Obstacle Detection for Image-Guided Surface Water Navigation

by

Tanmana Sadhu

B. Tech, West Bengal University of Technology, 2013

A Thesis Submitted in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Electrical and Computer Engineering

University of Victoria

Obstacle Detection for Image-Guided Surface Water Navigation

by

Tanmana Sadhu

B. Tech, West Bengal University of Technology, 2013

Supervisory Committee

Dr. Alexandra Branzan Albu, Supervisor

Department of Electrical and Computer Engineering,

University of Victoria

Dr. Maia Hoeberechts, Supervisor

Department of Computer Science,

University of Victoria

Abstract

An issue of concern for maritime safety when operating a small to medium-sized sailboat is that the presence of hazards in the navigational route in the form of floating logs can lead to a severe collision if undetected. As a precautionary measure to prevent such a collision with a log, a 2D vision-based detection system is proposed. We take a combined approach involving predictive mapping by linear regression and saliency detection. This approach is found to overcome specific issues related to the illumination changes and unstructured environment in the dataset. The proposed method has been evaluated using precision and recall measures. This proof of concept demonstrates the potential of the method for deployment on a real-time onboard detection system. The algorithm is robust and of reasonable computational complexity.

Contents

List of Figures

List of Tables

Glossary

SVR: Support Vector Regression

GLM: Generalized Linear Model

MOG: Mixture of Gaussians

LRC: Linear Regression Classification

GLCM: Gray Level Co-occurrence Matrix

LBP: Local Binary Pattern

MSSS: Maximum Symmetric Surround Saliency

NN: Neural Network

GMM: Gaussian Mixture Model

GPS: Global Positioning System

IR : Infra-red

CIELAB : Commission Internationale de l'éclairage LAB

Acknowledgements

I would like to thank my supervisors, **Dr. Alexandra Branzan Albu** and **Dr. Maia Hoeberechts** for their continued guidance, support and patience. I would also like to thank my lab colleagues **Frédéric Jean**, **Trevor Beugeling**, **Jeremy Svendsen**, **Kawthar Moria** and **Amanda Dash** for their friendship, help and encouragement. I am grateful to **Eduard Wisernig** and **Dr. Brian Wyvill** for their help with providing the database and their guidance for this work.

1 Introduction

This section provides an overview of the field of computer vision and its applications in the field of obstacle detection and recognition. Section 1.1 will provide a general overview of the uses of computer vision in obstacle detection. Section 1.2 provides a brief discussion of some of the main challenges associated with the task. Section 1.3 outlines the main contributions of the thesis.

1.1 Computer Vision for Obstacle Detection and Avoidance

The discipline of Computer Vision has confirmed its place in the scientific and technological fields of Cognitive Science and Artificial Intelligence for the development of automated visual perception. Computer vision algorithms aim to understand a 3D scene from 2D images using the properties of the structures present in the scene. Often, an interdisciplinary approach is needed to mimic human vision, combining knowledge across domains of computer science, electrical engineering, mathematics, physics, physiology and biology. The developed models are used to solve real-world problems using image and video data as input data. Applications are many and varied, including medical imaging, automation, event detection, object detection and recognition, tracking and restoration, to

name a few.

For self-navigating vehicles, computer vision plays a major role in path planning and obstacle avoidance. With the introduction of self-driving cars as a real-world application, this area of research has seen remarkable progress in a short period of time [4], [5]. Such systems often rely on a variety of sensing technologies such as stereo cameras and lasers in addition to monocular cameras. Designing an autonomously navigating robotic system often requires a detailed understanding of the environment through which the system will navigate. Vision sensors such as optical and infra-red (IR) cameras can provide additional information about the environment and the system.

Vision-guided autonomous navigation for air and land-based vehicles have been more actively explored compared to autonomous water navigation. Although technological navigational aids such as GPS localization are currently used, precise obstacle maps are almost impossible to generate owing to the unpredictable appearance of obstacles such as floating logs, crab traps, buoys and other small vessels. Obstacle avoidance, area survey analysis, threat assessment and surveillance is at present dependent mostly on man-in-the-loop situational awareness [6]. A US Coast Guard report from 2014 [7] notes that accidents such as collision with floating debris during the operation of a sailboat often occur due to attention deficit or misjudgement due to inexperience [7].

1.2 Design Challenges

With safety being a primary concern for an operator during the operation of a sailing vessel, the need to understand the surrounding environment and awareness about any

hazards on the water surface, are crucial. On-the-fly detection of obstacles and subsequent issuance of warning using only a monocular camera device and limited on-board processing would offer a significant advantage in such a case. When sailing around shores and inlets, floating debris, especially logs which are half-submerged, are prone to appearing unexpectedly, often requiring the operator to maneuver the boat in time to prevent a collision. Few attempts have been made to implement a system for water-based navigation which may generate a warning on an impending collision situation with the aid of computer vision technology (see section 2.0.2). The task is complicated by the visually noisy, unstructured scene consisting of the shoreline, reflections and the rapidly changing water features. Widely varying illumination conditions also make it difficult to differentiate between obstacles and other floating objects which do not qualify as obstacles (birds, seaweed, etc.).

The figure 1.1 below shows example frames containing log obstacles, spanning the variety of the landscape types, lighting and reflection conditions. Identifying such obstacles is challenging due to the complex nature of the background present in the images. The frame in figure 1.1a represents an image with non-uniform illumination and occurrence of a bright blue patch in the bottom-left of the image. Figure 1.1b shows a frame with strong shore reflections and lighting changes on the water surface. The frame in figure 1.1c shows a small-sized log occurring at a large distance from the camera. Figure 1.1d shows an image with very strong shore reflections with a log oriented at a small slanting angle from the camera. The frame in figure 1.1e is representative of dark, cloudy weather conditions, and shows a log highly contrasted with the background.

(a) Non-uniform illumination

(b) Large reflected area on water

(c) Obstacle located far from the camera

(d) Irregular horizon, strong reflections

(e) Dark appearance of background

Figure 1.1: Some examples of challenging scenarios for object detection in sample frames from the database

1.3 Focus of Thesis

To detect floating debris, specifically logs, which are deemed to be the most common and hazardous obstacles in the course of a small sailing vessel operating on the west coast of British Columbia, an automated approach using a visual processing system has been presented in this thesis. It is a step towards solving the debris detection and alarm generation problem. We take a modular approach based on classification by simple linear regression and saliency-based techniques in order to generate a combined probability map, which is further thresholded and analyzed by post-processing steps to perform the detection. We evaluate a combination of the regression-based predictive map and contrast-based saliency map for the goal of obstacle detection on our locally acquired database. The major contribution of this work is the development of an original static frame-based approach combining high-level visual recognition through the supervised linear regression module with the saliency detection module using low-level visual features. Although the work presented here focuses solely on the detection of a particular type of obstacle and is intended for a small sailing vessel, it can provide a background for extension to other types of obstacles.

2 Related Work

Computer vision for obstacle detection and avoidance is a broad field of research which includes a wide range of methods and applications. In this section, we will give an overview of the major focus areas in this field. First we discuss machine learning methods for object recognition and then we take a an in-depth look into saliency methods for object detection. In the end, we summarize the key points learned from the literature review.

2.0.1 Air and Land-based Vehicle Navigation

Vision-based navigation for air and land based vehicles has been much more extensively researched than the domain of water navigation ([8], [9], [10], [11], [12], [13]).

Terrestrial autonomous mobile systems have been well-researched in recent years with the increased availability of computing resources. Vision research for such systems has moved outside of controlled experimental setup [14]. The problem of path planning for a robotic land vehicles is addressed [15] in which trajectories are found by a combination of 2D visual detection and 3D mapping based on stereo depth. The authors in [16] on the other hand, approach the obstacle detection problem using 2D monocular vision only, by an appearance-based method with online learning of color histograms. The simplicity of

such a method allows it to perform well in real time. One of the most recent methods of autonomous navigation for self-driving cars is presented in [17], where the ego-motion of the car is estimated using point correspondences. They use a multi-camera system to match keypoints extracted from the surroundings and estimate a motion model.

Most recent vision-based obstacle detection methods rely on machine learning techniques. In [18], a vision-based obstacle detection system for small unmanned aerial vehicles (UAVs) is presented. Obstacles are detected by segmenting the image into sky and non-sky regions by a binary classifier and non-sky regions are treated as obstacles. The feasibility of this approach is demonstrated by using the vision output to steer a small unmanned aircraft to fly towards an obstacle.

2.0.2 Water-based Vehicle Navigation

Navigation sensors such as Doppler sonars, acoustic positioning systems, magnetic compasses, and pressure depth sensors are used for most surface and underwater navigation systems. SONAR-based mapping and navigational systems have been developed with reasonable success in [19] and [20]. However, the use of optical cameras for image-guided water navigation is an area that still remains relatively unexplored, and could significantly contribute to the development of autonomous navigation systems for surface vehicles.

Obstacle detection using cameras for marine environments has been attempted in [21], [22] and [23]. The method in [22] aims to find objects of interest in still images of the sea surface using adaptive thresholding on texture properties. In the presence of widely varying texture patterns in different parts of the water surface, an accurate segmentation of the object of interest against the water surface textures poses significant challenges.

The authors in [21] focus upon the detection of marine vehicles in open sea environments. The method is however, dependent on horizon detection as a primary step of the algorithm, to delineate water from sky prior to object detection.

The method in [23] is different from the above-described methods ([21], [22]) as it operates on IR images for locating floating mines. The authors have compared a number of background subtraction and motion-based algorithms for detection on an agitated sea surface using IR imagery. Background subtraction as a problem of saliency has been presented in [24] and the model has been demonstrated to work well with moving target objects against a dynamic background.

Research for visual obstacle avoidance within a context similar to ours has been presented in [25] and [6] for riverine navigation. Their method is based on the computation of optical flow. These systems return estimates of all obstacle regions using distinctness in object structure (such as corners) and/or motion patterns. Preliminary results for debris detection using a recognition module only for our sailboat navigation system have also been previously presented in [26].

2.0.3 Machine Learning Methods for Object Recognition

In the domain of machine learning, recognition tasks can be implemented by supervised or unsupervised learning algorithms, wherein supervised learning infers characteristics of labelled training data, and unsupervised learning deals with unlabelled data. Supervised algorithms analyze the training data, consisting of an input and a corresponding known output value, to infer a function. This function is then used to predict outputs for unknown test data. Unsupervised methods such as clustering find patterns in the data

to form labels. See figure 2.1 for a taxonomy of different branches of machine learning (supervised and unsupervised) and commonly employed algorithms in each branch.

To address the task of detecting and recognizing logs on the water surface, we studied several methods for object recognition. Considering the challenge of recognizing a specific object category with invariance to small changes in appearance, lighting and diverse backgrounds, it became imperative to select robust features as inputs for training a classifier to recognize our target objects.

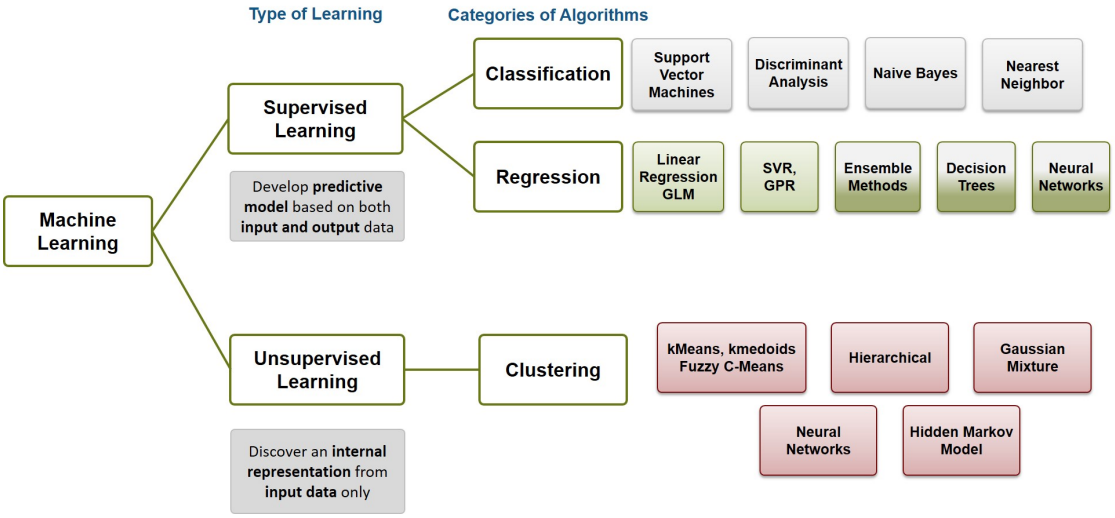


Figure 2.1: An overview of divisions of various machine learning algorithms [1]

In many applications such as face recognition, learning-based methods operating on low-level local features have been quite successful [27], [28]. Most applications rely on classification or regression techniques for object recognition [29], such as Support Vector Machines, multi-layer neural networks, ensemble methods such as boosting and bagging,

mixture models, and more recent methods like decision trees and convolutional neural networks [30], [27], [31], [32]. Although some of the more complex classification methods perform well on our dataset, our application requires the recognition module to be fast and reliable at the same time. A complex model often requires greater computational time and memory, which is undesirable in our case as we aim to deploy a real-time system with minimal computational load. Ideally, the system should be suitable for staging on off-the-shelf or dedicated low-cost hardware.

To gain a detailed understanding of machine learning methods for obstacle detection in robotic navigation, we studied the methods in [33], [34] and [35], which describe supervised and unsupervised paradigms. The work in [34] presents an end-to-end learning-based obstacle avoidance system for off-road mobile robots. Their learning process consists of a single convolutional network, inspired by the biological visual system. Convolutional neural nets represent the most recent advance in machine learning methodologies, and have been successfully employed in many vision applications including object recognition. The authors have constructed feature maps, which comprise inputs from a small neighborhood of the previous layer maps. The same weights are shared between all units in each feature map. Therefore, each feature map can be seen as convolving the feature maps of the previous layers with small-size kernels, and passing the sum of those convolutions through sigmoid functions as inputs for the next layer. This process automatically extracts local features at various scales which are then used to learn steering angles for the vehicle using back propagation with stochastic gradient minimization. This helps the robot to effectively learn its environment and steer away from potential obstacle regions. A convolutional net is also applied in [36] to estimate depths from monocular images for path planning.

The supervised technique for driving over desert terrain in [35] is shown to perform monocular road detection in desert terrains. The technique which is invariant to lighting conditions, involves extracting close range road location from sensor data. This helps in the subsequent removal of sky and shadow areas from the visual field. Once the self separation is done, a visual model of the nearby road is learned. The road pixels are modeled as a mixture of Gaussians (MOG). Every point in the image (excluding sky and shadows which have already been removed from consideration) is then assigned a class score using the learned MOG model.

The authors in [33] present an on-line learning method which can make accurate predictions of the traversability properties of complex terrain. Their method is based on autonomous training data collection which exploits the robot's experience in navigating its environment to train classifiers to recognize terrain classes without human intervention. This is in contrast to other learning methods in which training data is collected manually. The traversability learning method is tested on an unmanned ground vehicle (UGV) in realistic outdoor environments.

We constrain ourselves to the problem of obstacle (log) detection for alert generation only in this work as a step towards a more general obstacle detection and avoidance system. On comparing and conducting research into the different machine learning paradigms applied to the problem of autonomous vehicular navigation as discussed above, we concluded that to detect our known targets(logs) from background, we would need a supervised method of classification. Similar to the methods discussed we rely on a machine learning algorithm to help us recognize object features. We take a cue for our recognition module from the method in [37], which presents an approach where supervised learning is used to estimate depths from single monocular images. A simple linear

regression algorithm is trained to identify distances to the closest obstacles, while navigating in unstructured outdoor environments. The algorithm is able to learn monocular vision cues computed from texture energies and gradients to estimate the relative depths of obstacles in a scene. The authors go on to develop a control policy for their robotic car using reinforcement learning, that predicts a steering direction as a function of the vision system's output. We focus on obstacle detection only for this work, training our recognition module to distinguish between log features and background (comprising water, shore and sky) features.

In [38], the linear regression algorithm is posed as a classification task. It is based on the fundamental concept that patterns from a single-object class lie on a linear subspace. The inverse problem is solved using the least-squares method and the decision is ruled in favor of the class with the minimum reconstruction error. The Linear Regression Classification (LRC) algorithm falls in the category of nearest subspace classification. The authors establish the efficacy of their method by achieving a high recognition rate. The simple architecture of their approach makes it computationally efficient and suitable for real time video-based face recognition. Since we have a binary classification problem of distinguishing logs from background, we construct our linear regression targets as 2 different classes, such that it is an LRC problem with the assumption that samples from a specific object class lie on the same linear subspace. The efficiency of the design of linear regression as a classification method for recognizing visual features, along with the low computational requirements motivated its use in our recognition task.

2.0.4 Salient Object Detection

Saliency has been recognized as an important cue for visual obstacle avoidance in indoor and outdoor environments in many works ([39], [40], [41]). We realized that the property of saliency might be useful for our application considering that the floating logs to be identified often appear significantly contrasted from the surrounding water regions. The authors in [42] proposed a saliency model which is an adaptation for obstacle detection in all-terrain environments of the biologically inspired model originally proposed by [43]. Their method is based on the idea that the higher the saliency value of a given region, the more likely it is for the visual field to contain an object of significance. Location-based and object-based saliency are considered to be the two broad categories of saliency methods. Object-based approaches model saliency of objects by breaking down an image into regions or small clusters called superpixels. These methods often generate object-level saliency maps by measuring the irregularity of each region/ superpixel [32], [33], [34]. Conversely, location-based methods aim to create accurate saliency maps by assigning a saliency value to each pixel based on local or global features such as contrast. Such methods follow a bottom-up framework using low-level pre-attentive image features such as color, luminance or motion.

Top-down approaches for saliency detection often focus on finding high-level features such as faces, text, people, cars and so on [44], [45]. In [46], the authors have combined top-down and bottom-up approaches to predict human eye fixation. In experiments it was determined that extracting visually salient top-down features from our database of natural images is infeasible owing to the complex background, as the object-based saliency methods often select large shore or cloud regions rather than logs making it inaccurate for our object detection.

Learning-based methods for detecting saliency are proposed in [43], [47] and [48]. The method in [43] is perhaps one of the most influential. In their work, center-surround excitation responses of normalized feature maps are calculated at different frequencies, scales and orientations and fed into a neural network which outputs a saliency measure. The center-surround concept in saliency computation models is a representation of the mechanism of biological vision, where elementary visual features are used to determine how different a stimulus is from its surroundings, in many submodalities and scales.

A probabilistic saliency model is decribed in [49], wherein saliency as a Bayesian formulation is derived from common statistical properties of natural images such as Difference of Gaussians, obtained in advance from a collection of natural images. In [50], a multi-task learning approach is proposed to infer multiple saliency models simultaneously.This method builds on prior knowledge from scene clusters. The framework is constructed from multiple scene-specific saliency models, to select optimal features. The primary advantage of this method is that by multi-task learning each model is trained on a subset of samples selected from the whole dataset. Training each learner on a different subset is found to improve generalization by preventing overfitting. Although they report comparable or better performance than several other saliency algorithms, the need to use a clustering step (K-Means) followed by solving an Expectation Maximization (EM) problem makes it computationally intensive. More recent learning-based approaches as in [51] attempt to learn image features at various scales by utilizing convolutional neural networks. Global-context models for deep learning, however, cannot capture detailed information in local neighborhoods. Thus multi-context deep learning ([52]) would be more useful to capture salient objects.

Another class of saliency detection methods compute decision-theoretic saliency such

as in [53], where the authors argue that the most discriminant salient features are those that best distinguish a class of interest from all other classes. A similar hypothesis is validated in [54] through eye-fixation prediction and background subtraction, which inspired us to select discriminant saliency features like intensity and color to formulate our one-vs-all classification task.

For computing location-based saliency models, bottom-up approaches are used which compute saliency through pre-attentive features integrated without prior knowledge. These models can be roughly categorized into three domains: spatial, spatiotemporal and the transform domains. These categories are discussed in detail below to explain the rationale behind our selected approach in the transform domain.

A) Spatial domain

In the spatial domain, saliency models usually compute visual saliency by detection of irregularities in input visual stimuli either locally, globally, or both. These are based on the observation that high spatial frequencies are more discriminatory than low spatial frequencies for contrast, edge-content and chromaticity [55]. A boolean map approach proposed in [56] is another bottom-up approach, which repeatedly thresholds all color channels of an image, leveraging the center-surround cue to enable figure-ground segregation. However, these approaches fail to detect saliency for small scale objects such as in the case of our floating logs seen from a large distance from the camera.

B) Spatiotemporal domain

Using center-surround features, saliency as a measure of "surprise" was calculated for videos with the combination of spatial contrast and temporal evolution in [57]. This can be considered an example of spatio-temporal domain saliency measurement. Temporal information extracted from our videos lacked reliability primarily due to noisy background movement in the natural outdoor setting, complicated by the unpredictable motion of the water regions.

C) Transform domain

In the transform domain, bottom-up models include spectrum analysis based models, which find salient regions by extraction of spectral residuals in the amplitude spectrum of Fourier transform [58]. In [59], the authors introduce a frequency-tuned approach to estimate center-surround contrast using color and luminance features that offers three advantages over existing methods: uniformly highlighted salient regions with well-defined boundaries, full resolution, and computational efficiency. However, this method fails to perform in case of larger salient objects or complex backgrounds. The method in [3] improves on the previous method by varying the bandwidth of the center surround filter near image borders using symmetric surrounds, thus adapting to the scale of an object. We found this method to provide sufficiently high accuracy of saliency detection for our database, while also generating full-resolution maps.

2.0.5 Summary

In this literature review, we found that obstacle detection methods are highly task-dependent. Methods that perform detection and subsequent path planning vary according to the environment in which navigation is expected to take place. Understanding the background to isolate foreground objects is a popular approach. Considering the complexity of our background, approaches such as texture segmentation or keypoint extraction of object and background are not very successful in distinguishing our target logs from the surrounding water and shores. As the sailboat navigates close to shores in vastly different natural lighting conditions, background subtraction is also not an option. Optical flow based methods try to leverage the difference in motion patterns between water and other objects. Although we took initial interest in this approach, we realized that our target objects do not always have a strikingly different motion pattern from the surrounding waters, especially in calm conditions and when located further away from the camera. Hence the motion features are not found to be informative enough to apply optical flow methods.

In the recent methods surveyed for land, air and water-based navigation, a clear separation of either roads from objects, or land and sky or shore and water is a requirement. Horizon detection to isolate land and sky or shore and water is often a primary step. In our navigational environment, such a separation was found to be difficult to achieve and segmentation results failed mostly due to presence of reflections close to shore. Horizon detection as a preliminary processing step would help improve our approach, however the irregularity of our horizon line and large reflection patches pose difficulties. To combat the challenges of bright patches and reflections for our detection of logs on water, we decided to take up a supervised learning approach to maximize correct detections.

(a) Test image (b) Detected objects

Figure 2.2: An example of top-down "objectness" measure [2] shows failure to detect our target object

We also employ a transform domain bottom-up saliency method to reinforce our target detection. For our database, top-down approaches by region/superpixel segmentation often failed to discriminate between water regions and shore regions owing to the presence of strong reflections in many frames. Further, these methods failed to identify target objects appearing small from a distance as a separate region than the surrounding regions, preventing a robust object-based saliency detection. In [2], an "objectness" prior is calculated using low-level cues such as contrast, edge density and superpixels to learn if a certain region of the image contains an object or not (figure 2.2).

Bottom-up approaches rely on local or global unique characteristics for identifying salient regions. The biologically-inspired saliency models use pre-attentive features for computing center-surround response feature maps. The bottom-up frequency-tuned approach with an adaptive center-surround selection described in [3] was able to capture the local details of the logs even when they appear small. This method utilizes the property of contrast as well as location to determine the saliency value at each image pixel. A higher contrast measure as well as a more central location for a pixel relative to its

neighbouring region would lead to a higher saliency value assignment for it.

The goal of our system is to enable a computationally inexpensive obstacle detection system for a future real-time implementation on an on-board processing unit. The detection system output should be subsequently directed to a sailboat maneuvering system to generate alerts. The details of our approach are given in the next chapter.

3 Proposed Algorithm

This research is focused on the detection of obstacles using a computer vision based module for an autonomous water navigation system. As the nature of potential obstacles on the water surface in natural surroundings can be highly varied, in this work we focus exclusively on the detection of logs, identified to be one of the most common and hazardous obstacles for the navigation of small-sized sailboats. The intended application of this analysis is to develop a system which recognizes logs, often not clearly distinguishable on the water surface, being half-submerged or far from the observation point. The goal is to reduce watch-keeping efforts and the risk of collisions. The main objective of this work is to analyze a database of water navigation videos in a variety of lighting conditions and background scenes to detect log obstacles in each frame. An alert is generated for an identified obstacle persistent over a fixed number of consecutive frames.

The method used to build the detection algorithm is based on training a regression model to identify regions of a video frame where log obstacles may be present, reinforced by a saliency detection technique. The details of the algorithm are described in the following section.

3.1 Algorithm Overview

Keeping in mind the aim to achieve real-time performance while not being too resource-intensive, a novel algorithmic approach was developed. The proposed algorithm does not consist of highly computationally intensive modules and yet achieves a reasonable degree of accuracy and robustness.

A combination of vision techniques was used. The sequence of steps is shown in the overview block diagram (see figure 3.1). A linear regression model is implemented as the predictive mapping by regression analysis step. The saliency detection module applied in combination with regression allows for a reduction in the number of false positives, which occur mainly due to the presence of the shoreline in many frames, along with reflections and light patterns on the water surface. In experiments, detection by thresholding the regression map only yielded many more false positives, and thresholding only the saliency map led to a large increase in false positives and false negatives, as compared to the combined approach. Previous methods that use a combination of saliency with other methods for detection have been reported in ([60], [61]). The main contribution of our work is the implementation of a novel method combining linear regression with saliency for obstacle detection. Our method is a combination of high-level visual recognition through the regression-based learning module with a bottom-up attention mechanism through the saliency detection module.

Each step of the proposed algorithm is described in detail in the following sections.

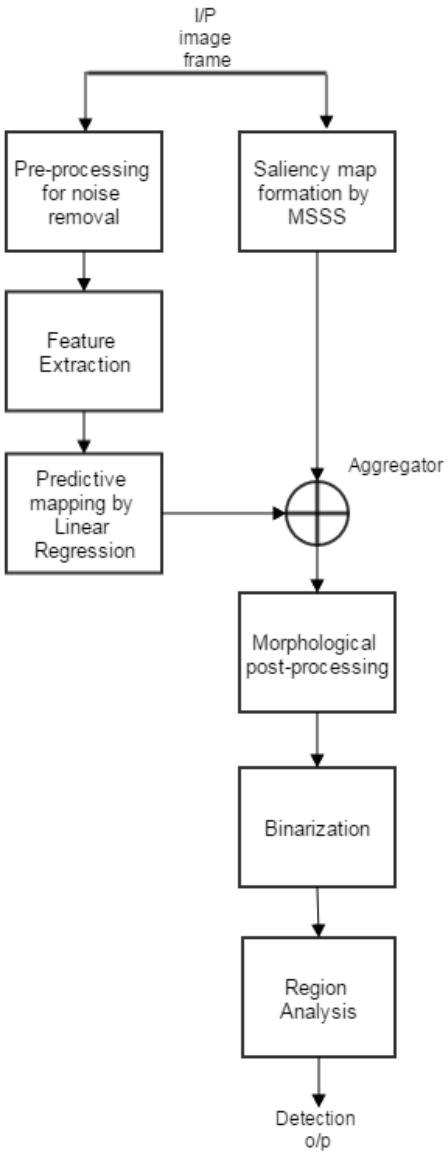


Figure 3.1: Block diagram

3.1.1 Preprocessing for Noise Removal

This step is performed to reduce noise effects prior to computing of the feature values. Each image frame is convolved with a rotationally symmetric 2-D Gaussian filter of size 3×3 which has an overall smoothing effect. A standard deviation value $\sigma = 0.5$ for the kernel was empirically found to be optimal taking into consideration the smallest size of the object to be detected (300 pixels approximately) such that the texture information would be preserved. Figures 3.3 and 3.4 show the effect of gaussian filtering on an image frame and extracted patches, respectively. With standard deviation being small, the effect of filtering is subtle to visualize.

Figure 3.2: Original image without filtering

Figure 3.3: Gaussian filtered image

(a) Image patch without filtering

(b) Gaussian filtered image patch

(c) Image patch without filtering

(d) Gaussian filtered image patch

Figure 3.4: Gaussian filtering on image patches

3.1.2 Feature Extraction

The features are extracted from each image on a pixel-by-pixel basis for classification. We consider appearance-based features and group them into two different categories: (1) intensity and color-based and (2) texture-based. The first category comprises one intensity channel (I) and two color channels (RG,BY), and the second comprises the

texture features.

Intensity and Color features

The color channels constructed are a representation of the "color double-opponent system" in the visual cortex [43]. From a center-surround perspective, neurons in the center of the receptive field are excited by one color and inhibited by another, while the reverse is applicable in the surround. This spatial and chromatic opposition is true for red/green, green/red and blue/yellow color pairs in human primary visual cortex [62]. The intensity (I), red/green (RG) and blue/yellow (BY) channels are computed as in eqns. (1-3) below [49]:

$$I = (r + g + b)/3 \tag{3.1}$$

$$RG = r - g \tag{3.2}$$

$$BY = b - \frac{r + g}{2} - \frac{min(r, g)}{2} \tag{3.3}$$

The above features are extracted for each pixel in a frame along with texture features.

Texture features

The texture of water surface regions are significantly different from other regions (for instance ripples on the water surface versus logs). Texture analysis quantifies the qualitative appearance of objects intuitively described as rough or smooth, grainy or silky, as a function of the spatial variations in pixel intensity values. This is leveraged to differentiate between textures of the logs vs. the background or other objects. As texture is a spatial property, the statistical moments of the gray level histogram of the image are calculated to define the texture features. A 2D dependence matrix called gray-level co-occurrence matrix (GLCM) is well-established for texture analysis [63]. The GLCM $p(i, j)$ is defined by a displacement vector $d = (dx, dy)$. All pairs of pixels having gray levels i and j separated by d are counted. A position operator is specified, according to which the pairs are constructed, for example (1,1) would indicate a neighbor pixel located to the right and below a reference pixel.

When there is no specific structure to the distribution of pixels, the GLCM is expected to be uniformly populated. The entropy is one of the quantities that can be measured from the GLCM, used to measure the randomness in the intensity distribution. It is represented in the eqn. (3.4) below :

$$E = -\sum_{i=1}^{N_g}\sum_{j=1}^{N_g} p(i,j)log(p(i,j)) \tag{3.4}$$

where E denotes entropy, $p(i, j)$ is the $(i, j)th$ entry in the normalized gray tone spatial dependence matrix $P(i, j)$, and N_g is the number of gray levels in the quantized image. The entropy feature value is calculated for each pixel in the image frame by considering

a 5×5 neighbourhood (chosen empirically considering the best performance measures in evaluation phase). Each output pixel contains the entropy value of the neighborhood around the corresponding pixel in the input image (figure 3.5 , 3.6). The highly textured areas of the images, such as shores and their reflections on water, are seen to correspond to high entropy values in the heat maps whereas smoother areas typically have lower values of entropy.

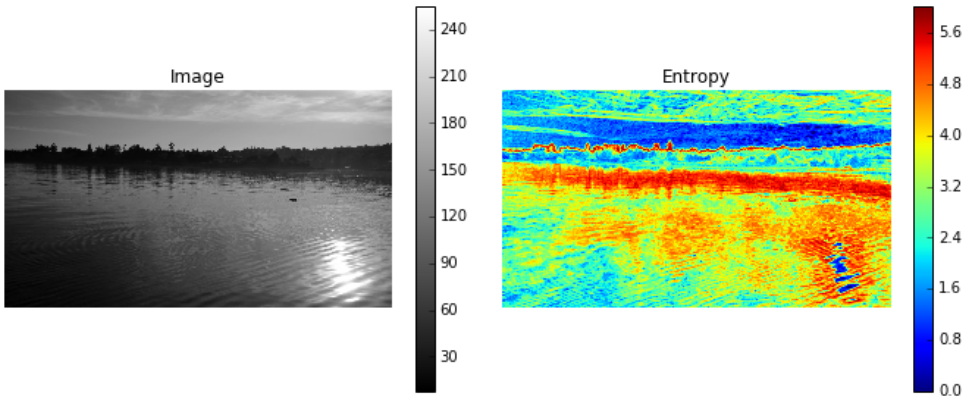


Figure 3.5: Entropy features for test image 1

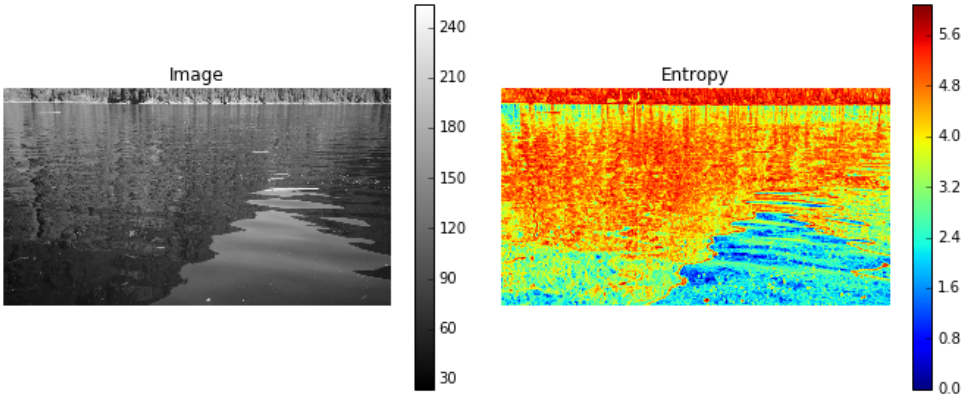


Figure 3.6: Entropy features for test image 2

The other set of texture features computed are the Local Binary Pattern (LBP) [64] values at each pixel neighborhood. The LBP measure encodes the relationship between a pixel and its neighbors into a binary word, allowing the representation of patterns. Its simplicity of computation and robustness to monotonic gray-scale changes (due to changes in illumination, distortion or occlusion) makes it well-suited to our application.

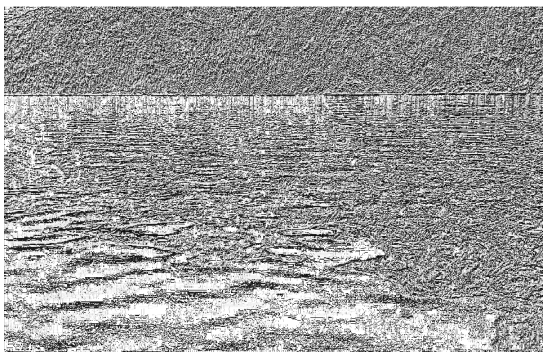
With the help of LBP, 2D surface textures are represented by the local spatial patterns and the gray scale contrast. Originally, the LBP labelled image pixels by thresholding a pre-defined neighborhood of a reference pixel at the centre, and converting the result into a binary number. In our application we use a flat radially symmetric neighborhood. The bilinear interpolation is used to fill in values for non-integer pixel coordinates in the neighborhood. Next, the histograms computed over each pixel neighborhood are used as texture descriptors. The LBP is represented by the following equation:

$$LBP(x_i, y_j) = \sum_{n=0}^{N-1} f(g_n - g_{i,j})2^n \tag{3.5}$$

where $g_{i,j}$ is the gray value of center pixel at (i, j) location in a circular neighborhood,g_n is the gray value of its neighbor. The function f is defined as $f(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$.

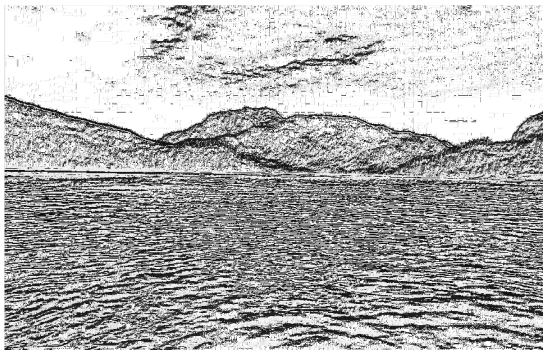
The figure 3.7 shows visualizations for the LBP features computed for different test image frames. The shores and reflected areas with strong gradients tend to show dense patterns of LBP features while less dense patterns are characteristic of areas with weaker gradients such as the sky in figure 3.7d.

(a) Test image 1



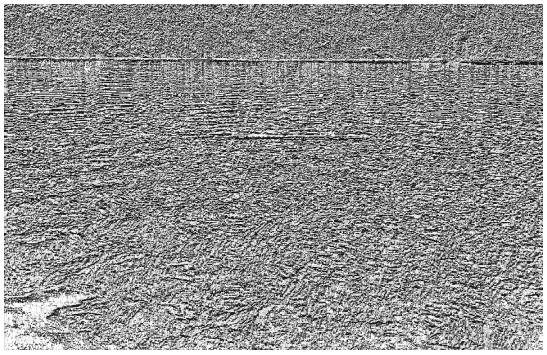
(b) Visual representation of LBPs extracted from test image 1

(c) Test image 2



(d) Visual representation of LBPs extracted from test image 2

(e) Test image 3



(f) Visual representation of LBPs extracted from test image 3

Figure 3.7: Visualization of the Local Binary Pattern (LBP) features obtained from sample test images

3.1.3 Predictive Mapping by Linear Regression Model

In this step we apply a supervised learning technique based on the multiple linear regression model. Regression analysis involves fitting a model to data and has been used previously in computer vision for a variety of applications such as face detection and land-based autonomous vehicular navigation [38], [65], [37].

The goal of application of simple linear regression at this step of our algorithm is to obtain a map which assigns a measure to each pixel in the image that predicts how likely it is for the pixel to belong to a target object. This is in essence a linear regression classification problem [38] with the probabilities calculated from the regression scores rendered as the real-valued outputs normalized in the range [0,1]. The Linear Regression Classifier (LRC) is essentially a discriminative classifier, belonging to a class of models used to model dependency of a target variable y on an observation variable x. Within a probabilistic framework, this is done by modeling the conditional probability distribution $P(y|x)$, which can be used for predicting y from x.

Based on our observations we form the assumption that features from our target class (logs) form patterns that can be mapped to the same linear subspace, we apply the classical linear regression technique for class probability prediction, which involves linear combinations of the input feature vectors [66]. To estimate the coefficients of the regression model, we consider the number of classes $L = 2$, corresponding to target and background, while m is the number of training images. We model the training set as:

$$S_{(a\times b)} = \bigcup_i q_i^{(c)}, \qquad c = 1..L, i = 1..m \tag{3.6}$$

where q_i represents each training image of the size $a \times b$, and $i = 1...m$.

The set of all feature vectors can be represented as:

$$W_{(g\times h)} = \bigcup_i \mathbf{w}_i^{(c)}, \qquad c = 1..L, i = 1..m \tag{3.7}$$

where each feature vector $\mathbf{w}_i$ (corresponding to each training image) is of size $g \times h$, $g = ab$ and h is the number of features.

The training image set is transformed to the feature vector set as $S_{(a\times b)} \rightarrow W_{(g\times h)}$ by concatenation of each feature vector $\mathbf{w}_i$.

The feature vectors are further normalized and stacked together to form the vector X_c as follows:

$$X_c = [\mathbf{w}_1^{(c)}, \mathbf{w}_2^{(c)}, ...\mathbf{w}_m^{(c)}]^T, \qquad c = 1..L \tag{3.8}$$

Each vector X_c is specific to a class c , and is also called the regressor for the class. Let our class-specific dependent target variable be denoted by Y_c, given by a deterministic output function $y(X_c, \beta_c)$ where β_c is the optimal set of parameters. This parameter set is estimated through the training phase as $\tilde{\beta}_c$ and is used to model the linear dependency between the independent vector of regressors X_c and scalar dependent target variable Y_c. The eqn.(3.9) below shows the relationship between dependent target and independent regressor vectors:

$$Y_c = X_c\beta_c + \epsilon, \qquad c = 1...L. \tag{3.9}$$

where a disturbance term ϵ represents an unobserved random variable, which adds noise to the linear model. The model parameter vector β_c can be estimated by the least squares estimation method [67]:

$$\tilde{\beta_c} = (X_c^T X_c) X_c^T Y_c \tag{3.10}$$

In the least squares method the sum of squares of errors error function E_D is minimized as follows:

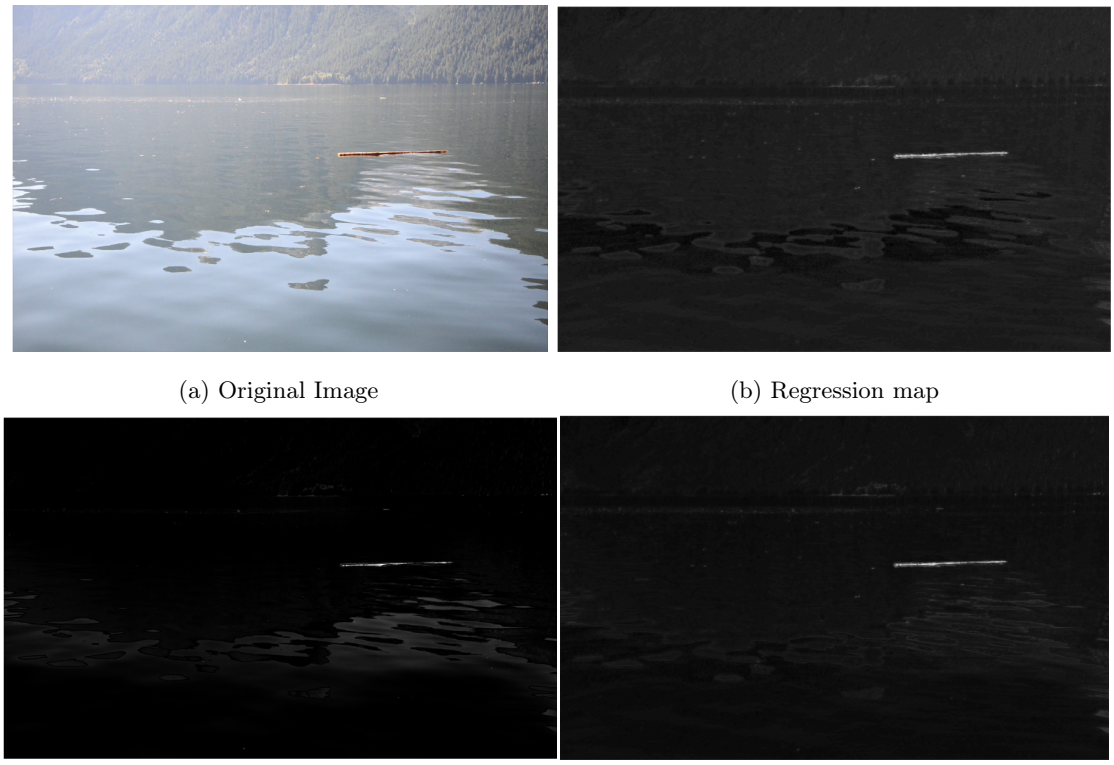
$$E_D = \frac{1}{2} \left\| Y_c - \tilde{\beta_c} (X_c) \right\|^2 \tag{3.11}$$

Next we construct the test feature vector Z corresponding to each test image of size $g \times h$, similar to the training phase.

The posterior probability Z of the observation X to belong to a class $c = 1...L$ is calculated using the above linear regression model. The regression map R is constructed as in eqn. 3.12 :

$$R = P(Z(c = 1)|X, \tilde{\beta}) \tag{3.12}$$

Here $c = 1$ corresponds to the target class and R is computed for each test image frame, represented by visualization in figure 4.3b .



(a) Original Image (b) Regression map

(c) Saliency map (d) Final map

Figure 3.8: Comparison between the (a) original image, (b) regression map, (c) saliency map and (d) final combined map.

3.1.4 Saliency detection using Maximum Symmetric Surround Saliency (MSSS)

Visual saliency is the perceived quality that focuses attention on a distinct region, object or pixel in an image, with respect to color, brightness or some other characteristics. A pixel is usually considered salient if it belongs to an area of image that has a unique appearance, both locally and globally [68]. Capturing of attention is dependent on the

selection of bottom-up or top-down processes, or their combination. From observation of our video database, the logs often appear highly contrasted to the surrounding water regions in our frames, even in varying illumination. This observation led us to consider applying a pre-attentive, bottom-up approach which is driven by the low-level attributes. We chose a saliency detection algorithm that would be relatively fast to compute and improve the detection capability of the algorithm (by eliminating false positive occurrences) from the previous step.

The Maximum Symmetric Surround Saliency (MSSS) technique proposed in [3] proved to be reasonably fast and accurate for our detection task. It also outputs a full resolution saliency map, which can be easily combined with our probability map R. It is based on the frequency tuned center-surround saliency measure introduced in [59]. The method computes the center-surround saliency measure for each pixel in CIELAB color space, which includes a wide range of perceivable colors. The "surround" is a symmetric region around the "center" whose dimension decreases as we move the center closer to the border of the image (figure 3.9). It is based on the idea that the more central a pixel is within the salient object, the smaller has to be the low-frequency cut-off for detecting it. However, how central a pixel can be inside an object is limited by how far the pixel is from the boundary. A pixel belonging to a salient object near the boundary will be less central inside the object. Here an assumption is made that the salient object is fully within the image, and the bandwidth of the center-surround filter is varied by increasing the low-frequency cut-off near the image borders. This would mean that object regions central to the image frame are considered comparatively more salient than the other regions. This becomes a limitation in our application in situations where the logs do not occur in the front field of view of the camera, but more towards the sides.

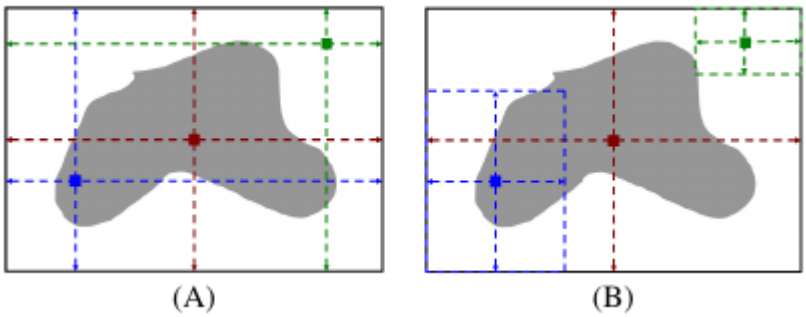


Figure 3.9: The method in [3] uses surround regions (sub-images) that are symmetric w.r.t the pixel whose saliency needs to be computed. This leads to varying center-surround bandwidths depending on the distance of the pixel from the image borders. A pixel in the center (red) has a global surround while one near the border(green) has a much smaller surround.

For an image of width w and height h, the symmetric surround saliency value at the given pixel location (r, t) is obtained as:

$$S_{MSSS}(r,t) = \left\| \tilde{I}(r,t) - I_{lp}(r,t) \right\| \tag{3.13}$$

where $\tilde{I}(r,t)$ is the average value of CIELAB vector for the subimage with center pixel at location (r,t) given by eqn. (14) as follows:

$$\tilde{I}(r,t) = \frac{1}{W} \sum_{m=r-r_0}^{r+r_0} \sum_{n=t-t_0}^{t+t_0} I(m,n) \tag{3.14}$$

where $r_0 = min(r, w-r)$; $t_0 = min(t, h-t)$; $W = (2r_0+1)(2t_0+1)$;
The figure 3.8c shows the saliency map visualization as obtained from above eqn. 3.14.

3.1.5 Aggregation

The predicted output map R and the saliency map S_{MSSS} are normalized such that all the values lie within the same range between [0,1]. Based on the method of *weighted pixel averaging* for aggregation or fusion of images ([69], [70]), regression matrix R and saliency map S are combined using a scalar weight μ. (See eqn. 3.15). The linear combination of these two matrices is justified by the fact that the information contained in each of them is independent from the other. This pixel averaging method is easily implemented and fast to execute [71].

$$F = \mu * R + (1 - \mu) * S_{MSSS} \tag{3.15}$$

where F represents the final probability map. The weight μ is set to 0.7, found to yield the best results from experiments (see section 4.0.3). This parameter value is tuned by computing the precision and recall values at each different combination of μ and $(1 - \mu)$ for each test set. The average precision and recall values over each set is computed to determine the value for which precision and recall are the highest, with more weight given to the recall value to account for few missed detections (see section 4.0.3). The figure 3.8d shows the final map obtained from this step.

3.1.6 Post-processing

The final combined map F obtained from the above step is normalized and thresholded to obtain a binary map with non-target pixels set to 0 (black) and target region pixels set to 255 (white). We use a threshold of 70% of the maximum value of F on the basis

of observation of the entire test dataset, which can be varied by about 10% of its value without significantly affecting the final segmentation. The binary map thus obtained contains many noisy connected component regions which need to be removed through the post-processing steps. At this stage, the top 1/4 of the binary image is masked out to remove noisy components (mostly parts of sky and shore), as this part never contains log obstacles in any of the frames in the dataset. Next, a number of morphological processing steps are performed as described below.

a) **Morphological closing** - This is performed on the binary map to fill holes and remove noise elements. The closing or filling operation is a combination of dilation and erosion operations performed stepwise in that order. We use a circular structuring element (B) of dimension 8 to perform this step on the binarized image (A). The closing operation is represented by the following eqn. 3.16:

$$A \bullet B = (A \oplus B) \ominus B \tag{3.16}$$

where represents $\oplus$ dilation and $\ominus$ represents erosion.

b) **Removal of connected components associated with the image border**- Connected-component labeling is an algorithmic application of graph theory, where subsets of connected components are uniquely labeled based on a given heuristic. It is used to detect connected regions in binary images. This step, in some cases, helps to exclude shore components connected to the border from the binary map.

We further impose constraints on each of the following region properties, which can be considered as parameters, and their values have been empirically determined by experimenting over the entire test dataset. They are as follows:

c) The **area** of the region (A) in pixels is in the range $300 \leq A \leq 40000$ to approximately limit the area between the smallest and largest pixel dimensions, respectively, of the target objects in our database.

d) **Eccentricity** $e = \frac{Distancebetweenfoci}{Majoraxislength}$, is a measure of the deviation of shape from circularity such that for a circle $e = 0$. An ellipse whose eccentricity is 1 is a line segment. We restrict this measure as $0.9 \leq e \leq 1$ based on the rationale that values closer to 1 would indicate long, thin objects likely to be logs. This restriction would prevent us from considering logs appearing at a vertical angle from the camera as the eccentricity in such a case would be close to zero.

e) **Solidity(s)** is the measure of the convexity (or concavity) of a shape defined by $s = A_s/H$ where A_s represents the area of the shape and H is the convex hull area. Restricting its value range as $0.5 \leq s \leq 1$ helps to remove component regions with very rough edges, which are unlikely to be our targets.

f) **Euler number (EN)** indicates the difference between number of contiguous parts (P) and number of hole regions (D) such that $EN = P - D$. Its value in our case must be 1 as the logs to be detected generally do not have large hole regions.

The table 3.1 show results from the region analysis step performed on binary maps obtained as a result of applying the proposed method and post-processing steps on a few sample image frames (figure 3.10). The **area**, **eccentricity**, **Euler number** and **solidity** measures calculated for all objects detected in each frame are tabulated. Test frames 3 and 4 3.10f show multiple objects detected, including false detections. In table 3.1, rows 3(a), 3(b) and 3(c) correspond to the leftmost, middle and rightmost objects, while rows 4(a) and 4(b) correspond to the top and bottom objects respectively.

The final objective of the detection system is to generate an alert once an obstacle is located. To achieve that, in our algorithm we evaluate a series of 10 successive frames at a time. If an obstacle is detected consistently over all 10 frames, an alert flag is set to be high.

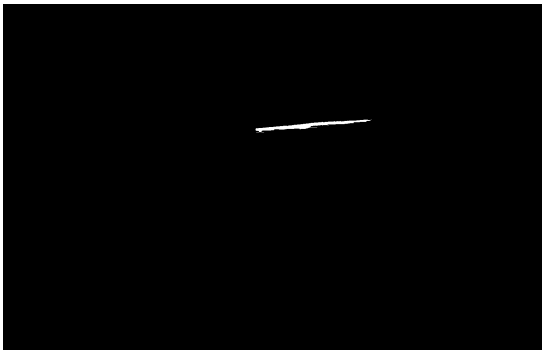
Table 3.1: Region analysis results after post-processing

Frame	Area	Eccentricity	Euler No	Solidity
1	1714	0.9996	1	0.6892
2	4023	0.9995	1	0.6868
3	a)469	0.9973	1	0.6523
	b)445	0.9903	1	0.6825
	c)596	0.9957	1	0.8187
4	a) 463	0.9960	1	0.6512
	b) 2158	0.9989	1	0.5930

(a) Test frame 1

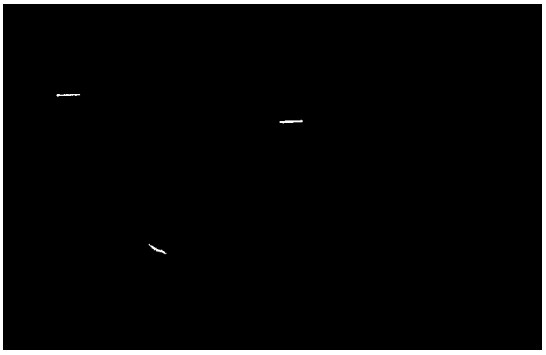
(b) Test frame 1 after binarization and post-processing

(c) Test frame 2



(d) Test frame 2 after binarization and post-processing

(e) Test frame 3



(f) Test frame 3 after binarization and post-processing

(g) Test frame 4

(h) Test frame 4 after binarization and post-processing

Figure 3.10: Morphological post-processing performed on binary image frames

4 Evaluation and Results

4.0.1 Experimental Database

The data used in this research consist of videos and images acquired using a forward looking monocular camera (Canon 7D, using a 15-85mm Canon stabilized lens) from the bow of a sailing vessel (figure 4.1). The data was collected and kindly provided for this research by Eduard Wisernig, from the Department of Computer Science, University of Victoria.

Figure 4.1: Sailing vessel “Moonshadow”

The experimental database comprises a set of 100 training frames per class and 2000 test frames with high spatial resolution of 1920×1080, obtained from 5 different videos sequences of at least 3 minutes length each. Table 4.1 provides details of the dataset construction.

Table 4.1: Dataset construction

Set No.	No. of frames	Background conditions
1	400	Sunny daylight
2	400	Normal daylight
3	400	Subdued daylight/twilight
4	400	Dark/cloudy
5	400	Normal to cloudy daylight

4.0.2 The Linear Regression Model

The Linear Regression model obtained from the predictive mapping step (see Section 3.2.3) of the algorithm has 5 coefficients (shown in Table 4.2) such that the regression equation can be formulated as (eqn. 4.1), where y is the response variable, (x_1,...x_5) are predictor variables and (β_0,...β_5) are the coefficients while ϵ is the intercept.

Estimates of the coefficients and F-statistic vs. constant model (F) are tabulated in

table 4.2. F calculated on the regression model tests for a significant linear regression relationship between the response and predictor variables. The higher values of x_1 , x_2 and x_5 thus show a more significant relationship with y as compared to the other predictors x_3 and x_4.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \epsilon \tag{4.1}$$

Table 4.2: Regression coefficients

Coefficient	Estimate	F
β	241.92	-
x_1	-0.34721	1.2817e+05
x_2	0.91038	20255
x_3	-0.32175	4875.2
x_4	0.053693	880.14
x_5	9.9088	8545.1

An intercept of 241.92 is estimated along with the coefficients to the model. The table 4.3 tabulates the values of obtained error measures. The root mean squared error ($RMSE$) is 0.387, which is the square root of the mean squared error, and estimates the standard deviation of the error distribution. This value is obtained for a normalized range [0,1], and lower values indicate a better fit. The R-squared ($R^2 = 0.758$) measure

is the coefficient of determination, calculated as the square of the sample correlation coefficient (R) between the outcomes and their predicted values. It can be interpreted as the percentage of the response variable variation that is explained by a linear model and thus explains approximately 75% of the variability in the response variable y around its mean. In general a higher R-squared value indicates a better fit of the model to the data. A 100% variance would mean that the fitted values always equal the observed values and all data points fall on the regression line. The variability and error obtained for our model suggests it to be a moderately good fit for generalizing over the test data, but could perhaps benefit from a larger and more representative training data set.

Table 4.3: Error measures

Error measure	R^2	$RMSE$
Value	0.758	0.387

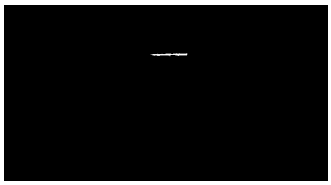
4.0.3 Evaluation and Discussion

We have designed the obstacle detection as a one-vs-all classification problem.The training set consists of 100 images per class to train a binary classifier to recognize logs belonging to target class while the other class consists of sky, shore and water samples. The test set is constructed using 2000 image frames extracted from 5 different video sequences representing a variety of background and lighting conditions (see table 4.1).

Performance of the proposed method is highly dependent on the characteristics of the test images. For this reason, the test frames were divided into 5 different sets of

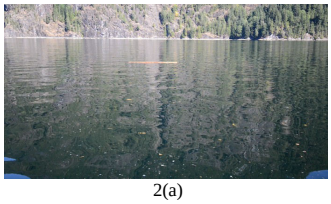
400 frames each, with each set representing different background and/or illumination conditions (please see table 4.1). Set 1 (see figure 4.1) consists of frames which show a log present at close proximity to the shore. The water almost entirely reflects the shore. The appearance of water in set 2 varies within each image frame, with large salient patches which differ from the rest of the water. Set 3 consists of frames with a prominent shoreline and low level of illumination. The irregular horizon line in this set makes it especially difficult to identify and outline. Set 4 is representative of cloudy conditions with the water appearing darker in colour than in other sets. The log in set 5 is further away from the camera and thus hard to detect. Examples of detection results (marked by red bounding boxes) are shown in figure 4.1. Correct detection results are highlighted by red solid line arrows while false detections are shown with broken line arrows. In the example detection results in figure 4.1, the logs are detected accurately in all of the sample images, however, figure 4.1.7(c), 4.1.8(c), 4.1.9(c) and 4.1.13(c) show some detected false positives caused by ripples in water and part of the shore.

1(a)



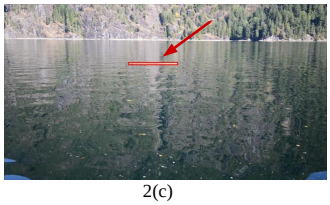
1(b)

1(c)

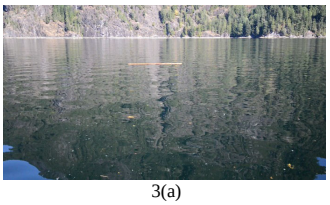


2(a)

2(b)



2(c)

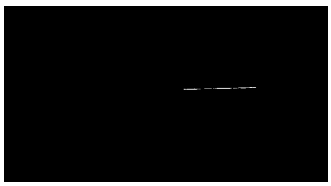


3(a)

3(b)

3(c)

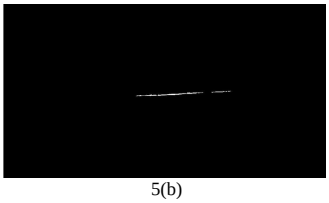
4(a)



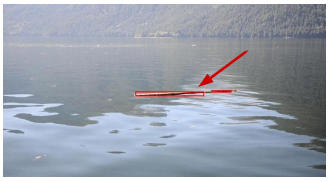
4(b)

4(c)

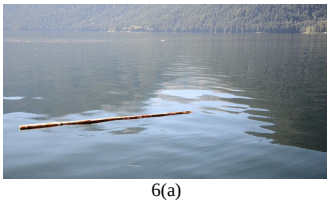
5 (a)



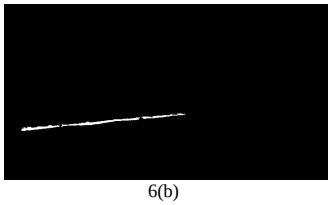
5(b)



5(c)

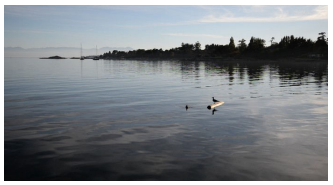


6(a)

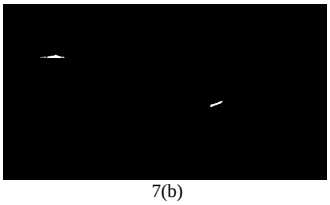


6(b)

6(c)



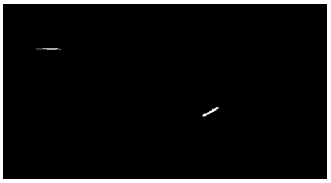
7(a)



7(b)

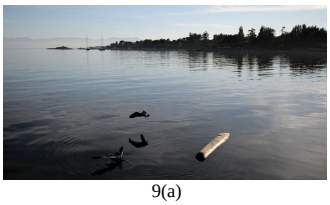
7(c)

8(a)

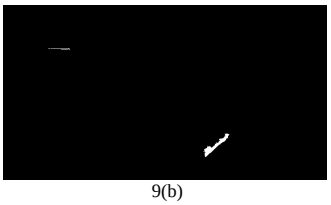


8(b)

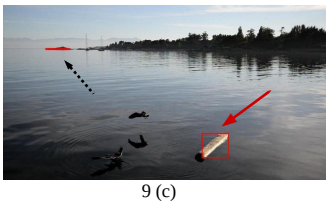
8 (c)



9(a)

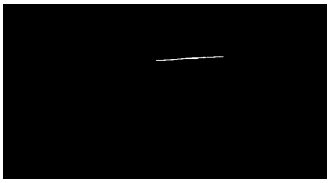


9(b)



9 (c)

10 (a)



10(b)

10(c)

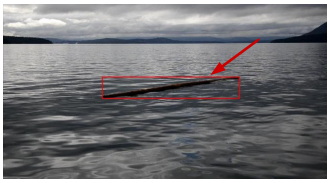
11(a)

11 (b)

11(c)

12(a)

12(b)



12(c)

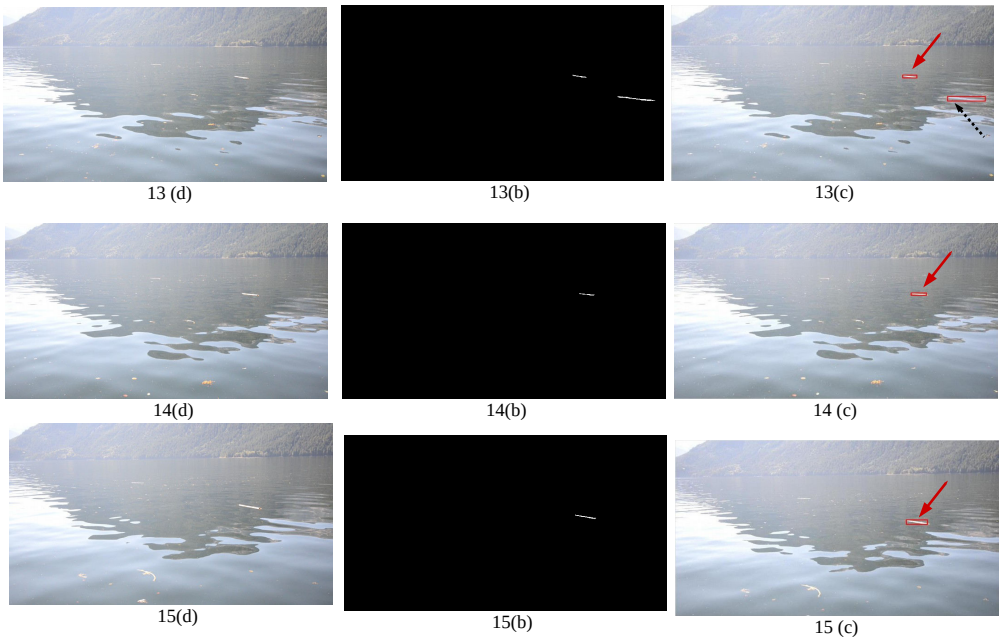


Figure 4.1.: Left hand column shows original image frame, middle frame shows binary image and right hand column shows detection results highlighted by red bounding boxes. 4.1.1(a)-4.1.3(c) Set 1 consists of frames where the shore is strongly reflected. 4.1.4.(a)-4.1.6(c) Set 2 image frames have large salient patches 4.1.7(a)- 4.1.9(c)Set 3 frames with strong reflections and varying illumination 4.1.10(a)-4.1.12(c) Set 4 with cloudy conditions 4.1.13(a)-4.1.15(c) Set 5 frame has small-sized obstacle further away from camera, thus hard to detect. [Correct detections shown by solid red line arrows, false detections shown by broken black line arrows.]

To analyze the performance of the supervised learning by linear regression module without the incorporation of saliency detection, we evaluate the precision (P) and recall (R) values for $\mu = 1$ condition. The table 4.4 presents the results of this analysis for each of the 5 test sets considered. The results show that the learning module by itself has an overall low precision and recall due to large number of false positives and false negative occurrences.

The results of saliency detection by MSSS only for the special case $\mu = 0$ has been provided in table 4.5 for comparison. The recall values for the regression module employed independently are seen to be comparatively higher while the precision values for MSSS are significantly higher for set 1.

Table 4.4: Detection results for regression ($\mu = 1$)

Set no.	Precision(P)	Recall(R)
1	44.6%	31.5%
2	71.2%	30.2%
3	14.8%	6.00%
4	25.8%	2.53%
5	34.7%	6.73%

Table 4.5: Detection results for MSSS ($\mu = 0$)

Set no.	Precision(P)	Recall(R)
1	90.0%	11.2%
2	56.0%	29.7%
3	13.3%	0.25%
4	23.5%	0.17%
5	18.5%	4.25%

The P and R measures are given by :

$$Precision(P) = \frac{(TP)}{(TP) + (FP)} \tag{4.2}$$

$$Recall(R) = \frac{(TP)}{(TP) + (FN)} \tag{4.3}$$

where TP and FP denote True Positives and False Positives respectively, while FN denotes False Negatives.

4.0.4 Comparative Analysis

In the evaluation phase, the proposed method is compared to three other state-of-the-art supervised methods at the classification step, a feed-forward neural network (NN), a Gaussian Mixture Model (GMM) and the Adaboost ensemble classifier. The saliency

detection step (MSSS) has been applied consistently in combination with all of the classification methods as it was found to yield good detection results for our dataset. This is done to determine the best predictive method to use in conjunction with the saliency detection for obtaining higher accuracy and lower number of false alarms. The neural network(NN) ([72]) is trained to find a predicted probability map similar to the regression map R calculated in the previous section combined with the MSSS (optimal no. of hidden layers = 3, $\mu = 0.6$). The Gaussian Mixture Model (GMM) where model parameters are estimated by Maximum Likelihood Estimation [73] is similarly used in combination with the MSSS (maximum no. of iterations=100,$\mu = 0.7$). In a similar way, the posterior probability scores obtained for the pixel features of each test image on classification by Adaboost are used to construct a predicted probability map which is then combined with the MSSS method. The optimal parameters selected for the Adaboost+MSSS method are $\mu = 0.7$ and number of trees=50.

The test sets have been evaluated by precision (P) and recall (R) measures (eqns. 4.2,4.3).

True Positives in our case refer to logs present in the frame which are detected correctly. **False Positives** correspond to objects detected as logs that are not. **False Negatives** are the cases in which logs are present but not detected, in other words, missed detections.
From the application perspective the number of false negatives should be minimized to avoid potential collisions. Table 4.6 presents the evaluation results for our method in comparison to other supervised methods combined for predictive mapping, keeping the saliency detection step (MSSS) the same.

The results show comparable precision and higher recall values for the proposed method in most cases, in comparison to the other methods for the test dataset (see tables 4.6,

4.5, 4.4). For set 3, although the log is detected accurately, the number of false positives is high, causing the precision and recall to be much lower in comparison to the other sets. This occurs due to the detection of a part of the shore similar in appearance to the logs (see figure 4 (e)). This limitation may be addressed in a future implementation by accurately delineating water region from shore. Set 1 and 2 show the highest precision and recall values in most cases. In general, the linear regression classification yields less false positives and false negatives than the saliency detection module applied by itself (tables 4.5,4.4). However, saliency detection helps to suppress many of the false positives and false negatives(multiple false negatives may occur due to multiple logs present in the frame) that occur with the linear regression approach. For the proposed method, the average precision and recall obtained is maximum for the value of $\mu = 0.7$ at 75.96% and 84.44% respectively. This allows greater influence of the classification module in the combined approach. Some of the precision values obtained for the other classification methods combined with saliency are higher than those for the proposed method, for example, Adaboost+MSSS show significantly higher precision values for Set 1. However, the recall values of the proposed method in comparison are much higher for the set, which is desirable for our application for reducing false negatives.

The average precision and recall curves for each test set are computed and plotted against variations in values of the aggregator parameter μ to evaluate the performance of the proposed method compared with different classification methods in combination with the saliency detection (figure 4.3). The plots show highest precision and recall for the proposed method at $\mu = 0.7$, which is selected as the optimum value of the parameter. The NN+MSSS method has a peak precision and recall at $\mu = 0.6$ while the other compared methods (GMM+MSSS, Adaboost+MSSS) have highest precision and recall values at $\mu = 0.7$. The highest precision values are comparable for all the methods

compared as can be seen in figure 4.3f, the recall curve however shows significantly higher value at its peak and in general for the proposed method (figure 4.3f). This is an important indication of the success of the proposed method as higher recall is suggestive of higher number of relevant instances that are retrieved, or a relatively low number of false negatives compared to the number of true positives. A high precision, on the other hand, suggests a large number of retrieved instances that are relevant, or a low number of false positive detections. The recall measure is considered to be of more importance for the application as the objective is to achieve as many correct detections as possible, keeping the number of missed detections to a minimum.

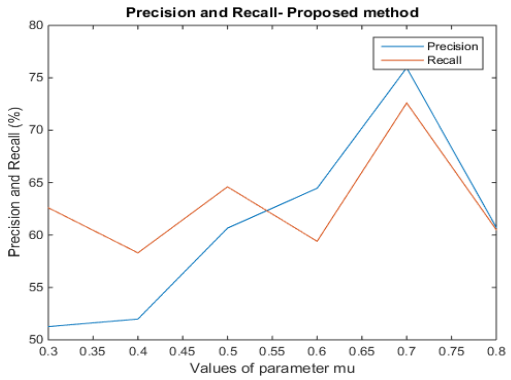
The average computational time for our algorithm is calculated to be 2.6 seconds per frame on a 1.8 GHz Intel Core i5-3337 processor using MATLAB®version 8.5. Keeping in mind that the application that the algorithm will be used to drive is to operate in real-time, the computational requirements must be kept minimum. The average computational time for each frame have been compared for each of the methods evaluated to deduce the best classifier for the application (shown in table 4.7). The proposed method is found to be faster in this per-frame evaluation of computational times. However, the proposed method's computational time is still considered high for a real-time system. This limitation is could be addressed in future implementations by code optimization and/or implementation in a programming language that supports fast processing speeds, such as C/C++ or in a parallel processing environment.

Table 4.7: Average computational time per frame (in seconds)

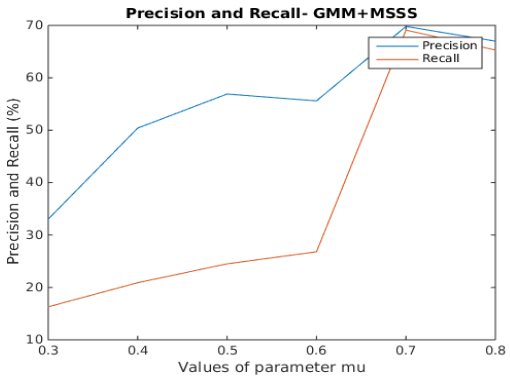
Method	Proposed method	GMM+MSSS	NN+MSSS	Adaboost+MSSS
Avg. Computational Time	2.64	5.97	9.61	7.39

Table 4.6: Experimental results

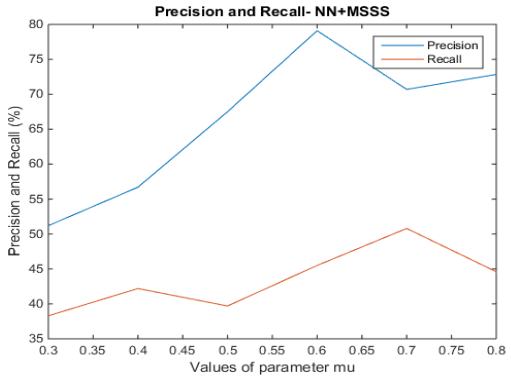
μ	Set no.	Proposed method		GMM+MSSS		NN+MSSS		Adaboost+MSSS	
		Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
0.3	1	94.9%	36.5%	50.8%	5.61%	20.2%	91.2%	66.6%	21.7%
	2	90.0%	58.5%	71.6%	57.7%	97.5%	40.0%	100%	49.7%
	3	16.3%	14.3%	1.92%	0.24%	7.69%	3.41%	7.39%	8.50%
	4	35.8%	17.4%	21.2%	1.75%	100%	20.5%	69.2%	8.81%
	5	25.8%	88.0%	19.6%	16.03%	30.5%	36.6%	66.3%	16.0%
0.4	1	89.9%	58.0%	88.0%	13.7%	21.8%	90.5%	100%	15.7%
	2	62.3%	67.0%	100%	63.3%	100%	50.5%	97.7%	50.7%
	3	15.4%	14.5%	4.32%	2.52%	2.54%	11.2%	8.13%	21.7%
	4	50.5%	23.0%	30.0%	5.81%	100%	18.2%	78.7%	37.2%
	5	41.8%	78.1%	29.9%	18.7%	59.3%	40.5%	67.7%	29.5%
0.5	1	89.0%	61.5%	91.7%	14.5%	100%	55.5%	100%	16.0%
	2	77.2%	67.0%	95.8%	66.0%	98.1%	75.0%	100%	42.8%
	3	15.4%	35.7%	3.51%	2.52%	18.9%	5.52%	7.32%	21.7%
	4	60.1%	47.7%	40.4%	22.5%	98.0%	25.0%	73.7%	42.7%
	5	61.6%	57.0%	53.4%	16.8%	22.7%	37.5%	52.7%	9.75%
0.6	1	89.0%	88.0%	18.8%	16.5%	100%	87.7%	100%	1.52%
	2	68.0%	80.0%	85.7%	67.5%	74.6%	31.0%	98.4%	16.0%
	3	14.9%	24.3%	4.37%	2.52%	100%	5.51%	22.7%	21.6%
	4	61.5%	48.0%	46.34%	33.5%	52.1%	88.0%	83.3%	42.7%
	5	88.9%	82.1%	53.4%	14.2%	68.8%	15.5%	68.0%	81.50%
0.7	1	88.4%	95.0%	88.0%	22.6%	71.2%	77.6%	100%	87.0%
	2	97.0%	85.6%	91.0%	93.2%	94.8%	31.0%	100%	47.1%
	3	14.9%	58.0%	4.37%	88.3%	8.60%	60.1%	34.0%	20.7%
	4	85.9%	93.6%	82.4%	78.1%	98.6%	72.0%	87.6%	43.5%
	5	93.6%	90.0%	83.5%	63.0%	80.5%	13.4%	75.3%	16.0%
0.8	1	70.2%	94.3%	80.8%	14.3%	100%	78.0%	79.0%	84.2%
	2	89.7%	79.5%	78.2%	84.0%	74.2%	31.0%	16.9%	48.0%
	3	12.3%	75.2%	8.1%	87.2%	27.1%	62.4%	27.3%	16.4%
	4	55.4%	78.6%	79.8%	86.2%	78.0%	34.5%	58.6%	39.4%
	5	76.0%	45.5%	58.2%	55.0%	39.9%	17.3%	81.2%	12.3%



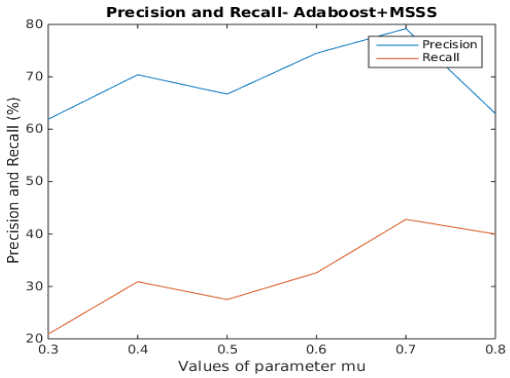
(a) Proposed method



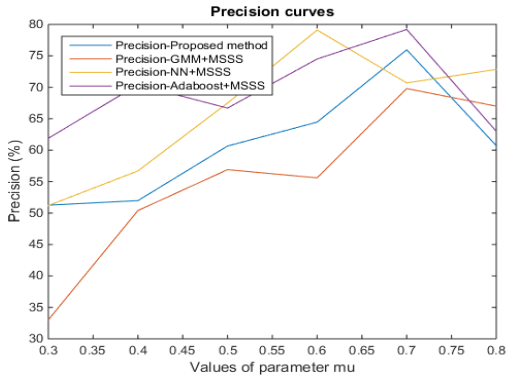
(b) GMM+MSSS



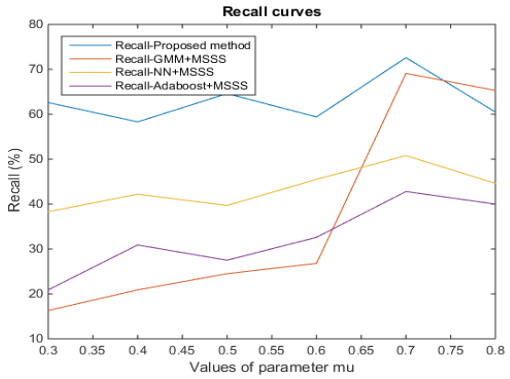
(c) NN+MSSS



(d) Adaboost+MSSS



(e) Precision curves



(f) Recall curves

Figure 4.3: Comparison of precision and recall curves for different methods

5 Conclusions

In this study, a novel method of achieving obstacle detection (specifically logs) and alert generation is presented along with the results of performance evaluation. The algorithm has been designed keeping in mind the requirements for deployment on a real-time system for sailboat navigation. In order to achieve the maximum number of accurate detections and more importantly the minimum number of missed detections, frame-by-frame processing is done. The outlook for the presented algorithm is integration into an autonomous system that is well-suited to navigation of a sailboat close to shores or open sea environments.

The major contribution of this thesis is an algorithm developed to solve the task of obstacle detection for vision-based autonomous water navigation. Our model can be considered to be a basic representation of biological vision as a combination of high-level cognitive and low-level attention mechanisms, through regression-based learning and saliency detection respectively.

Our method draws inspiration from the study of various obstacle avoidance methods spanning land, air and water. Saliency detection used in conjunction with a recognition module is shown to have a reasonable success rate for obstacle detection. A supervised

linear regression method for target recognition along with the Maximum Symmetric Surround Saliency detection algorithm is found to achieve a high success rate in comparison to other classification methods.

The key advantages of the algorithm are as follows:

A. Using the stateless frame-based approach wherein each frame from a video input is analyzed individually has the advantage of zero dependency on previous frame inputs. This allows fast processing and robustness to failures in case of false detections or missed detections.
B. Low complexity of computation of low-level features (intensity, color and texture) and algorithm design allows for lower computational times and easier integration into hardware.
C. The recall values for the proposed method were higher on average in comparison to other methods evaluated.

The identified limitations are:
A. Due to our shape constraints, we are not able to consider logs which may be at a perpendicular orientation to the image plane. The eccentricity measure would most likely eliminate such a detection as the value can be expected to be closer to 0 than 1.
B. The Maximum Symmetric Surround Saliency method assigns higher saliency values to more centrally located pixels in an image frame. While this assumption might be true from the saliency computation perspective, it does not account for logs that may occur away from the centre, nearer to the image borders.

From the results of evaluation we show that our method is relevant for the application of log detection and could be extended to obstacle detection in general for future

applications. The comparison of performance of the linear regression with other supervised machine learning methods at the classification step establishes the suitability of the proposed algorithm for the application. This research has been presented at the 13th Canadian Conference on Computer and Robot Vision (CRV), 2016, and awaits publication in the conference proceedings [74].

5.1 Future Directions

In its current implementation, our method performs only alert generation through object detection. A tracking module can be integrated into the system in a future implementation to detect the exact position of the obstacle at each frame. Tracking an object may be performed from frame-by-frame detection, or involve more efficient ways of predicting the new location of the object being tracked in the next frame using estimated dynamics. Even for dynamic tracking, robust detection is required as a primary step.

For the aggregation step (section 3.1.5) in the proposed algorithm, more complex and efficient image fusion techniques could be used instead of a fixed parameter. This would ensure a more automated approach and better detection.

The training and test set could be augmented for the classification module for further evaluations. Also, different classification and/or saliency approaches could be considered for enhancing performance, leaving scope for further research.

For an intended real-time implementation, using a GPU for performing some of the image processing tasks might significantly improve computational efficiency and reduce computational times.

We could also consider eliminating the shoreline from the frames before the detection process by integrating location information from standard terrestrial charts.

Bibliography

[1] P. Pilotte. Analytics-driven embedded systems part2, developing analytics and prescriptive controls. https://channels.theinnovationenterprise.com/articles/analytics-driven-embedded-systems-part-2-developing-analytics-and-prescriptive-controls. [Online; accessed February 2016].

[2] Y. Jia and M. Han, "Category-independent object-level saliency detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 1761–1768.

[3] R. Achanta and S. Süsstrunk, "Saliency detection using maximum symmetric surround," in *Image Processing (ICIP), 2010 17th IEEE International Conference on*. IEEE, 2010, pp. 2653–2656.

[4] D. A. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neural Computation*, vol. 3, no. 1, pp. 88–97, 1991.

[5] J. Guivant, E. Nebot, and S. Baiker, "Autonomous navigation and map building using laser range sensors in outdoor applications," *Journal of robotic systems*, vol. 17, no. 10, pp. 565–583, 2000.

[6] F. D. Snyder, D. D. Morris, P. H. Haley, R. T. Collins, and A. M. Okerholm, "Autonomous river navigation," in *Optics East*. International Society for Optics and Photonics, 2004, pp. 221–232.

[7] R. B. Matthew D. Plumlee, Roland Arsenault and C. Ware, "Chart of the future project: Maximizing mariner effectiveness through fusion of marine & visualization technologies," Washington, DC, November 16 - November 17 2004. [Online]. Available: www.chc2012.ca

[8] V. Graefe and K.-D. Kuhnert, "Vision-based autonomous road vehicles," in *Vision-based Vehicle Guidance*. Springer, 1992, pp. 1–29.

[9] Y. He, H. Wang, and B. Zhang, "Color-based road detection in urban traffic scenes," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 5, no. 4, pp. 309–318, 2004.

[10] S. Glaser, B. Vanholme, S. Mammar, D. Gruyer, and L. Nouveliere, "Maneuver-based trajectory planning for highly autonomous vehicles on real road with traffic and driver interaction," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 11, no. 3, pp. 589–606, 2010.

[11] M. Bertozzi, A. Broggi, and A. Fascioli, "Vision-based intelligent vehicles: State of the art and perspectives," *Robotics and Autonomous systems*, vol. 32, no. 1, pp. 1–16, 2000.

[12] J. Valasek, K. Gunnam, J. Kimmett, J. L. Junkins, D. Hughes, and M. D. Tandale, "Vision-based sensor and navigation system for autonomous air refueling," *Journal of Guidance, Control, and Dynamics*, vol. 28, no. 5, pp. 979–989, 2005.

[13] J. Conroy, G. Gremillion, B. Ranganathan, and J. S. Humbert, "Implementation of wide-field integration of optic flow for autonomous quadrotor navigation," *Autonomous robots*, vol. 27, no. 3, pp. 189–198, 2009.

[14] L. Fletcher, N. Apostoloff, J. Chen, and A. Zelinsky, "Computer vision for vehicle monitoring and control," in *Proc. Australian Conference on Robotics and Automation*, 2001, pp. 67–72.

[15] K. Yamaguchi, T. Kato, and Y. Ninomiya, "Moving obstacle detection using monocular vision," in *Intelligent Vehicles Symposium, 2006 IEEE*. IEEE, 2006, pp. 288–293.

[16] I. Ulrich and I. Nourbakhsh, "Appearance-based obstacle detection with monocular color vision," in *AAAI/IAAI*, 2000, pp. 866–871.

[17] G. Hee Lee, F. Faundorfer, and M. Pollefeys, "Motion estimation for self-driving cars with a generalized camera," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2746–2753.

[18] T. G. McGee, R. Sengupta, and K. Hedrick, "Obstacle detection for small autonomous aircraft using sky segmentation," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 4679–4684.

[19] A. Elfes, "Sonar-based real-world mapping and navigation," *Robotics and Automation, IEEE Journal of*, vol. 3, no. 3, pp. 249–265, 1987.

[20] H. K. Heidarsson and G. S. Sukhatme, "Obstacle detection and avoidance for an autonomous surface vehicle using a profiling sonar," in *Robotics and Automation*

(ICRA), 2011 IEEE International Conference on. IEEE, 2011, pp. 731–736.

[21] S. Fefilatyev, "Detection of marine vehicles in images and video of open sea," Ph.D. dissertation, University of South Florida, 2008.

[22] P. Mulassano and L. L. Presti, "Object detection on the sea surface, based on texture analysis," in *Electronics, Circuits and Systems, 1999. Proceedings of ICECS'99. The 6th IEEE International Conference on*, vol. 2. IEEE, 1999, pp. 855–858.

[23] A. Borghgraef, O. Barnich, F. Lapierre, M. Van Droogenbroeck, W. Philips, and M. Acheroy, "An evaluation of pixel-based methods for the detection of floating objects on the sea surface," *EURASIP Journal on Advances in Signal Processing*, vol. 2010, p. 5, 2010.

[24] V. Mahadevan and N. Vasconcelos, "Background subtraction in highly dynamic scenes," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on.* IEEE, 2008, pp. 1–6.

[25] T. El-Gaaly, C. Tomaszewski, A. Valada, P. Velagapudi, B. Kannan, and P. Scerri, "Visual obstacle avoidance for autonomous watercraft using smartphones," in *Proceedings of the Autonomous Robots and Multirobot Systems workshop (ARMS 2013, at AAMAS 2013).*

[26] E. Wisernig, T. Sadhu, C. Zilinski, B. Wyvill, A. Branzan Albu, and M. Hoeberechts, "Augmented reality visualization for sailboats (ARVS)," in *2015 International Conference on Cyberworlds (CW).* IEEE, 2015, pp. 61–68.

[27] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection,"

Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 20, no. 1, pp. 23–38, 1998.

[28] E. Osuna, R. Freund, and F. Girosi, "Training support vector machines: an application to face detection," in *Computer vision and pattern recognition, 1997. Proceedings., 1997 IEEE computer society conference on.* IEEE, 1997, pp. 130–136.

[29] M. Vidal-Naquet and S. Ullman, "Object recognition with informative features and linear classification." in *ICCV*, vol. 3, 2003, p. 281.

[30] K. Okuma, A. Taleghani, N. De Freitas, J. J. Little, and D. G. Lowe, "A boosted particle filter: Multitarget detection and tracking," in *Computer Vision-ECCV 2004*. Springer, 2004, pp. 28–39.

[31] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.

[32] Y. LeCun, K. Kavukcuoglu, C. Farabet *et al.*, "Convolutional networks and applications in vision." in *ISCAS*, 2010, pp. 253–256.

[33] D. Kim, J. Sun, S. M. Oh, J. M. Rehg, and A. F. Bobick, "Traversability classification using unsupervised on-line visual learning for outdoor robot navigation," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 2006, pp. 518–525.

[34] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. L. Cun, "Off-road obstacle avoidance through end-to-end learning," in *Advances in neural information processing systems*,

2005, pp. 739–746.

[35] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G. R. Bradski, "Self-supervised monocular road detection in desert terrain." in *Robotics: science and systems*, vol. 38. Philadelphia, 2006.

[36] M. Mancini, G. Costante, P. Valigi, and T. A. Ciarfuglia, "Fast robust monocular depth estimation for obstacle detection with fully convolutional networks," *arXiv preprint arXiv:1607.06349*, 2016.

[37] J. Michels, A. Saxena, and A. Y. Ng, "High speed obstacle avoidance using monocular vision and reinforcement learning," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 593–600.

[38] I. Naseem, R. Togneri, and M. Bennamoun, "Linear regression for face recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 11, pp. 2106–2112, 2010.

[39] D. Meger, P.-E. Forssén, K. Lai, S. Helmer, S. McCann, T. Southey, M. Baumann, J. J. Little, and D. G. Lowe, "Curious george: An attentive semantic robot," *Robotics and Autonomous Systems*, vol. 56, no. 6, pp. 503–511, 2008.

[40] J. Morén, A. UDE, A. Koene, and G. Cheng, "Biologically based top-down attention modulation for humanoid interactions," *International Journal of Humanoid Robotics*, vol. 5, no. 01, pp. 3–24, 2008.

[41] Y. Yu, G. K. Mann, and R. G. Gosine, "A task-driven object-based attention model for robots," in *Robotics and Biomimetics, 2007. ROBIO 2007. IEEE International*

Conference on. IEEE, 2007, pp. 1751–1756.

[42] P. Santana, M. Guedes, L. Correia, and J. Barata, "A saliency-based solution for robust off-road obstacle detection," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on.* IEEE, 2010, pp. 3096–3101.

[43] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 11, pp. 1254–1259, 1998.

[44] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 9, pp. 1627–1645, 2010.

[45] P. Viola and M. Jones, "Robust real-time object detection," *International Journal of Computer Vision*, vol. 4, 2001.

[46] A. Borji, "Boosting bottom-up and top-down visual features for saliency estimation," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on.* IEEE, 2012, pp. 438–445.

[47] T. Judd, K. Ehinger, F. Durand, and A. Torralba, "Learning to predict where humans look," in *Computer Vision, 2009 IEEE 12th international conference on.* IEEE, 2009, pp. 2106–2113.

[48] M. Cerf, J. Harel, W. Einhäuser, and C. Koch, "Predicting human gaze using low-level saliency combined with face detection," in *Advances in neural information processing systems*, 2008, pp. 241–248.

[49] L. Zhang, M. H. Tong, T. K. Marks, H. Shan, and G. W. Cottrell, "SUN: A Bayesian framework for saliency using natural statistics," *Journal of vision*, vol. 8, no. 7, p. 32, 2008.

[50] J. Li, Y. Tian, T. Huang, and W. Gao, "Multi-task rank learning for visual saliency estimation," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 21, no. 5, pp. 623–636, 2011.

[51] M. Kümmerer, L. Theis, and M. Bethge, "Deep gaze I: Boosting saliency prediction with feature maps trained on imagenet," *ICLR 2015*, forthcoming.

[52] R. Zhao, W. Ouyang, H. Li, and X. Wang, "Saliency detection by multi-context deep learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1265–1274.

[53] D. Gao, S. Han, and N. Vasconcelos, "Discriminant saliency, the detection of suspicious coincidences, and applications to visual recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 6, pp. 989–1005, 2009.

[54] D. Gao, V. Mahadevan, and N. Vasconcelos, "The discriminant center-surround hypothesis for bottom-up saliency," in *Advances in neural information processing systems*, 2008, pp. 497–504.

[55] L. Elazary and L. Itti, "Interesting objects are visually salient," *Journal of vision*, vol. 8, no. 3, pp. 3–3, 2008.

[56] J. Zhang and S. Sclaroff, "Saliency detection: A boolean map approach," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 153–160.

[57] L. Itti and P. Baldi, "A principled approach to detecting surprising events in video," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 631–637.

[58] X. Hou and L. Zhang, "Saliency detection: A spectral residual approach," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE, 2007, pp. 1–8.

[59] R. Achanta, S. Hemami, F. Estrada, and S. Susstrunk, "Frequency-tuned salient region detection," in *Computer vision and pattern recognition, 2009. CVPR 2009. IEEE conference on*. IEEE, 2009, pp. 1597–1604.

[60] L.-K. Wong and K.-L. Low, "Saliency-enhanced image aesthetics class prediction," in *Image Processing (ICIP), 2009 16th IEEE International Conference on*. IEEE, 2009, pp. 997–1000.

[61] J. Han, K. N. Ngan, M. Li, and H.-J. Zhang, "Unsupervised extraction of visual attention objects in color images," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 16, no. 1, pp. 141–145, 2006.

[62] S. Engel, X. Zhang, and B. Wandell, "Colour tuning in human visual cortex measured with functional magnetic resonance imaging," *Nature*, vol. 388, no. 6637, pp. 68–71, 1997.

[63] R. M. Haralick, "Statistical and structural approaches to texture," *Proceedings of the IEEE*, vol. 67, no. 5, pp. 786–804, 1979.

[64] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture mea-

sures with classification based on featured distributions," *Pattern recognition*, vol. 29, no. 1, pp. 51–59, 1996.

[65] X. Chai, S. Shan, X. Chen, and W. Gao, "Locally linear regression for pose-invariant face recognition," *Image Processing, IEEE Transactions on*, vol. 16, no. 7, pp. 1716–1725, 2007.

[66] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[67] S. M. Stigler, "Gauss and the invention of least squares," *The Annals of Statistics*, pp. 465–474, 1981.

[68] S. Goferman, L. Zelnik-Manor, and A. Tal, "Context-aware saliency detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 10, pp. 1915–1926, 2012.

[69] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2. IEEE, 2005, pp. 60–65.

[70] Y. Chen, W. Chen, X. Yin, X. Ye, X. Bao, L. Luo, Q. Feng, X. Yu *et al.*, "Improving low-dose abdominal ct images by weighted intensity averaging over large-scale neighborhoods," *European Journal of Radiology*, vol. 80, no. 2, pp. e42–e49, 2011.

[71] M. I. Smith and J. P. Heather, "A review of image fusion technology in 2005," in *Defense and Security*. International Society for Optics and Photonics, 2005, pp. 29–45.

[72] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by

back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[73] G. McLachlan and D. Peel, *Finite mixture models*. John Wiley & Sons, 2004.

[74] T. Sadhu, A. Branzan Albu, M. Hoeberechts, E. Wisernig, and B. Wyvill, "Obstacle detection for image-guided water navigation," in *13th Canadian Conference on Computer and Robot Vision (CRV)*. IEEE, 2016, (in press).