scoring examples.

4.3 Results

There are a number of parameters that need to be tuned in the proposed model. To list the important ones:

- the initial gaussian blur parameters used in pre-processing the image for the MST model
- which color spaces is used for the boundary dissimilarity map
- the boundary width to be used for the boundary dissimilarity map and the strength of the uniform component used in the GMM model
- the tree filter blurring parameters
- which color space is used for modeling the background in the GMM Model
- the amount of eroding to do on the seed node map
- the strength of the markov random field constraints placed on the posterior and prior

As explained previously, these parameters were tuned on half of our data set and final test scores were generated on the other half. Note that we did not do a full exhaustive search over the parameter space and mainly just optimized each parameter independently. In the future it may be beneficial to write an automated script to automatically set and test all possible parameter combinations, though I expect the gains in accuracy to be fairly minimal. Additionally, it does not make sense to spend too much time tuning parameters when our training set is so small as we will likely just end up overfitting an image set that is not fully representative of all possible scenarios.

We present the train and test scores for the MST and GMM models 34 separately and compare those with the results for the combined model:

TRAIN RESULTS	GMM	MST	Combined
Recall	0.682148	0.859317	0.898317
Precision	0.672417	0.62339	0.788667
F-Score	0.578884	0.63582	0.789418
MABO	0.414265	0.498212	0.566062
(F-Score+MABO)/2	0.4965745	0.567016	0.67774

TEST RESULTS	GMM	MST	Combined
Recall	0.691111	0.822778	0.848556
Precision	0.705573	0.568585	0.720382
F-Score	0.614284	0.540659	0.705184
MABO	0.445373	0.441816	0.535263
(F-Score+MABO)/2	0.5298285	0.4912375	0.6202235
95% Confidence Interval:	[0.4693, 0.5903]	[0.4139, 0.5686]	[0.5605, 0.6799]
85% Confidence Interval:	[0.4349, 0.5475]	[0.4858, 0.5739]	[0.5768, 0.6637]

Figure 34: Train and Test results for models

Both the the GMM Model and MST Model perform similarly on the test and train data with the GMM Model performing 3.9% better than the MST Model on test data. There is quite a large difference between the train and test results for the MST model (7.6%) which suggests that the MST model is more subject to overfitting the training data when tuning parameters. Indeed the gap between the train and test result for the combined model is notable as well at 5.7%.

Based on the value of the average alone, the test result for the combined model is significantly better than the GMM model by a margin of 9%. Unfortunately there is some overlap between the 95% confidence intervals of our models though the width of our intervals in large part has to do with the severe lack of test data. With an 85% confidence interval, there is no more overlap, thus we can say that we are 85% confident that our combined method performs more accurately than the separate MST and GMM models. It would have been ideal to have more data to train and test on in order to establish a higher level of confidence. Regardless, I believe that the results we have obtained show that our method is quite promising.

We also some present the resulting of the bounding boxes for each of the three methods in the appendix for qualitative comparison.

4.4 Performance

On average each frame takes roughly a quarter second to fully process though this varies depending on the number of EM iterations it takes to converge (max 5). We can say that as of now the frame rate is at 4 fps running on a MacBook Air with a 2.2 GHz Intel Core i7 Processor and 4 GB 1600 Mhz DDR3 memory. This indeed is not very fast as for real time performance we would require at least 30 fps. However, as will be detailed in the next section we may not need to run the full method continuously.

5 Next Steps

This project will be continued in Spring 2017. In this section I will discuss improvements that need to be made and overall goals for the upcoming semester.

5.1 Minor implementation improvements

- We should apply adaptive blurring during pre-processing by blurring more near the boat and less near the horizon so that we can remove high frequency wave features near the boat while not making smaller farther away boats near the horizon too hard to detect.
- Distance to the horizon line can be used to determine what a reasonable size for a bounding box proposal is. For example, smaller boxes farther away from the horizon line can be likely be ignored.
- We currently erode the seed node map until a certain percent of the seed node map is marked. More accurately however, we should be eroding around each potential object for a minimum amount. The risk otherwise is that the percent of the image marked as potential object pixels will be greater than the preset seed node percentage and no eroding will occur at all, which may result in missed proposals.
- As a hardware fix, a polarizing lens on the camera should be added to filter out specular lighting effects from the glare of the sun off of the water's surface. These are otherwise difficult to deal with selectively in software35.

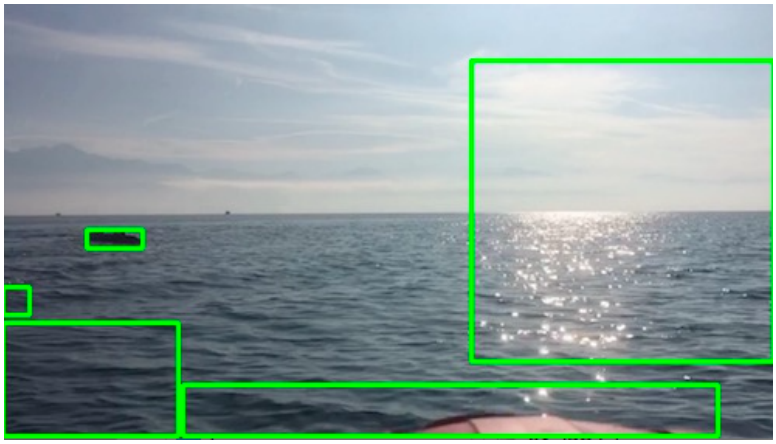


Figure 35: Specular reflections off of waters sirface result in false obstacle detections

5.2 Using temporal information from video

- Update GMM model across frames instead of treating each frame as an independent image, improving confidence in our background models over time.
- Bounding box proposals over features in the water would probably not remain for very long across frames. Bounding boxes that do not stay in roughly the same location across frames can be eliminated. In addition only bounding boxes that enter at the edges of our scene need to be considered as new obstacles since we do not expect obstacles to be surfacing from below.
- A single bounding box may encompass multiple obstacles that can split at any point (two boats next to each other). Approaches such as multi hypothesis kalman filtering may be helpful here.
- Multiple bounding boxes that share the same trajectory for many frames may be combined into one as they likely belong to the same boat.

5.3 Next semester major goals

5.3.1 Machine learning to filter/find bounding boxes

If issues with detecting water as obstacles still exist an additional classifier based method could be used to filter out the non-water areas from the water areas, perhaps with methods such as from *Water Region Detection Supporting Ship Identification in Port Surveillance*[2]. In addition it may be interesting to test out the accuracy and performance of promising machine learning based methods such as YOLO[14] and domain adaptation[19].

5.3.2 Performance Optimization

Parrallelization has not been explored and certainly GPU processing could be helpful in greatly speeding up global image operations. Indeed OpenCV provides implementations for some matrix operations using CUDA. On the boat itself we only plan on running the algorithm at most once a second as tracking of proposed boxes between these runs can be done via other less computationally expensive means. Still we would like to get the algorithm to be faster especially considering that it will likely be run on a system that is much slower than the MacBookAir (likely a Raspberry Pi). The exact desired number of frames per second remains to be seen once more testing has been done.

5.3.3 Robust Horizon Line Estimation

The method requires a robust estimate of the horizon line. The attempt with canny edge detection and hough lines showed that the technique by itself was not sufficient. Improvements can be made by taking into account the orientation of the boat using other sensors such as a gyroscope, so that we can perform a constrained hough line scheme only considering lines with slopes that correspond to the current measured gyroscope angle. In addition, since we know that the horizon line location will not change much from frame to frame we can also eliminate false positives and choose the detected horizon line that is closest to the previous frame's.

5.3.4 Frame to frame tracking

Once good bounding boxes have been determined we must track the boxes from frame to frame. If the method were capable of running at a full 30 fps

we could run the method continuously. However we don't see this as a viable option given the current performance and the on-board hardware. Instead, we will track the bounding boxes between runs of the method using object tracking methods such as CamShift, particle filtering or kalman filtering. The process may be sped up as well be using only the binary output from an edge detection method such as frei-chen in determining what contents to track. Using the binary image would allow us to perform fast binary operations when comparing frames, though this remains to be explored further. The results from tracking may also provide information that may contribute to the obstacle detection results of our current algorithm by providing some measure of confidence in the next set of predictions.

5.3.5 Distance from monocular video

In order to apply the results of our obstacle detection to collision avoidance we would ideally like to create a map of boats and obstacles in the water in a radar like fashion. However, due to weight constraints we are limited to just using the existing on board camera alone thus using two cameras for stereo depth detection is not an option. It may be possible to apply a stereo algorithm between frames as the boat moves, though a generic stereo algorithm assumes that objects in the scene themselves are not moving which is of course not the case here. In addition, it will be difficult to test the accuracy of our method since it is difficult to acquire actual labeled distance data for training and testing.

5.3.6 Obstacle avoidance and path planning

If a good distance/location information for each obstacles is found and the type of obstacle is identified (boat/buoy) then we can generate a desired trajectory. At the very least we can respond to imminent collisions by slowing down or moving the boat out of harm's way.

5.3.7 Hardware/Boat integration and testing

No testing was done on-board and all data was taken from on-shore photos. Thus the method needs to be tested on the boat itself or at least with better data. Details like removing information of our own boat from the scene should be done as well.

6 Conclusion

In this study we have provided an image based method to determine the presence of obstacles and competitors in the water using primarily an on-board camera on an unmanned surface vehicle. We have tested a multitude of existing methods, with focus on the methods provided by [18] and [10]. From the results obtained we can make a couple major points.

- Firstly that taking advantage of structural information in our images to determine a measure of object salience, as was done in the MST Model, can be an effective way to determine the presence of obstacles in a maritime scene.
- Secondly that making assumptions of the structure of the scene as we did in the GMM Model can aid greatly in accurately modeling the appearance of background regions in a scene.

Indeed we have found that a method that combines the strengths of these two methods in an original way out performs either separate method by a notable margin of 9% with 85% confidence. This confidence will likely improve with more training/test data as well as with the addition of improvements listed in 5.1. We find that while the performance of the current implementation is a little slower than desired there is still room for improvement. For now however, what we have found is a method that provides us with a very reasonable set of bounding boxes over objects of interest in the scene. These bounding boxes can be further refined in the later stages of the project and will eventually be used to achieve the our ultimate goal of automated maritime collision avoidance and path planning.

References

[1] Bao, Linchao, Yibing Song, Qingxiong Yang, Hao Yuan, and Gang Wang. "Tree Filtering: Efficient Structure-Preserving Smoothing With a Minimum Spanning Tree." IEEE Transactions on Image Processing 23, no. 2 (2014), 555-569. doi:10.1109/tip.2013.2291328.

[2] Bao, Xinfeng, Svitlana Zinger, Rob Wijnhoven, and Peter H. De With. "Water Region Detection Supporting Ship Identification in Port Surveillance." Advanced Concepts for Intelligent Vision Systems, 2012, 444-454. doi:10.1007/978-3-642-33140-4_39.

[3] Bloisi, Domenico D., Luca Iocchi, Andrea Pennisi, and Luigi Tombolini. "ARGOS-Venice Boat Classification." 2015 12th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 2015. doi:10.1109/avss.2015.7301727.

[4] Corbane, Christina, Laurent Najman, Emilien Pecoul, Laurent Demagistri, and Michel Petit. "A complete processing chain for ship detection using optical satellite imagery." International Journal of Remote Sensing 31, no. 22 (2010), 5837-5854. doi:10.1080/01431161.2010.512310.

[5] Daniel, Rakos. "Frei-Chen edge detector." rastergrid. Last modified 2011. rastergrid.com/blog/2011/01/frei-chen-edge-detector/.

[6] Duan, Zhanzhan, Lanfang Miao, and Hui Wang. "Binarized normed gradients for object detection." 2015 IEEE Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2015. doi:10.1109/iaeac.2015.7428721.

[7] Heidarsson, Hordur K., and Gaurav S. Sukhatme. "Obstacle detection from overhead imagery using self-supervised learning for Autonomous Surface Vehicles." 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011. doi:10.1109/iros.2011.6048233.

[8] Islam, Mohammad M., Mohammed N. Islam, K. V. Asari, and Mohammad A. Karim. "Anomaly based vessel detection in visible and infrared images." Image Processing: Machine Vision Applications II, 2009. doi:10.1117/12.805513.

[9] Kadyrov, Alexander, Hui Yu, and Honghai Liu. "Ship Detection and Segmentation Using Image Correlation." 2013 IEEE International Conference on Systems, Man, and Cybernetics, 2013. doi:10.1109/smc.2013.532.

[10] Kristan, Matej, Vildana Sulic Kenk, Stanislav Kovacic, and Janez Pers. "Fast Image-Based Obstacle Detection From Unmanned Surface Vehicles." IEEE Transactions on Cybernetics 46, no. 3 (2016), 641-654. doi:10.1109/tcyb.2015.2412251.

[11] Libe, Tzvika, Evgeny Gershikov, and Samuel Kosolapov. "Comparison of Methods for Horizon Line Detection in Sea Images." Paper presented

at The Fourth International Conference on Creative Content Technologies, 2012.

[12] Mttyus, G. "Near Real-Time Automatic Marine Vessel Detection On Optical Satellite Images." ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XL-1/W1 (2013), 233-237. doi:10.5194/isprsarchives-xl-1-w1-233-2013.

[13] Otsu, Nobuyuki. "A Threshold Selection Method from Gray-Level Histograms." IEEE Transactions on Systems, Man, and Cybernetics 9, no. 1 (1979), 62-66. doi:10.1109/tsmc.1979.4310076.

[14] Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi. "You Only Look Once: Unified, Real-Time Object Detection." 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. doi:10.1109/cvpr.2016.91.

[15] Santana, Pedro, Ricardo Mendonca, and Jose Barata. "Water detection with segmentation guided dynamic texture recognition." 2012 IEEE International Conference on Robotics and Biomimetics (ROBIO), 2012. doi:10.1109/robio.2012.6491235.

[16] Selvi, M. U., and S. S. Kumar. "Sea Object Detection Using Shape and Hybrid Color Texture Classification." Communications in Computer and Information Science, 2011, 19-31. doi:10.1007/978-3-642-24043-0_3.

[17] Suzuki, Satoshi, and Keiichi Abe. "Topological structural analysis of digitized binary images by border following." Computer Vision, Graphics, and Image Processing 29, no. 3 (1985), 396. doi:10.1016/0734-189x(85)90136-7.

[18] Tu, Wei-Chih, Shengfeng He, Qingxiong Yang, and Shao-Yi Chien. "Real-Time Salient Object Detection with a Minimum Spanning Tree." 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. doi:10.1109/cvpr.2016.256.

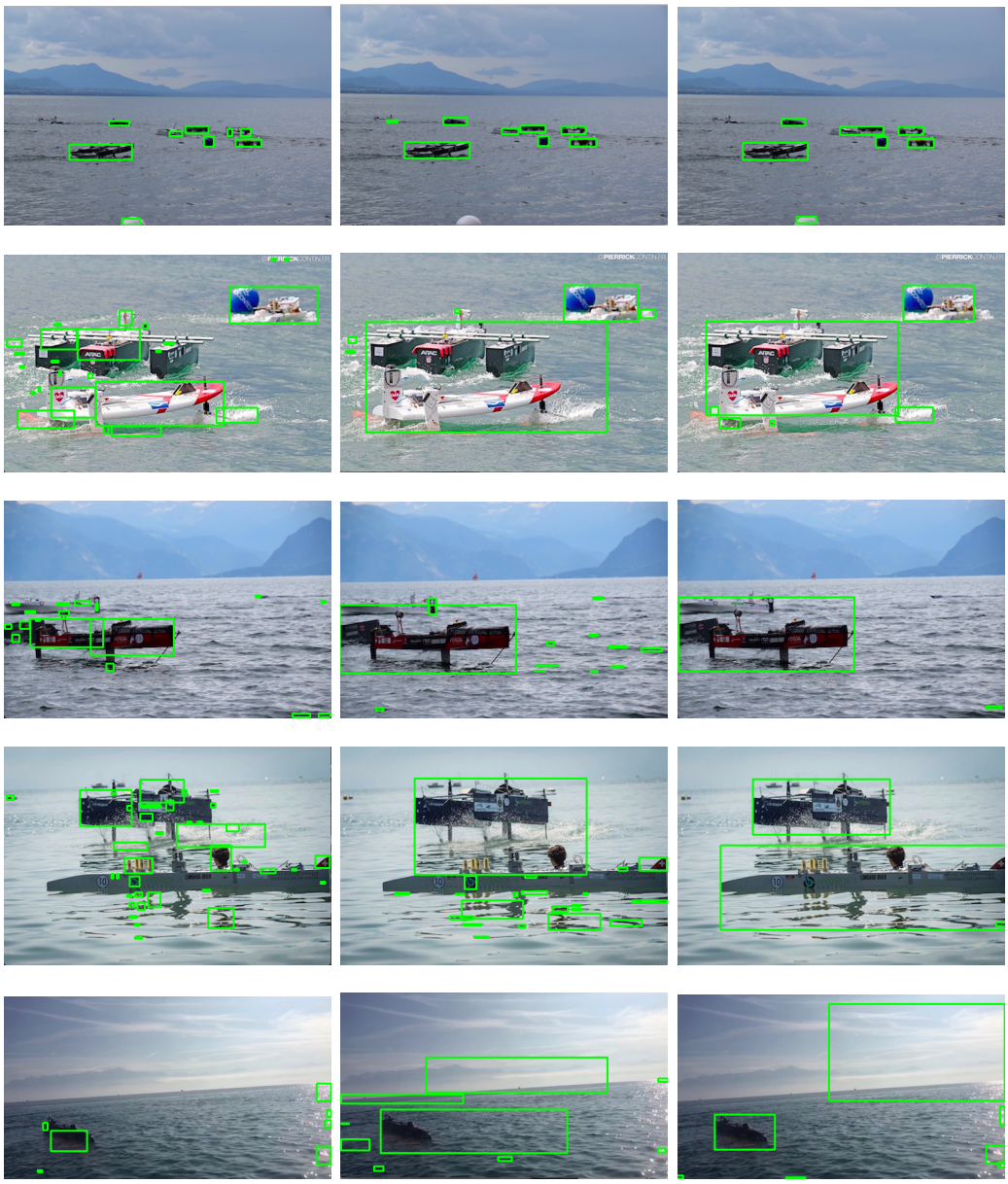
[19] Tzeng, Eric, Judy Hoffman, Trevor Darrell, and Kate Saenko. "Simultaneous Deep Transfer Across Domains and Tasks." 2015 IEEE International Conference on Computer Vision (ICCV), 2015. doi:10.1109/iccv.2015.463.

[20] Van de Sande, Koen E., Jasper R. Uijlings, Theo Gevers, and Arnold W. Smeulders. "Segmentation as selective search for object recognition." 2011 International Conference on Computer Vision, 2011. doi:10.1109/iccv.2011.6126456.

[21] Yildirim, Gkhan, and Sabine Ssstrunk. "FASA: Fast, Accurate, and Size-Aware Salient Object Detection." Computer Vision – ACCV 2014, 2015, 514-528. doi:10.1007/978-3-319-16811-1_34.

Appendices

GMM Model	MST Model	Combined Model



GMM Model	MST Model	Combined Model