University of Southern Queensland

Faculty of Health, Engineering & Sciences

Automated Shark Detection using Computer Vision

A dissertation submitted by

K. Byles

in fulfilment of the requirements of

ENG4112 Research Project

towards the degree of

Bachelor of Mechatronic Engineering (Honours)

Submitted: October, 2016

Abstract

With the technological advancements of UAVs, researchers are finding more ways to harness their capabilities to reduce expenses in everyday society. Machine vision is at the forefront of this research and in particular image recognition. Training a machine to identify objects and differentiate them from others plays an integral role in the advancement of artificial intelligence. This project aims to design an algorithm capable of automatically detecting sharks from a UAV. Testing is performed by post-processing aerial footage of sharks taken from helicopters and drones, and analysing the reliability of the algorithm.

Initially this research project involved analysing aerial photography of sharks, dissecting the images into the individual colour channels that made up the RGB and HSV colour spaces and identifying methods to detect the shark blobs. Once an adaptive threshold of the brightness channel was designed, filters were curated specific to the environments presented in the obtained aerial footage to reject false positives. These methods were considerably successful in both rejecting false positives and consistently detecting the sharks in the video feed.

The methods produced in this dissertation leave room for future work in the shark detection field. By acquiring more reliable data, improvements such as using a kalman filter to detect and track moving blobs could be implemented to produce a robust shark detection and tracking system.

University of Southern Queensland

Faculty of Health, Engineering & Sciences

ENG4111/2 *Research Project*

Limitations of Use

The Council of the University of Southern Queensland, its Faculty of Health, Engineering & Sciences, and the staff of the University of Southern Queensland, do not accept any responsibility for the truth, accuracy or completeness of material contained within or associated with this dissertation.

Persons using all or any part of this material do so at their own risk, and not at the risk of the Council of the University of Southern Queensland, its Faculty of Health, Engineering & Sciences or the staff of the University of Southern Queensland.

This dissertation reports an educational exercise and has no purpose or validity beyond this exercise. The sole purpose of the course pair entitled "Research Project" is to contribute to the overall education within the student's chosen degree program. This document, the associated hardware, software, drawings, and other material set out in the associated appendices should not be used for any other purpose: if they are so used, it is entirely at the risk of the user.

Dean

Faculty of Health, Engineering & Sciences

Certification of Dissertation

I certify that the ideas, designs and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.
I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

K. Byles
0061009769

Acknowledgments

I would like to thank my friends and family for providing positive criticism on my work throughout the duration of this project. Dr Tobias Low was always able to offer constructive feedback, and didn't leave me feeling abandoned or lost in any way.

Most of all I'd like to acknowledge my loving girlfriend Katherine, for I would not be where I am today without her undoubted support.

K. Byles

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Background

Shark attacks have haunted every swimmer at some point in their life. The ominous shadows of the deep lurking below them, the object that brushed past their feet or the cool chill of something not being right has often reminded people of the horror of being a big fish's lunch. In the last 100 years in Australia alone there have been 543 unprovoked shark attacks, with 135 of those people losing their lives (Taronga 2016). Although for a statistician those aren't overwhelming odds, the long-term trend indicates an increasing number of shark attacks which is most likely due to an increasing number of people swimming at the beach. To try and reduce the frequency of shark attacks there have been a few control methods put in place such as:

- Shark nets
  - These are used to entangle sharks and prevent them from reaching swimmers and surfers. The nets are typically placed 500 metres offshore and are 150m wide and 6m tall (Sealifetrust 2015) and submerged in water of depths around 10m. Shark nets have been known to kill marine life such as dolphins and turtles (Sumpton, Taylor, Gribble, Mcpherson & Ham 2011)

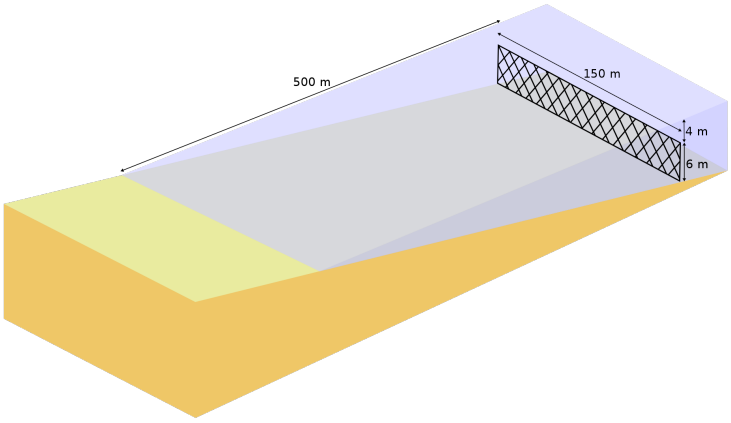

Figure 1.1: Diagram of shark net.

- Drumline
  - Consists of baited hooks that are set in waters 6m to 12m deep and approximately 600m offshore (Sumpton et al. 2011). The hooks are checked and re-baited every 15-20 days by contracted fisherman, who thoroughly record each catch.

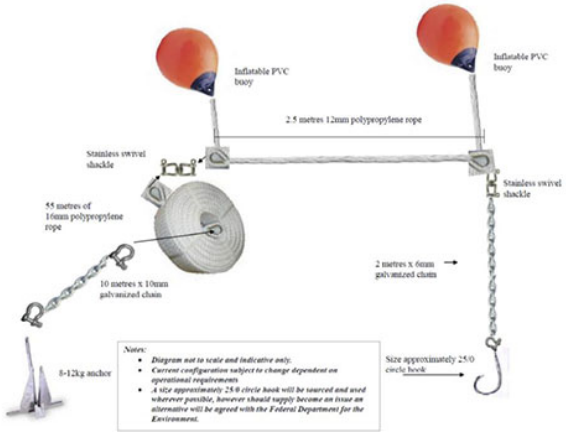

Figure 1.2: Diagram of shark drumline.

- Aerial Patrol
  - These have been used to monitor the beaches from small planes and helicopters. A recent study in 2014 recorded the success rate of identifying artificial shark analogues placed at an average 2.5m below the surface of the ocean was 12.5% for fixed-wing and 17.1% for helicopters (Robbins, Peddemors, Kennelly & Ives 2014). A typical aerial patrol can involve a sweep of nearly 300km of coastline, where the plane or helicopter is only offering a couple minutes of real-time security for popular beaches(Robbins et al. 2014).

Figure 1.3: Shark seen off Boudlers beach near Ballina from a helicopter.

- Tagging
  - Sharks are tagged and monitored over a certain time period, where the data obtained is typically used to determine migration patterns. Other useful data can be the depth at what the shark dives to for a given time period and the temperature of the water (Holmes, Pepperell, Griffiths, Jaine, Tibbetts & Bennett 2014). Tagged sharks can offer real-time tracking (News.com.au 2015) but it is not feasible to track down every shark and install a tag on them.

Figure 1.4: A great white shark tagged with both acoustic (front) and pop-up satellite (rear) tags. The acoustic tag is detected when the shark swims within 250m of a listening station, while the pop-up satellite tag records information about location, temperature and depth - and relays it to the laboratory when the tag releases itself from the shark.

- Spotters
  - Professional spotters that scan the water from a high point such as a cliff or tower and radio back to the control station if a shark is spotted.

In light of these expensive control methods in place, shark attacks are still occurring. To combat this the author is proposing a shark detection system based on aerial photography, which could greatly reduce funding needed for shark control methods and offer real-time surveillance and alerts of sharks in dangerous or threatening positions.

1.2 Computer Vision

Computer vision is a broad topic that involves many areas and is apart of everyday life. Medical professionals use computer vision for x-rays and MRI's to better diagnose their patients. Architects and engineers use computer vision for 3D modelling. Toll points and parking lots use computer vision to recognise license plates on vehicles. Production plants use computer vision for automatic inspection of goods on assembly lines to check for imperfections, and the list goes on. As science and engineering progress, computer vision is establishing a greater impact on society.

Computer vision can be a very complex area, and in the words of Richard Szeliski, "despite all advances in computer vision, the dream of having a computer interpret an image at the same level as a two-year old (for example, counting all the animals in a picture) remains elusive". This is because vision is an inverse problem (Szeliski 2010), in which we seek to recover some unknowns given insufficient information to fully specify the problem. We must therefore resort to physics-based models and probabilistic models to disambiguate between potential solutions.

Computers interpret images as arrays of numbers and cannot be taught to identify objects by what shape they are. Instead different mathematical techniques are used to determine how the relative pixel of varying densities are arranged to each other. An edge might be detected by a sharp sudden change in pixel density, whilst a shape may be detected by an arrangement of such edges in a circular, mathematical fashion.

Dr Adrian Rosebrock, author of Practical Python and OpenCV, defines a pixel as "a colour or the intensity of light that appears in a given place in an image". Digital cameras are quite often ranked for their resolution, or by how many pixels are in the photos that the camera produces. The resolution is the product of the rows of pixels multiplied by the columns, because images are digitally stored as arrays of pixel intensities that represent the colour or the brightness of the details of the image. For 8-bit images, pixel values are

ranked from 0 to 255 with 0 being no colour at all (black) and 255 being the maximum (white). In RGB (colour) images there are three layers of arrays, each relating to the 3 different colour channels red, green and blue. Every pixel in the arrays are given a value of intensity (or how much of that colour the pixel is trying to represent) out of 0 to 255. For example, a bluish almost purple might have pixel values of 200 for blue, 100 for red and 50 for green.

Figure 1.5: Example of different colour channels compiled together.

1.3 Colour Spaces

A colour space or model is a way to mathematically represent the formulation of colours from the combination of primary colours. Additive colour models are primarily designed for electronic systems such as televisions, computers and digital photography. There are 3 dominant models that are typically used in computer vision.

RGB colour space is the most commonly used colour space and is based on the addition

of 3 primary colours, red, green and blue. In order to create a colour with RGB, three light coloured beams (red, green and blue) must be superimposed (Insights 2016). With no intensity, each of the 3 colours is perceived as black, while full intensity is perceived as white. Varying the intensities varies the output colour.

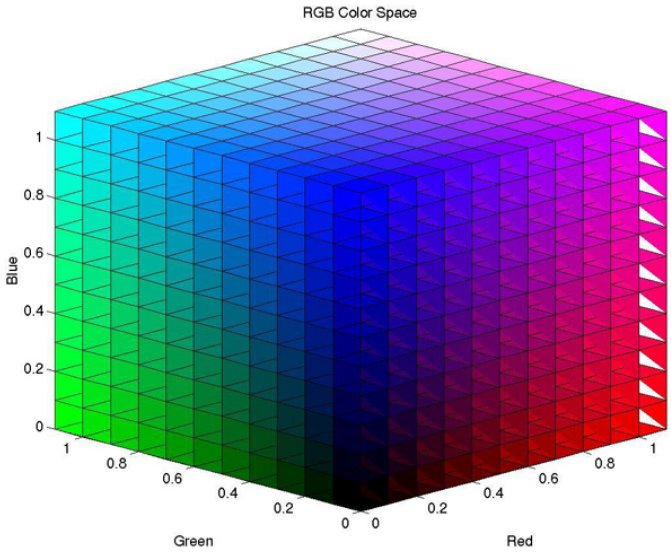

Figure 1.6: An example of the RGB colour space.

HSV (hue, saturation and value) colour space depicts 3D colour. HSV seeks to describe relationships between colours, and improve on the RGB colour model. HSV can be illustrated as a wheel where the centre axis goes from white at the top to black at the bottom, with other neutral colours in between. The angle from the axis represents the hue, the distance from the axis depicts the saturation, and the distance along the axis represents the value (Insights 2016).

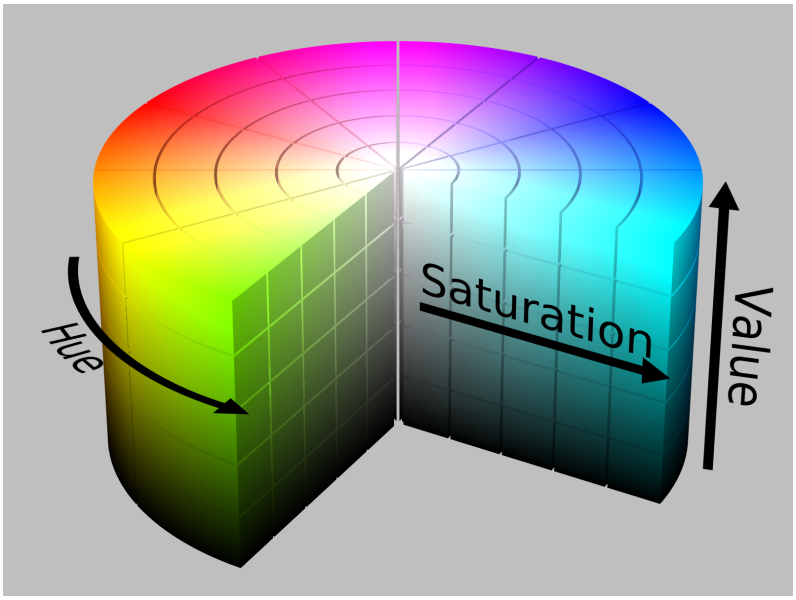

Figure 1.7: An example of the HSV colour space.

LAB colourspace is designed to approximate human vision. Unlike RGB, LAB is not device-dependant. In this 3D model, the 'L' stands for the lightness of the colour, with 0 producing black and 100 producing a diffused white. The 'A' is the redness vs. greenness and the 'B; is the yellowness vs. blueness (Insights 2016).

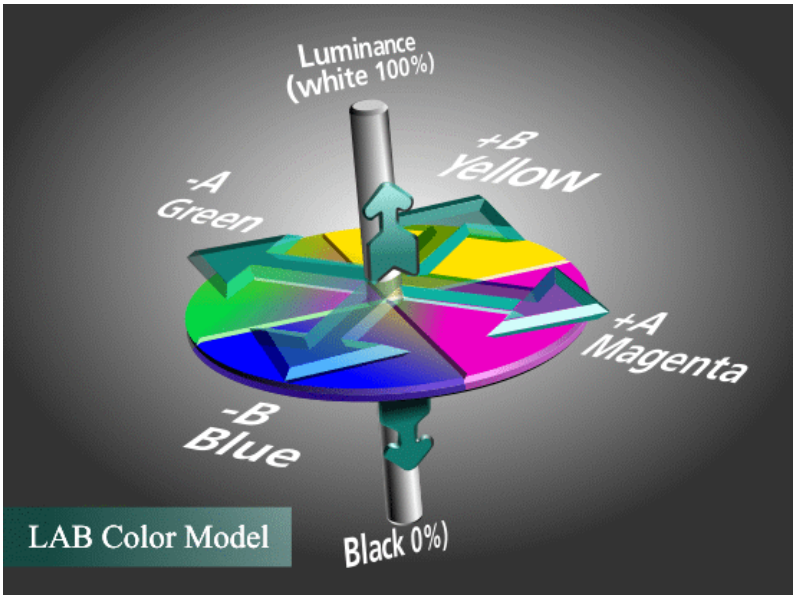

Figure 1.8: An example of the LAB colour space.

1.4 Image Processing

Image processing involves editing and manipulating digital images to enhance its visual effects or to make it more user-friendly for complex computer vision applications. A common use of image processing is in advertising or media, where models can be digitally enhanced to improve their attractiveness or for products to seem more appealing. Image processing is typically the first step in a computer vision algorithm, where features can be transformed, enhanced, blurred and more to make the image more user-friendly for the algorithm to process. For example, a colour recognition algorithm might begin with an image being blurred so that insufficient details are removed, making the probability of a blob of a certain predetermined colour being detected higher. Filters used for smartphone applications such as 'Snapchat' and 'Instagram' are good examples of image processing as they enhance or warp images when applied.

Figure 1.9: An example of image processing being used to enhance the beauty of a model.

1.5 Edge Detection

Edge detection is a process which involves the computer automatically highlighting all the edges in the image through a series of operations. These operations typically involve filtering, differentiation and detection. One of the most successful and commonly used edge detectors is the Canny edge detector, developed by John F. Canny in 1986. Canny had specified performance criteria for the detector to meet (Shah 1997) such as:

1. Good Detection. The optimal detector must minimise the probability of false positives as well as false negatives.

2. Good Localisation. The edges detected must be as close as possible to the true

edges

3. Single Response Constraint. The detector must return one point only for each edge point.

The reason for this criteria was to help minimise unwanted noise that was apparent in other detectors such as the Marr-Hildreth detector where the Laplacian of the Gaussian was used to locate zero crossings which indicated an edge. The steps in the Canny edge detector are:

1. Smooth image with Gaussian filter $g(x, y)$

$$g(x, y) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-x^2+y^2}{2\sigma^2}} \tag{1.1}$$

$$S = I * g(x, y) \tag{1.2}$$

2. Compute derivative of filtered image

$$\nabla S = \nabla(g * I) = \begin{vmatrix} g_x \\ g_y \end{vmatrix} * I \tag{1.3}$$

3. Find magnitude and orientation of gradient

   (S_x, S_y) Gradient Vector

   magnitude $= \sqrt{(S_x^2 + S_y^2)}$

   direction $= \theta = \tan^{-1} \frac{S_y}{S_x}$

4. Apply "non-maximum suppression"

   The pixels that are perpendicular to the identified edge must be suppressed so that thick edges are minimised. This is done by locating a point (x, y) and determining whether the points either side have a steeper gradient.

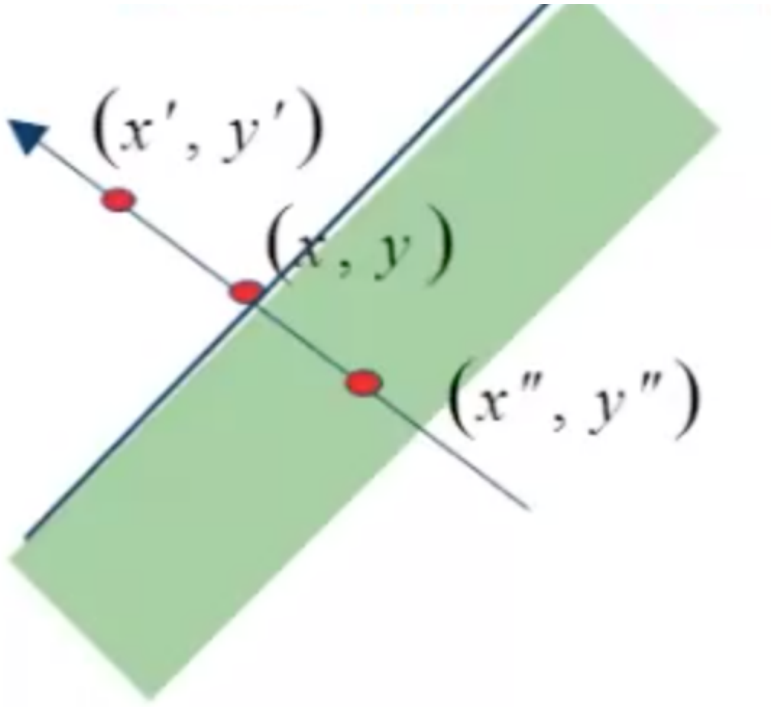

Figure 1.10: Illustration of finding the local maxima for an edge

$$M(x,y) = \begin{cases} |\nabla S|(x,y) & \text{if} \quad |\nabla S|(x,y) > |\nabla S|(x',y') \\ & \& \quad |\nabla S|(x,y) > |\nabla S|(x'',y'') \\ 0 & \text{otherwise} \end{cases} \tag{1.4}$$

5. Apply "hysteresis threshold"

   If the gradient at a pixel is

   -above 'high', declare it as an edge pixel

   -below 'low', declare it as a non-edge pixel

   -between 'low' and 'high', consider its neighbours iteratively then declare it an edge pixel if it is connected to an edge pixel directly or via pixels between 'low' and 'high'.

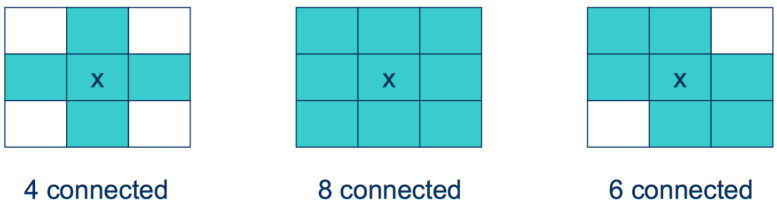

Figure 1.11: Illustration of neighbouring pixels

Scan the image from left to right, top to bottom. If the gradient magnitude at the pixel is above the high threshold, declare that as an edge point. Then recursively look at its neighbours. If the gradient magnitude is above the low threshold, declare that as an edge point.

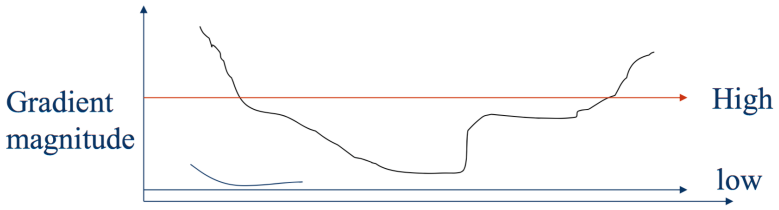

Figure 1.12: Illustration of detected edge points within the predetermined thresholds. As shown, edges remain consistent within the low to high boundary if they are connected to edge points that eventually go above the high threshold level. The line below the low threshold represents noise in the data.

Edge detection is quite often used as the first step in blob detection, where the outlines of objects are detected. Further image processing techniques involving dilation, fill and erosion help to isolate the object. An example of edge detection can be seen in Figure 1.14 where canny edge detection has been applied to a photo of a figurine holding a computer. The edges were then dilated for illustration purposes.

Figure 1.13: Image of figurine imported into Matlab and converted to grayscale.

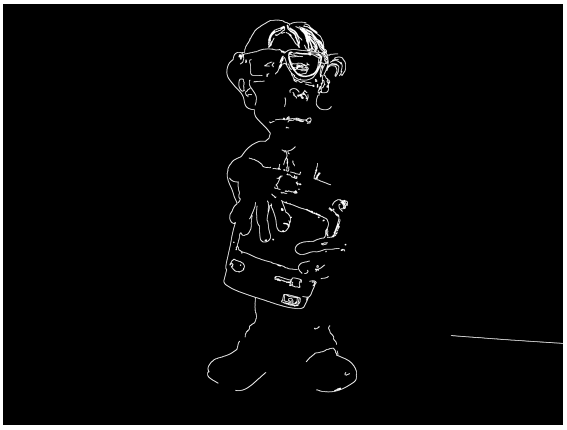

Figure 1.14: Canny edge detection applied to the figurine and edges dilated.

1.6 Image segmentation

Image segmentation has been widely applied in image analysis for various areas such as biomedical imaging, intelligent transportation systems and satellite imaging (Choong, Kow, Chin, Angeline, Tze & Teo 2012). The main goal of image segmentation is to cluster pixels into salient image regions, i.e. regions corresponding to individual surfaces, objects or natural parts of objects (Bodhe 2013). After regions are successfully segmented into individual regions they can then be identified automatically as seen in pest detection systems for horticulture (Bodhe 2013).

Mathworks have offered several examples that explain the basics of image segmentation. Detecting a shark in the water, which in theory should be a controlled environment depending on the amount of light available, could be formulated using edge detection and basic morphology. The example as published by Mathworks (Mathworks 2016*h*) for cell detection could prove to be a good starting point in developing a algorithm for identifying sharks. The example works in six steps:

Figure 1.15: Original image of cell.

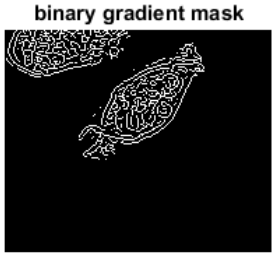

Figure 1.16: Edge detection applied to image.

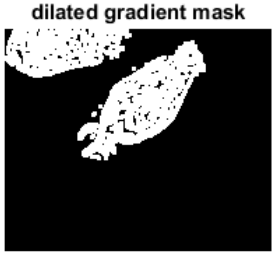

Figure 1.17: Edges dilated with 'imdilate' function in Matlab.

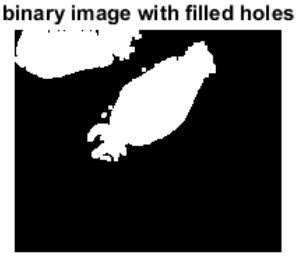

Figure 1.18: Images filled with 'imfill' function.

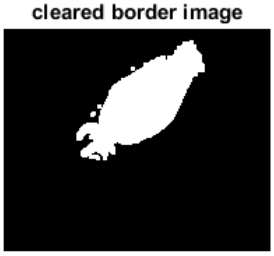

Figure 1.19: Images connected to border are removed.

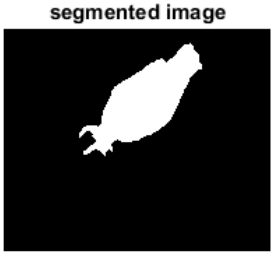

Figure 1.20: Blobs not connected to the main blob are 'eroded'.

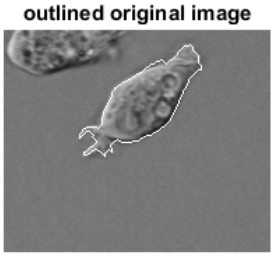

Figure 1.21: The perimeter of the segmented image is overlayed on the original image.

This is a very good example of how the blob of the shark might be obtained. Further analysis could determine whether the blob is of a 'shark' size by identifying how many pixels make up the blob. Although the depth of image is unknown, if an assumed operating height of the drone is known then relative blob sizes can be calculated.

1.7 Object Tracking

Object tracking involves the detection of objects and tracking their movement over multiple frames. It is typically used in surveillance type systems and can be broken down into 3 main steps:

- Object detection
- Object classification

- Object tracking

1.7.1 Object Detection

Object detection is the process of finding instances of real-world objects such as faces, bicycles and buildings in images or videos (Mathworks 2016*i*). Object detection is still very limited in everyday use in society as it is extremely difficult for a computer to automatically detect objects, as explained earlier, a computer can only define an object by the arrangement and intensities of the pixels in the image. However, face detection and people tracking systems are becoming more popular and plenty of research is being undertaken to develop more complex surveillance systems.

To locate a moving object in consecutive video frames the following 3 methods can be applied:

- Frame Differencing
  - Frame differencing works by checking the difference between 2 consecutive frames. It employs the input as 2 image frames of video and produces the output as the difference of the pixel values. This is obtained by subtracting pixel values of the second frame from the first frame (Kothiya 2015). As a result, moving objects can be easily detected in videos with static backgrounds.
- Optical Flow
  - Optical flow is a technique that presents an apparent change in the moving object's location between consecutive frames of the video (Kothiya 2015). It employs the motion field that represents the direction and velocity of each point in every frame. It is a computationally expensive method that is more suitable for multiple moving object detection.
- Background Subtraction
  - The background subtraction method involves removing the background so that only the foreground moving objects are left. Consecutive images are compared to find the difference in pixel values so that moving objects can be identified. This method is ideal for static backgrounds and can deal with multiple moving objects.

1.7.2 Object Classification

Once a moving object has been detected it must be classified to identify what the object is. The following methods are typically used for object classification:

- Shape based classification
  - The shape information of a moving object can be retrieved from the representation of the blob. It can then be compared through matching techniques using complex SIFT or SURF features, or by testing for a certain size or shape. Matching techniques generally require a large database with thousands of test images, both positive and negative to compare against.
- Motion based classification
  - This involves analysing the velocity of the pixels related to an object and classifying them with respect to their known speed. Such applications could involve traffic cameras detecting cars travelling faster than the speed limit.
- Colour based classification
  - Objects can easily be identified that are of a known consistent colour. This method works best when the object of interest is a considerably different colour to the background.
- Texture based classification
  - Texture can be used as a feature by identifying the variation in pixel intensities for a given region of interest. Texture classification involves learning and recognition. Learning consists of identifying the texture features for a given object and recognition attempts to match the texture features with the subsequent frames.

1.7.3 Object Tracking

The typical methods of object tracking are:

- Point based tracking

  - Point based tracking is a complex problem as it can result in false detections and the occlusion of objects in frames (Kothiya 2015). The various methods of point based tracking are:
    * Kalman filter
      · The Kalman filter is based on the probability density function. It is a relatively complex method that is designed to return the optimal solution.
    * Particle filter
      · Particle filtering generates all models for one state variable before it moves to the next state variable. The trajectory of the tracked object can be determined by taking the particle with the highest weight of the particle set at each time step (Hess 2009).
    * Multiple hypothesis tracking
      · This process is also known as an iterative algorithm that begins with a set of existing track hypothesis (Kothiya 2015). The object's position in every frame is made for each hypothesis. Cham and Rehg's (Cham & Rehg 1999) study of tracking people dancing demonstrates that this method is capable of handling occlusion and can track multiple objects.

- Kernel based tracking
  - Kernel based tracking relates to the appearance of an objects shape. The kernel can be elliptical or rectangular and is able to track objects in translation and rotation. Simple template matching, mean shift method, support vector matching and layered based matching are all examples of kernel based tracking (Kothiya 2015).

- Silhouette based tracking
  - Silhouette based tracking involves using the information encoded inside an object region for tracking. This is ideal for objects that have complex shapes such as shoulders, fingers and hands that cannot be described adequately by simple geometric shapes. Silhouette tracking is typically achieved by contour and shape matching.

1.8 Project Aim

This dissertation aims to research object classification by means of computer vision and determine whether an automated shark detection system can be established. The approach taken to design the shark detection algorithm will be from an aerial perspective, with the end goal being an Unmanned Aerial Vehicle (UAV) patrolling the coastal waters and alerting a control station or lifeguards if a shark is detected.

On the 29th of February 2016, the NSW premier Mike Baird unveiled a $250,000 military grade shark-spotting drone called the "Little Ripper" (Chang 2016) which states that it uses real-time sensor and pattern recognition algorithms. Although it is still in the testing phase, the drone is said to be able to identify sharks better than the naked eye.

The proposal of drones policing the coastal waters for sharks offers an inexpensive early shark detection method in comparison to having helicopter and fixed-wing surveillance. If the shark detection algorithm can be taught to identify sharks with minimal error, such as false positives or missing targets, it may pave the way for a foolproof shark surveillance system that eradicates the need for expensive piloted aerial patrols.

1.9 Research Objectives

For the system to be feasible, it needs to show that it can identify sharks and have minimal false positives. To reach this state, the following research objectives must be met:

- Carry out a literature review that is relevant to image segmentation, object recognition, object classification and object tracking.
- Obtain aerial videos of sharks in coastal waters to begin designing the algorithm with.
- Write a basic computer vision program that can detect the 'blob'.
- Expand on the basic blob detection algorithm by trying to filter out all of the blobs that aren't relative to a shark.
- If the algorithm is unsuccessful across the dataset, tune it for each individual photo/video and record what changes made it successful (if any).

Chapter 2

Literature Review

The areas that have been researched for this project are computer vision and techniques relevant to object classification, object recognition and tracking.

2.1 Segmentation methods of marine wildlife from UAV's

Recent studies have proved limited success in detecting wildlife from UAV's. In 2013 a study was conducted by students from QUT (Queensland University of Technology) titled 'Detection of Dugongs from Unmanned Aerial Vehicles' (Maire, Mejias, Hodgson & Duclos 2013). The dugongs were segmented by identifying them by colour thresholding and determining the red-ratio of pixels. To limit false positives 'whitecaps' were identified and 'inpainted' to reduce the confusion of similar shaped bright pixels. The whitecaps were detected by identifying a relationship between the mean of the colour channel 'c' in the image and determining whether the a pixel was greater than that scaled mean.

To further classify the dugongs and limit false positives, a binary map was made of the detected blob and overlayed on a template to determine the similarity. It was discovered that a blob was more likely to relate to a dugong if its shape was elliptical, so by extracting information such as the 'MajorAxisLength' and 'MinorAxisLength' of the detected blob and calculating its elliptical area, it could be compared to the area of the blob template. The closer to 1 this relationship was the higher the probability of the detected blob being a dugong.

The study concluded that the performance of the algorithm was very sensitive to sea

conditions as ever-changing waves and reflections would result in a lot of false positives.

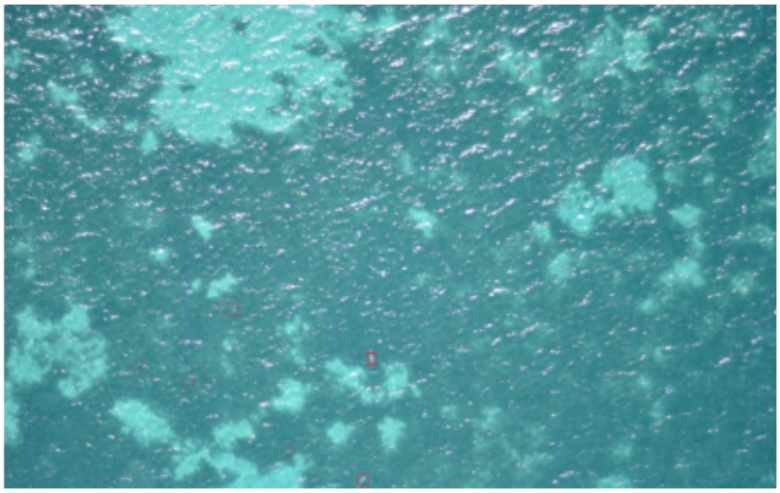

Figure 2.1: Results of dugong detection study illustrating 7 out of 13 dugongs detected, no false positives.

2.1.1 Dugongs and Machine Learning

Machine learning has been another method researchers have investigated to perform automatic analysis of aerial video footage. CNNs (Convolutional Neural Networks) have been the key component in pattern recognition systems, and are predominantly used in face detection and OCR (Optical Character Recognition) systems. Research from the University of Queensland investigated whether a CNN could be used to apply automatic detection to dugongs (Maire, Mejias & Hodgson 2014). By compiling two LeNet convolutional layers, a hidden layer and then a logistic regression layer, (Maire et al. 2014) was able to predict whether there were any dugongs in a given frame.

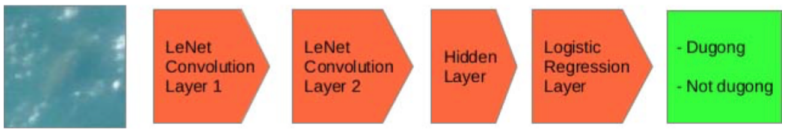

Figure 2.2: Layers of CNN used to detect dugongs.

The success rate of the CNN was found to be feasible, but was limited to lower resolution images as the classifier had been trained on 28 x 28 input windows due to a small training set. It was concluded that it was difficult to compare to methods previously used because the bounding boxes of dugongs were typically 100 x 100 (Maire et al. 2013). A larger training set would need to be obtained to effectively train a CNN with a larger input

window.

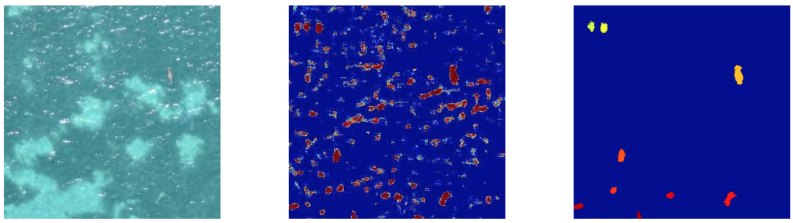

Figure 2.3: Blob analysis using a CNN filtering out blobs based on confidence scores.

2.2 Underwater Fish Detection

The estimation of fish population and species classification has been an interesting application of blob analysis (Fabic, Turla, Capacillo, David & Naval 2013). The study, conducted by the University of the Philippines, utilised an underwater camera to capture local schools of fish inhabiting coral. The basis of this study was to determine a method using computer vision to count and determine the species of fish. Such information would be helpful to marine biologists to track and monitor fish species and their habitats.

Figure 2.4: Blobs detected after edge detection, coral blackening and blob filling.

The algorithm first used histogram comparison to identify coral and 'blacken' it out. Once the coral had been removed, edge detection was applied to the frame to identify contours related to fish. These contours were then filled to represent 'blobs', and then filtered and categorised by size. This allowed the vision system to differentiate between the two fish species within the frame, the parrot fish and the sturgeon fish. The system proved to be relatively successful, producing little to no false negatives and very few false positives.

Video at 00:05		Machine	
	Human Count	Machine Count	Correctly Identified
Identifiable Fish	5	5	4
Sturgeon Fish	1	2	1
Parrot Fish	4	3	3
Video at 00:27		Machine	
	Human Count	Machine Count	Correctly Identified
Identifiable Fish	6	7	5
Sturgeon Fish	3	5	2
Parrot Fish	3	2	1
Video at 00:27		Machine	
	Human Count	Machine Count	Correctly Identified
Identifiable Fish	6	7	5
Sturgeon Fish	3	5	2
Parrot Fish	3	2	1

Figure 2.5: Success rate of vision system.

2.3 Marine object detection

Other algorithms, such as jellyfish and sea snake detection, use what is known as a hybrid detection method to correctly identify objects of interest (Zhou, Llewellyn, Wei, Creighton & Nahavandi 2015). This hybrid method is composed of statistical learning techniques, such as Gaussian mixture modelling, and feature based approaches.

One of the main problems with dealing with marine object detection is being able to remove all the pixels related to the water, and just leave the objects of interest. To achieve this in the jellyfish and snake detection, a Gaussian mixture model which could be trained automatically to recognise background pixels (water) (Zhou et al. 2015). Automatic collection of training pixels was segregated into different areas of the training frame, due to the lighting distribution. Histogram models were made of each area of the frame, and bin thresholding was applied to determine which pixels were related to the background.

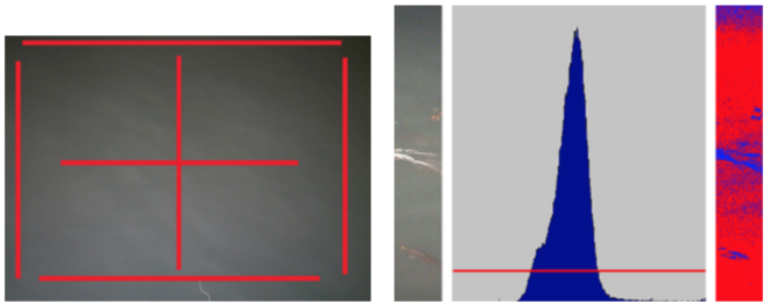

Figure 2.6: Frame divided into different areas and histogram bin thresholding applied.

The GMM was then trained on these background pixels, after being partitioned into

K clusters using the Orchard-Bouman algorithm (Orchard & Bouman 1991). Image segmentation is then applied to differentiate between the foreground and background groups, based on the Otsu algorithm (Yu, Dian-ren, Yang & Lei 2010). Standard morphology operations such as fill holes, eroding and dilating help remove any noise and leave the blobs of the sea life defined.

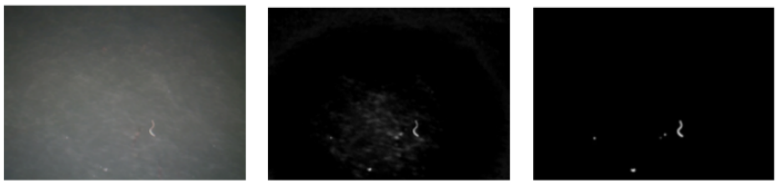

Figure 2.7: Effects of GMM and background subtraction.

Blob analysis was then performed to classify the snakes and jellyfish. Blob selection criteria such as blob height to width ratio, area to bounding box ratio and circularity (using Heywood's circularity factor) allowed the snakes and jellyfish to be correctly classified. The disclosed results determined a detection rate of over 90% over a sample size of 243 images (frames with snakes or jellyfish in them) collected over two years.

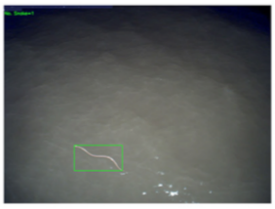

Figure 2.8: Frame showing snake detection.

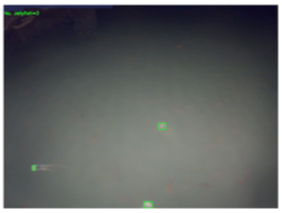

Figure 2.9: Frame showing jellyfish detection.

2.3.1 LiDAR Detection and Classification of Subsurface Objects

LiDAR detection systems have also been investigated for marine object detection (Cianciotto 1997), due to their resilience to weather conditions as opposed to traditional computer vision techniques. Airborne LiDAR systems have been used for research in:

- Detection of submerged objects
- Oil spill detection and identification
- Marine biology and biomass estimation

- Atmospheric pollution surveillance
- Stratospheric dust measurements
- Subsurface ocean temperature measurements

The advantages of these LiDAR systems is that LiDAR is typically able to see objects 3-5 times deeper than conventional computer vision systems, which would be ideal for locating sharks that are too submerged for a camera to identify. LiDAR systems are however quite expensive, leaving this possible avenue unavailable for the author.

2.4 Blob Analysis

Blob analysis has also been widely used in vehicle counting, pedestrian tracking and traffic sign recognition. By identifying what makes an image unique, pixel thresholding can be performed to remove pixels that are unrelated to the region of interest. This can involve manipulating colour spaces such as RGB, HSV and LAB. Research from the University of Ostrava demonstrated pixel thresholding in the HSV colour space when attempting to detect and classify traffic signs (Zavadil & Tuma 2012). By 'ANDing' pixel conditions for all three channels (hue, saturation and value), he was successful in identifying the pixels most likely related to the traffic signs. From there, analysis of the blob properties was performed to determine the geometry of the blob, and the Mahalanobis distance was computed to determine the similarity score against a reference set of images. The testing was performed on a robot race track, and images were captured within 2m of the upcoming traffic symbol. Of the 321 positive frames, only 15 false negatives were obtained and 0 false positives. The following graph illustrates the results:

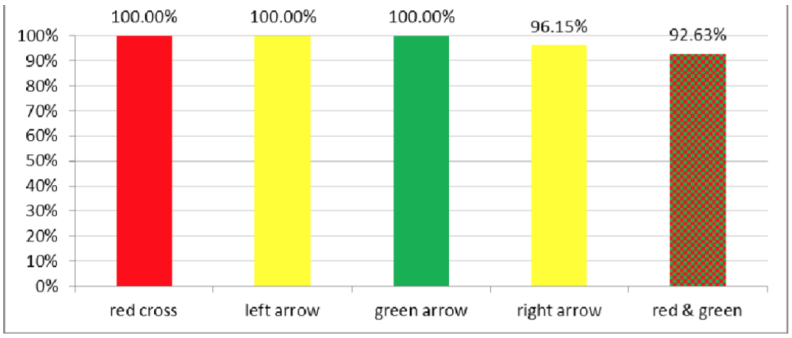

Figure 2.10: Success rate of recognition within 2m of sign.

Vehicle counting is another good demonstration of blob analysis, where background subtraction is utilised to remove all stationary pixels. Therefore stationary cameras can be set up at busy intersections, and be programmed to count each car (blob) that is moving. This is a much more effective way to record traffic flow information than to have people continually noting down the cars driving past. Research from the GMR Institute of Technology, India, illustrated how background subtraction can be utilised, and blob analysis can be performed to allow the vision system to identify light coloured cars driving past (Telagarapu 2012).

Figure 2.11: Sample frame of traffic.

Figure 2.12: Background subtraction and blob analysis performed to identify white cars.

Background subtraction has also been widely used in people detection and counting systems. Research from the Pozna University of Technology demonstrated that people entering a building could be identified, counted and tracked, by removing the background and applying blob analysis to the detected moving pixels (Tchn, Kfnqhsglr, Mc, Shnmr, Marciniak, Dąbrowski, Chmielewska, Nowakowski, Wkh, Ri, Lq, Uhvhdufk & Uhvxowlqj 2012) . Although background subtraction can be a useful tool in elementary object detection solutions, it is certainly not feasible for a shark detection system where the waves are constantly moving and changing.

2.5 Project Area of Research

Based on the findings of the literature review this project will be focused on determining an appropriate pixel thresholding method to identify the shark blobs within the frame and use extensive blob analysis techniques to classify the detected blobs as sharks.

These methods will involve converting the image to alternate colour spaces for analysis (Zavadil & Tuma 2012), applying pixel thresholding techniques and determining blob attributes such as elliptic ratio (Maire et al. 2013), and investigating other unique properties of the shark blobs to enable the shark detection algorithm to filter out false positives with blob analysis techniques (Fabic et al. 2013).

Chapter 3

Methodology

This chapter discusses the approach that will be taken to design, develop and evaluate the shark detection algorithm on its feasibility.

3.1 Project Methodology

Computer vision can be an elusive task that is prone to error, and unfortunately the solution to one object detection problem can not simply be applied to another. That is why the methodology for this project will involve many alterations. The development stage of the algorithm will involve countless changes and tweaking to first, attempt to isolate the shark from the image and then optimise the isolation and tracking process.

To further discuss the methodology of the project, the project has been segregated into tasks:

1. Determine the tools required such as software packages and electronics
2. Obtain useful videos of sharks in coastal waters
3. Determine trends in pixel variations and colour spaces
4. Develop an adaptive thresholding method for identifying regions of interest
5. Morphological operations
6. Filtering of false positives

7. Develop tracking methods

8. Determine feasibility and highlight why the algorithm is successful or not

3.2 Task Analysis

The following sections highlight the key components of the designated tasks.

3.2.1 Determine the tools required such as software packages and electronics

The programming environment for the development of the shark detection algorithm has been chosen to be MATLAB. This software offers some very useful computer vision and image processing functions and its matrix representation makes it incredibly easy to analyse images. The MATLAB suite combined with the computer vision toolbox is $155.

All other necessary equipment such as a laptop with Microsoft office are already owned by the author.

3.2.2 Obtain useful videos of sharks in coastal waters

The author has already reached out to sharkspotting groups on facebook and marine biologists but has been unable to acquire any aerial videos of sharks. However, there are some videos on youtube that may be suitable for image processing. The following images are snapshots from two of the videos:

Figure 3.1: Screenshot from drone footage of shark in coastal waters

Figure 3.2: Screenshot from drone footage of shark in coastal waters

Videos where the drone is 'sweeping' the coast may prove difficult to work with, so only necessary snippets of the videos where the drone is hovering above a shark will be used.

3.2.3 Determine trends in pixel variations and colour spaces

The first step of the algorithm will be blob analysis. Colour thresholding can be performed to remove unwanted objects in an image such as removing dark objects such as roads and converting the resulting image to a binary map (Telagarapu & Suresh 2012). To identify what colour thresholds to implement to localise the sharks, RGB and HSV colour channels will analysed to determine reliable trends in the pixel representation of the sharks. If a reliable trend can be identified, a threshold will be applied that sets all pixels outside the desired range of the pixel to zero. The result of this will be transformed into a binary map which shows the blobs of interest.

3.2.4 Develop an adaptive thresholding method for identifying regions of interest

The problem with computer vision in an uncontrolled environment is how sensitive the system can become due to changes in illumination. To actively set a threshold to locate the regions of interest, a relationship will have to be determined between lighting variation from the sun's exposure. The colour of the ocean can quite often change, so setting a fixed threshold for pixel values will be inadequate. Instead, an adaptive threshold will be put in place that determines the threshold value due to average pixel values in the frame. It can be assumed that the vast majority of the frame will be the ocean colour.

3.2.5 Morphological Operations

Morphology is basically the manipulation of blobs. These methods include:

1. Smoothing of blobs
2. Removing any blobs under a determined pixel area
3. Filling holes in blobs
4. Removing blobs connected to the border of the frame

5. Dilating the blobs

6. Retrieving data such as area, perimeter, major and minor axis lengths and orientation of the blobs

Using the above tools the identified blobs can be operated on to further filter out any unwanted objects. This is necessary when dealing with a dynamic environment such as the ocean where the constant change of waves and shadows could easily fool a computer vision system.

3.2.6 Filtering of false positives

False positives are undoubtedly going to make their way through the system. To help prevent the false classification of sharks, filters will be put in place that test the attributes of each detected blob to determine whether it is actually a shark. These filters will involve methods such as:

1. Calculating the brightness ratio of pixels to the area of the bounding box. Blobs associated with too many bright pixels will be rejected. This is to eliminate false detections such as surfers, who typically where black wetsuits and ride bright surfboards.

2. Calculating the variance of the brightness values in the bounding box. A minimum threshold will be determined to prevent dark patches of water from being classified as a shark.

3. Calculating the elliptic ratio. Shark blobs tend to have an elliptic shape. A maximum threshold will be determined to prevent false positives.

3.2.7 Develop tracking methods

Tracking of the identified blobs will help filter out false positives that have made it past all the previous steps. Assuming a constant operating height of a UAV, when a blob is detected the UAV can hover and track the movement of the blob using its centroid. If the detected blob remains stationary (such as driftwood floating with the current), or the detected blob is travelling at a speed impossible of sharks, the blob can be ignored.

This will be difficult to effectively implement for the footage obtained, as the shark footage has been sourced from the internet and rarely has moments where the UAV is hovering above the shark. For the purpose of this project, a dynamic array will be created that will record the binary maps of the last five frames. Using this dynamic array the detected blob will be compared to the last five frames to determine if it has been consistently detected and allowing it to pass a 'reliability threshold'. This should further filter out any sudden changes in the wave formations that may trick the system. If time permits the movement of the centroids can be calculated relative to the UAV movement.

3.2.8 Determine feasibility and highlight why the algorithm is successful or not

This task involves critically reviewing the work accomplished in the previous tasks and determining feasibility of a shark detection algorithm. Areas such as why the algorithm performed the way it did should be reviewed in depth to gain understanding what could have made it better or what kind of data would produce optimum results. If the algorithm has failed to detect sharks on all accounts, the author should attempt to explain why it may have failed and what needs to change for it to produce better results next time.

3.3 Resource Analysis

To understand the financial requirements of the project all the required resources mentioned in the methodology section must be tabulated and calculated so that necessary funds are available.

Table 3.1: Project Resource Requirements

Resource	**Quantity**	**Cost**	**Source**
Laptop	1	Owned	N/A
LaTeX Package	1	Free	Online
MATLAB with Computer Vision Toolbox	1	$155	Mathworks Website
	Total	$155	

The total cost for the project has been calculated at $155. The total for this project is

considered reasonably affordable and some of the resources will have other uses outside of the project such as MATLAB.

The critical items of the project that would threaten the viability of the whole project are the Laptop and the MATLAB package. For these reasons all project documents and MATLAB scripts are backed up to a personal cloud which then syncs with Google Drive. If the laptop was to breakdown or be stolen all the project work to date would still be accessible from another computer and MATLAB could be re-downloaded through the Mathworks account page.

3.4 Project Consequential Effects

For the project to be undertaken in conjunction with the standards set by Engineers Australia, the sustainability, safety issues, ethics and risks involved must be highlighted and discussed.

3.4.1 Sustainability

If the project is to be successful and the shark detection system affordably rolled out across popular beaches in Australia, it holds the potential to rethink current shark control methods. Shark nets could potentially be discontinued, which would sharply reduce the unnecessary deaths of marine life such as dolphins and turtles.

3.4.2 Ethics

For the project to maintain its integrity the Code of Ethics (Australia 2010) published by Engineers Australia must be closely adhered to. The key aspect highlighted is all the work published by the author must be truthful and not fabricated to proclaim success. At all times the author must clearly define how the code is preforming its analysis of the shark video and why it was either successful or unsuccessful. The author shall not alter the code to detect a shark under false pretences.

3.5 Risk Assessment

As with any project, the risks involved must be identified so that they may be minimised or eliminated all together. The following risk matrix ranks the likelihood of an incident occurring against the possible consequences involved.

0 – 5 = Low Risk 6 – 10 = Moderate Risk 11 – 15 = High Risk 16 – 25 = extremely high unacceptable risk		Severity of the potential injury/damage				
		Insignificant damage to Property, Equipment or Minor Injury **1**	Non-Reportable Injury, minor loss of Process or slight damage to Property **2**	Reportable Injury moderate loss of Process or limited damage to Property **3**	Major Injury, Single Fatality critical loss of Process/damage to Property **4**	Multiple Fatalities Catastrophic Loss of Business **5**
Likelihood of the hazard happening	Almost Certain **5**	5	10	15	20	25
	Will probably occur **4**	4	8	12	16	20
	Possible occur **3**	3	6	9	12	15
	Remote possibility **2**	2	4	6	8	10
	Extremely Unlikely **1**	1	2	3	4	5

Figure 3.3: Risk Assessment Matrix.

3.5.1 Risks Identified

Due to the nature of the project there are very limited risks involved. This project consists of post processing data obtained from the internet, so the only risks associated are the preservation of work and equipment. The following risks were identified and scored on the matrix:

1. Failing to back up data: Risk Sore - 9

    - If the computer had a breakdown and the progress made on the algorithm had not been saved to an external hard drive or cloud storage, a considerable amount of time would be lost. Therefore the directory containing all the Matlab source files will be synced to a cloud storage to avoid this possible catastrophe.

1. Incorrect Seating: Risk Sore - 6

    - Hours are going to be spent sitting at a desk attempting to write the shark detection algorithm. To avoid back problems the author will sit in an ergonomic chair that promotes good back posture. Back problems are a common occurrence for the author.

Chapter 4

Pixel Trends, Colour Spaces and Thresholds

4.1 Initial test and data display

To determine how to segment the image the pixel values related to sharks need to be identified. To begin with, a test image was used that portrayed only the shark surrounded by water. The red, green and blue arrays were individually extracted from the RGB image, and then the image was converted to HSV and the hue, saturation and value arrays were saved to separate variables. This is essentially the control for the experiment as there are not other objects in the frame that could skew the data.

Figure 4.1: Test image.

The image was imported into MATLAB using the 'imread' function and the pixel intensities were displayed as a 3D surface model to better illustrate the pixel densities. The plots were viewed from an angle of 0 degrees and 90 degrees. The following pseudo-code represents how the pixel densities are extracted and plotted:

Algorithm 1 Display Surf Models of Pixel Densities

1: **procedure** IMREAD($frame$)
2: *Red* ← (:,:,1) *frame*
3: *Green* ← (:,:,2) *frame*
4: *Blue* ← (:,:,3) *frame*
5: *framehsv* ← rgb2hsv *frame*
6: *Hue* ← (:,:,1) *framehsv*
7: *Saturation* ← (:,:,2) *framehsv*
8: *Value* ← (:,:,3) *framehsv*
9: **Display** *Surface Models Side View and Aerial View*

The original code to the above algorithm can be found in the Appendix.

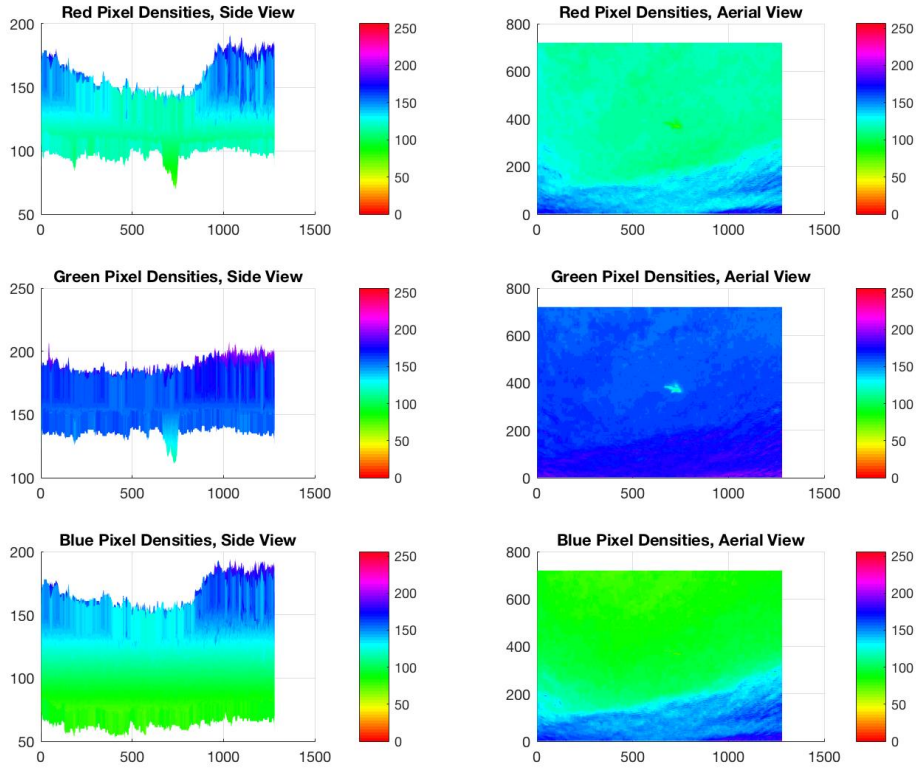

Figure 4.2: RGB representation of current test image. The red, green and blue data arrays are saved to individual variables and displayed as a surface model. This will help identify methods for pixel thresholding.

As illustrated above, the shark can be located by its red and green pixel values. There is a considerable peak which clearly represents the shark when looking at the colourmap from an aerial view. It is easier to determine from the green pixel density with the human eye, but from the side view it is clear that the red and green values have similar peaks.

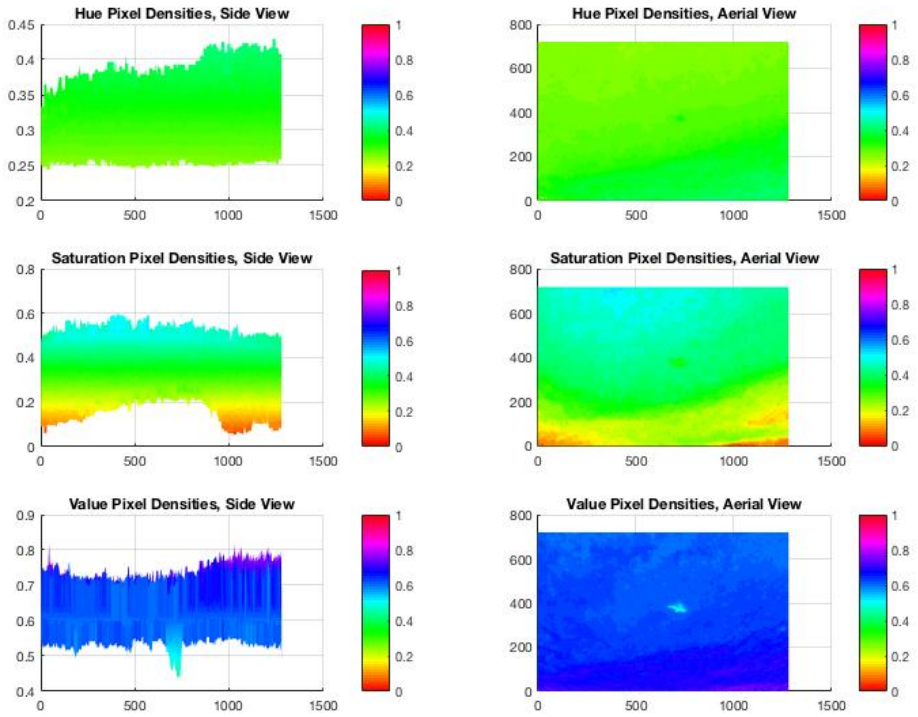

Figure 4.3: HSV representation of current test image. The hue, saturation and value data arrays are saved to individual variables and displayed as a surface model. This will help identify methods for pixel thresholding.

As illustrated above, the shark can only be determined from the 'value' pixel density which represents the brightness of the pixel. If it can be assumed that the shark will always be darker than the water around it, theoretically because of the shadow it would cast under the water, this may be a suitable way to analyse the data and apply a threshold. The value pixel density distribution is very similar to the green pixel density distribution.

4.2 Identifying strong trends

It can be determined that the pixel values representing the shark in the test image are of a lower value in the Red, Green and Value arrays. More complex frames were tested to determine if this relationship was consistent across different water colour conditions to decide on which colour space to apply thresholding to. The following four images were compared:

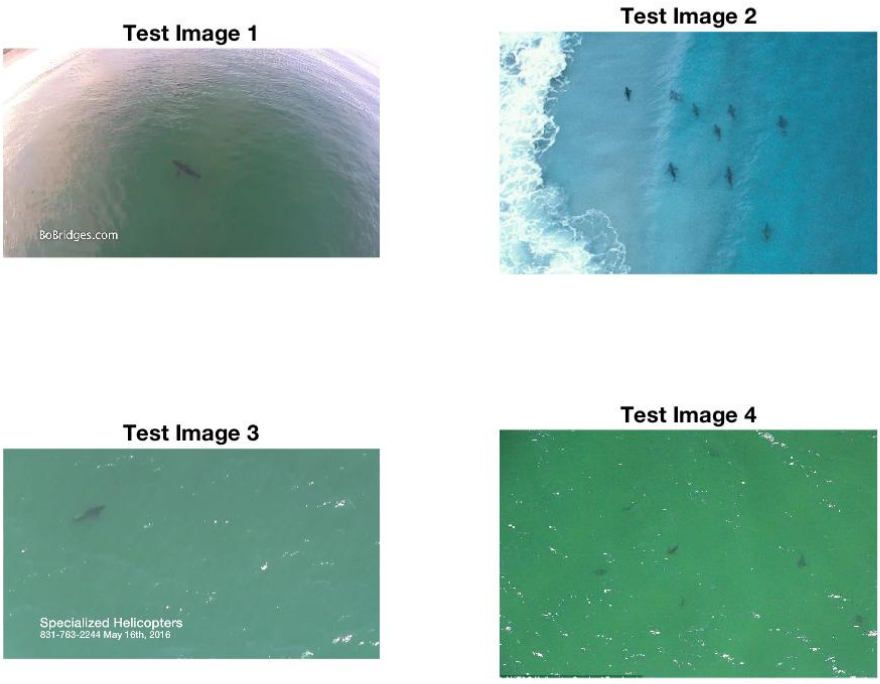

Figure 4.4: Frames used to test the relationship of lower pixel values for identified sharks in the Red, Green and Value arrays.

The first comparison was for 'Test Image 1' which illustrated the 3D surface model for the Red, Green and Value arrays. It can be seen in this image that the frame is from a camera with a wide angle lens. The shark can be seen clearly in the middle of the frame and is the darkest object in the frame. The upper sides of the image contain some glare and there is reflection on the water due to the angle the camera is making with the water and the sky above. The water is a much darker green than the original test image and the contrast between the shark and the surrounding water is less vivid.

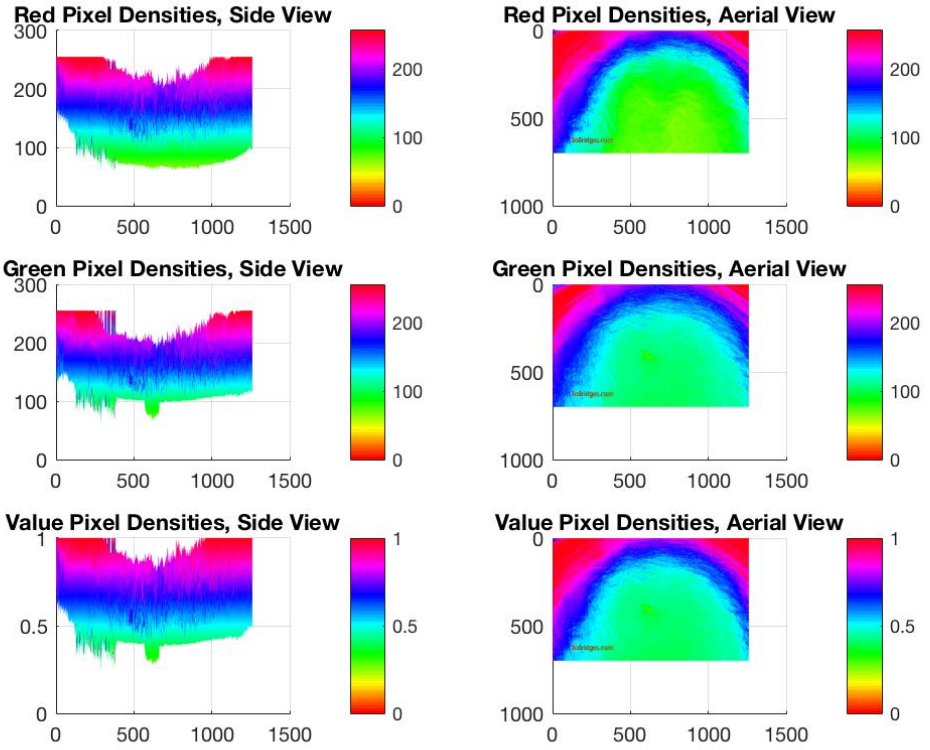

Figure 4.5: 3D surface model for Red, Green and Value arrays for test image 1 at side and aerial views.

As illustrated above the Red model does not display the peak where the shark is located in the image. However the Green and Value display similar results and illustrate where the peak for the shark is. There are also narrower peaks in the Green and Value plots that represent the shadows in the waves.

'Test Image 2' contains nine sharks swimming in a light blue coloured ocean. There is some whitewash to the left of the frame and the water has a darker shade of blue towards the right side of the frame.

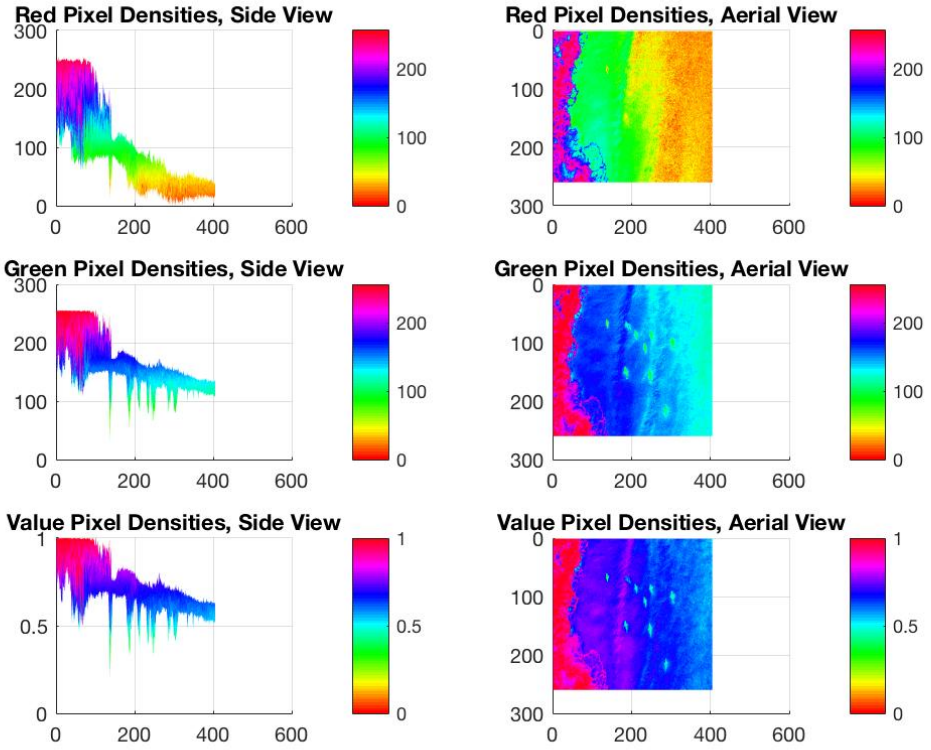

Figure 4.6: 3D surface model for Red, Green and Value arrays for test image 2 at side and aerial views.

Once again the Red model has failed to display the peaks where the sharks are, whereas the Green and Value models depict the sharks location by the lighter shades of blue in the colour map. The Value model appears to depict the sharks more clearly than the Green model.

'Test Image 3' contains one shark in the top left corner. The water is a greeney-blue colour and there is some reflection from the sun on the tops of the waves. There is also some overlaying writing which needs to be ignored.

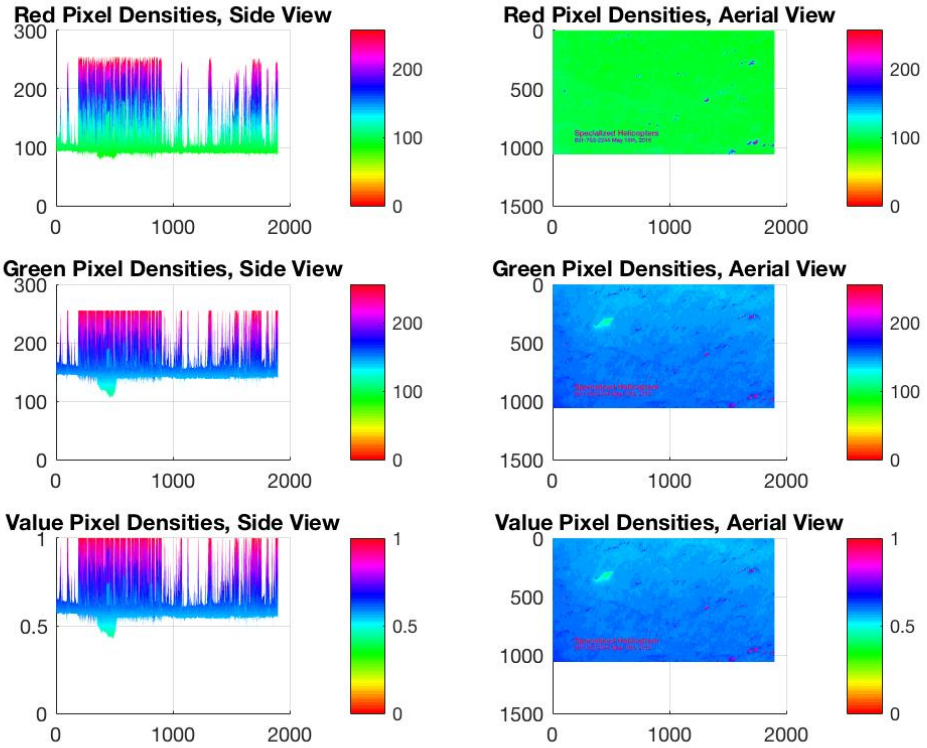

Figure 4.7: 3D surface model for Red, Green and Value arrays for test image 3 at side and aerial views.

The Red model once again proves to be not of any use. The Green and Value models both clearly depict the shark.

'Test Image 4' contains seven sharks scattered throughout the frame. The water is a dark green colour and there is some reflection from the sun on the tops of the waves

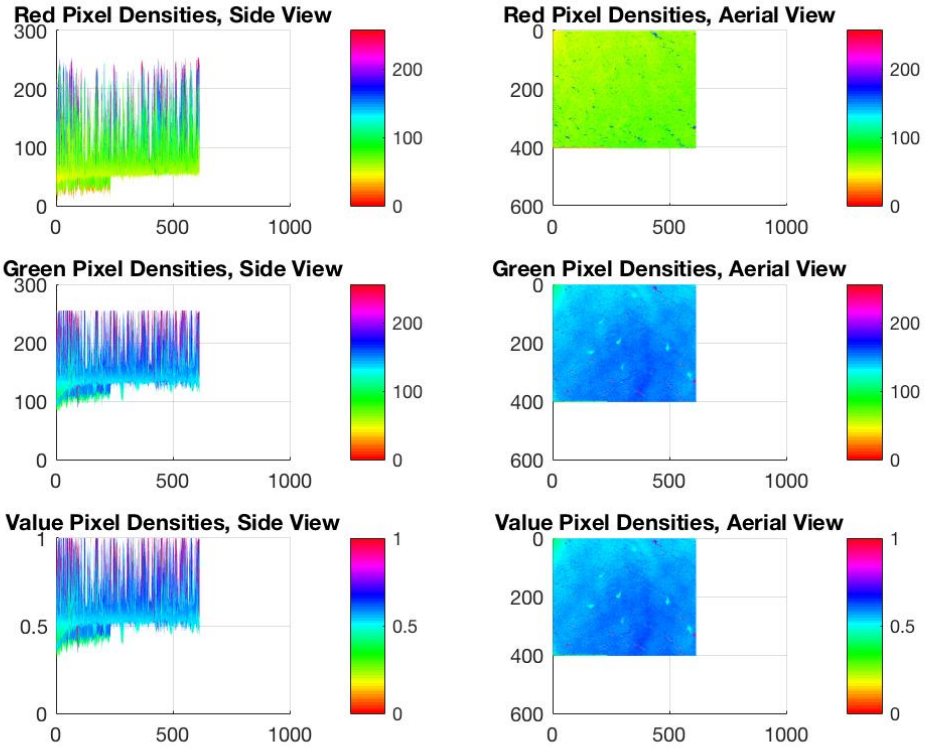

Figure 4.8: 3D surface model for Red, Green and Value arrays for test image 4 at side and aerial views.

The Red model is clearly inadequate for using as a thresholding condition to detect the shark blobs. The Green and Value channels appear to be the most reliable channels suitable for pixel thresholding.

4.3 Colour Space Thresholding

From the tests performed in the previous section it can be assumed that Value pixel density thresholding can be implemented to segment the shark blobs. Although this might allow for some unwanted blobs, further blob analysis techniques can be utilised to reject the false detections. To test the colour space thresholding for the test images, threshold values were chosen after analysing the 3D surface models. The thresholds differ for each image because of the different environments and colouring of the water. The threshold values used were:

1. Test Image 1 = 0.387

2. Test Image 2 = 0.551
3. Test Image 3 = 0.543
4. Test Image 4 = 0.504

The following pseudo code shows how anything over the cut-off pixel value in the Value array is set to 0. It then converts the image to a binary map, performs some basic morphology operations and then overlays the outline of the detected blobs over the original image:

Algorithm 2 Display Blob Outlines on Original Image

```
    procedure IMREAD(frame)
2:      framehsv ← rgb2hsv frame
        Value ← (:,:,3) framehsv
4:      for i = 1:size(Value,1) do
            for j = 1:size(Value,2) do
6:              if Value(i,j)>threshold then Value(i,j) = 0
        BW ← imbinarize(Value)
8:      BW ← imdilate(BW)
        BW ← imfill(BW)
10:     BW ← bwperim(BW)
        BW ← imdilate(BW)
12:     frame(BW) ← 255
        Display Test Images with detected blob outlines
```

The original code to the above algorithm can be found in the Appendix.

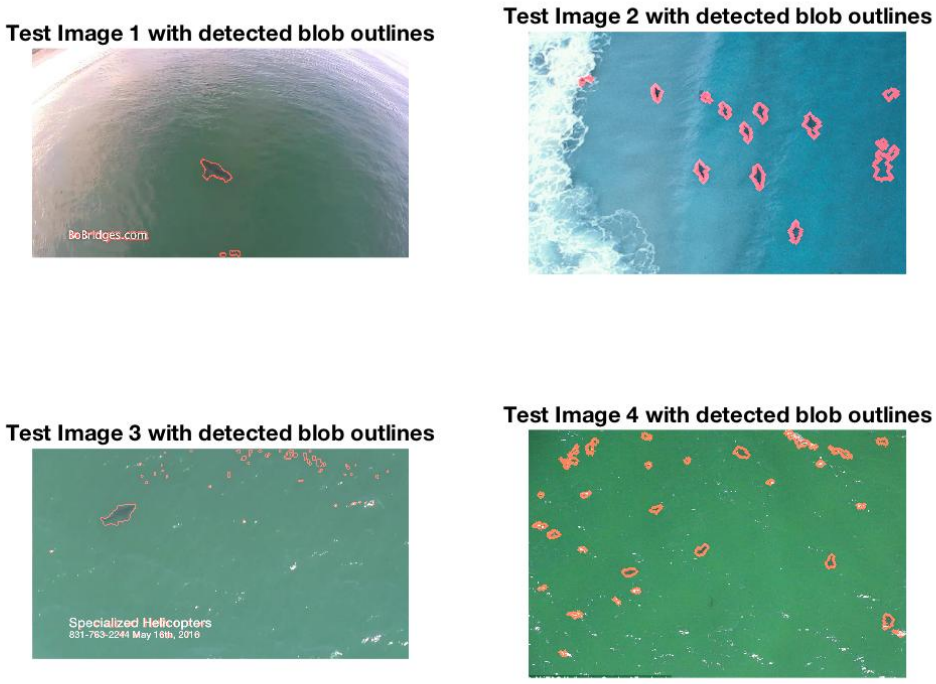

Figure 4.9: Detected blob outlines overlaid on original image using Value pixel thresholding.

By determining an appropriate threshold for the Value pixel value, the majority of the image can be ignored and only dark coloured blobs are left. The Value pixel is a number between zero and one, where the closer to the zero the value is the darker the pixel is. So everything above the threshold is therefore set to zero, and when the resulting image is 'binarized', all pixels with a value greater than zero are set to one. This is very basic blob analysis and can be effectively used in a controlled environment where the object of interest is a unique colour to that environment.

However, there are plenty of blobs that have been detected that fell below the desired threshold that are not sharks. This can be because of shadows in the waves or rocks in the water. To help filter out all the unwanted blobs, further morphological operations can be put in place to discard the false positives. Functions such as determining the minimum blob area size and maximum area size are particularly useful for clearing up noisy data.

The above demonstrations illustrate that by applying a threshold to the Value array from the HSV colour model of the frame, sharks can be detected and their blobs parsed to the next stage of the algorithm for further filtering of false positives. The remainder of the

project will concentrate on first detecting all the blobs under a given threshold for each frame.

4.4 Adaptive Thresholding

The problem with applying a constant threshold to the data is that with each frame of the video the environment can change. An adaptive threshold method was determined and applied to automatically calculate the appropriate threshold for each frame. This was achieved by representing the data from the Value array as a histogram and calculating when the derivative was greater than a certain value starting from 0 and determining the forward difference. Because this method was tested with different pictures and videos with different resolutions, an adaptive derivative formula was determined to calculate the derivative threshold for each frame:

$$DerivativeThreshold = \frac{framewidth * frameheight}{-1500} \tag{4.1}$$

The pseudo code for this loop can be represented as follows:

Algorithm 3 Calculate Threshold for Regions of Interest

```
    procedure H = IMHIST(Value)
        D ← zeros length(H,1)
3:      for i = 1:length(H) do
            D(i) ← H(i)-H(i+1)
            if i == 255 then
6:              break
            if D(i)<(size(Value,1)*size(Value,2))/-1500 then
        Threshold ← i/256
```

This method was used because it isolates the darker objects in the image. From the data obtained, it was noticed that sharks were consistently a darker colour than the surrounding water. This threshold calculation works well when the frame is made up of bright scenes such as water, whitewash and sand. Due to the lack of data this method was unable to be tested when there were cliffs or rocks in the scene. The calculated threshold for the above test images can be visualised with the following histograms:

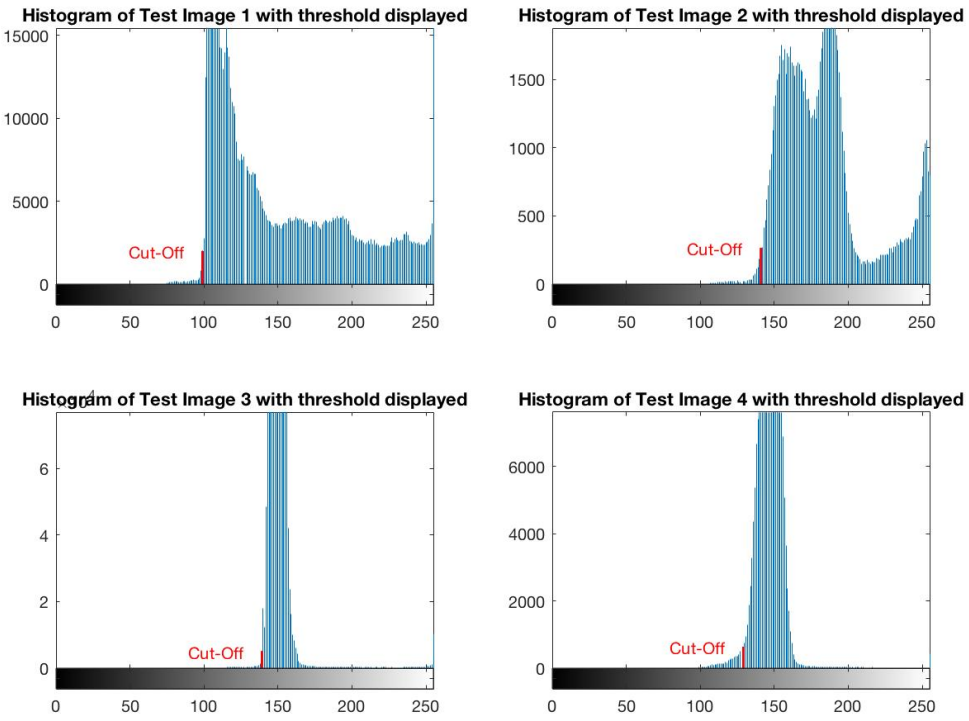

Figure 4.10: Histograms of Test Images displaying cut-off threshold. Most populated bins represent pixels related to the water. Everything to the right of the cut-off is ignored.

Once the threshold is applied, all values in the array above the threshold are set to zero and the array is converted to a binary representation. The clusters of one's in the image that represent the pixels under the threshold are referred to as 'blobs'. This is where the blob analysis begins.

Chapter 5

Blob Analysis

5.1 Morphology

Image segmentation is the process of identifying and segregating regions of interest for further processing and decision making. After pixel thresholding has been applied and the leftover scene has been converted to binary, there is often a lot of noise and unwanted data that needs to be filtered out. By using some basic morphological functions, the binary map can be filtered to remove blobs under a certain area, remove blobs connected to the border and erode or dilate blobs. The following plots are what is left when the Value array is first converted to binary after thresholding:

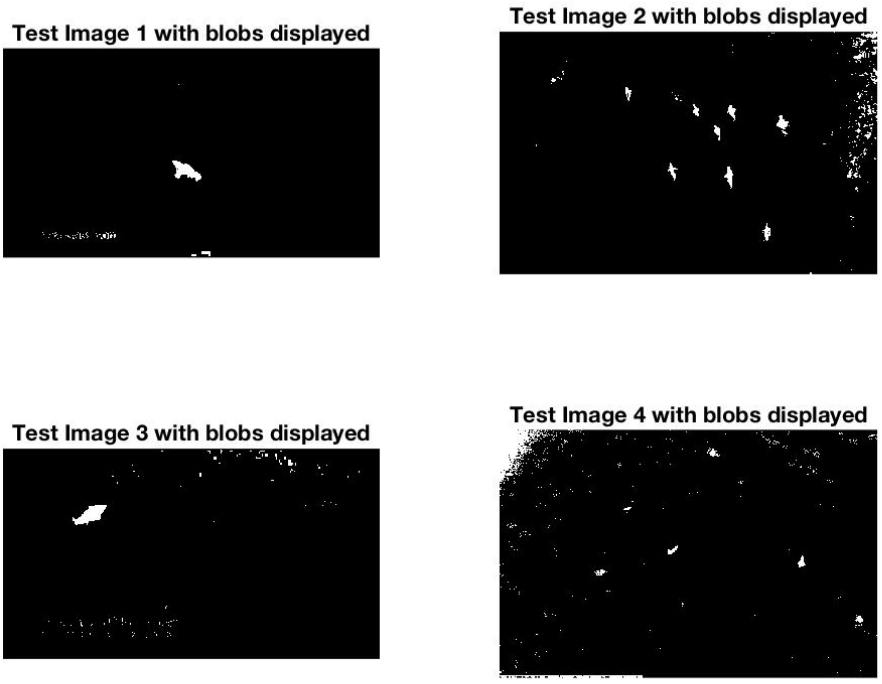

Figure 5.1: Binary maps of test images after thresholding. No morphology performed yet.

As illustrated above, there is some considerable noise in the binary maps. For the vision system to automatically detect sharks the images need to be filtered as much as possible to remove any objects that might confuse the system. By implementing these filters the system's false positive rate will be greatly reduced.

To begin filtering the image an analysis must be performed on the above plots to determine the regions of interest that the system will want to ignore. The outline of the shark is clearly visible to the human eye, but to a computer it looks like this:

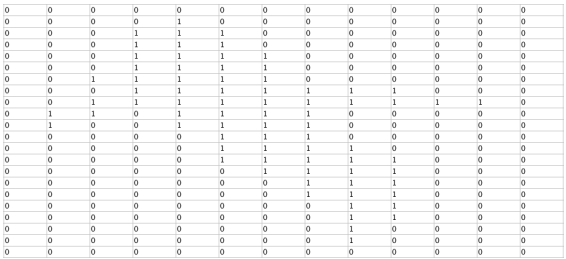

Figure 5.2: How a computer 'sees' a shark.

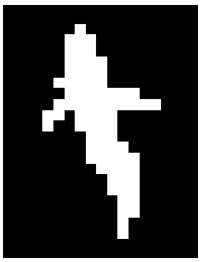

Figure 5.3: Binary map of shark.

To remove unwanted noise the first step was remove any blob that was under a certain pixel area size. Using the function 'bwareaopen' in Matlab, the program is able to calculate the area of each individual blob and if it does not meet the minimum area the blob is removed. Because the test images have different frame sizes, the minimum area was determined by a ratio dependant on the frame area. The following before and after plots illustrate removing blobs under the minimum area and any blobs that are connected to the border:

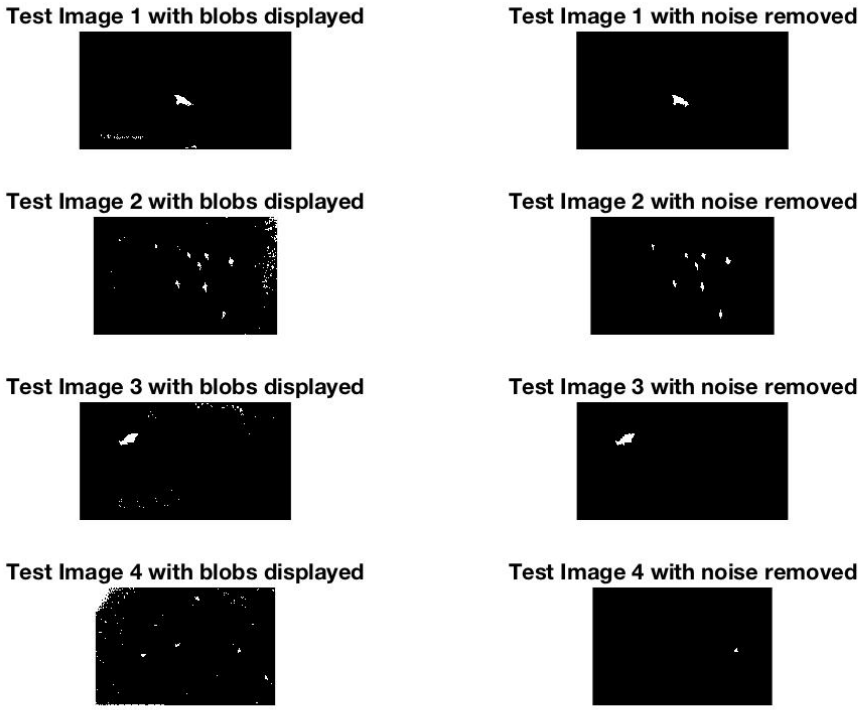

Figure 5.4: Binary maps of test images before and after morphology to remove noise.

As illustrated above, the basic morphology steps to clear the image work effectively. However, it can be seen in 'Test Image 4' that the minimum area was too large and has consequently removed most of the shark blobs. Although this factor can be easily tuned, it would be more desirable to assume a constant operating height of the UAV when it is on patrol. This way the blob area size can nearly be predicted to help narrow down false positives.

It would appear that by applying these simple filters that the vision system would perform quite well in recognising sharks in coastal waters. However, in its current state the system

will detect any dark object over a certain pixel area size. The following figure depicts a shark swimming near a surfer. The surfer is wearing a black wetsuit, and by analysing the Value array the pixel values are below the sharks. After performing the same morphology process as above, the binary map of the frame looks like:

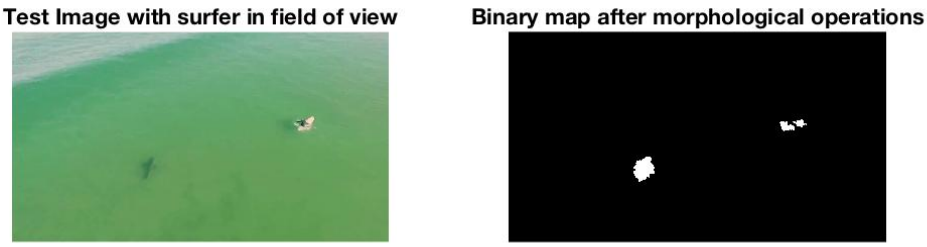

Figure 5.5: Morphology applied to frame with a surfer in the field of view.

To help filter out false positives such as surfers an analysis was performed on the detected blob region. By making the assumption that surfers always ride a bright coloured surfboard, additional conditions can be implemented to exclude blobs that contain a certain ratio of bright pixels. This type of exclusion would also remove objects such as:

1. Dark shapes creating a wake. The brightness of the white-caps could be tested and used to disregard such blobs. This could also be useful for ignoring false positives such as dolphins and whales jumping out of the water.

2. Boats that have large dark patches on them are more than likely to have some bright coloured components.

3. Rocks that have water crashing over them.

5.2 Region of Interest Filtering

5.2.1 Removing Bright Detections

As illustrated in the above sections, sharks tend to be the darkest object in the water when spotted swimming just below the surface. With the current algorithm, all dark objects over a certain pixel area will be detected and classified as possible sharks. In this section 'Region of Interest Filtering' is introduced to further filter out the detected false

positives. This is achieved by calculating the amount of bright pixels in a 'bounding box' relative to each blob. A 'bounding box' is a box that can be plotted over the detected blob that encases the limits of the blob. To extract bounding box coordinates relative to each blob the 'regionprops' function can be called. The following pseudo code depicts how the blobs are excluded for having a large bright pixel ratio:

Algorithm 4 Region of Interest Filtering

```
procedure BW = IMBINARIZE(Value)
    rp ← regionprops (BW,'BoundingBox')
    for i = 1:size(rp,1) do
        box ← rp(i).BoundingBox
        bright ← imcrop(Value,box)
        threshold ← min(Value)+(max(Value)-min(Value))*0.75
        [x,y] ← find(bright>threshold)
        brightest ← size(x,1)/(area of bounding box)
        if brightest > 0.01 then
            BW(box) ← 0
```

By applying this filter after the morphological operations the algorithm is able to remove surfers that would otherwise result in a false positive. The following figure is the output from the above filter:

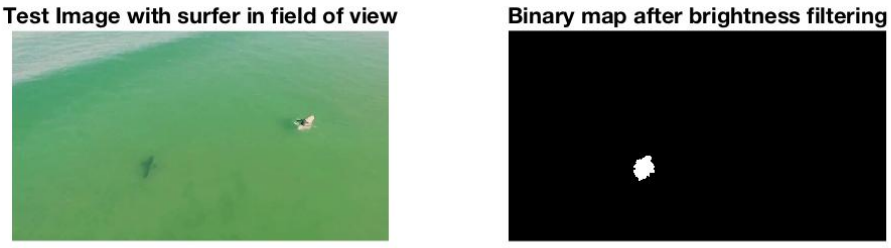

Figure 5.6: Morphology applied to frame with a surfer in the field of view.

It can be assumed that sharks swimming in the area are not going to be leaving any kind of wake or white water in their trail. By removing all the objects that are closely related to bright pixels the system will be able to reduce false positives.

5.2.2 Removing Dark Patches of Water

Throughout the simulations it was noticed that dark patches of water that fell below the pixel value threshold, were greater than the minimum area and were not connected to any bright pixels that were being detected. To filter these false positives a simple test was used to determine the change in pixel values within the bounding box of the blob. This is because when the bounding box is plotted over the blob, it encases the blob area in the smallest possible rectangle. This is relevant because there will be some pixels belonging to the surrounding water within the bounding box. If the blob detection happens to be a shark, there will be a considerable difference between the brightness values of the bounding box. However, if the detection is just simply a darker patch of water that has just made it through the pixel thresholding, there will be relatively no change in the pixel values. The following plot illustrates an example of this:

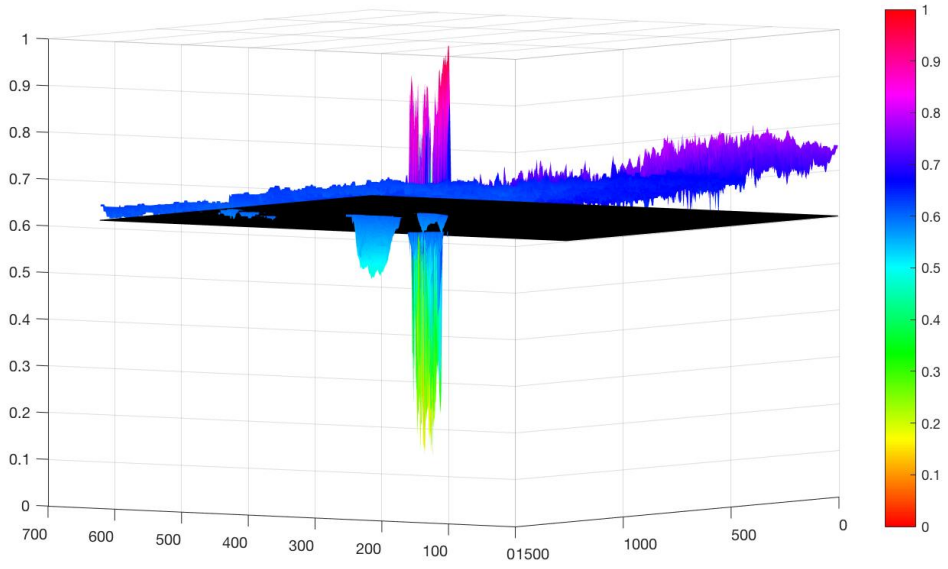

Figure 5.7: 3D surface model depicting dark patches of water being falsely detected.

In figure 5.7 the pixel threshold has been plotted over the 'Value' array to help illustrate which regions of the image fall below the threshold. There are three distinct objects:

1. A surfer who's black wetsuit and white surfboard cause a large spike in the data.
2. The first shark which can distinctly be made out and causes a considerable change in pixel values.
3. The second shark whose shadow can just visualised.

To the left of these objects are some dark patches of water that are just below the pixel threshold. The following pseudo code represents how these unwanted regions of interests are removed:

Algorithm 5 Remove Dark Patches of Water

```
procedure BW = IMBINARIZE(Value)
    rp ← regionprops (BW,'BoundingBox')
    for i = 1:size(rp,1) do
        box ← rp(i).BoundingBox
        DW ← imcrop(Value,box)
        difference ← max(DW)-min(DW)
        if difference < 0.1*max(DW) then
            BW(box) ← 0
```

The following figure illustrates the dark water being filtered out:

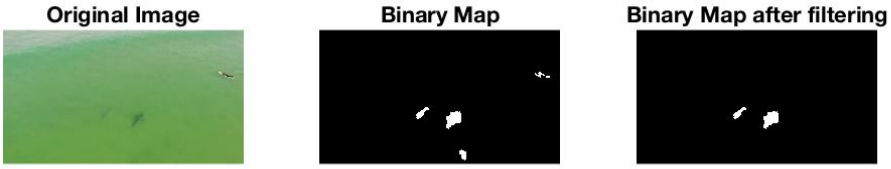

Figure 5.8: Original Image and stages in filtering out unwanted regions of interest.

5.3 Elliptic Ratio

Elliptic ratio thresholds have been used for dugong detection filters because "a blob is more likely to relate to a dugong if its shape is elliptical" (Maire et al. 2013). Reviewing the blobs that were used to make the database, it was noticed that the mean elliptic ratio was 1.18 with a standard deviation of 0.085. The elliptic ratio of the sharks was relatively close to one (perfect ellipse) and determined an accurate method to help identify sharks. The elliptic ratio threshold was therefore adopted as a filter to help further reduce false positives.

To determine a suitable threshold for the elliptic ratio, the elliptic ratios for the blob database were calculated and plotted as a histogram:

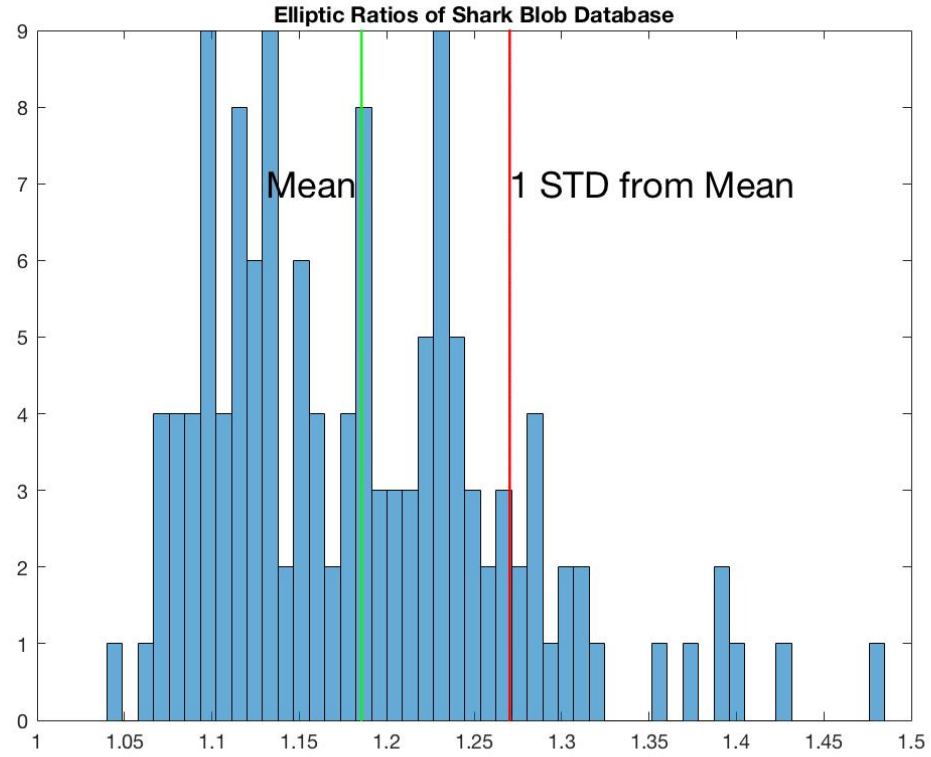

Figure 5.9: Histogram of Elliptic Ratios of Blob Database.

As illustrated, there are a few outliers that don't have a strong elliptic ratio. These outliers resemble blobs where the shark has been rotated in the out of plane direction and only one pectoral fin has been picked up. The elliptic ratios were calculated by:

$$EllipticRatio = \frac{\pi * MajorAxisLength * MinorAxisLength}{4 * Area} \tag{5.1}$$

A suitable threshold for the elliptic ratio was determined to be one standard deviation from the mean. This threshold will help further reduce false positives. It was noticed in the testing stages that the elliptic ratio for swimmers and surfers was typically around 1.6. With this method and the previous blob comparison ratio, the false positive rate should be adequately reduced.

5.4 Noise Filtering

With all these filters in place, it is still possible for false positives to occur. The variability of the current frame, due to the lighting and continuous wave motion, is infinite. Object tracking can be an effective method, where the consistency and trajectory of the object

detected is considered and classified accordingly. However, object tracking using computer vision has had the best results when the camera has been stationary. By using a kalman filter, an object's movement can be predicted by the recorded velocity or acceleration of the object, so that if occlusion happens or if the object is unable to be detected in the current frame, its location can still be tracked. Due to the nature of this project, where the data has been taken from a UAV that is constantly moving, the kalman filter has been unable to be investigated for its effectiveness in detecting and tracking of sharks.

A method for noise filtering, which compares the last five frames for the object detected, has been introduced to further reduce false positives. This filter serves two purposes:

1. The object must have been detected for a minimum of four consecutive frames before the system classifies the object as a shark.
2. If the object has been consistently detected and then the system fails to detect it for one frame, the system will still identify the shark.

This method is achieved by initialising a structure that contains five empty binary maps, equal to the size of the video frames. For each frame, each binary map in the structure is shifted upwards to replace the previous recorded binary maps. For example, the 5th map becomes the 4th, the 2nd map becomes the 1st and so on. With the last five binary maps saved, the current blob detected is put through a loop to test whether it has been detected in the last five frames. This is achieved by testing whether the centroid of the blob is equal to 1 in each frame. For each successful detection, a variable is incremented which flags the system to classify the shark once it reaches a certain value. If the system fails to detect the object for a couple frames but then detects the object again, a different variable is incremented which will eventually flag the system to ignore the classification.

The following pseudo-code represents how random blobs are filtered out. It begins by initialising the variables needed to store the last five binary maps, and demonstrates how the structure is updated and used to determine whether to classify the object or not.

Algorithm 6 Noise Filtering

```
for i = 1:5 do
    noise(i) = zeros(700,1260)
procedure MAIN LOOP(currentframe)
    BW ← BlobAnalysis(frame)
    maps = [ ]
    field = 'maps'
    for j = 1:4 do
        noise(j) = noise(j+1)
    noise(5) = BW
    maps = struct(field,noise)
    rp ← regionprops (BW,'Centroid','BoundingBox')
    for i = 1:size(rp,1) do
        centroid ← rp(i).Centroid
        for j = 1:5 do
            if BW(centroid(2),centroid(1))==1 && maps(j).maps(centroid(2),centroid(1))==1
then
                OnFlag = OnFlag+1
            elseOffFlag = OffFlag+1
        if OnFlag>4 then
            On = 1
        if OffFlag>3 then
            On = 0
        if On == 1 then
            box ← rp(i).BoundingBox
            rectangle('Position',box,'EdgeColor','y')
            text(box(1),box(2)-10,'SHARK','FontSize',16,'BackgroundColor','y')
```

This method proves to be an effective way to filter out momentary shadows that have passed through the other thresholds in place.

Chapter 6

Testing and Results

Three videos were sourced from Youtube to test the sensitivity of the algorithm. These three videos were ideal because they were aerial footage of sharks, taken at different locations under different conditions. A simulation was set up in MATLAB where the algorithm iteratively processed each frame. Red outlines have been placed over the blobs that have been detected in each frame, and if a shark was detected, a yellow box was also placed over the blob.

6.1 Analysis of performance on Testing Video 1

Testing video 1 had the most reliable footage of the shark, and also introduced other objects that could possibly give false positives. The footage was smooth and followed the shark at a consistent height. Different angles of the camera in relation to the water proved to be the biggest problem, where the reflection of the horizon would disrupt the variance of the pixel intensity distribution leading to the cut-off threshold being premature. The video was shot at 30 frames per second.

6.1.1 Detecting the shark

The opening frame of the test video shows the shark swimming towards the south-east of the frame. There are two surfers that are relatively close to the shark, with other surfers further back in the frame. The algorithm is able to identify the shark, by utilising the thresholding methods and filters mentioned in the previous section.

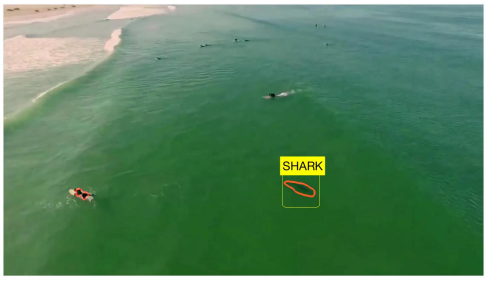

Figure 6.1: Frame 33: Shark detected in frame.

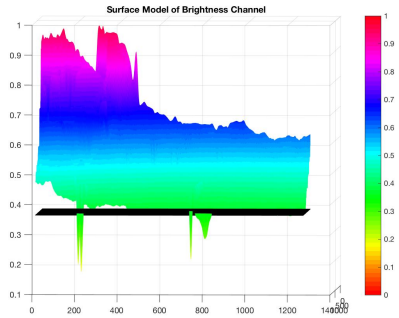

Figure 6.2: 3D surface model of frame 33.

Table 6.1: Frame 33: Values of blob elements

Blob	**Brightness (<0.01)**	**IQR (>0.02)**	**Elliptic Ratio (<1.27)**
Surfer	**0.0417**	0.139	1.12
Shark	0.00	0.034	1.02

As illustrated, the shark has been correctly labelled. The 3D surface model portrays the brightness values for each pixel. The calculated threshold, using the adaptive threshold model, is represented by the black plane. All pixels above this black plane are ignored after the pixel thresholding step, and all remaining blobs are filtered to reject false positives. In this particular frame, there is one other blob that could potentially register as a shark. This blob represents the surfer in the left of the frame, whose dark wetsuit tricks the system into identifying the object as a region of interest. However, the brightness ratio filter prevents the false positive.

The following frame shows where the detection has failed. A combination of the angle that the camera makes with the water and a wave rolling in disrupts the brightness pixel distribution and leaves the shark too submerged for the system to detect it.

Figure 6.3: Frame 84: False negative.

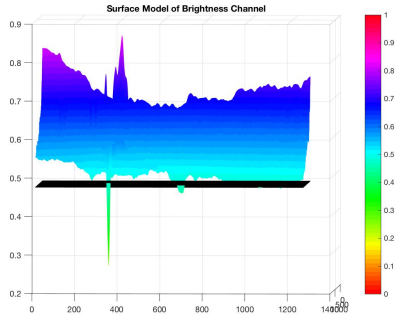

Figure 6.4: 3D surface model of frame 84.

Table 6.2: Frame 84: Values of blob elements

Blob	**Brightness (<0.01)**	**IQR (>0.02)**	**Elliptic Ratio (<1.27)**
Shark	0.00	**0.017**	1.02

As illustrated in the 3D surface model, there are two small blobs that are just below the brightness threshold. The first blob has an area that is calculated to be too small to be a shark. The second blob, which represents the shark, has a variance of 0.017 which does not meet the threshold used to determine dark water patches . This prevents the shark from being detected.

From the shark's first detection, in frame 33, until the second shark comes into view in frame 369, the system has consistently detected the shark. It only fails to classify the shark from frames 78 to 90. With a rate of 30 frames per second, the shark has been correctly detected for 11 seconds.

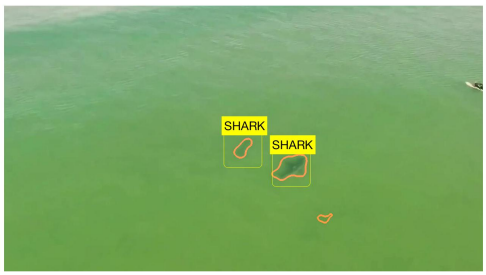

Figure 6.5: Frame 369: Two sharks detected in frame.

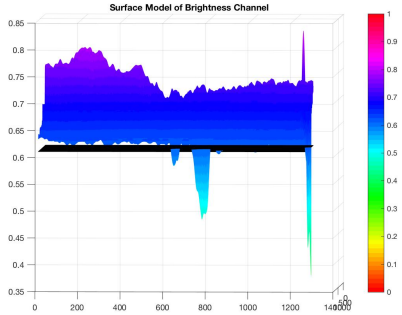

Figure 6.6: 3D surface model of frame 369.

Table 6.3: Frame 369: Values of blob elements

Blob	**Brightness (<0.01)**	**IQR (>0.02)**	**Elliptic Ratio (<1.27)**
Shark	0.00	0.027	1.05
Shark	0.0073	0.063	1.06
Shadow	0.00	**0.007**	1.19

In this frame the first shark is well defined, with its brightness values well beneath the threshold. Looking closely at the 3D surface model, it can be noted that the second shark (smaller peak to the left of the dominant peak) is barely below the cut-off value. This is because the second shark is still quite submerged, and therefore it is difficult for the system to detect. The variance for the second shark is 0.027, which is just above the

threshold of 0.02. It can be observed that the angle the camera makes with the water is favourable as there is little to no reflection. Also the brightness of the water allows for a strong contrast so that the shark can be detected.

Although the shark detection algorithm appears to be strongly identifying the shark, it can exhibit false negatives where the filters in place are potentially too sensitive. In the following frame the system has failed to detect both sharks. It is clear from the 3D surface model that the pixels relative to the shark are beneath the cut-off threshold.

Figure 6.7: Frame 498: Missed detection of two sharks.

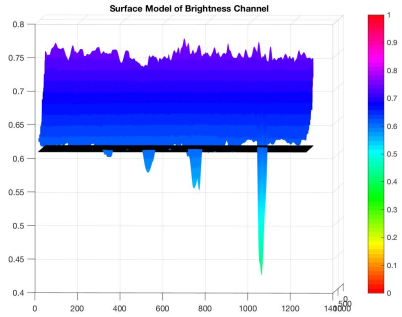

Figure 6.8: 3D surface model of frame 498.

Table 6.4: Frame 498: Values of blob elements

Blob	**Brightness (<0.01)**	**IQR (>0.02)**	**Elliptic Ratio (<1.27)**
Shadow	0.00	**0.008**	1.02
Shark	**0.04**	0.021	1.03
Shark	**0.0573**	0.03	1.02
Shadow	0.00	**0.008**	1.09
Surfer	**0.5066**	0.097	1.12

The brightness filter appears to be too sensitive. With a cut-off threshold of 0.01, the calculated values for the brightness ratio's of the shark blobs is too high. The system then rejects the blobs as being sharks.

In the following frame, the main shark that was being consistently detected in earlier frames has been missed whilst the second shark has been correctly detected. This is due to the variance of the pixel values not being above the threshold. The wave is rolling over the shark, submerging the shark and making it more difficult for the system to identify.

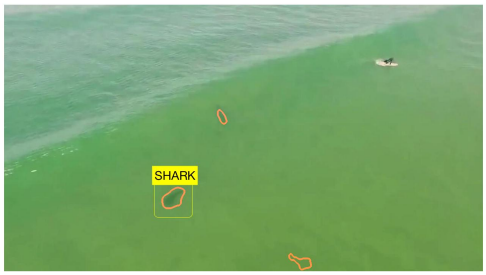

Figure 6.9: Frame 595: One true positive, One false negative and One true negative.

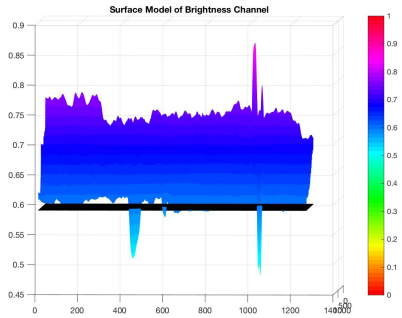

Figure 6.10: 3D surface model of frame 595.

Table 6.5: Frame 595: Values of blob elements

Blob	**Brightness (<0.01)**	**IQR (>0.02)**	**Elliptic Ratio (<1.27)**
Shark	0.00	0.045	1.02
Shark	0.00	**0.019**	1.01
Shadow	0.00	**0.004**	1.26

Although the dark water filter is proving to be a little too sensitive, the sharks are still being detected the majority of the time. It can be noted that the system has not made any false detections yet. This is due to the success of the different filters being applied to reject any false positives. However it is possible that the system may be too heavily filtered because of the false negatives. As mentioned earlier, a weighted system may have more success in identifying the sharks using the last known location.

It can be seen from the 3D surface model that part of the shark is just below the threshold, but because of the minimal area, it is rejected as being a blob that could represent a shark.

When the shark starts to get too far away from the UAV, and the angle that the camera makes with the water becomes too shallow, the system has trouble detecting the shark. In the following frame, the shark can barely be identified by the human eye but because we know there is a shark in the area an element of bias is put against the system. There is nothing definitive about the small dark shadow, and if there was no prior knowledge of a shark in the area, it could be mistaken for seaweed or a rock.

Figure 6.11: Frame 711: False negative.

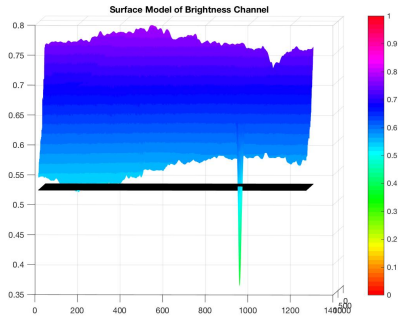

Figure 6.12: 3D surface model of frame 711.

Table 6.6: Frame 711: Values of blob elements

Blob	**Brightness (<0.01)**	**IQR (>0.02)**	**Elliptic Ratio (<1.27)**
Shadow	0.00	**0.002**	1.05

By analysing the 3D surface model, it is clear why the system failed to identify the shark. The brightness properties of the pixels have less variance, and the small portion of pixels dark enough to be below the threshold (a handful at x=200, z=0.53) are ignored because of the minimum area and dark water filtering.

In frame 883, the value for the elliptic ratio is 1.04. It is clear from the 3D surface model that the shark blob is well beneath the threshold, and there is plenty of variance of brightness values within the bounding box. However, the brightness ratio is too sensitive, and therefore rejects the shark detection.

Figure 6.13: Frame 883 False negative.

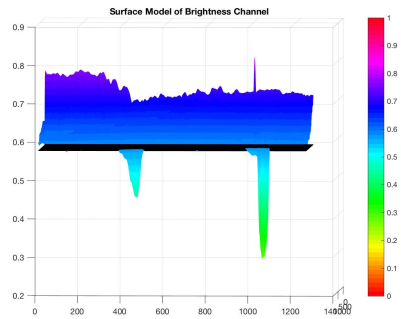

Figure 6.14: 3D surface model of frame 883.

Table 6.7: Frame 883: Values of blob elements

Blob	Brightness (<0.01)	IQR (>0.02)	Elliptic Ratio (<1.27)
Shark	**0.0144**	0.027	1.04
Surfer	**0.3151**	0.04	1.30

The algorithm rejects the surfer as being a shark by the high value for the brightness ratio. Future adjustments and improvements may involve lifting the brightness ratio to prevent false negatives of the sharks.

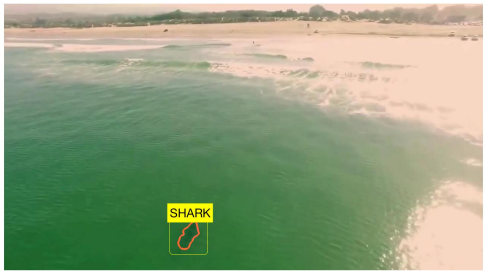

Figure 6.15: Frame 1545: Shark successfully detected.

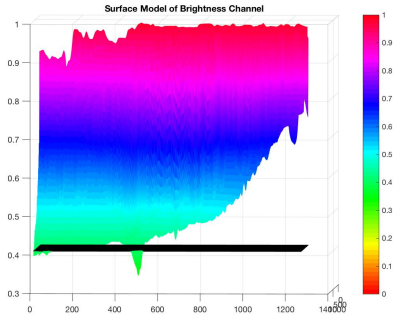

Figure 6.16: 3D surface model of frame 1545.

Table 6.8: Frame 1545: Values of blob elements

Blob	Brightness (<0.01)	IQR (>0.02)	Elliptic Ratio (<1.27)
Shark	0.00	0.037	1.03

Testing video 1 has provided some good results, proving that the algorithm can adequately reject false positives and detect sharks in the frame. Factors such as waves rolling in, the angle the camera makes with the water and the distance of the UAV to the shark have been noted to negatively influence the detection of the shark.

6.2 Analysis of performance on Testing Video 2

Testing video 2 shows a shark swimming in choppy green water. It has been taken from a helicopter that is being buffeted around by the wind, and has various zoom lengths. The rolling whitecaps over the shark cause the brightness ratio to be too high in some cases, which results in the system rejecting the detection. The buffeting of the helicopter also causes a problem which prevents the shark from being detected.

6.2.1 Detecting the shark

The shark is quickly detected in the video footage. The view of the camera allows the shark to be of an acceptable area, there are no other objects in the frame and there is no reflection from the horizon to skew to brightness values. The following figure illustrates the blob that has been detected and the shark that has been classified.

Figure 6.17: Frame 4: Shark detected in frame.

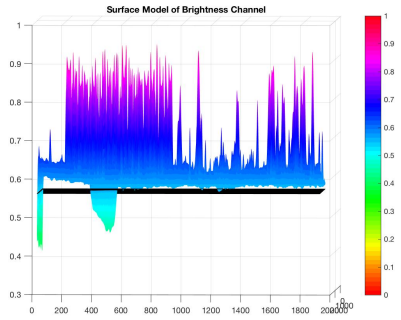

Figure 6.18: 3D surface model of frame 4.

Table 6.9: Frame 0004: Values of blob elements

Blob	**Brightness (<0.01)**	**IQR (>0.02)**	**Elliptic Ratio (<1.27)**
Shark	0.00	0.056	1.08

The filter thresholds used seem to be ideal for this environment. The brightness values are low, there is enough variance and the elliptic ratio value is well within the limits. It can be noted from the 3D surface model that the shark is the only object dark enough to be below the cut-off, proving that the adaptive threshold is working appropriately.

In frame 297, the shark detection is lost. It can be seen from the 3D surface model that the pixels relevant to the shark are below the threshold, but because the area is too small they have been ignored. The shark has been consistently detected up until this point, which equates to 10 seconds.

Figure 6.19: Frame 297: False negative.

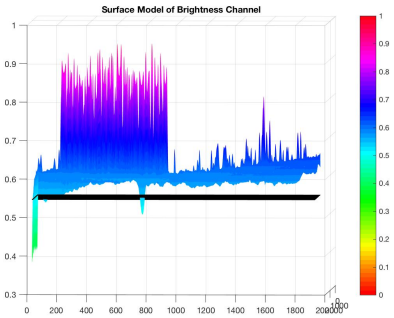

Figure 6.20: 3D surface model of frame 297.

If this shark detection algorithm was to be installed into UAVs for real time patrolling of beaches, a maximum operating height would have to be set so that the detected shark blobs would not be rejected for their minimal size.

In frame 360, the camera has zoomed out further so the beach the shark is swimming near can be seen. The shark is a small black dot in the middle of the frame, and their are dark patches of water that may be related to seaweed scattered about the frame.

Figure 6.21: Frame 360: False negative.

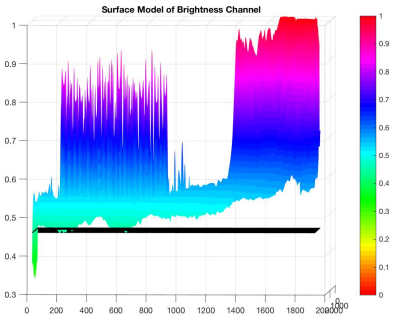

Figure 6.22: 3D surface model of frame 360.

Table 6.10: Frame 360: Values of blob elements

Blob	**Brightness (<0.01)**	**IQR (>0.02)**	**Elliptic Ratio (<1.27)**
Shadow	0.00	**0.006**	1.19

There has been one blob detected in the upper left corner of the frame that relates to a dark patch of water. The variance of the blob is very low and therefore the blob has been rejected as being a shark.

Midway through the video, there are some control features that the original editor was

using that pop up on the screen. Because these control features are displayed over the video, and are black in colour, the pixel distribution for the brightness channel is negatively skewed. This results in the shark blob not being detected for the duration that these control features are displayed.

Figure 6.23: Frame 472: False negative.

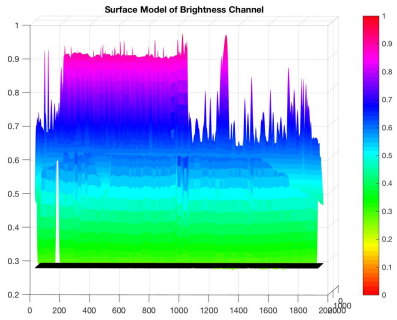

Figure 6.24: 3D surface model of frame 472.

It can be seen from the 3D surface model why the shark blob has not been detected. The cut-off threshold is at the bottom of the plot, meaning that every pixel that has a brightness value lighter than the control bar has been ignored.

The shark unfortunately is not detected for the rest of the video, which is another 12 seconds. The lighting is favourable, there is enough variance in the blobs detected and the elliptic ratio is below the threshold.

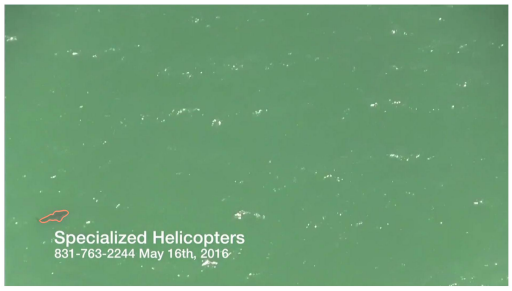

Figure 6.25: Frame 529: False negative.

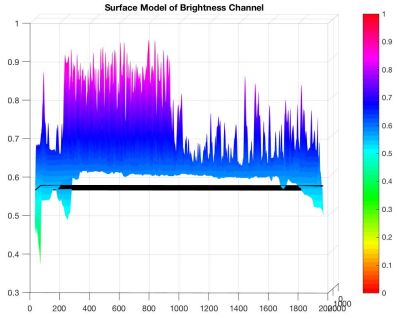

Figure 6.26: 3D surface model of frame 529.

Table 6.11: Frame 529: Values of blob elements

Blob	**Brightness (<0.01)**	**IQR (>0.02)**	**Elliptic Ratio (<1.27)**
Shark	0.00	0.064	1.10

The reason that the shark has not been detected is because the noise filter is rejecting the

detection. The camera is not steady enough for the centroids to line up over the past five frames. Therefore the system thinks that the blobs are randomly appearing and are not related to a consistent object. A closer zoom and less movement would allow the blobs to overlap, which would then result in the successful detection of the shark.

The following frame shows the blob of the shark being detected, but because of the noise filter, the detection has been rejected.

Figure 6.27: Frame 800: False negative.

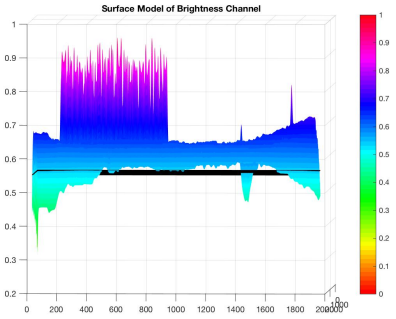

Figure 6.28: 3D surface model of frame 800.

Table 6.12: Frame 800: Values of blob elements

Blob	**Brightness (<0.01)**	**IQR (>0.02)**	**Elliptic Ratio (<1.27)**
Shark	0.00	0.054	1.04

Testing video 2 has provided some good results. The filters are working successfully, as there has not been any false positives. The noise filter however prevented the majority of the successful shark detections. This means that if a UAV was to be in windy and rough conditions, it would have problems detecting the shark. Possible remedies to this may be to reduce the sensitivity of the filter, or to set the operating height of the UAV to a shorter distance so the shark blob is bigger, and therefore greater disturbance forces are needed to prevent the overlap of blobs between frames.

6.3 Analysis of performance on Testing Video 3

Testing video 3 is a recent video taken from a helicopter hovering over a beach in Kingscliffe, NSW. It is suspected that this particular shark is related to an attack that occurred at Lighthouse Beach in Ballina, NSW, on the 25th of September 2016. The footage is somewhat steady, and exhibits some different zoom lengths.

6.3.1 Detecting the shark

In the opening frames of the video, the person controlling the camera takes a moment to zoom in on the shark and hold the camera steady. The shark blob is detected by the system, but because of the movement of the camera, the noise filter rejects the blob as being classified as a shark.

Figure 6.29: Frame 65: False negative.

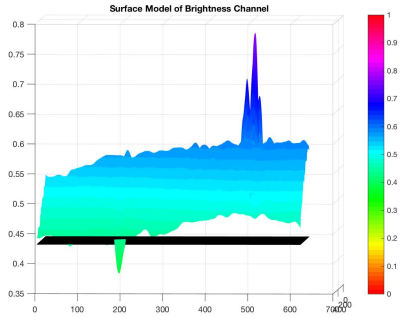

Figure 6.30: 3D surface model of frame 65.

Table 6.13: Frame 65: Values of blob elements

Blob	**Brightness (<0.01)**	**IQR (>0.02)**	**Elliptic Ratio (<1.27)**
Shark	0.00	0.053	1.03

It can be seen from the 3D surface model that the adaptive threshold is working properly. The adaptive threshold was designed so that it could calculate the desired threshold for the shark blob for any frame size or lighting condition. The three testing videos used have all been of a different resolution, and the adaptive threshold has not needed to be adjusted once.

A few frames later the camera has steadied enough for the noise filter to allow the shark detection. The contrast between the shark blob and the surrounding water is not the strongest, but the system is still able to successfully detect the shark. The angle the camera makes with the water is complimenting the system, as it reduces the amount of reflection on the water.

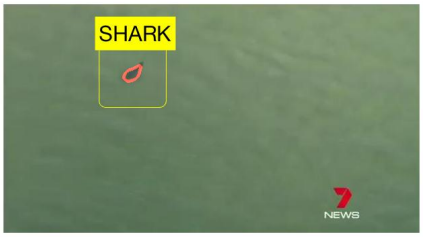

Figure 6.31: Frame 71: Shark detected in frame.

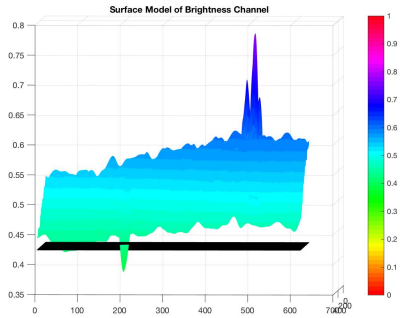

Figure 6.32: 3D surface model of frame 71.

Table 6.14: Frame 71: Values of blob elements

Blob	**Brightness (<0.01)**	**IQR (>0.02)**	**Elliptic Ratio (<1.27)**
Shark	0.00	0.051	1.03

The variance of the blob is well above the threshold, the elliptic ratio of the blob is well below the threshold and the brightness ratio is 0. These qualities and thresholds are proving to be successful descriptors for blobs related to sharks.

The camera is then zoomed in, enabling a close up view of the shark. The shark is quite submerged in the water, making it difficult to detect. A wave rolling over the shark distorts its shadow underneath, separating the blob into two blobs.

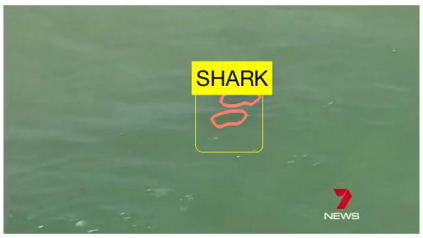

Figure 6.33: Frame 184: Two blobs representing one shark.

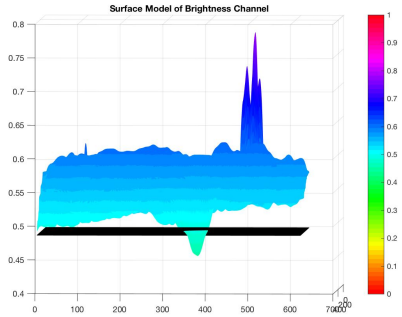

Figure 6.34: 3D surface model of frame 184.

Table 6.15: Frame 184: Values of blob elements

Blob	**Brightness (<0.01)**	**IQR (>0.02)**	**Elliptic Ratio (<1.27)**
Shark	0.00	**0.012**	1.02
Shark	0.00	0.034	1.06

One of the blobs detected does not have a variance that meets the threshold and is therefore rejected. If there was in fact two sharks present, the noise filter would need two consistent centroids over the past five frames to allow a double detection. However, because there is only one shark, the noise filter would disregard the second blob if its variance was above the threshold.

The system successfully detects the shark for 10 seconds, demonstrating that the filters are working properly to reject any blobs that are not related to the shark.

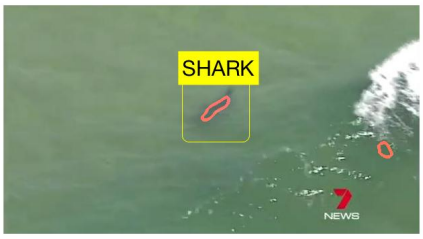

Figure 6.35: Frame 402: Shark detected in frame.

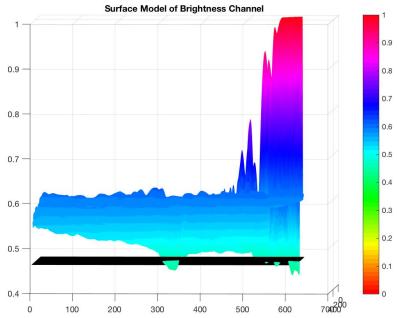

Figure 6.36: 3D surface model of frame 402.

Table 6.16: Frame 402: Values of blob elements

Blob	**Brightness (<0.01)**	**IQR (>0.02)**	**Elliptic Ratio (<1.27)**
Shark	0.00	0.046	1.05
Shadow	0.00	**0.008**	1.02

The 3D surface model illustrates that the adaptive threshold is quite low, and only just detecting the shark. This could be due to the whitewash that is present in the frame. The following frame shows how the whitewash has skewed the pixel distribution enough so that no blobs have been detected.

Figure 6.37: Frame 433: Whitewash rejects shark detection.

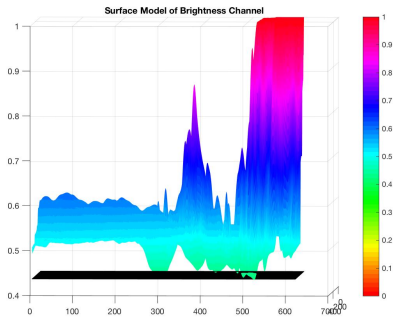

Figure 6.38: 3D surface model of frame 433.

The skewing of the pixel distribution from the whitewash might mean that the camera view is too close, and to obtain optimum results for the shark detection system the UAV would have to be zoomed out further or operating at a greater height. Another possible explanation may be that the whitecaps over the shark have disrupted the dark pixel count, and therefore the threshold has been lowered.

The presence of whitewash does not automatically mean that the system will not detect the shark. The following frame shows the shark being detected once the whitecaps over the shark have disappeared.

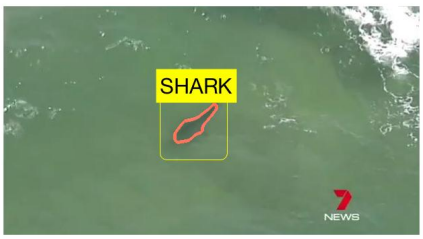

Figure 6.39: Frame 453: Shark detected in frame.

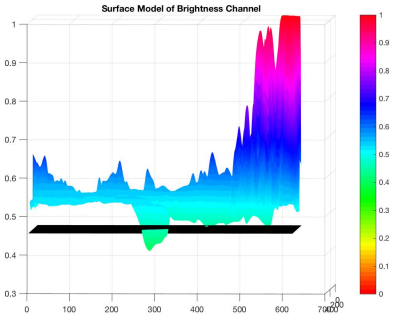

Figure 6.40: 3D surface model of frame 453.

Table 6.17: Frame 453: Values of blob elements

Blob	**Brightness (<0.01)**	**IQR (>0.02)**	**Elliptic Ratio (<1.27)**
Shark	0.00	0.072	1.10

As illustrated by the 3D surface model, the threshold calculation has performed better after the whitecaps over the shark blob have disappeared.

The following frame shows how the camera has been zoomed out, and the shark blob has too small of an area to be considered as a shark.

Figure 6.41: Frame 885: Area of shark too small for detection.

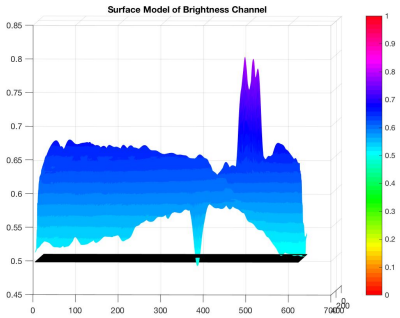

Figure 6.42: 3D surface model of frame 885.

Table 6.18: Frame 885: Values of blob elements

Blob	**Brightness (<0.01)**	**IQR (>0.02)**	**Elliptic Ratio (<1.27)**
Shadow	0.00	**0.004**	1.18

The 3D surface model shows the small peak of the shark beneath the threshold. This means that the shark blob has been detected, but the area was too small. A dark patch of water in the top right corner has also been detected. The dark water filter has successfully rejected this blob by calculating its variance.

The camera zooms back in, allowing the system to detect the shark as its area is over the minimum area threshold. The dark water filter continues to reject dark patches of water as being sharks by calculating their variance.

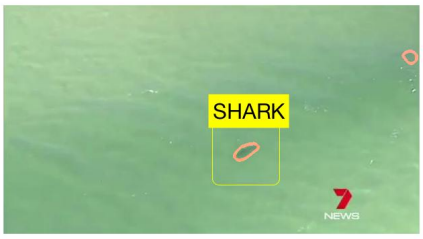

Figure 6.43: Frame 1103: Shark detected in frame.

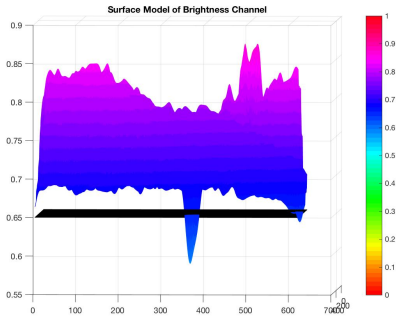

Figure 6.44: 3D surface model of frame 1103.

Table 6.19: Frame 1103: Values of blob elements

Blob	**Brightness (<0.01)**	**IQR (>0.02)**	**Elliptic Ratio (<1.27)**
Shark	0.00	0.048	1.01
Shadow	0.00	**0.012**	1.01

The 3D surface model illustrates the difference between the variance values of the detected blobs. The blob classified as a shark has a greater negative peak than the blob related to a dark patch of water.

In the following frame, the camera has been zoomed out leaving the shark unidentifiable. This leads to the algorithm's first false positive, where the darks shadows underneath the water have passed all the criteria to be classified as a shark.

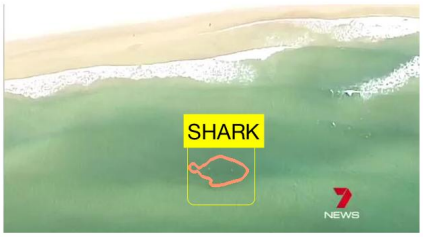

Figure 6.45: Frame 3663: False positive.

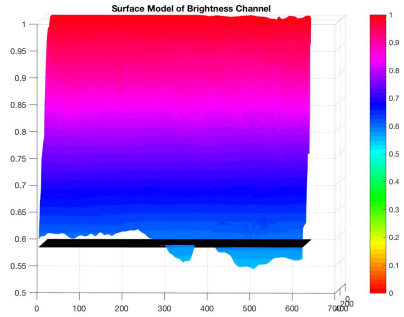

Figure 6.46: 3D surface model of frame 3663.

Table 6.20: Frame 3663: Values of blob elements

Blob	**Brightness (<0.01)**	**IQR (>0.02)**	**Elliptic Ratio (<1.27)**
Shadow	0.00	0.023	1.07

This false positive suggests that the system would detect any dark object in the water that passes all the thresholds. Due to the nature of the project and the quality of the data, stationary objects have not been able to be differentiated from moving objects. To help prevent false positives related to these instances, kalman filters would be used to track the path of the object. If the object was found to not be moving, it would be rejected as being a shark.

For the last minute of the video, the camera is zoomed in on the shark as it is swimming through very shallow waters. The reflection from the sand underneath causes the frame to be very bright, which results in the brightness filter rejecting the shark detection.

Figure 6.47: Frame 5000: Brightness ratio rejects shark.

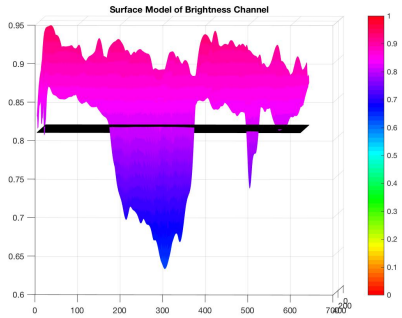

Figure 6.48: 3D surface model of frame 5000.

Table 6.21: Frame 5000: Values of blob elements

Blob	**Brightness (<0.01)**	**IQR (>0.02)**	**Elliptic Ratio (<1.27)**
Shark	**0.4412**	0.105	1.07
Logo	**0.6139**	0.056	1.16

It can be seen from the 3D surface model and the red outline on the frame that the shark blob was clearly detected. However the brightness filter has not been designed to adapt to the lighting conditions of the current scene, and is therefore rejecting the shark blob. The calculated value of 0.4412 for the brightness ratio is excessively high compared to the cut-off limit of 0.01.

Testing video 3 has provided some good results and identified parts of the algorithm that could be improved to ensure better detection rates. Key elements such as adding a kalman filter to track the movement of blobs, and a brightness filter that can adapt to the current lighting conditions of the frame, could enable the shark detection algorithm to be robust enough to be used in universal environment conditions.

Chapter 7

Conclusions and Future Work

7.1 Achievement of Project Objectives

The following objectives have been addressed:

- **Carry out a literature review that is relevant to image segmentation, object recognition, object classification and object tracking:**

  Background information on common computer vision applications has been presented in Chapter 1. A review of previous research on marine object detection and tracking using computer vision has been presented in Chapter 2. This also includes research relevant to object recognition and blob analysis outside the marine environment.

- **Obtain aerial videos of sharks in coastal waters to begin designing the algorithm with.**

  Aerial videos were sourced on the internet and downloaded from Youtube. These aerial videos provided a stable testing method for the shark detection algorithm.

- **Write a basic computer vision program that can detect the 'blob'.**

  Chapter 4 discussed the analysis of the different colour spaces and channels and how the shark blobs could be depicted from them. The brightness channel was selected from the HSV colour space as an initial thresholding technique to reject blobs over a determined brightness. Chapter 4 also discussed the design of an adaptive threshold so that the optimum threshold value could be calculated for each frame, leaving the

blobs of interest for further filtering.

- **Expand on the basic blob detection algorithm by trying to filter out all of the blobs that are not relative to a shark.**

  Chapter 5 discussed the design of four key filters used to identify false positives. These filters were:

  - **Brightness ratio filter**

    This filter was used to reject blobs that were associated with bright pixels. It was originally designed to filter out surfers that were being detected because of their dark wetsuits. The bright pixels associated with surfboards and the white wash created was used to reject these blobs. This filter was predominantly successful and only caused problems in the last testing video when the shark was in very shallow water. The system would benefit from an adaptive brightness ratio filter to suit the current lighting conditions of the environment.

  - **Variance filter**

    This filter was used to reject blobs that were associated with shadows and dark patches of water. The ever changing reflection on the water caused the brightness values of the water to fluctuate. This filter calculated the variance for each blob detected, and rejected the blob if it was not over a determined threshold. This filter was predominately successful and only caused problems when the shark became too submerged, or if a wave was rolling over the shark.

  - **Elliptic ratio filter**

    This filter was used to reject blobs that did not have an elliptic shape. By processing the first 10 seconds of testing video 1 and calculating the elliptic ratio for each shark blob, a relationship was discovered that suggested shark blobs have a strong elliptic resemblance. A threshold limit was determined from the data, and each blob detected was tested against this limit to determine if its shape was elliptic. The filter was successful in rejecting irregular blob shapes and did not cause any false negatives.

  - **Noise Filter**

    This filter was used to reject blobs that were randomly detected from the changing reflections in the waves. It compared the last five binary maps and overlapped consistent blobs to determine whether they were random or not. This allowed random detections that were not related to consistent objects in

the frame to be rejected. This filter was successful in reducing false positives and maintaining constant detection of the shark.

- **If the algorithm is unsuccessful across the dataset, tune it for each individual photo/video and record what changes made it successful (if any).**

  The adaptive threshold designed in Chapter 4 allowed the algorithm to have a high detection rate across the different testing videos. The only adjustment that might prove beneficial is deriving a formula to calculate an adaptive threshold for the brightness ratio. Towards the end of testing video 3 it was proven that the brightness ratio was excessively sensitive for the current lighting conditions.

7.2 Future Work

To really test and improve on the shark detection algorithm, a drone would need to be built capable of withstanding high wind speeds for flying out over the coast and recording data. Communications would need to be established between the drone and a control station for the imagery and detections to be relayed back to. On board analysis could be explored or left to an advanced computer in the control station.

Applications such as a kalman filter cannot be implemented until the drone can be programmed to first identify a blob of interest, and then hover so the drone can imitate a stationary platform. This would allow the shark to be tracked relative to the drone, and then could be identified whether the blob of interest is moving.

This future work could also pave the way for more complicated machine learning algorithms, used to differentiate between marine life such as sharks, whales and dolphins. By using the kalman filter to track and identify how a shark swims, it could be used to segment the identified shark blob to extract the tail movements and analyse them.

References

Atkins, S., Cliff, G. & Pillay, N. (2013), 'Short Note Multiple Captures of Humpback Dolphins (Sousa plumbea) in the KwaZulu-Natal Shark Nets , South Africa', *Aquatic Mammals* **39**(4), 397–400.

Australia, E. (2010), 'Code of Ethics', `https://www.engineersaustralia.org.au/western-australia-division/code-ethics`. [Online; accessed 2016-05-24].

Bodhe, T. S. (2013), Selection of Color Space for Image Segmentation in Pest Detection, *in* 'Advances in Technology and Engineering (ICATE), 2013 International Conference', pp. 1–7.

Cham, T.-j. & Rehg, J. M. (1999), A Multiple Hypothesis Approach to Figure Tracking, *in* 'Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference', number Cvpr 99, pp. 239–245.

Chang, O. (2016), 'NSW premier Mike Baird has launched a $ 250 , 000 shark-spotting drone', `http://www.businessinsider.com.au/nsw-premier-mike-baird-has-launched-a-250000-shark-spotting-drone-2016-2`. [Online; accessed 2016-05-11].

Chen, X. & Chen, H. (2010), A Novel Color Edge Detection Algorithm in RGB Color Space, *in* 'IEEE 10th INTERNATIONAL CONFERENCE ON SIGNAL PROCESSING PROCEEDINGS', pp. 793–796.

Choong, M. Y., Kow, W. Y., Chin, Y. K., Angeline, L., Tze, K. & Teo, K. (2012), Image Segmentation via Normalised Cuts and Clustering Algorithm, *in* 'Control System, Computing and Engineering (ICCSCE), 2012 IEEE International Conference', pp. 430–435.

Cianciotto, F. T. P. (1997), Detection and Classification of Subsurface Objects in a Marine

Environment bv the use of a Lidar Svstem ", *in* 'Aerospace and Electronics Conference, 1997. NAECON 1997., Proceedings of the IEEE 1997 National', pp. 463–468.

Connell, C. P. O., Andreotti, S., Rutzen, M., Meÿer, M., Matthee, C. A. & He, P. (2014), 'Journal of Experimental Marine Biology and Ecology Effects of the Sharksafe barrier on white shark (Carcharodon carcharias) behavior and its implications for future conservation technologies', *Journal of Experimental Marine Biology and Ecology* **460**, 37–46.

Fabic, J. N., Turla, I. E., Capacillo, J. A., David, L. T. & Naval, P. C. (2013), Fish Population Estimation and Species Classification from Underwater Video Sequences using Blob Counting and Shape Analysis, *in* 'Underwater Technology Symposium (UT), 2013 IEEE International Conference', pp. 1–6.

Farmer, M. E. & Jain, A. K. (2005), A Wrapper-Based Approach to Image Segmentation and Classification, *in* 'IEEE Transactions on Image Processing', Vol. 14, pp. 2060–2072.

Feng, L., Xiaoyu, L. & Yi, C. (2014), An efficient detection method for rare colored capsule based on RGB and HSV color space, *in* 'Granular Computing (GrC), 2014 IEEE International Conference', pp. 175–178.

Fletcher, L., Apostoloff, N., Chen, J. & Zelinsky, A. (2001), 'Computer Vision for Vehicle Monitoring and Control', (November), 14–15.

Hess, R. (2009), 'Particle Filter Object Tracking', `http://blogs.oregonstate.edu/hess/code/particles/`. [Online; accessed 2016-05-25].

Holmes, B. J., Pepperell, J. G., Griffiths, S. P., Jaine, F. R. A., Tibbetts, I. R. & Bennett, M. B. (2014), 'Tiger shark (Galeocerdo cuvier) movement patterns and habitat use determined by satellite tagging in eastern Australian waters', pp. 2645–2658.

Images, G. (2006), 'Sharkdrum', `www.dpc.wa.gov.au`. [Online; accessed 2016-05-11].

Images, G. (2009*a*), 'Sharknet', `https://en.wikipedia.org/wiki/Shark{_}net`. [Online; accessed 2016-05-11].

Images, G. (2009*b*), 'Sharktag', `http://news.stanford.edu/news/2009/november2/gifs/sharks{_}doubletag{_}news.jpg`. [Online; accessed 2016-05-11].

Images, G. (2015), 'Sharkpatrol', http://www.goldcoastbulletin.com.au/lifestyle/beaches-and-fishing/aerial-shark-patrols-have-been-discredited-by-a-study-but-northe news-story/ec8de0448b04393ee18da455d8a5f0a5. [Online; accessed 2016-05-11].

Images, G. (2016*a*), 'HSV_color_solid_cylinder_alpha_lowgamma @ upload.wikimedia.org', https://upload.wikimedia.org/wikipedia/commons/0/0d/HSV{_}color{_}solid{_}cylinder{_}alpha{_}lowgamma.png. [Online; accessed 2016-05-18].

Images, G. (2016*b*), 'LAB-Color-Model @ archive.xaraxone.com', http://archive.xaraxone.com/guest/guest58/LAB-Color-Model.png. [Online; accessed 2016-05-18].

Images, G. (2016*c*), 'photoshop-body-evolution @ www.slrlounge.com', https://www.slrlounge.com/wp-content/uploads/2015/01/photoshop-body-evolution.1.jpg. [Online; accessed 2016-05-18].

Images, G. (2016*d*), 'Risk Scoring matrix', http://3.bp.blogspot.com/-Pu8ELmJOQ30/UX3ebsAXNKI/AAAAAAAAAU4/JkGmt7iLAD4/s1600/RiskScoringmatrix.jpg. [Online; accessed 2016-05-23].

Images, G. (2016*e*), 'what-is-color-space-RGB @ www.arcsoft.com', http://www.arcsoft.com/images/topics/darkroom/what-is-color-space-RGB.jpg. [Online; accessed 2016-05-18].

Insights, D. (2016), 'Understanding Color Models and Spot Color Systems', http://www.designersinsights.com/designer-resources/understanding-color-models. [Online; accessed 2016-05-18].

Jun, G., Kragh, M., Kryger, P. & Ahrendt, P. (2016), 'Automatic behaviour analysis system for honeybees using computer vision', *Computers and Electronics in Agriculture* **122**, 10–18.

Kothiya, S. V. (2015), A Review on Real Time Object Tracking in Video Sequences, *in* '2015 International Conference on Electrical, Electronics, Signals, Communication and Optimization (EESCO)', pp. 1–4.

Lu, G., Peng, L., Zhang, L., Li, W. & Zhang, Y. (2015), Two-Step Pedestrian Searching Method Based on Lab Color Space, *in* 'Industrial Electronics and Applications (ICIEA), 2015 IEEE 10th Conference', pp. 1707–1711.

Maire, F., Mejias, L. & Hodgson, A. (2014), A Convolutional Neural Network for Automatic Analysis of Aerial Imagery, *in* 'Digital lmage Computing: Techniques and Applications (DlCTA), 2014 International Conference', pp. 1–8.

Maire, F., Mejias, L., Hodgson, A. & Duclos, G. (2013), Detection of Dugongs from Unmanned Aerial Vehicles, *in* '2013 IEEE/RSJ International Conference on Intelligent Robots and Systems', pp. 2570–2756.

Mathworks (2016*a*), 'CellSegmentationExample_01 @ au.mathworks.com', `http://au.mathworks.com/help/examples/images{_}product/CellSegmentationExample{_}01.png`. [Online; accessed 2016-05-21].

Mathworks (2016*b*), 'CellSegmentationExample_02 @ au.mathworks.com', `http://au.mathworks.com/help/examples/images{_}product/CellSegmentationExample{_}02.png`. [Online; accessed 2016-05-21].

Mathworks (2016*c*), 'CellSegmentationExample_03 @ au.mathworks.com', `http://au.mathworks.com/help/examples/images{_}product/CellSegmentationExample{_}03.png`. [Online; accessed 2016-05-21].

Mathworks (2016*d*), 'CellSegmentationExample_04 @ au.mathworks.com', `http://au.mathworks.com/help/examples/images{_}product/CellSegmentationExample{_}04.png`. [Online; accessed 2016-05-21].

Mathworks (2016*e*), 'CellSegmentationExample_05 @ au.mathworks.com', `http://au.mathworks.com/help/examples/images{_}product/CellSegmentationExample{_}05.png`. [Online; accessed 2016-05-21].

Mathworks (2016*f*), 'CellSegmentationExample_06 @ au.mathworks.com', `http://au.mathworks.com/help/examples/images{_}product/CellSegmentationExample{_}06.png`. [Online; accessed 2016-05-21].

Mathworks (2016*g*), 'CellSegmentationExample_07 @ au.mathworks.com', `http://au.mathworks.com/help/examples/images{_}product/CellSegmentationExample{_}07.png`. [Online; accessed 2016-05-21].

Mathworks (2016*h*), 'Detecting a Cell Using Image Segmentation'.

Mathworks (2016*i*), 'Object Detection', `http://au.mathworks.com/discovery/object-detection.html`. [Online; accessed 2016-05-24].

Miller, G., Fels, S. & Oldridge, S. (2011), A Conceptual Structure for Computer Vision, *in* 'Computer and Robot Vision (CRV), 2011 Canadian Conference', pp. 168–174.

Murahira, K. & Taguchi, A. (2012), Hue-Preserving Color Image Enhancement in RGB Color Space with Rich Saturation, *in* 'Intelligent Signal Processing and Communications Systems (ISPACS), 2012 International Symposium', number Ispacs, pp. 266–269.

News.com.au (2015), 'An app that follows sharks in real-time amazes beachgoers and scientists', `http://www.news.com.au/technology/an-app-that-follows-sharks-in-realtime-amazes-beachgoers-and-scientists/news-story/7d9b842396cae64008a7002c6f956358`.

Orchard, M. T. & Bouman, C. A. (1991), 'Color Quantization of Images', **39**(12).

Parikh, M. (2013), 'Animal Detection Using Template Matching Algorithm', **1**(April), 26–32.

Queensland Government (2006), Report on Queensland Shark Safety Program, Technical report.

Recky, M. & Leberl, F. (2010), Windows Detection Using K-means in CIE-Lab Color Space, *in* 'Pattern Recognition (ICPR), 2010 20th International Conference', pp. 356–359.

RGB_Cube_Show_lowgamma_cutout_b @ upload.wikimedia.org (2016), `https://upload.wikimedia.org/wikipedia/commons/8/83/RGB{_}Cube{_}Show{_}lowgamma{_}cutout{_}b.png`.

Robbins, W. D., Peddemors, V. M., Kennelly, S. J. & Ives, M. C. (2014), 'Experimental Evaluation of Shark Detection Rates by Aerial Observers', *PLoS ONE* **9**(2).

Sage, K., Young, S. & Hill, S. (1998), Computer vision for security applications, *in* 'Security Technology, 1998. Proceedings., 32nd Annual 1998 International Carnahan Conference on', pp. 210–215.

Sathish, B. S. (2014), HSV Color Space Based Segmentation of Region of Interest in Satellite Images, *in* 'Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2014 International Conference', pp. 101–105.

Sealifetrust (2015), 'Shark nets in Australia what are they and how do they work?', `http://www.sealifetrust.org.au/news/latest/shark-nets-australia-work`. [Online; accessed 2016-05-11].

Shah, M. (1997), 'Funadementals of Computer Vision'.

Shah, M. (2012), 'CAP 5415 Computer Vision Fall 2012 Edge Detection'.

Sugitha, T. S. & T, G. J. K. (2014), Optical Flow Based On Feature Match and Super Pixel Segmentation, *in* 'Advances in Computing and Communications (ICACC), 2014 Fourth International Conference', pp. 243–246.

Sumpton, W. D., Taylor, S. M., Gribble, N. A., Mcpherson, G. & Ham, T. (2011), 'Gear selectivity of large-mesh nets and drumlines used to catch sharks in the Queensland Shark Control Program', **33**(1), 37–43.

Szeliski, R. (2010), *Computer Vision : Algorithms and Applications*.

Taronga (2016), 'Latest Figures', `https://taronga.org.au/animals-conservation/conservation-science/australian-shark-attack-file/latest-figures`. [Online; accessed 2016-05-05].

Tchn, M., Kfnqhsglr, K., Mc, M., Shnmr, O., Marciniak, T., Dąbrowski, A., Chmielewska, A., Nowakowski, M., Wkh, H., Ri, Q., Lq, S., Uhvhdufk, F. & Uhvxowlqj, D. Q. G. (2012), MS @ U RO @ Real-Time Bi-Directional People Counting Using Video Blob Analysis, *in* '2012 Joint Conference New Trends In Audio & Video And Signal Processing: Algorithms, Architectures, Arrangements And Applications (NTAV/SPA)', pp. 161–166.

Telagarapu, P. (2012), A Novel Traffic-Tracking System Using Morphological and Blob Analysis, *in* '2012 International Conference on Computing, Communication and Applications', pp. 1–4.

Telagarapu, P. & Suresh, G. (2012), A novel traffic-tracking system using morphological and Blob analysis, *in* '2012 International Conference on Computing, Communication and Applications1-4'.

Wales, S., Africa, N. S. & Dudley, S. F. J. (1997), 'A comparison of the shark control programs of New South Wales and Queensland (Australia) and KwaZulu- Natal (South Africa) S. F. J. Dudley', **34**(1), 1–27.

Xu, L. (2011), A new method for license plate detection based on color and edge information of Lab space, *in* 'Multimedia and Signal Processing (CMSP), 2011 International Conference 99-102'.

Yilmaz, A. & Javed, O. (2006), 'Object Tracking : A Survey', **38**(4).

Youtube (2015*a*), 'Testing Video 1', `https://www.youtube.com/watch?v=q2U3gjwJfS4`. [Online; accessed 2016-04-23].

Youtube (2015*b*), 'Testing Video 2', `https://www.youtube.com/watch?v=Yz-rgpWqj10`. [Online; accessed 2016-05-23].

Youtube (2016*a*), 'Screenshot from drone footage', `https://www.youtube.com/watch?v=q2U3gjwJfS4`. [Online; accessed 2016-05-22].

Youtube (2016*b*), 'Screenshot from drone footage', `https://www.youtube.com/watch?v=0kz0Ai{_}DHgg`. [Online; accessed 2016-05-22].

Youtube (2016*c*), 'Testing Video 3', `https://www.youtube.com/watch?v=613RAAt8ZRY`. [Online; accessed 2016-09-26].

Yu, C., Dian-ren, C., Yang, L. & Lei, C. (2010), Otsu ' s Thresholding Method Based on Gray Level-Gradient Two-Dimensional Histogram, *in* 'Informatics in Control, Automation and Robotics (CAR), 2010 2nd International Asia Conference', pp. 282–285.

Zavadil, J. & Tuma, J. (2012), Traffic signs detection using blob analysis and pattern recognition, *in* 'Carpathian Control Conference (ICCC), 2012 13th International Conference', pp. 776–779.

Zhang, B. (2010), Computer Vision vs . Human Vision 1, *in* 'Cognitive Informatics (ICCI), 2010 9th IEEE International Conference', number 2004, pp. 3–3.

Zhou, H., Llewellyn, L., Wei, L., Creighton, D. & Nahavandi, S. (2015), Marine object detection using background modelling and blob analysis, *in* 'Systems, Man, and Cybernetics (SMC), 2015 IEEE International Conference', pp. 430–435.

Appendices

Appendix A

Matlab Source Files

A.1 Display Pixel Densities

```
1   \begin{lstlisting}
2   Shark = ...
        imread('/Users/kanebyles/Documents/MATLAB/Youtubeimages1/frame_0200.jpg');
3
4   Red = Shark(:,:,1);
5   Green = Shark(:,:,2);
6   Blue = Shark(:,:,3);
7
8   Sharkhsv = rgb2hsv(Shark);
9
10  Hue = Sharkhsv(:,:,1);
11  Sat = Sharkhsv(:,:,2);
12  Val = Sharkhsv(:,:,3);
13
14  figure(1)
15  subplot(3,2,1)
16  h1 = surf(Red);
17  set(h1,'LineStyle','none');
18  caxis([0,256])
19  colorbar
20  colormap hsv
21  view(0,0)
22  title('Red Pixel Densities, Side View')
23  subplot(3,2,2)
```

```
24      h2 = surf(Red);
25      set(h2,'LineStyle','none');
26      caxis([0,256])
27      colorbar
28      colormap hsv
29      view(0,90)
30      title('Red Pixel Densities, Aerial View')
31      subplot(3,2,3)
32      h3 = surf(Green);
33      set(h3,'LineStyle','none');
34      caxis([0,256])
35      colorbar
36      colormap hsv
37      view(0,0)
38      title('Green Pixel Densities, Side View')
39      subplot(3,2,4)
40      h4 = surf(Green);
41      set(h4,'LineStyle','none');
42      caxis([0,256])
43      colorbar
44      colormap hsv
45      view(0,90)
46      title('Green Pixel Densities, Aerial View')
47      subplot(3,2,5)
48      h5 = surf(Blue);
49      set(h5,'LineStyle','none');
50      caxis([0,256])
51      colorbar
52      colormap hsv
53      view(0,0)
54      title('Blue Pixel Densities, Side View')
55      subplot(3,2,6)
56      h6 = surf(Blue);
57      set(h6,'LineStyle','none');
58      caxis([0,256])
59      colorbar
60      colormap hsv
61      view(0,90)
62      title('Blue Pixel Densities, Aerial View')
63
64      figure(2)
65      subplot(3,2,1)
66      h10 = surf(Hue);
```

```
67      set(h10,'LineStyle','none');
68      caxis([0,1])
69      colorbar
70      colormap hsv
71      view(0,0)
72      title('Hue Pixel Densities, Side View')
73      subplot(3,2,2)
74      h20 = surf(Hue);
75      set(h20,'LineStyle','none');
76      caxis([0,1])
77      colorbar
78      colormap hsv
79      view(0,90)
80      title('Hue Pixel Densities, Aerial View')
81      subplot(3,2,3)
82      h30 = surf(Sat);
83      set(h30,'LineStyle','none');
84      caxis([0,1])
85      colorbar
86      colormap hsv
87      view(0,0)
88      title('Saturation Pixel Densities, Side View')
89      subplot(3,2,4)
90      h40 = surf(Sat);
91      set(h40,'LineStyle','none');
92      caxis([0,1])
93      colorbar
94      colormap hsv
95      view(0,90)
96      title('Saturation Pixel Densities, Aerial View')
97      subplot(3,2,5)
98      h50 = surf(Val);
99      set(h50,'LineStyle','none');
100     caxis([0,1])
101     colorbar
102     colormap hsv
103     view(0,0)
104     title('Value Pixel Densities, Side View')
105     subplot(3,2,6)
106     h60 = surf(Val);
107     set(h60,'LineStyle','none');
108     caxis([0,1])
109     colorbar
```

```
110     colormap hsv
111     view(0,90)
112     title('Value Pixel Densities, Aerial View')
```

A.2 Compare Test Images Pixel Densities

```
Shark1 = ...
    imread('/Users/kanebyles/Documents/MATLAB/Youtubeimages2/frame_1982.jpg');
Shark2 = imread('/Users/kanebyles/Documents/MATLAB/shark3.jpg');
Shark3 = ...
    imread('/Users/kanebyles/Documents/MATLAB/Youtubeimages3/frame_0010.jpg');
Shark4 = imread('/Users/kanebyles/Documents/MATLAB/shark2.jpg');
Shark1 = imcrop(Shark1, [10 10 (size(Shark1(:,:,3),2))-21 ...
    (size(Shark1(:,:,3),1))-21]);
Shark2 = imcrop(Shark2, [10 10 (size(Shark2(:,:,3),2))-21 ...
    (size(Shark2(:,:,3),1))-21]);
Shark3 = imcrop(Shark3, [10 10 (size(Shark3(:,:,3),2))-21 ...
    (size(Shark3(:,:,3),1))-21]);
Shark4 = imcrop(Shark4, [10 10 (size(Shark4(:,:,3),2))-21 ...
    (size(Shark4(:,:,3),1))-21]);
Red1 = Shark1(:,:,1);
Green1 = Shark1(:,:,2);
Sharkhsv1 = rgb2hsv(Shark1);
Val1 = Sharkhsv1(:,:,3);
Red2 = Shark2(:,:,1);
Green2 = Shark2(:,:,2);
Sharkhsv2 = rgb2hsv(Shark2);
Val2 = Sharkhsv2(:,:,3);
Red3 = Shark3(:,:,1);
Green3 = Shark3(:,:,2);
Sharkhsv3 = rgb2hsv(Shark3);
Val3 = Sharkhsv3(:,:,3);
Red4 = Shark4(:,:,1);
Green4 = Shark4(:,:,2);
Sharkhsv4 = rgb2hsv(Shark4);
Val4 = Sharkhsv4(:,:,3);

figure(1)
subplot(3,2,1)
h1 = surf(Red1);
set(h1,'LineStyle','none');
caxis([0,256])
colorbar
colormap hsv
view(0,0)
```

```
34 title('Red Pixel Densities, Side View')
35 subplot(3,2,2)
36 h2 = surf(Red1);
37 set(h2,'LineStyle','none');
38 caxis([0,256])
39 colorbar
40 colormap hsv
41 view(0,-90)
42 title('Red Pixel Densities, Aerial View')
43 subplot(3,2,3)
44 h3 = surf(Green1);
45 set(h3,'LineStyle','none');
46 caxis([0,256])
47 colorbar
48 colormap hsv
49 view(0,0)
50 title('Green Pixel Densities, Side View')
51 subplot(3,2,4)
52 h4 = surf(Green1);
53 set(h4,'LineStyle','none');
54 caxis([0,256])
55 colorbar
56 colormap hsv
57 view(0,-90)
58 title('Green Pixel Densities, Aerial View')
59 subplot(3,2,5)
60 h5 = surf(Val1);
61 set(h5,'LineStyle','none');
62 caxis([0,1])
63 colorbar
64 colormap hsv
65 view(0,0)
66 title('Value Pixel Densities, Side View')
67 subplot(3,2,6)
68 h6 = surf(Val1);
69 set(h6,'LineStyle','none');
70 caxis([0,1])
71 colorbar
72 colormap hsv
73 view(0,-90)
74 title('Value Pixel Densities, Aerial View')
75
76 figure(2)
```

```
77  subplot(3,2,1)
78  h1 = surf(Red2);
79  set(h1,'LineStyle','none');
80  caxis([0,256])
81  colorbar
82  colormap hsv
83  view(0,0)
84  title('Red Pixel Densities, Side View')
85  subplot(3,2,2)
86  h2 = surf(Red2);
87  set(h2,'LineStyle','none');
88  caxis([0,256])
89  colorbar
90  colormap hsv
91  view(0,-90)
92  title('Red Pixel Densities, Aerial View')
93  subplot(3,2,3)
94  h3 = surf(Green2);
95  set(h3,'LineStyle','none');
96  caxis([0,256])
97  colorbar
98  colormap hsv
99  view(0,0)
100 title('Green Pixel Densities, Side View')
101 subplot(3,2,4)
102 h4 = surf(Green2);
103 set(h4,'LineStyle','none');
104 caxis([0,256])
105 colorbar
106 colormap hsv
107 view(0,-90)
108 title('Green Pixel Densities, Aerial View')
109 subplot(3,2,5)
110 h5 = surf(Val2);
111 set(h5,'LineStyle','none');
112 caxis([0,1])
113 colorbar
114 colormap hsv
115 view(0,0)
116 title('Value Pixel Densities, Side View')
117 subplot(3,2,6)
118 h6 = surf(Val2);
119 set(h6,'LineStyle','none');
```

```
120 caxis([0,1])
121 colorbar
122 colormap hsv
123 view(0,-90)
124 title('Value Pixel Densities, Aerial View')
125
126 figure(3)
127 subplot(3,2,1)
128 h1 = surf(Red3);
129 set(h1,'LineStyle','none');
130 caxis([0,256])
131 colorbar
132 colormap hsv
133 view(0,0)
134 title('Red Pixel Densities, Side View')
135 subplot(3,2,2)
136 h2 = surf(Red3);
137 set(h2,'LineStyle','none');
138 caxis([0,256])
139 colorbar
140 colormap hsv
141 view(0,-90)
142 title('Red Pixel Densities, Aerial View')
143 subplot(3,2,3)
144 h3 = surf(Green3);
145 set(h3,'LineStyle','none');
146 caxis([0,256])
147 colorbar
148 colormap hsv
149 view(0,0)
150 title('Green Pixel Densities, Side View')
151 subplot(3,2,4)
152 h4 = surf(Green3);
153 set(h4,'LineStyle','none');
154 caxis([0,256])
155 colorbar
156 colormap hsv
157 view(0,-90)
158 title('Green Pixel Densities, Aerial View')
159 subplot(3,2,5)
160 h5 = surf(Val3);
161 set(h5,'LineStyle','none');
162 caxis([0,1])
```

```
colorbar
colormap hsv
view(0,0)
title('Value Pixel Densities, Side View')
subplot(3,2,6)
h6 = surf(Val3);
set(h6,'LineStyle','none');
caxis([0,1])
colorbar
colormap hsv
view(0,-90)
title('Value Pixel Densities, Aerial View')

figure(4)
subplot(3,2,1)
h1 = surf(Red4);
set(h1,'LineStyle','none');
caxis([0,256])
colorbar
colormap hsv
view(0,0)
title('Red Pixel Densities, Side View')
subplot(3,2,2)
h2 = surf(Red4);
set(h2,'LineStyle','none');
caxis([0,256])
colorbar
colormap hsv
view(0,-90)
title('Red Pixel Densities, Aerial View')
subplot(3,2,3)
h3 = surf(Green4);
set(h3,'LineStyle','none');
caxis([0,256])
colorbar
colormap hsv
view(0,0)
title('Green Pixel Densities, Side View')
subplot(3,2,4)
h4 = surf(Green4);
set(h4,'LineStyle','none');
caxis([0,256])
colorbar
```

```
206 colormap hsv
207 view(0,-90)
208 title('Green Pixel Densities, Aerial View')
209 subplot(3,2,5)
210 h5 = surf(Val4);
211 set(h5,'LineStyle','none');
212 caxis([0,1])
213 colorbar
214 colormap hsv
215 view(0,0)
216 title('Value Pixel Densities, Side View')
217 subplot(3,2,6)
218 h6 = surf(Val4);
219 set(h6,'LineStyle','none');
220 caxis([0,1])
221 colorbar
222 colormap hsv
223 view(0,-90)
224 title('Value Pixel Densities, Aerial View')
225
226 figure(5)
227 subplot(2,2,1)
228 imshow(Shark1)
229 title('Test Image 1')
230 subplot(2,2,2)
231 imshow(Shark2)
232 title('Test Image 2')
233 subplot(2,2,3)
234 imshow(Shark3)
235 title('Test Image 3')
236 subplot(2,2,4)
237 imshow(Shark4)
238 title('Test Image 4')
```

A.3 Show Blob Outline on Original Image

```
1   clear all
2   close all
3   clc
4
5   Shark1 = ...
        imread('/Users/kanebyles/Documents/MATLAB/Youtubeimages2/frame_1982.jpg');
6   Shark2 = imread('/Users/kanebyles/Documents/MATLAB/shark3.jpg');
7   Shark3 = ...
        imread('/Users/kanebyles/Documents/MATLAB/Youtubeimages3/frame_0010.jpg');
8   Shark4 = imread('/Users/kanebyles/Documents/MATLAB/shark2.jpg');
9   Shark1 = imcrop(Shark1, [10 10 (size(Shark1(:,:,3),2))-21 ...
        (size(Shark1(:,:,3),1))-21]);
10  Shark2 = imcrop(Shark2, [10 10 (size(Shark2(:,:,3),2))-21 ...
        (size(Shark2(:,:,3),1))-21]);
11  Shark3 = imcrop(Shark3, [10 10 (size(Shark3(:,:,3),2))-21 ...
        (size(Shark3(:,:,3),1))-21]);
12  Shark4 = imcrop(Shark4, [10 10 (size(Shark4(:,:,3),2))-21 ...
        (size(Shark4(:,:,3),1))-21]);
13
14  Sharkhsv1 = rgb2hsv(Shark1);
15  Val1 = Sharkhsv1(:,:,3);
16
17  Sharkhsv2 = rgb2hsv(Shark2);
18  Val2 = Sharkhsv2(:,:,3);
19
20  Sharkhsv3 = rgb2hsv(Shark3);
21  Val3 = Sharkhsv3(:,:,3);
22
23  Sharkhsv4 = rgb2hsv(Shark4);
24  Val4 = Sharkhsv4(:,:,3);
25
26  for q = 1:4
27  if q==1
28  B = Val1;
29  Shark = Shark1;
30  elseif q==2
31  B = Val2;
32  Shark = Shark2;
33  elseif q==3
```

```
34      B = Val3;
35      Shark = Shark3;
36      elseif q==4
37      B = Val4;
38      Shark = Shark4;
39      end
40      D = zeros((size(B,1)),(size(B,2)));
41      D1 = zeros((size(B,1)),(size(B,2)));
42      for i = 1:size(B,1);
43      for j = 1:size(B,2);
44      D1(i,j) = B(i,j) * 256;
45      end
46      end
47
48
49      D = uint8(D);
50      D1 = uint8(D1);
51
52      t = imhist(D1);
53      der = zeros(length(t),1);
54      for e = 1:length(t)
55      der(e) = t(e)-t(e+1);
56      if e == 255;
57      break;
58      end
59      if der(e) < ((size(B,1)*size(B,2))/-1500);
60      break;
61      end
62      end
63
64      for i = 1:size(B,1);
65      for j = 1:size(B,2);
66      if B(i,j) > ((e/256)*1) ;
67      B(i,j) = 0;
68      end
69      end
70      end
71
72      for i = 1:size(B,1);
73      for j = 1:size(B,2);
74      D(i,j) = B(i,j) * 256;
75      end
76      end
```

```
77
78      se = strel('diamond',2);
79      BW = imbinarize(D);
80      BW = imdilate(BW,se);
81      BW = bwareaopen(BW,60,4);
82      BW2 = imclearborder(BW,4);
83      BW3 = imfill(BW2,'holes');
84      BW3 = bwperim(BW3);
85      BW3 = imdilate(BW3,se);
86
87
88      if q==1
89      Shark1(BW3) = 255;
90      elseif q==2
91      Shark2(BW3) = 255;
92      elseif q==3
93      Shark3(BW3) = 255;
94      elseif q==4
95      Shark4(BW3) = 255;
96      end
97      fprintf('Threshold Value = %0.3f\n',e/256)
98      end
99      figure(1)
100     subplot(2,2,1)
101     imshow(Shark1)
102     title('Test Image 1 with detected blob outlines')
103     subplot(2,2,2)
104     imshow(Shark2)
105     title('Test Image 2 with detected blob outlines')
106     subplot(2,2,3)
107     imshow(Shark3)
108     title('Test Image 3 with detected blob outlines')
109     subplot(2,2,4)
110     imshow(Shark4)
111     title('Test Image 4 with detected blob outlines')
```

A.4 Shark Detection Algorithm

```
1  %   ERP2016 - Automated Shark Detection using Computer Vision
2  %
3  %   Author: Kane Byles
4  %   Student Number: 0061009769
5  %
6  %   This program is used for post processing of aerial shark footage.
7  %   It loops through the video, frame by frame, and attempts to identify
8  %   all the sharks in the frame. Yellow bounding boxes are placed around
9  %   sharks, whilst red blob outlines are placed over all blobs detected
10 %   that meet the adaptive threshold conditions. This algorithm then
11 %   saves the processed data to a '.avi' format for reviewing and
12 %   analysing.
13
14 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
15 %%   Clear all workspace variables and close all figures
16 clear all
17 close all
18 clc
19 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
20 %%  Initalise Variables
21 OffFlag = 0;
22 On = 0;
23 OnFlag = 0;
24
25 for i = 1:5
26 Noise(i) = {zeros(700,1260)};
27
28 end
29 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
30 %% Initalise video object files
31 v = VideoWriter('Post_Processed.avi');
32 open(v)
33 source='/Users/kanebyles/Documents/MATLAB/Shark_footage.mp4';
34 vidobj=VideoReader(source);
35 frames=vidobj.Numberofframes;
36 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
37 %% Main Loop
38 for f=1:frames
39 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Read Current Frame, crop to remove border and convert to HSV
thisframe=read(vidobj,f);
fprintf('Frame %d\n',f);
Shark = imcrop(thisframe, [10 10 (size(thisframe(:,:,3),2))-21 ...
(size(thisframe(:,:,3),1))-21]);
Sharkhsv = rgb2hsv(Shark);
%*************************************************************%
% Extract Value channel, blur Value array and save data to histogram
Value = Sharkhsv(:,:,3);
Value1=Value;
ValueBlur = imgaussfilt(Value,4);
HistData = ValueBlur.*256;
HistData = uint8(HistData);
ValueHistogram = imhist(HistData);
%*************************************************************%
% Determine adaptive threshold limit for calculating bin derivative
BinDerivative = zeros(length(ValueHistogram),1);
for e = 1:length(ValueHistogram)
BinDerivative(e) = ValueHistogram(e)-ValueHistogram(e+1);
if e == 255;
break;
end
if BinDerivative(e) < ((size(ValueBlur,1)*size(ValueBlur,2))/-1500);
break;
end
end
%*************************************************************%
% Write threshold limit to array for 3D surface model
% Set all values of brightness array above limit to zero
Limit = zeros(size(ValueBlur,1),size(ValueBlur,2));
Limit(:,:) = e/256;
ValueBlur(ValueBlur>(e/256))=0;
%*************************************************************%
% Scale values of brightness array to 0-255
% Convert to binary map
% Perform morphological operations on binary map
ScaledValue = ValueBlur.*255;
se = strel('disk',1);
de = strel('disk',2);
BW = imbinarize(ScaledValue);
BW = bwareaopen(BW,round(((size(ValueBlur,1)*...
size(ValueBlur,2))/2100)),4);
BW = imclearborder(BW,4);
```

```
83 BW = imdilate(BW,se);
84 BW = imfill(BW,'holes');
85 BlobOutline = bwperim(BW);
86 BlobOutline = imdilate(BlobOutline,de);
87 %************************************************************%
88 % Extract blob properties
89 % Move index of noise filter back one
90 BlobProperties = regionprops(BW, 'BoundingBox', 'Area','Centroid',...
91 'MajorAxisLength','MinorAxisLength','Orientation','Perimeter');
92 maps = [];
93 field1 ='maps';
94 for c = 1:4
95 Noise(c) = Noise(c+1) ;
96 end
97 %************************************************************%
98 % Loop through detected blobs
99 % Brightness ratio filter
100 % Variance filter
101 % Elliptic ratio filter
102 % Set blob elements to zero if they don't pass threshold
103 % Print out values for filters
104 for j = 1:size(BlobProperties,1)
105 box = BlobProperties(j).BoundingBox;
106 EllipticRatio = (pi*BlobProperties(j).MajorAxisLength*...
107 BlobProperties(j).MinorAxisLength)/(4*BlobProperties(j).Area);
108 CropBB = imcrop(Value1,box);
109 BrightLimit = min(min(ValueBlur))+(max(max(ValueBlur))...
110 -min(min(ValueBlur)))*0.60;
111 [x,y] = find(CropBB>BrightLimit);
112 BrightnessRatio = size(x,1)/(size(CropBB,1)*size(CropBB,2));
113 if BrightnessRatio>0.01
114 BW(round(box(2)):round((box(2)+box(4))),round(box(1)):...
115 round((box(1)+box(3)))) =0;
116 end
117 W = imcrop(ValueBlur,box);
118 Wvector = reshape(W,[],1);
119 IQR = iqr(Wvector);
120 if IQR<0.02
121 BW(round(box(2)):round((box(2)+box(4))),round(box(1)):...
122 round((box(1)+box(3)))) =0;
123 end
124 if EllipticRatio > 1.27
125 BW(round(box(2)):round((box(2)+box(4))),round(box(1)):...
```

```
126 round((box(1)+box(3)))) =0;
127 end
128 fprintf('Elliptic Ratio is: %0.2f\n',EllipticRatio)
129 fprintf('Diff = %0.3f\n',IQR);
130 fprintf('Brightest = %0.4f\n\n',BrightnessRatio);
131 end
132 %************************************************************%
133 % Plot outline of detected blobs on original frame
134 Shark(BlobOutline) = 255;
135 figure(1)
136 imshow(Shark)
137 BlobProperties = regionprops(BW, 'Centroid','BoundingBox');
138 %************************************************************%
139 % Update noise filter memory
140 % Test whether blobs are random occurences (noise)
141 % Plot yellow bounding box labelling shark if noise filter passed
142 Noise(5) = {BW};
143 maps = struct(field1,Noise);
144 for i = 1:size(BlobProperties,1)
145 BlobCentroid = BlobProperties(i).Centroid;
146 BlobCentroid = round(BlobCentroid);
147 for g = 1:5
148 if BW(BlobCentroid(2),BlobCentroid(1)) == 1 &&...
149 maps(g).maps(BlobCentroid(2),BlobCentroid(1)) == 1
150 OnFlag = OnFlag+1;
151 OffFlag = OffFlag-1;
152 if OffFlag<0
153 OffFlag = 0;
154 end
155 else
156 OffFlag = OffFlag+1;
157 end
158 end
159 if OnFlag > 4
160 On = 1;
161 end
162 if OffFlag > 2
163 On = 0;
164 end
165 if On == 1
166 box = [BlobCentroid(1)-50 BlobCentroid(2)-50 100 100];
167 rectangle('Position', box,'Curvature',0.2, 'EdgeColor', 'y');
168 text(box(1),box(2)-10,'SHARK','FontSize',16,'BackgroundColor','y');
```

```
OnFlag = 0;
OffFlag = 0;
end
end
%****************************************************************%
% Get current frame and write to video
% Loop back to start of main loop to get new frame
F = getframe;
writeVideo(v,F);
end
%****************************************************************%
% Close video file
close(v)
```

Appendix B

Project Specification

ENG 4111/2 Research Project

Project Specification

For:	**Kane Byles**
Title:	Automated Shark Detection using Computer Vision
Major:	Mechatronic Engineering
Supervisors:	Dr Tobias Low
Enrolment:	ENG4111 - EXT S1, 2016
	ENG4112 - EXT S2, 2016
Project Aim:	To develop a computer vision algorithm that is capable of automatically detecting sharks from aerial video footage

Program:

1. Research the history of computer vision and understand the fundamental principles.
2. Research how computers can be taught to "see" and define objects.
3. Obtain data useful for constructing a shark detection algorithm.
4. Develop a shark detection algorithm based on the available data.
5. Develop filtering techniques to reject false positives
6. Test the algorithm on aerial shark footage and analyse successful and unsuccessful detections

Agreed:

Student Name:	Kane Byles
Date:	06/03/2016
Supervisor Name:	Dr Tobias Low
Date:	30/03/2016