DEEP VISION: FISH MONITORING

BY

LING YI JUN

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONS)

Faculty of Information and Communication Technology

(Kampar Campus)

AUGUST 2018

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: ____

____

____

Academic Session: ___________

I ____

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,

_________________________	_________________________
(Author's signature)	(Supervisor's signature)
Address:	
_________________________	_________________________
_________________________	Supervisor's name
_________________________	
Date: ____________________	**Date**:____________________

DEEP VISION: FISH MONITORING

BY

LING YI JUN

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONS)

Faculty of Information and Communication Technology

(Kampar Campus)

AUGUST 2018

DECLARATION OF ORIGINALITY

I declare that this report entitled "**DEEP VISION: FISH MONITORING**" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature	:	________________________
Name	:	________________________
Date	:	________________________

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisor, Dr. Lau Phooi Yee for giving me this opportunity to engage in this wonderful project, also providing guidance whenever necessary; thanks to her I was able to complete this project and have learnt a lot of thing.

Next, I would like to say thanks to all the people who have given me support in any away especially my family members and my best friend. Their support and encouragement has helped me throughout my course. They have given me suggestions; comfort and inspiration whenever I feel stuck or lost.

ABSTRACT

Aquaculture farm provides a solution for the overfishing issue. Maintaining big scale farm requires going through hours of video footages for meaningful information, a demanding task for human operators affected by biological restrictions. Therefore, using image processing techniques, meaningful information can be extracted from the videos automatically, assisting in the fish monitoring processes. This paper proposes a non-invasive method of counting fishes and estimating their sizes. The method proposed by this paper includes four main parts, to compensate the uneven illumination of input image, separating objects from background, fish detection, and lastly, decision making. Inconsistent lighting in the background is an issue for computer vision, therefore, input frame is needed to go through illumination compensation. An xy Gaussian technique is used to create a mask, which is later subtracted from the original footage. Objects are then being extracted out from the background, which is done with thresholding, and morphological operations. External contours are then extracted. Visual features are used to differentiate non-fish objects and fishes. Five visual features are used, namely: F1) Elongated Structure, F2) Downward Curve of Fish, F3) Upper Convexity Angle, F4) Downward Curve of Fish, F5) Differenced Pixel Value of a fish region with the background. A more detailed explanation would be further elaborated within the report. Later on, decision making module proceeds, a counter keeps count of the number of visual features a detected object match, when it reaches a certain threshold, the object is considered a fish and is counted and sized. This paper also includes discussion and experimental validation for the proposed method.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

OpenCV	Open Source ComputerVision
BGR	Blue, Green, Red
LoG	Laplacian of Gaussian

CHAPTER 1: INTRODUCTION

1.1Problem Statement

It is undeniable that fishes play an important role in the global food supply. According to an article,(Chu, W 2017) a group of international scientists have found out the several fish species that are at risk due to high demands of marine foods. The high population demands for fish proteins have led to an overfishing phenomenon which is not a sustainable way to satisfy consumer demands (Chu, W 2017). For example, the Atlantic Bluefin Tuna, which was once a species that can be found in abundance, is now considered a species that is at the risk of extinction (Kale, A n.d.).

Therefore, aquaculture, the practice of fish farming, is a more sustainable way to provide a continuous source of food and nutritional supplies (Rudolph, C n.d). Monitoring growth and the condition of fishes in an aquaculture environment is quite important since it gives information as to when the fish is ready to be harvested, when to transfer the fish to another environment, what grade a batch of fish is categorized into and also gives fish farmers insight of the health of their fishes. Before computer vision was introduced, surveillance camera footage of the underwater aquaculture farm is analysed using manual labour done by trained technicians; however, this method can be sometimes rather taxing and prone to human error, such as fatigue and counting bias. Since there are hours of footages that are needed to be analysed to extract meaningful information, which is a both physically and mentally demanding task. Therefore, developing an automated underwater imaging system to help track fish size and condition is a way to combat those issues. With minimal human intervention, machines can help overcome biological limitations faced by people.

1.2 Background and Motivation

With the global overfishing phenomenon of marine fisheries due to the ever increasing consumer demands for fishes, natural offshore fishing resources are increasingly depleted (Chow, L 2016). It is no longer sustainable to depend solely on natural offshore fishing resources to supply for the demands. Therefore, while great efforts have been made for the development of deep sea fisheries, aquaculture, also known as fish farms, is developing rapidly to help satisfy global demands for a more sustainable food supply. Fish farms involve raising fish in large quantities commercially in enclosures, mainly for the purpose of food (ScienceDaily n.d.). With a large amount of fishes to handle, Monitoring growth and the condition of fishes in an aquaculture environment is quite important since it gives information as to when the fish is ready to be harvested and when to transfer the fish to another environment.The traditional method of fish monitoring makes use of experienced divers to manually do the counting and information collecting. However, obviously, this method are subjected to many limitations, among them are the indeterminate waiting to obtained the service of a trained diver due to the shortage of them, possibility of human error occurring caused by diver fatigue and biological limitations, not only that, manual monitoring can be intrusive for the fishes as well. As technology improves and the cost of equipment being more accessible, underwater video imaging has become a popularly adopted technique to monitor underwater species without being intrusive, thus solving the intrusive divers issue.

Underwater video captured can act as a data source which has the ability to provide information about the condition of the fishes without having to physically diving in the fish enclosure. Until now, deriving video content was mostly done through manual observation, which is having a human watching the whole length of the video and recording down the findings (Varah, Sean 2012). Obviously, this method is not as efficient as it could be, since it is still manually done by a human being who is limited by many biological causes.

Previous industrial revolutions have freed mankind from relying solely on animal power, and enabled mass production possible. Now, comes forth the Fourth Industrial evolution, whose resulting shifts and interference can have an impact which would give birth to a dramatically improved efficiency of organizations and management of assets, and the impact of such changes can help with the regeneration of the natural environment (Shwab, K 2016), undoing the detrimental effects of previous industrial revolutions, in this case, help restore offshore environment which have been previously damaged by overfishing.

Fourth Industrial revolution have pressured a more advanced and automated way of fish farm management, in order to prevent being obsolete, and also to overcome problems such as the limitation of manual labour, it is an eventual occurrence that a more automated fish monitoring system be pushed forwards. In this case, computer vision needs to be implemented to extract meaning and data out from video automatically, thus replacing the need of human beings to manually observe long and lengthy video footages to monitor condition and number of the fishes within the fish farm enclosure.

Computer vision is the science that enables machine or computer to see visually the world around them much like how human utilize their eyes for such purpose. Computer vision is relevant with automated extraction, analysis and understanding of related information from either video footages, or a single image. Theoretical and algorithm basis development are needed to bring about automatic visual data extraction (BMVA n.d.). In the late 1960s, computer vision started out in universities that were developing artificial intelligence. It was meant to simulate the human visual system, as a starting step to provide robots with intelligent behaviour (Szeliski, R 2010). Studies in the 1970s formed the early foundations for many of the computer vision algorithms that exist today, including extraction of edges from images, this is done by comparing pixel by pixel the colour difference to determine whether a certain point in the image are considered edges(Szeliski, R 2010).

To extract out information from video, systems implementing computer vision methods would have to go through some usually taken steps, which is also taken by this project. The first step is image acquisition. A video would have to be broken down into frames with every single frame being transmitted to a local or remote PC, which the sequences of video frame is stored for further analysis (Xiao et al. 2011). The second step would be video pre-processing, in which the main goal of this step is to reduce noise present in the image as to not introduce false information. The next step is feature extraction, notable image feature are extracted from the image (Davies, ER 2005). Typical examples of such features are lines, edges and localized interest points such as blobs (Davies, ER 2005). A simple explanation of a blob can be said to be a group of connected pixels in an image that share common property, such as the gray scale value. For Figure 1.3 below, the dark spots are considered as blobs (Mallick, S 2015).

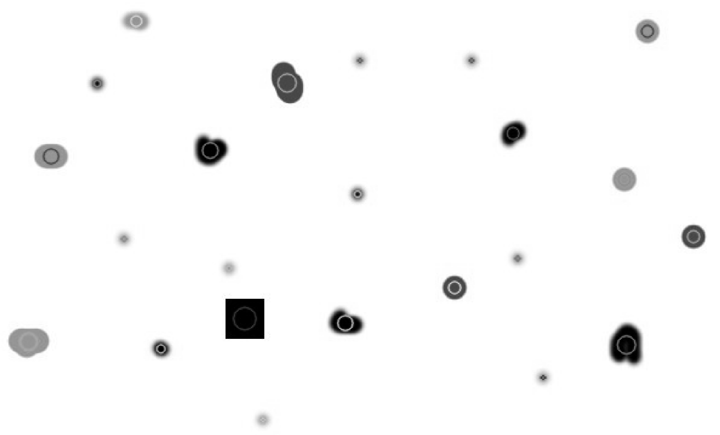


Figure 1.1: Sample of blobs

The second last step is detection/segmentation, where decisions in which image points of the image are relevant are made for further processing. The last step would be decision making, where final decisions are made. (Davies, ER 2005)

1.3 Project Objectives

The purpose of this project is to help fish farmers monitor their fishes more effectively. As mentioned earlier, employing manual divers to go underwater for fish inspection can be taxing and prone to error, therefore, an automated way to do that without the need of human labour can greatly ease the process of evaluating the fishes. With the automated underwater imaging system, fish farmers just have to analyze automatically generated outputs by the system, and make decisions based on the results. This process is fast and efficient and does not need any one to physically go underwater to observe the condition of the fishes.

The project aims to design a prototype system which can receive an input underwater video footage, process the video footage so it can be understood by a machine and extract out information from the video footages. The final deliverable is a prototype system that is able to detect fishes, output the number of fishes in that frame, and also output a roughly estimated area of multiple fishes in that frame. This system would be able to help fish farmers monitor the average amount of fishes and their sized within the sight of the underwater cameras.

The project will be focusing on image processing techniques, with the help of the OpenCV library. Ways to process input footage will be analysed and researched so that the input footage will eventually be able to be interpreted and understood by the computer to generate desired output results.

Hopefully in the future, the system will be further improved in different aspects. For example, the accuracy of the results can be improved, or even adding in new functionalities.

1.4 Proposed Approach

After acquiring the video footage, the first step is for the frame to go through illumination compensation due to the uneven illumination of underwater footages; the frames are blurred using a Gaussian kernel to get an estimated illumination gradient for the frame. Next, is to subtract the frame of interest with the estimated illumination gradient, to compensate for the uneven illumination, a range of Scalar threshold value is set where anything falls within the range will be scaled to black while anything outside the range would corresponds to the background, which would be scaled to black, creating a binary image. Noisy pixels and jagged edge removal is then done using morphological operations. Next, edge information is extracted to form contours, using a method proposed by (Suzuki and Abe, 1985), which connects a series of edge pixels to form a contour. Thus, outer contours are extracted. A few visual features are used to determine whether the detected contours can be considered a fish. A counter is used to keep track of how many features the foreground object matches, when it passes a certain threshold, the foreground object can be considered a fish and is included in the counting and sizing.

1.5 Report Organization

The following report will discuss further about the proposed approach for automated fish counting and sizing of underwater footages. The following chapter will discuss about existing work and also a review of computer vision techniques used in this proposed method. After the literature review chapters, would be the system design chapter, which would give an overview working of the system. A more detailed explanation of how each part of the system design works would be elaborated in a few chapter after the system review chapter, including the system's output evaluation. Finally, the last chapter would be to conclude the whole report, noting the achievements of the system and also personal insight to the total research experience, not to mention further developments and improvements that can be made for the system.

CHAPTER 2: LITERATURE REVIEW

There are some studies that have been made in this field, (Toh et al, 2009) have proposed a method to detect and count the fishes by initially obtaining blobs by using background subtraction, then go through noise and background object elimination and then counting the leftover blobs. Another method by (Fabic et al, 2013), is to detect fishes by using The Canny Edge Detector to detect fish contours, later filling up that contours to form blobs to allow blob counting. (Tan et al, 2014) proposed some techniques to detect marine species on underwater images. They differentiated lobsters and burrows by using visual features such as pixel intensity, structure and shape of contours extracted from the frames. There are also a few image processing techniques that are helpful in developing the proposed approach. Otsu's method (Otsu, 1979) creates a histogram of an image to find the optimum threshold value to convert an image into a binary image while preserving details of foreground objects. Contour detection algorithm proposed by Suzuki and Abe (1985) uses connected sequence of edge pixels to create contours. Efford, (2000) have explained the usage of a simple method called morphological closing while Poynton & Charles, (1998) explained gray scaling with colorimetric conversion.

2.1 Automated Fish Counting using Image Processing

In terms of which ways of implementing fish counting system is best, automated fish counting method proves to be a much superior alternative compared to the traditional way of manual fish counting. It is much more efficient, accurate and consistence since it is not bound by the limitations of the human body and mind. Manual fish counting with human operators are often time consuming, labour-intensive and prone to errors, human operators are susceptible to fatigue and other emotional factors, whereas, automated systems are not affected by those things. Therefore, the following method is proposed by Toh et al, (2009), describes how automated fish counting with underwater imaging is done.

Computer Vision is one of the good approaches to automated fish counting.

First of all, blobs need to be obtained. Frames from an underwater video are extracted and contrast is enhanced. Then the image undergo background subtraction, with a technique to acquire the background called background estimation, which close the frame morphologically with appropriate structuring element, leaving behind the estimated background.

After subtracting the background, the results go through a luminance threshold to be transformed into a pure black and white image. Pixels with luminance greater than a certain threshold are substituted with white pixels while the rest are substituted with black pixels, resulting in a pure black and white image.

For now, not all black pixels are considered a fish since there will be some noise. Therefore, the image must now go through noise and background elimination, by using an area threshold which is adjusted according to the expected sizes of the fish, anything below that threshold are considered noise and would be eliminated. Since there is a possibility that fishes intersect each other resulting in a bigger area, a threshold of five intersecting fishes are used, anything bigger than that area are considered background object and thus eliminated.

Figure 2.1 shows the transformation from a raw extracted frame to the resulting image after all the pre-processing the extracted frame has gone through.

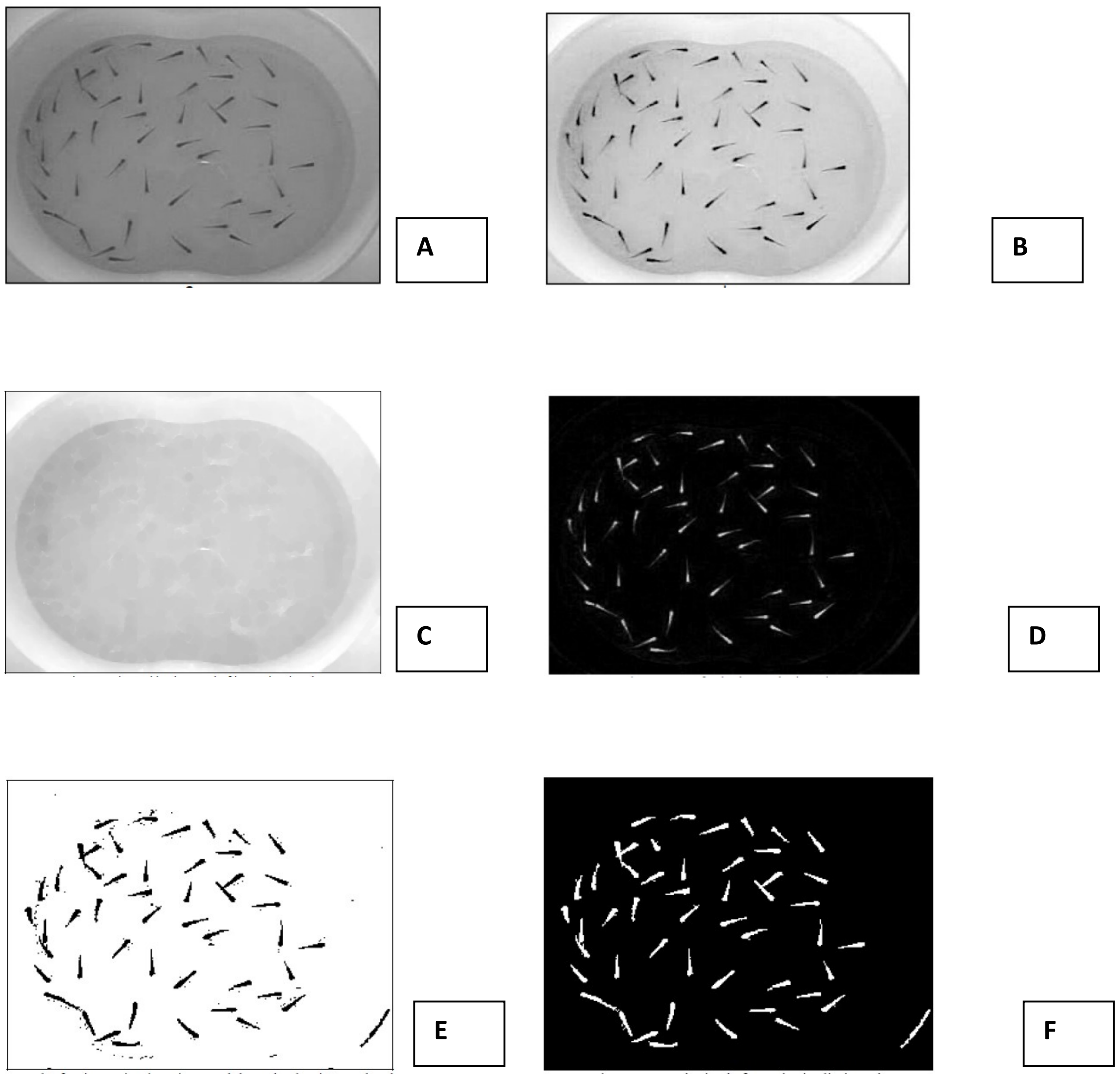


Figure 2.1 Image Preprocessing results
(Toh et al, 2009)

Image ‘A’ is the gray scaled image from the raw extracted frame; colour information is not needed due to the fish having higher intensity value compared to its surrounding. Image ‘B’ is after the image has gone through contrast enhancement. Image ‘C’ shows the estimated background. Image ‘D’ shows what the resulting image is like after subtracting the background. Image ‘E’ is a result after ‘D’ passes through a luminance threshold. And lastly, Image ‘F’ is what is obtained after noises are eliminated. Note that in Image ‘F’ lower right is part of the container that have not been eliminated due to imperfect background subtraction, thus, further manual adjustments are made to correct it.

Now what’s left of the blobs corresponds to only fishes. A standard area of fish is estimated by using all of the blob’s median area. Blob area which is lower than 140% median area is categorized as one fish. The reason to choose this percentage is so to fit in fishes that are slightly larger, yet small enough to have only little possibility to be intersecting fishes. Whereas remaining blobs are assumed to be intersecting fishes, thus the areas of these remaining blobs are divided by the blob median area to get number of fishes in each blob.

This method yields accurate results, although it has certain imperfections such as imperfect background subtraction that still has to be corrected manually. Also, background must be of good contrast relative to the fishes or else threshold-ing with luminance values would not work as smoothly. Moreover, background must also have relatively even illumination or else there would be some error when the frame passes the luminance threshold. Other than those mentioned shortcoming, this proposed method generates accurate results with an error percentage of not more than 6%.

2.2 Fish Population Estimation and Species Classification from Underwater Video Sequences using Blob Counting and Shape Analysis

Fish population estimation has always been an integral part for fish conservations. Steps can be taken to protect species which are facing endangerment or extinction by assessing fish abundance, distribution and diversity in marine environment (Fabic et al, 2013). This helps balance certain ecosystem and maintain nature's balance. Therefore fish population estimation is an important matter that needs to be addressed.

There have been many proposals that were made for assessing fish abundance, distribution and diversity in a more automated ways, this includes remotely operated vehicles, sonar containing towed equipment, resistivity counters and infrared beams (Fabic et al, 2013). However, live video imaging is considered advantageous due to its non-intrusive nature during data collection.

Lately, unattended video capture with embedded video cameras has been done by underwater observatories in mid-water and benthic habitats (Fabic et al, 2013). However, analyzing the collected video data manually is time consuming and tedious, thus if done by a human operator, concentration tends to dwindle and it is very prone to human error. Therefore, it is necessary to find an automated way to analyze the raw video data and extract out meaningful information needed for usage by related parties.

Fish movement, inconsistent illumination, camera motion etc proves to be challenges in designing a fish population estimating system. Luckily, advancement in technology has made it possible to overcome all these issues.

The method proposed here to detect fishes is an efficient way to undergo the fish count estimation process: Frames are extracted out from the Underwater Video Sequence (UWVS), goes through contour detection and later blob detection to detect where the fishes are (Fabic et al, 2013). For contour detection, The Canny Edge Detector is implemented in each frame to detect fish contours. Later on, the spaced between the contours are filled up to form blobs which enables blob counting.

According to Moeslund, (2009), the Process of Canny edge detection algorithm can be broken down to 5 different steps:

1. Apply Gaussian filter to smooth the image in order to remove the noise, since image noise can highly affect edge detection results. The point of Gaussian is to convolve with the image, which results in a somewhat smoother image.

   For Gaussian filter kernel size (2k+1)*(2k+1), the equation is:

$$H_{ij} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(i-(k+1))^2 + (j-(k+1))^2}{2\sigma^2} \right); 1 \le i, j \le (2k+1)$$

   An example of a 5x5 Gaussian filter (purpose is to generate the adjacent image):
   *=convolution operation
   $\sigma = 1.4.$

$$\mathbf{B} = \frac{1}{159} \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} * \mathbf{A}.$$

   Choosing Gaussian kernel size will affect detector performance, with a larger size lowering down detector’s noise sensitivity. Larger Gaussian filter kernel size would also increase slightly its localization error.

2. Find the intensity gradients of the image.

   There are varying directions of which an edge of an image can be pointed to, therefore four filters are used by the Canny algorithm to detect horizontal, vertical and two diagonal edges within a blurred image.

   Edge detection operator returns value for first derivative in:

   (G_x)= horizontal direction (G_y)= vertical direction

   With it edge gradient and direction may be derived with:

   $$\mathbf{G} = \sqrt{\mathbf{G}_x{}^2 + \mathbf{G}_y{}^2}$$

   $$\mathbf{\Theta} = \operatorname{atan2}(\mathbf{G}_y, \mathbf{G}_x)$$

   (G is computed with hypot function, while atan2 is arctangent function with two arguments)

   Angle of the edge direction is rounded off to either vertical (0°), horizontal (45°), and two diagonals (90°, 135°). For each colour region the edge direction falls, the edge direction will be set to fixed angle values. For example: θ in [0°, 22.5°] or [157.5°, 180°] maps to 0°

3. Apply non-maximum suppression to get rid of spurious response to edge detection.

   The edge extracted from gradient value by using gradient calculation is still rather blurred, therefore non-maximum suppression is needed for edge “thinning”, which helps suppress gradient values (setting them to zero) which are not the local maxima, this shows locations with sharpest change of intensity values. Every pixel of the gradient image has the algorithm which can be split into two parts. (1) Edge strength of current pixel and pixel of the positive and negative edge direction is compared. (2) If edge strength of current pixel is largest comparative to pixels in mask of same direction, value is maintained, else, value is suppressed.

4. Apply double threshold to determine potential edges

   After non-maximum suppression, there are still spurious responses, therefore it is important for pixels with weak gradient values to be filtered out and high gradient value pixels be maintained. High and low threshold values are selected to do this. For the value of edge pixel gradient higher than high threshold value, it is labelled as strong edge pixel. Else if value of edge pixel's gradient is bigger than low threshold value, yet smaller than high threshold value, it is labelled as weak edge pixel. Else, it will be suppressed.

5. Track edge by hysteresis: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.

Figure 2.2 shows a raw underwater video, while Figure 2.3 shows an underwater video after Canny Edge detection has been applied.

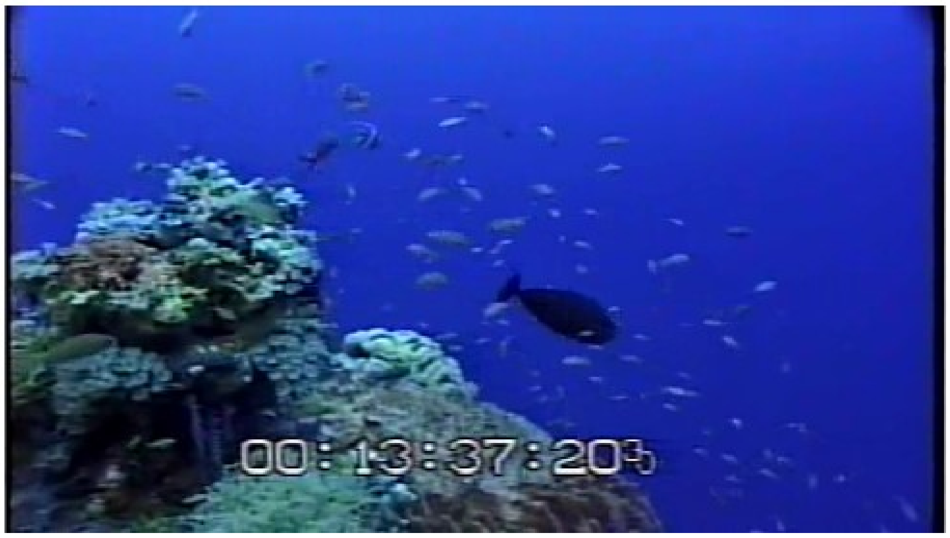


Figure 2.2
Raw Underwater Video
(Fabic et al, 2013)

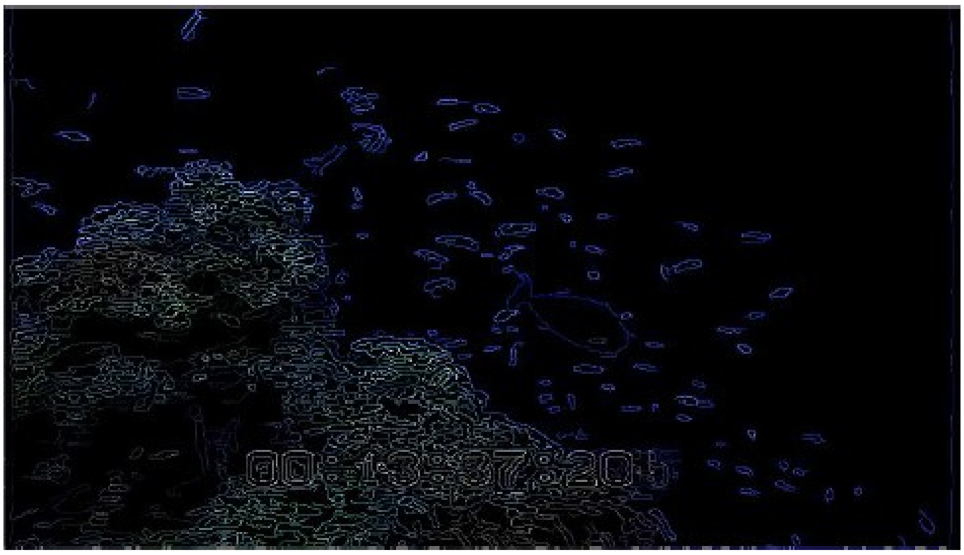


Figure 2.3
Underwater Video
with Canny
detection applied
(Fabic et al, 2013)

Next step after contours are detected is to fill them up to create blobs. Blob detector is based on the Laplacian of Gaussian (LoG), which can be described with the following equation:

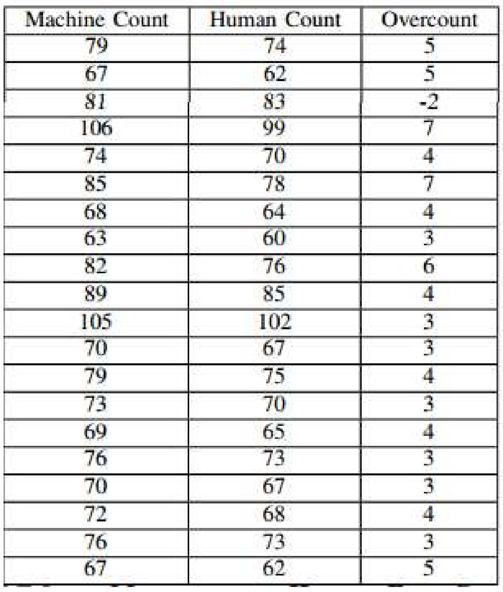
$$LoG \triangleq \triangle G_\sigma(x,y) = \frac{\partial^2}{\partial x^2}G_\sigma(x,y)+\frac{\partial^2}{\partial y^2}G_\sigma(x,y) = \frac{x^2+y^2-2\sigma^2}{\sigma^4}e^{-(x^2+y^2)/2\sigma^2}$$

, which convolves the image with a Gaussian kernel, which can be described with the following equation:

$$G_\sigma(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}}exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

Then, a Scale-normalized Laplacian operator is used to obtain multi-scale blob detector with automation scale selection. Results are later utilized to detect scale-space maxima or minima (Fabic et al, 2013).

However, there are still a small amount of inaccuracies not more than 7 over count present. The test results are shown in Figure 2.4.



Machine Count	Human Count	Overcount
79	74	5
67	62	5
81	83	-2
106	99	7
74	70	4
85	78	7
68	64	4
63	60	3
82	76	6
89	85	4
105	102	3
70	67	3
79	75	4
73	70	3
69	65	4
76	73	3
70	67	3
72	68	4
76	73	3
67	62	5

Figure 2.4 Machine and Human Fish Counts

2.3 Colorimetric (perceptual luminance-preserving) conversion to grayscale

Grayscaling is used extensively in image processing since colour information isn't useful for detecting important edges or other features, with exceptions of course.

According to Poynton and Charles (1998), colorimetry is common strategy to calculate grayscale values to have identical luminance as the source colour image.

According to Lindbloom(n.d), converting a colour from a colorspace using gamma compressed, which is considered non-linear, RGB colour model to its grayscale representation of its luminance, the RGB colourspace needs to be made linear by removing gamma compression function using gamma expansion (linearization), which can be defined by the equation below:

C_{srgb}= (R_{srgb}, G_{srgb}, and B_{srgb}, each in range [0,1])
C_{linear}= (R_{linear}, G_{linear}, and B_{linear}, each in range [0,1])

$$C_{\text{linear}} = \begin{cases} \frac{C_{\text{srgb}}}{12.92}, & C_{\text{srgb}} \le 0.04045 \\ \left(\frac{C_{\text{srgb}}+0.055}{1.055}\right)^{2.4}, & C_{\text{srgb}} > 0.04045 \end{cases}$$

Its purpose is to be able to apply suitable weighted sum to the linear colour components, $R_{\text{linear}}, G_{\text{linear}}, B_{\text{linear}}$, so that linear luminance Y_{linear} can be calculated, according to Stokes(n.d), the equation is as below.

$$Y_{\text{linear}} = 0.2126R_{\text{linear}} + 0.7152G_{\text{linear}} + 0.0722B_{\text{linear}}.$$

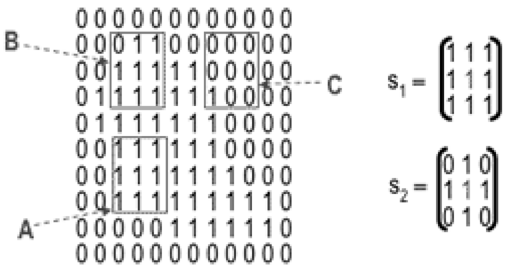
Green has greatest coefficient value due to human vision being most sensitive to green, while blue has the least value to human vision being least sensitive to blue. Then, values $Y_{\text{linear}}, Y_{\text{linear}}, Y_{\text{linear}}$ replaces $R_{\text{linear}}, G_{\text{linear}}, B_{\text{linear}}$ so that linear RGB are encoded in grayscale intensity. Then it needs to be gamma-compressed back to get back to its non-linear representation (Burger & Burge, 2010). Gamma-compressed is an inverse of gamma expansion with equation:

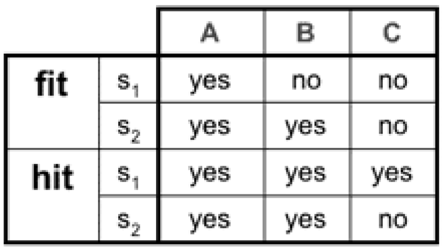
$$Y_{\text{srgb}} = \begin{cases} 12.92\, Y_{\text{linear}}, & Y_{\text{linear}} \le 0.0031308 \\ 1.055\, Y_{\text{linear}}^{1/2.4} - 0.055, & Y_{\text{linear}} > 0.0031308. \end{cases}$$

2.4 Morphological Closing

Efford,(2000) explained the basics of morphology operations.

Basically a template which is known as a structuring element, is positioned at every pixel of an image and it is then compared to all its possible neighboring pixels. Some of the comparison "fits" with its neighboring pixels while some "hits", also known as intersect. Figure 2.6 below shows an example of what is considered fits can what is considered hit.





		A	B	C
fit	s_1	yes	no	no
	s_2	yes	yes	no
hit	s_1	yes	yes	yes
	s_2	yes	yes	no

Figure 2.5: Fitting and hitting of a binary image with structuring elements s_1 and s_2.

Erosion can have the equation of: $g = f \ominus s$. With g denoting a new binary image, "f" is the original binary image, and "s" is the structuring element. For example, g(x,y) =1, means that "s" fits "f", if not, it is 0, this is repeated for every single pixel coordinate.

Dilation on the other hand has an equation of: $g = f \oplus s$. For example, g(x,y)=1 means that "s" hits ""'f, else if it doesn't hit it'll return a result of 0.

With that in mind, morphological closing has an equation of:

$$A \bullet B = (A \oplus B) \ominus B$$

A=binary image	$\oplus$ = dilation
B=structuring element	$\ominus$ =erosion

Morph close is useful for filling up holes to combine neighbouring region fragments.

2.5 Topological Structural Analysis of Digitized Binary Images by Border Following

Suzuki and Abe, (1985), have proposed an algorithm which would be used extensively now in the image processing field especially in contour extraction. They basically use connected sequence of edge pixels to create contours, the algorithm is explained below.

When the raster scan on the input binary image reaches the 1-pixel(which is circled in Figure 2.7 (a) below, it will check if it satisfies the condition to be the starting point of an outer border, if it is, a following number 2 is assigned to the border, since number 1 have been set aside to frame the picture. Then, it will start with the border following, changing border pixel value to 2 or -2 as shown in Figure 2.7 (b), when the entire border has been followed, raster scan continues. Circled pixel Figure 2.8(b) is where the next border following starts, and it satisfies condition shown in Figure 2.6 (b). Having a negative value means that the point cannot be the starting point for border following. The parent border for the number 3 border is an outer border of the number 2 border. When border following occurs, due to policy Figure 2.6 (b), it is shown that two pixel of value 1 have been changed to 3, which the other visited pixels stayed the same. These steps will go on and on until the final results are obtained.

This algorithm is useful for feature extraction of digital pictures.

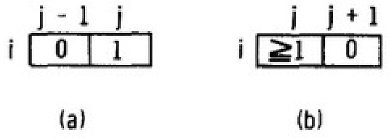


Figure 2.6 Conditions of border following, (a) for an outer border, (b) for a hole border

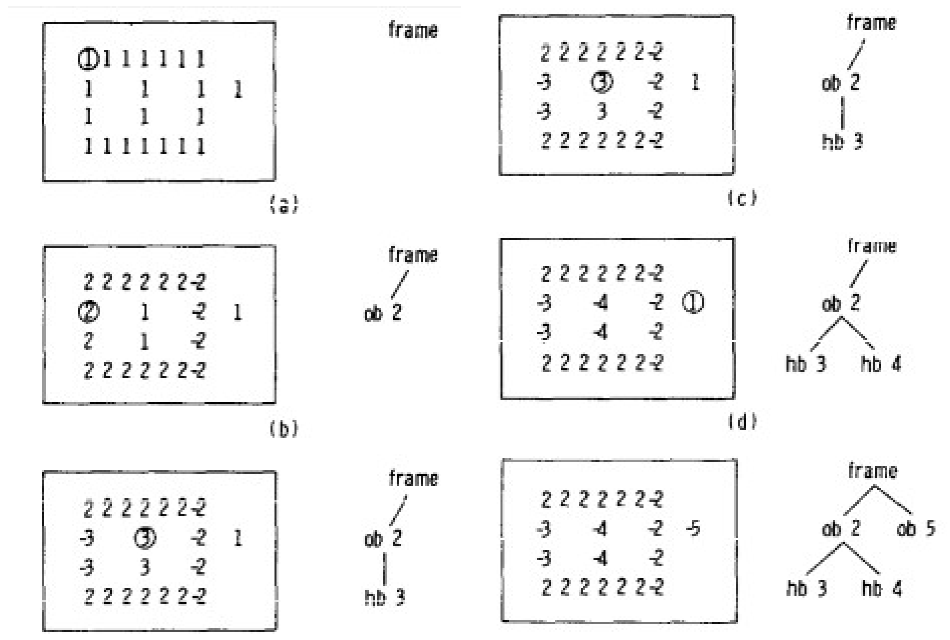
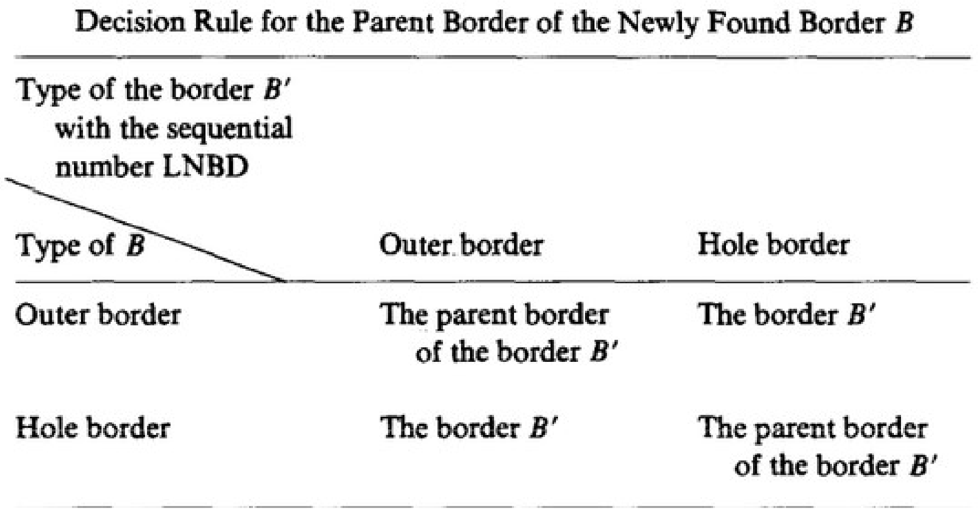


Figure 2.7 Illustration explaining border following process



Decision Rule for the Parent Border of the Newly Found Border *B*

Type of *B* \ Type of the border *B′* with the sequential number LNBD	Outer border	Hole border
Outer border	The parent border of the border *B′*	The border *B′*
Hole border	The border *B′*	The parent border of the border *B′*

Figure 2.8 Decision Rule for the Parent Border of Newly Found Border B

2.6 Otsu’s Method

Otsu’s method is used for performing clustering-based image thresholding automatically. Otsu’s algorithm basically starts with computing histogram and probabilities of every intensity level. Followed by setting up initial probabilities $\omega_i(0)$ and class means $\mu_i(0)$, go through all possible thresholds, update ω_i and μ_i, compute $\sigma^2_b(t)$, and the final desired threshold is the max $\sigma^2_b(t)$.

Weights ω_0 and ω_1 are the two class’s probablities which is seperated by threshold t, and σ^2_0 and σ^2_1 are their variances.

$$\sigma_w^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t)$$

Intra-class variance and inter-class variance minimization is the same as shown in the equation below.

$$\begin{aligned}\sigma_b^2(t) = \sigma^2 - \sigma_w^2(t) &= \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2 \\ &= \omega_0(t)\omega_1(t)[\mu_0(t) - \mu_1(t)]^2\end{aligned}$$

Otsu’s method limitation lies when object area is comparatively small compared to the background area, or when the image has a lot of noise in it.

CHAPTER 3: SYSTEM DESIGN

3.1 Software and Input Data

The system is developed using Visual Studio 2015 with the C++ language. OpenCV libraries are also useful in providing a general infrastructure for systems using computer vision and image processing techniques, thus, enhancing development procedures.

The input data used to test out and develop the system are sample videos of school of underwater fishes by Shutterstock. This is used because they provide a plethora of different underwater fishes footages of varying environments which is useful for analysis and performance testing of the proposed algorithm. The properties of the input data is in .MP4 file format, resolution of 596 X 336, frame rates ranging from 24 to 50 frames per seconds.

3.2 System Overview

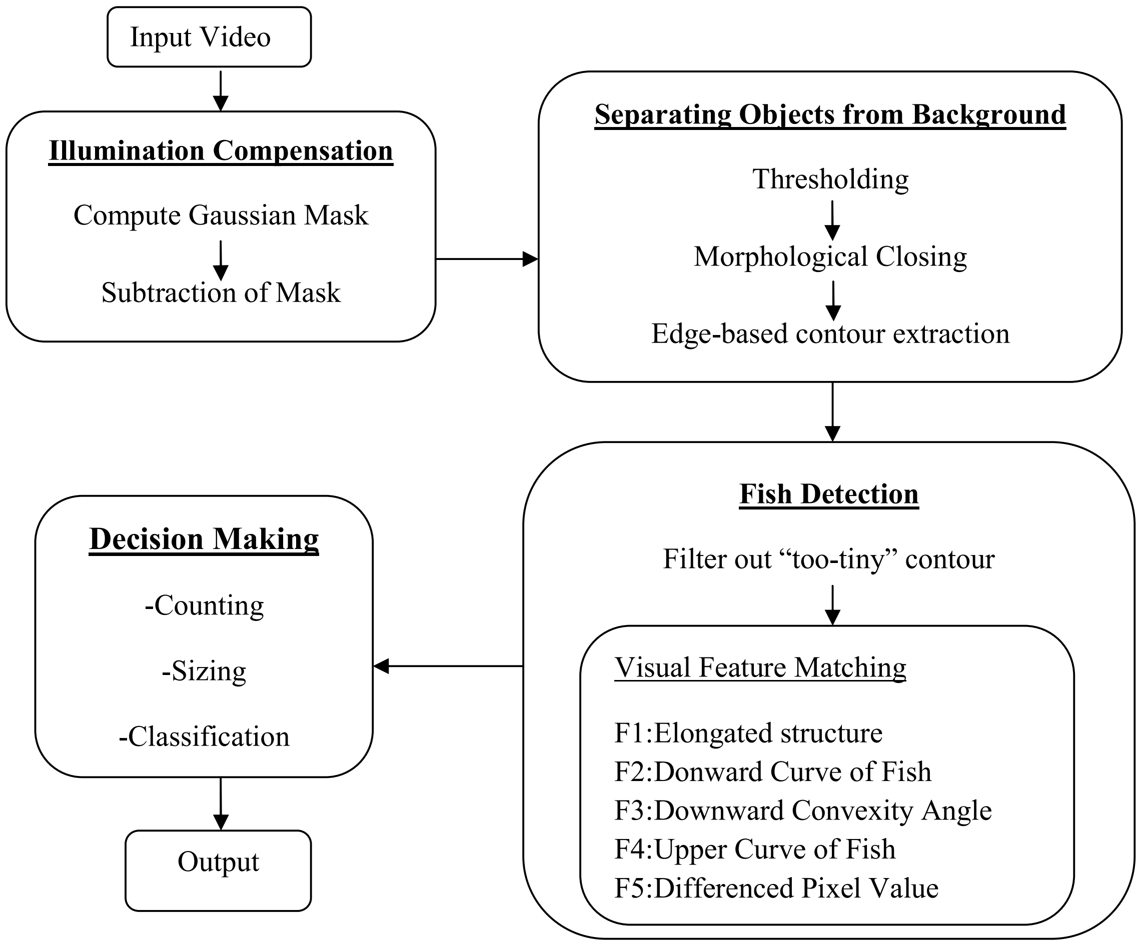


Figure 3.1 System Overview of Fish Monitoring System

3.2.1 Illumination Compensation

For the very initial step, Shutterstock sample video is being fed into the system, the system then processes the video using a frame-by-frame basis. Due to the fact that underwater footages usually have the issue of uneven lightning illumination, illumination compensation is needed as to not interfere with future processes, since computer vision is unlike human vision, which can interpret objects with ease despite being under very complex environment, the results produced using computer vision is sensitive to noise thus not addressing the uneven lighting issue would result in a very poor outcome.

Using a Gaussian Kernel, Gaussian blur of the x and y axis is done for the input frame to create a Gaussian mask, this is to predict the background illumination by blurring out the objects and retaining the estimated background to determine the illumination concentration. The original input image is then subtracted from the mask pixel by pixel.

3.2.2 Separating Objects from Background

Now that we have an evened out illumination, a threshold of a certain Scalar value is applied to change the image into a binary image,(further explanation on why and how it is done is provided in the following chapters) with black pixels signifying objects and the whites the background. Morphological operations are applied to smoothen out the edges of the objects. Last step for this part is to use an edge-based contour extraction algorithm to extract out contours from the smoothed binary image, the contour features are needed to differentiate between fish and non-fish entities.

3.2.3 Fish Detection

This step is to determine whether the detected object can be considered a fish. For the very first step in this part is to filter out contours that are way too tiny to be a fish, for contours with area under a certain threshold, they are immediately eliminated. What's left are contours with reasonable areas. Contour features are analysed to see if the particular contour matches the four visual features which corresponds to a fish, the features are: downward convexity angle, elongated structure, upper curve of fish, and lastly downward curve of fish. Visual features that are not related to the contour is the differenced pixel value for the region of interest and the mean background pixel values.

3.2.4 Decision Making

This is the part where the system establishes if the object have matched a sufficient amount of visual features to be classified as a fish. A counter have been made, for each detected object, when a visual feature matches, the counter will be incremented by one, once all features have been tested, for all object with counter value more than three, it passes and is classified as a fish, the object will also be included in the counting and sizing process.

3.3 Evaluation Method

Different footages would be tested out on the system, stopping at a few random frames. Human manual counting of the randomly selected frames would be done and compared to the system count. False negative and false positive are recorded down and the accuracy percentage is calculated.

The following chapters would describe in detail for each parts of the system, including pre-development analysis, further explanations and each part’s output.

CHAPTER 4: ILLUMINATION COMPENSATION

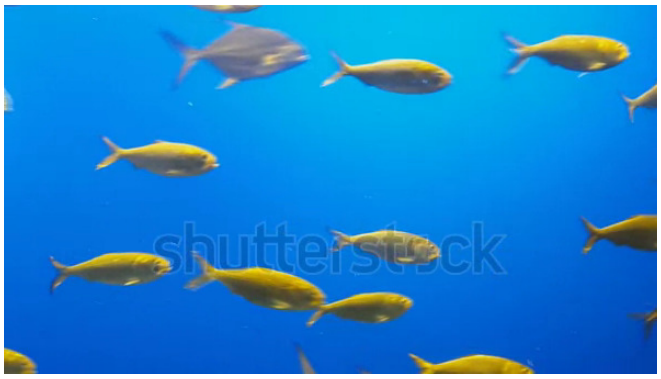


Figure 4.1 Input frame

As can be seen in figure 4.1, the underwater school of fishes' environment's illumination is not perfect. There is more illumination at the upper middle part of the frame, in which it gets darker and darker in a gradient way as it traverse to the bottom of the frame. Computer vision is very prone to be interfered with inconsistent lighting; therefore this issue has to be corrected, or else there would be a lot of noise present which would hinder future processes.

Gaussian blur method is used to create a Gaussian mask, in which its purpose is to define the gradient of illumination by blurring out the objects and retaining the background of uneven lighting. The reason Gaussian blurring is used is due to the fact that the results yield from it is far better in blurring out the scene evenly, unlike other techniques such as Median blur, which results in unsatisfactory blurring where the objects are still highly visible, hence not being able to describe the illumination gradient properly. After repeated testing on multiple footage and multiple frames, it is found that the Gaussian filter kernel which consists of 0.4 of input image size is best used to create a large convolution mask. Anything larger would result in the illumination description being blurred out as well, which would create a mask with no illumination gradient. While anything smaller would result in the object interfering in defining the illumination gradient of the mask. As can be seen in Figure 4.2, the mask corresponds to the estimated background illumination, with the objects blurred out, the illumination gradient can be seen with the middle upper part being lighter and it gets darker as it traverses down to the bottom part of the image.

This resulting mask is assigned to a Mat variable object and is later subtracted by the original input frame. The subtraction process is to subtract the values pixel by pixel, this can result in an evened illumination subtracted image. This works because, the Gaussian mask has the illumination gradient, in which the "lighter" part of an image have a certain pixel values that matches with the "lighter" part of the input image. When they are subtracted, they subtract each other out completely, and would result in a black pixel, if not completely, it is close to complete, which would result in a coloured but dark pixel. The objects are spared since they have been blurred out and therefore are not affected.

The result from this process is an image with evened illumination. The subtracted image is then assigned to another Mat object for further processing. Figure 4.3 shows the subtracted image, as can be seen there, the background no longer has a gradient illumination, and the pixels that have been subtracted completely results in a black background, or even the near complete ones results in a slightly coloured but still dark pixel, while the objects are clear.

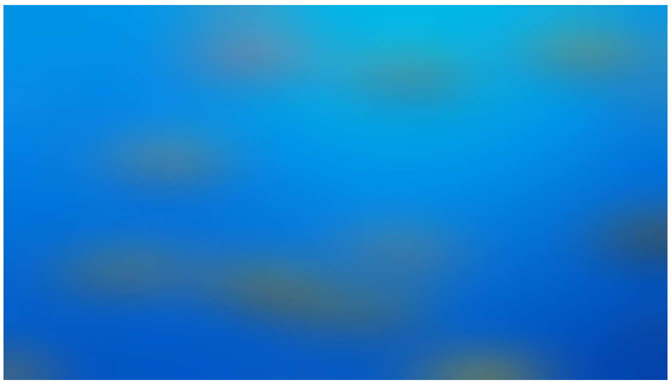


Figure 4.2 Gaussian Mask

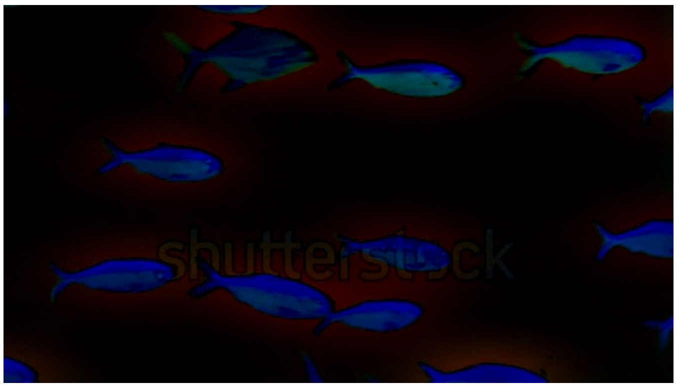


Figure 4.3 Subtracted Image

CHAPTER 5: SEPARATING OBJECTS FROM BACKGROUND

Analysing the Subtracted Image, it can be seen that the background had been converted to pure black, even if it is not pure black; it is a dark colour close to black. In this case, the image still have a reddish black around the objects, this is caused by blurring out the object in the Gaussian Blurring process to create the Gaussian Mask, and therefore when the original input frame is subtracted from the mask, the pixels surrounding the objects are not completely subtracted with each other, causing a non-pure black pixel, this gives the illusion that the colour of the objects "smeared" into its immediate surrounding causing the imperfect black background. In order to differentiate between the foreground objects and the background completely, thresholding process needs to be applied.

The threshold value used in the system has a range of Scalar value of pure black, which is in BGR value (0,0,0), to dark colours. Therefore, any pixel values that fall between that range would be converted to a white pixel, BGR(255,255,255), while the rest would be converted to a pure black pixel. What's left is a clear and direct representation of the background and foreground objects. This result can be seen in Figure 5.1.

Observing the threshold-ed image, the "blobs" representing the foreground objects are jagged and the edges are not defined enough, not to mention the noisy unreliable pixels that surround the object. This would cause some issue for the future processes such as contour extraction. Therefore, a method needs to be applied to join the jagged lines together to make it appear smoothened out; at the same time eliminate less reliable pixels. In this case, morphological closing is applied; the workings of morphological closing have been explained in the Literature Review in the previous chapter. The result after morphological closings are applied can be seen in Figure 5.2, the edges are smoothened out and less reliable pixels can no longer be seen.

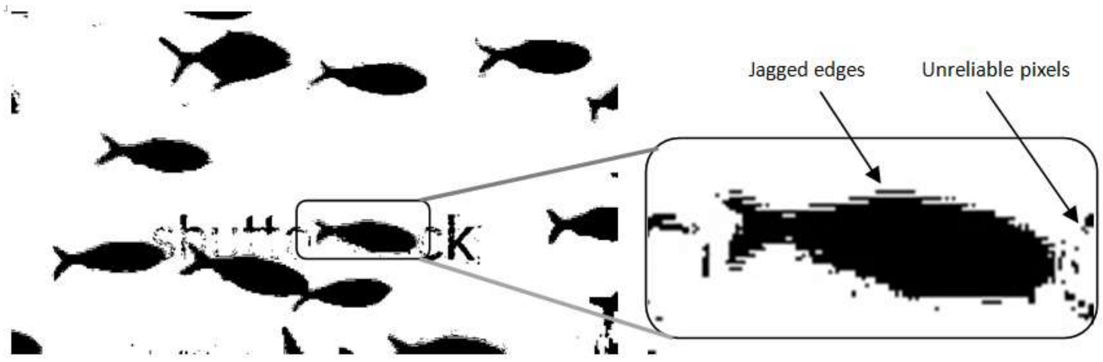


Figure 5.1 Threhsold-ed Image

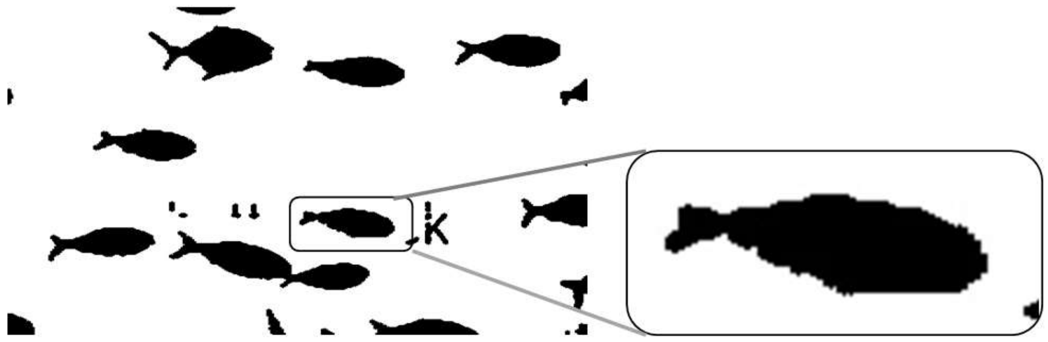


Figure 5.2 Morphological Closing result

What's left for this part is to define the contours of each object. The reason contours are extracted out is because the most obvious features that can be used for computer vision are the features of the contours. Contour features provide meaningful information such as the height and width of the contours, the curves of the contour and many more. This feature information is used to determine and differentiate whether the detected object can be considered a fish or not.

An edge based contour extraction algorithm is applied to define each object's contour. The workings of the edge based contour extraction algorithm have been explained in the Literature Review chapters. The system uses the most external contours only. The results can be seen in Figure 5.3.

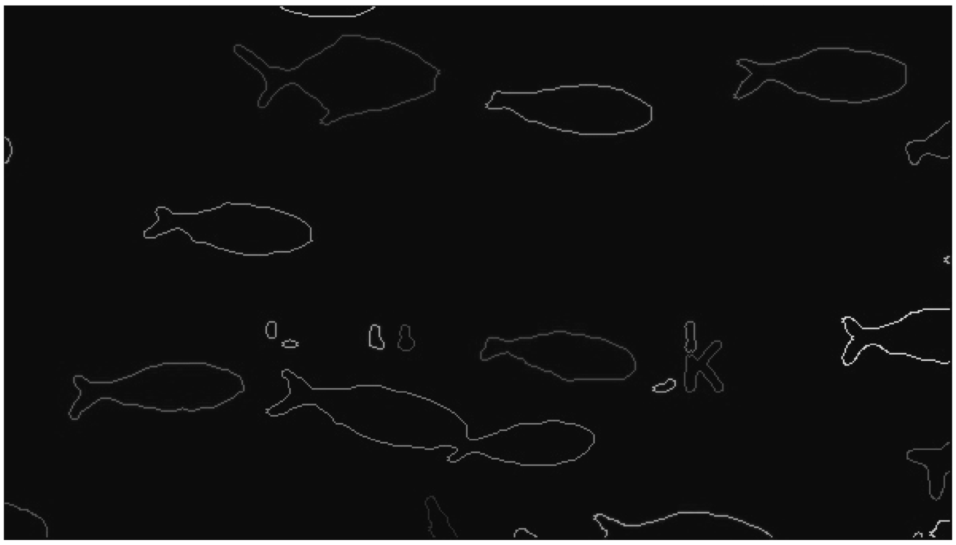


Figure 5.3 Extracted Contours

Each complete closed contour's pixel position is stored in an array. By the end of the contour extraction process, there would be an array of contours that can be looped across storing the pixels' xy-coordinates that makes up the particular contour. As can be seen, some contours do not belong to fishes.

For human vision, we can easily know which contour signifies a fish; however computer vision cannot do that. Therefore, further visual features needs to be analysed to determine whether the contours detected belongs to a fish.

CHAPTER 6 FISH DETECTION

Analysing the contours, there are a few visual features that corresponds with the fishes. What the system would be using are namely: F1) Elongated Structure, F2) Downward Curve of Fish, F3) Downward Convexity Angle, F4) Upper Curve of Fish, F5) Differenced Pixel Value. What the system, in short, does at this part is loop through the array of contours, each contour would then go through each visual feature one by one, each visual feature would produce a certain result from the contour, if the contour results falls between a specified range for a particular feature, the contour is said to have matched that specified visual feature, it will then be awarded with a score. Once the contour have gone through all the visual features, if the contour's accumulated score is more than a certain threshold, that contoured corresponds to a fish and is included in the later counting and sizing process.

Before all the process described above is implemented, the contours need to go through a very simple elimination round. For contour areas that are too little, they are eliminated and would not go through the visual feature matching process.

6.1 Elongated Structure

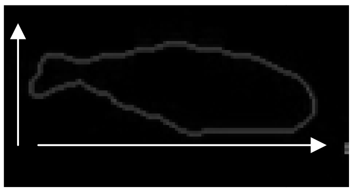


Figure 6.1 Elongated Structure of Fish

As seen in Figure 6.1, a fish contour has an elongated structure, with the x-axis being more than 2 times shorter than the y-axis. This is because a whole fish consists of its head, its body and its tail, all arranged in a horizontal manner, giving it its elongated structure. This holds true to most type of fishes found in nature, where the whole fish would represent an elongated structure, where some are more elongated than the rest. A bounding box is drawn around each contour, then, the length shorter side of the bounding box is compared with the length of the longer side of the bounding box, if the longer side's length is more than twice the length of the shorter side, the contour is said to have an elongated structure, and the contour matched this visual feature.

6.2 Downward Curve of Fish

Figure 6.2 shows how we can determine whether a line is indeed a curve or a straight line.

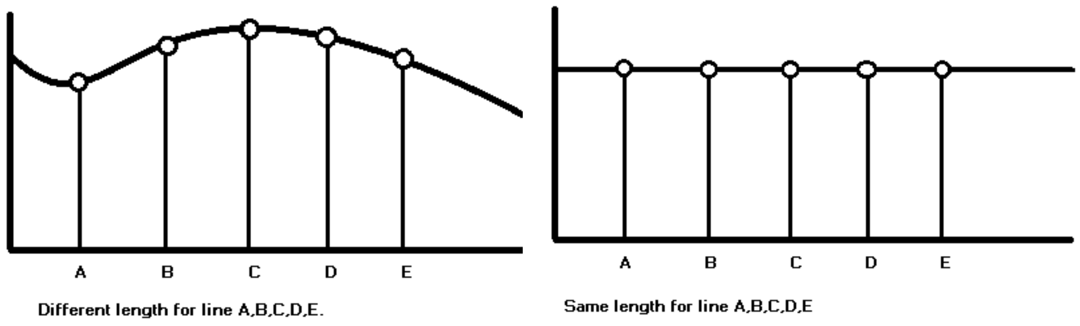

A
B
C
D
E
Different length for line A,B,C,D,E.
A
B
C
D
E
Same length for line A,B,C,D,E

Figure 6.2 Curve Detection

If the length of line connecting the x axis and the plotted points is different for line A,B,C,D and E, we can know that the line in which the points are plotted on is a curved line. The system will adopt this method to determine if the contour is indeed a curve as found in fishes, or not.

Basically, a few points would be assigned. The middle point of the both the shorter side of the bounding box is assigned as Equator point East and West. Then, an Equator Line is drawn to connect these two points. The Equator line is a line which cuts the fish horizontally in half, adjusted to the fish’s slight rotation.

Then, 5 points in the Equator line is assigned, the first one in the middle, second one in between the middle and Equator point East, third one being in the middle of second point and Equator point East, while the rest is a repetition of the above on the West side.

For each of these 5 points, each point coordinates forming the contours is looped to find and assigned a point on the contour which forms a 90 degree angle points in the equator line. Then, a line is drawn connecting the Equator line point and its corresponding contour point.

The angle is calculated with the formula below.

$$\alpha = \cos\theta = \frac{\bar{a} \cdot \bar{b}}{\|\bar{a}\| \cdot \|\bar{b}\|}$$

With vector 'a' being the point on the Equator line and vector 'b' being the points in the contour.

Then, the standard deviation of the length of the line is calculated. If the standard deviation is more than 2 points and not more than 50 points(because that would be impossible for the degree of curve of the fish), it is said that the contour matches this visual feature.

An overview of how this work is shown in Figure 6.3.

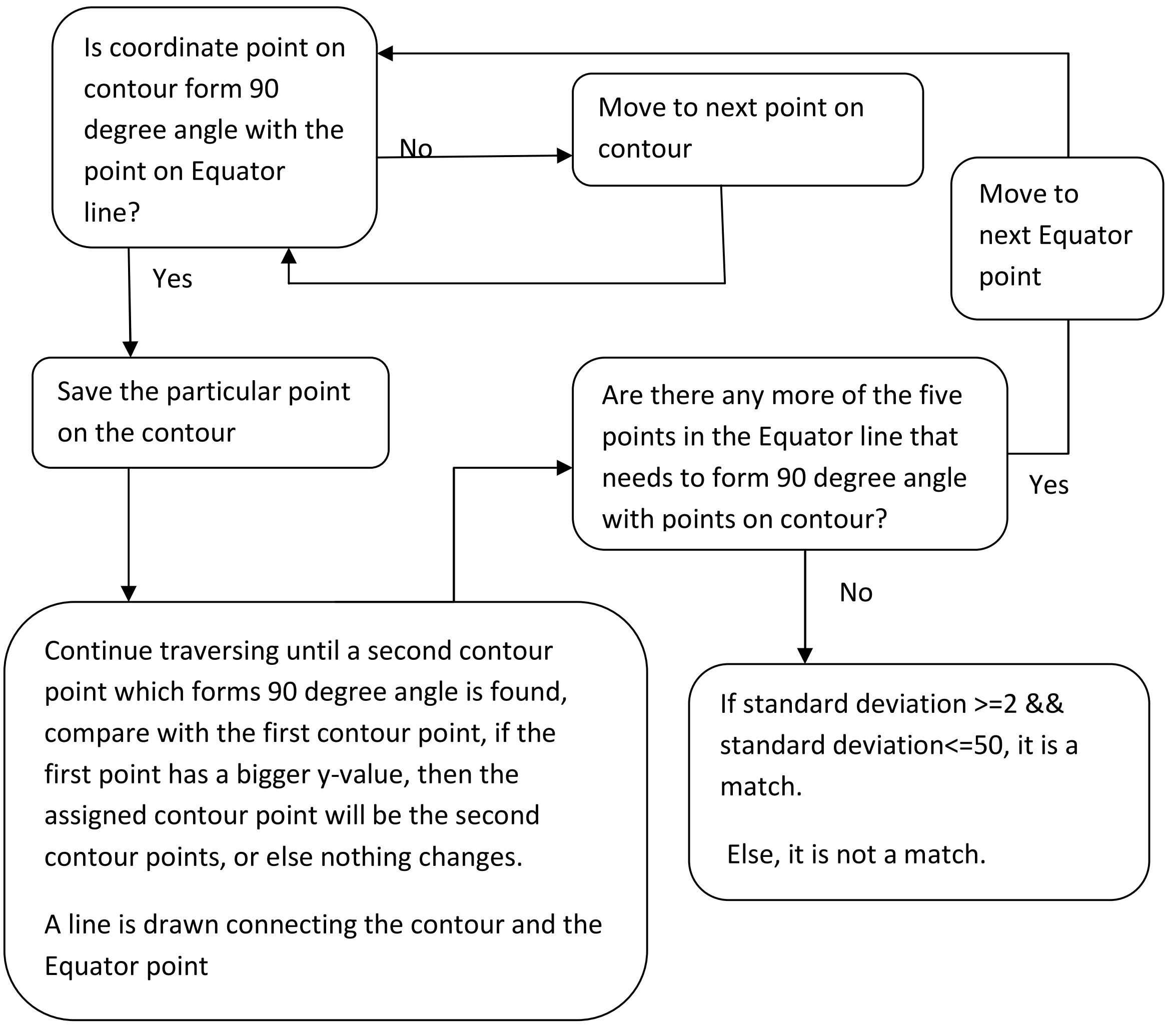

Is coordinate point on contour form 90 degree angle with the point on Equator line?
No
Move to next point on contour
Move to next Equator point
Yes
Save the particular point on the contour
Are there any more of the five points in the Equator line that needs to form 90 degree angle with points on contour?
Yes
No
Continue traversing until a second contour point which forms 90 degree angle is found, compare with the first contour point, if the first point has a bigger y-value, then the assigned contour point will be the second contour points, or else nothing changes.
A line is drawn connecting the contour and the Equator point
If standard deviation >=2 && standard deviation<=50, it is a match.
Else, it is not a match.

Figure 6.3 Overview of Curve Detection algorithm

Figure 6.4 shows the points drawn and also the lines drawn on the contours. Red line is the Equator line. Blue line and red lines form 90 degree angles.

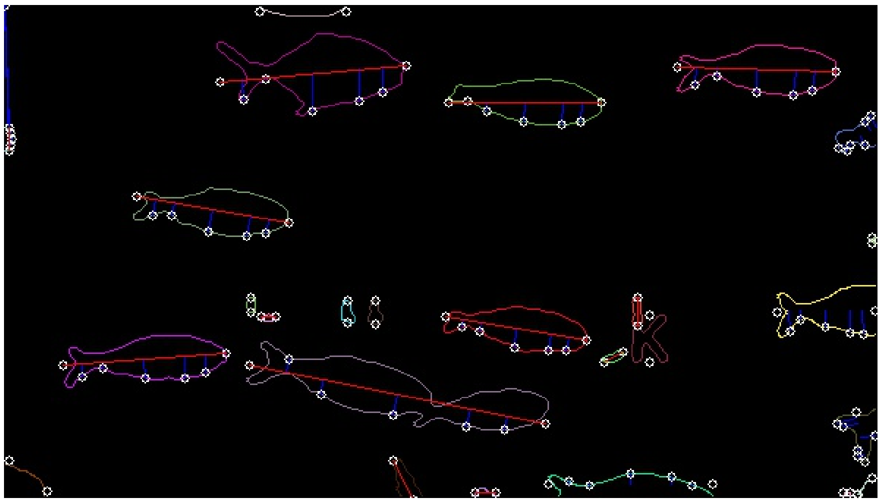


Figure 6.4 Output results showcasing Downward Curve of Fish

6.3 Downward Convexity Angle

A fish has a belly of certain volume which is downward convex in shape. Therefore, this visual feature can be implemented on the contour. However, it is noted that the downside of using this method is evident while detecting fishes with highly thin proportions, such as the Barracuda; however, most fishes found in nature do have an obvious and detectable downward curvature shape representing their belly.

Using points already established in the previous visual feature, the points are assigned as shown in Figure 6.5 below. The angle formed between the red lines are computed, after analysing what the usual angle is for a fish, it is decided that the angle lying in the range between 90 and 165 is suitable to be a visual match. Because, an angle larger than that value would mean that the volume that corresponds to the belly of the fish is little to non-existence, while anything smaller would form an acute angle, which is not possible to belong to a fish.

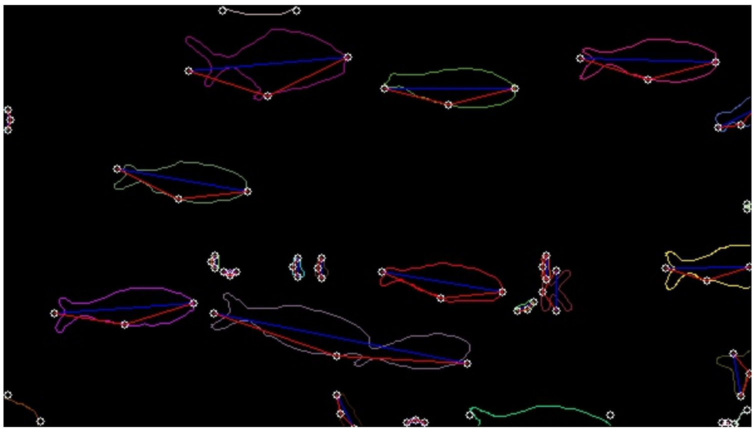


Figure 6.5 Downward Convexity Angle output

6.4 Upper Curve of Fish

Reusing the methods in Chapter 6.2, with the only difference is that the points assigned on the contour now gives priority to the points which has a smaller y-value, so that the points assigned all belong to the upper side of the fish. The output can be seen in Figure 6.6.

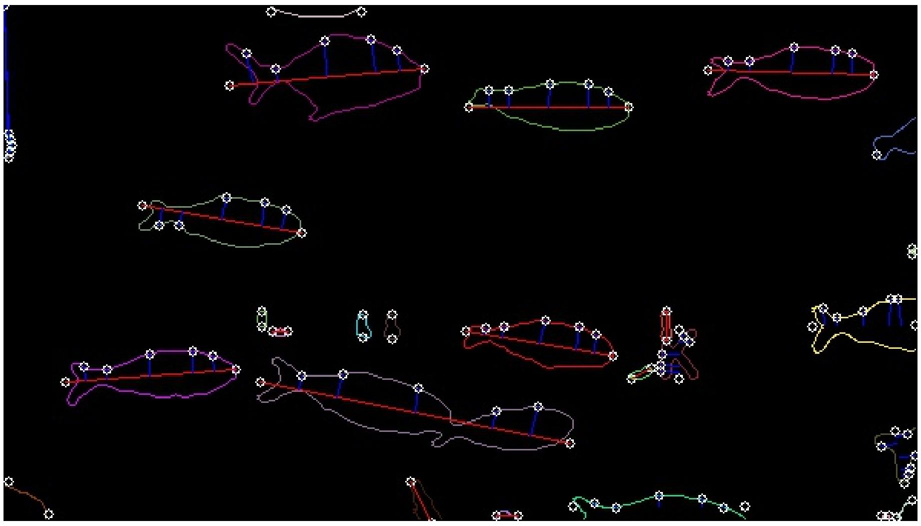


Figure 6.6 Output results showcasing Upward Curve of Fish

6.5 Differenced Pixel Value

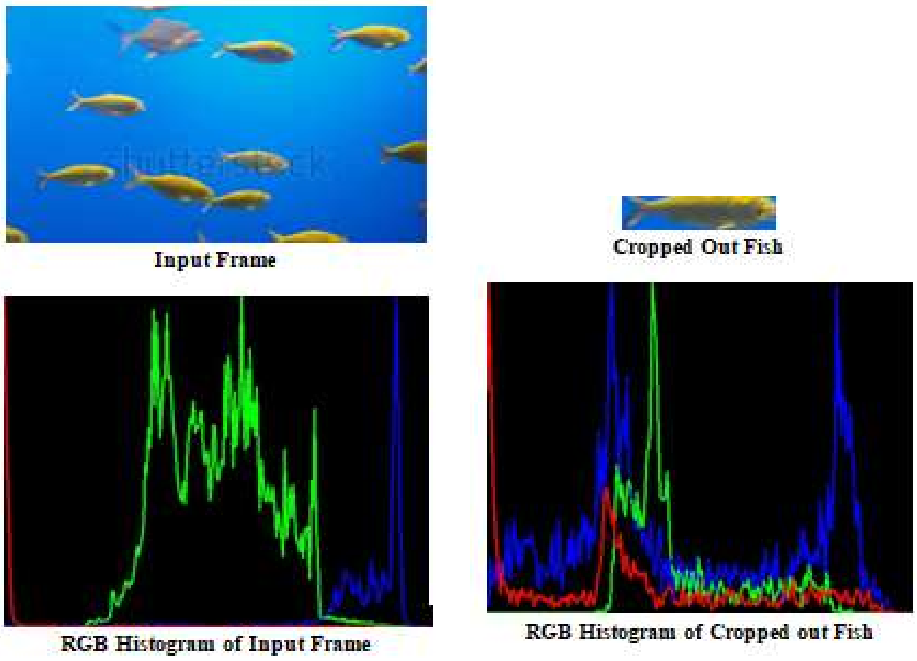


Figure 6.7 Histograms of Input Frame and Cropped out Fish

Analysing Figure 6.7, the histograms shows that the ROI containing a fish’s colour information is very different compared with the input image. The input image has a large amount of green pixels while the opposite is true for the cropped out ROI. Therefore, this visual feature is used to determine whether the ROI containing the contour, corresponds to the fish by comparing their summed up and mean pixel values. The differenced mean pixel value is calculated. An optimum threshold value is heuristically tested with multiple different types of footage. If the differenced pixel value is more than the specified threshold, the contour matched with the visual feature.

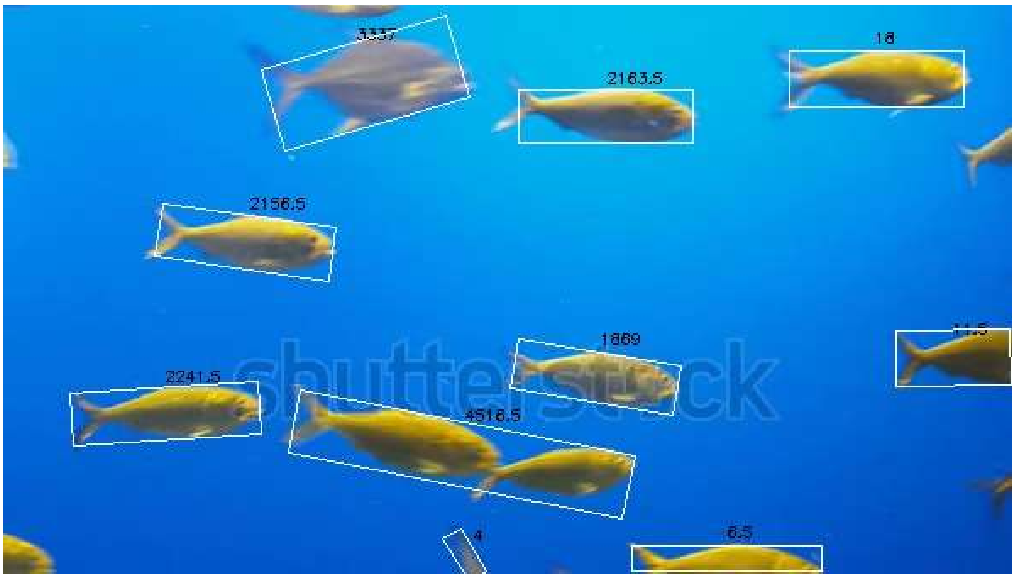
CHAPTER 7 DECISION MAKING

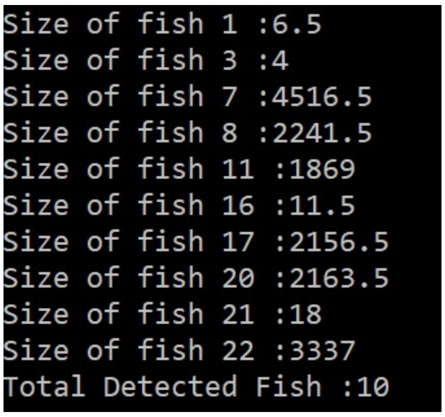
As mentioned earlier, there is an array of contour which is looped, and with each loop each contour would pass through five visual features for feature matching. Each visual matching algorithm would produce a result describing the contours; the results are then compared with a set threshold value to determine if the contour falls within the range of the threshold value to be considered a match with the visual feature.

For the decision making part of the system, a counter is initialized for every contour iteration, with an initial value of zero. Every time a contour matches a visual feature, the counter is incremented by one. After the contour have finished going through each visual feature, the accumulated counter value would be evaluated. For contour with counter value more than 3, which means the contour have matched more than half of the visual feature representing a fish, the contour is considered as a fish, and it would be included in the counting and sizing process, else, the contour would be ignored.

For the counting process, another counter is initialized with value zero, for each contour that passes the evaluation, the counter responsible for counting the fishes is incremented by one, and this would be repeated until the very last contour in the array have been iterated. For the contours that have passed the evaluation, they would be sized by counting the pixels that exists within the enclosed area of the contour. The results including the size and the number of fish of a frame is displayed every 10 frames of the video.

Figure 7.1 showcases the output result.



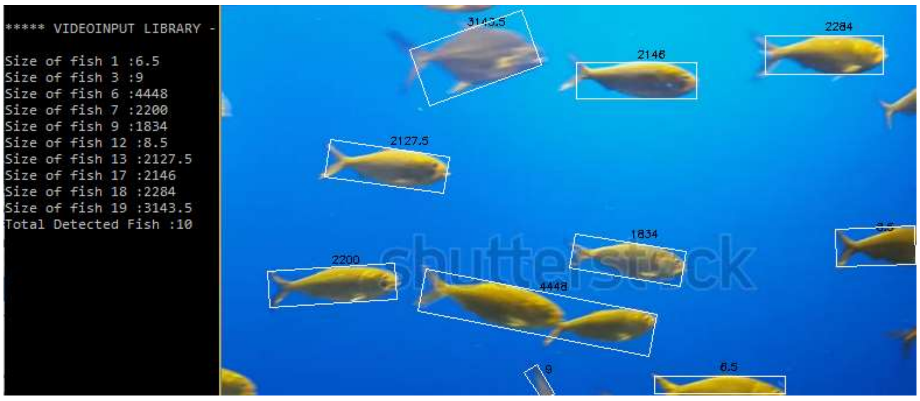


```
Size of fish 1 :6.5
Size of fish 3 :4
Size of fish 7 :4516.5
Size of fish 8 :2241.5
Size of fish 11 :1869
Size of fish 16 :11.5
Size of fish 17 :2156.5
Size of fish 20 :2163.5
Size of fish 21 :18
Size of fish 22 :3337
Total Detected Fish :10
```

Figure 7.1 Output results

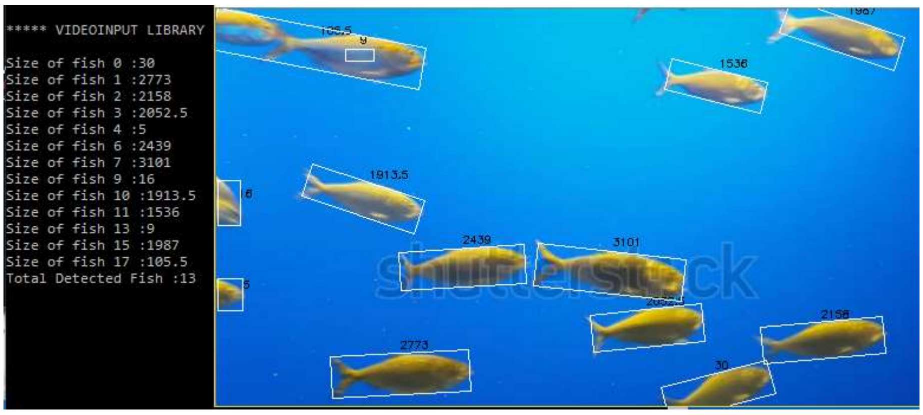
CHAPTER 8 RESULTS EVALUATION

Some footage of underwater fishes in different environment are tested. Sample frames are randomly chosen for testing and evaluation. The output results are show in Figure 8.1 to Figure 8.9.



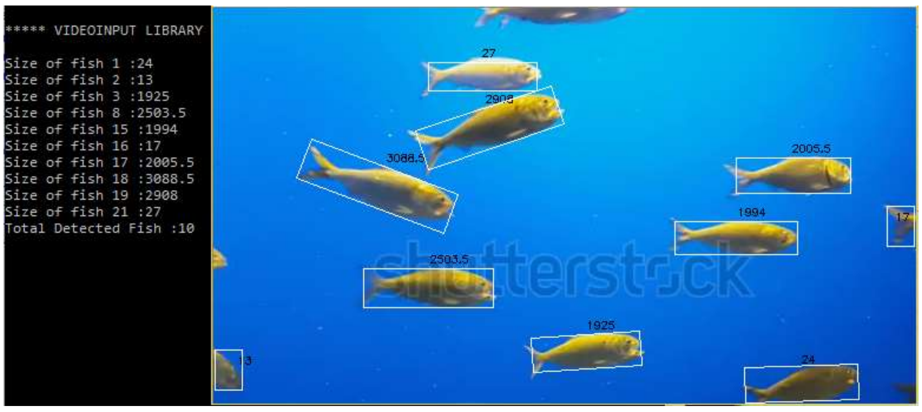
```
***** VIDEOINPUT LIBRARY -

Size of fish 1 :6.5
Size of fish 3 :9
Size of fish 6 :4448
Size of fish 7 :2200
Size of fish 9 :1834
Size of fish 12 :8.5
Size of fish 13 :2127.5
Size of fish 17 :2146
Size of fish 18 :2284
Size of fish 19 :3143.5
Total Detected Fish :10
```

Figure 8.1 Sample Output 1



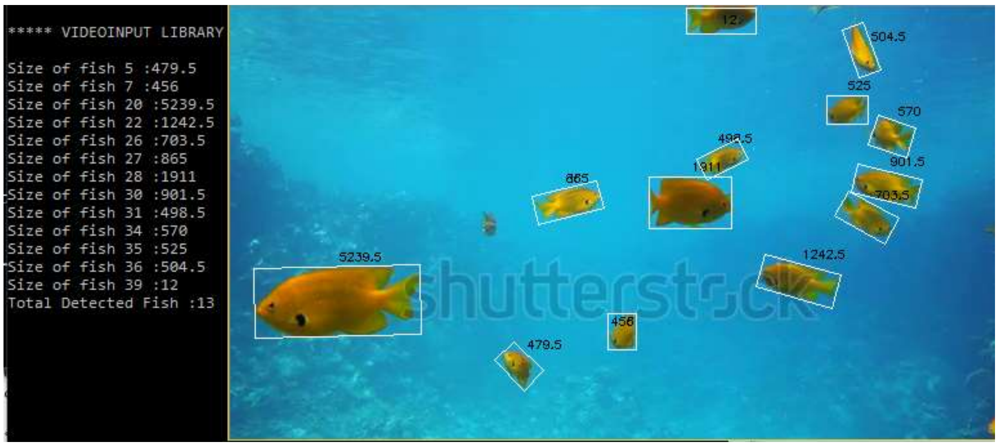
```
***** VIDEOINPUT LIBRARY

Size of fish 0 :30
Size of fish 1 :2773
Size of fish 2 :2158
Size of fish 3 :2052.5
Size of fish 4 :5
Size of fish 6 :2439
Size of fish 7 :3101
Size of fish 9 :16
Size of fish 10 :1913.5
Size of fish 11 :1536
Size of fish 13 :9
Size of fish 15 :1987
Size of fish 17 :105.5
Total Detected Fish :13
```

Figure 8.2 Sample Output 2



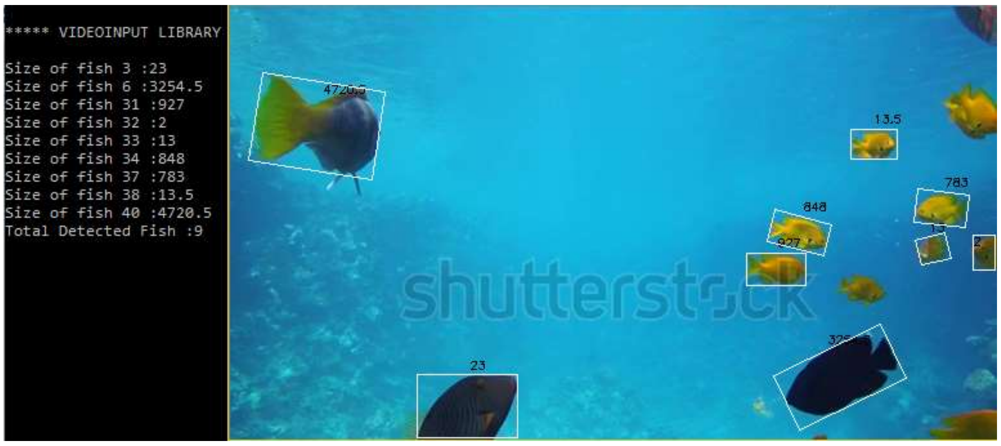
```
***** VIDEOINPUT LIBRARY

Size of fish 1 :24
Size of fish 2 :13
Size of fish 3 :1925
Size of fish 8 :2503.5
Size of fish 15 :1994
Size of fish 16 :17
Size of fish 17 :2005.5
Size of fish 18 :3088.5
Size of fish 19 :2908
Size of fish 21 :27
Total Detected Fish :10
```

Figure 8.3 Sample Output 3



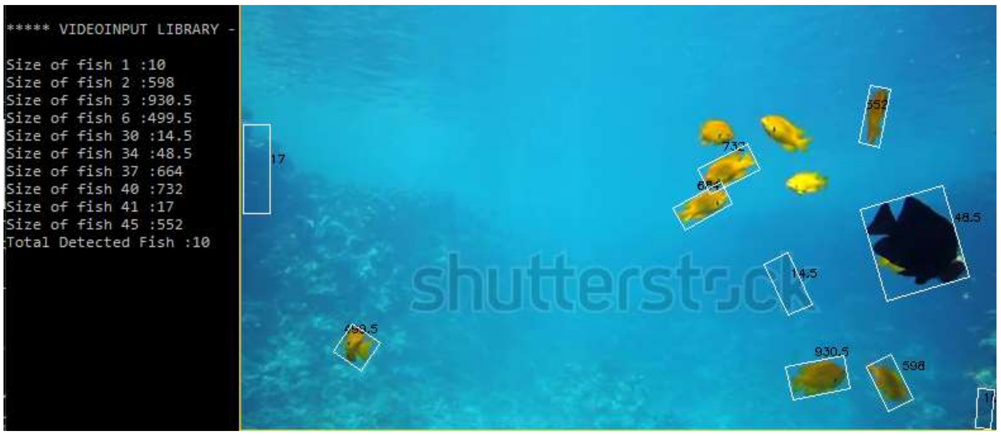
```
***** VIDEOINPUT LIBRARY

Size of fish 5 :479.5
Size of fish 7 :456
Size of fish 20 :5239.5
Size of fish 22 :1242.5
Size of fish 26 :703.5
Size of fish 27 :865
Size of fish 28 :1911
Size of fish 30 :901.5
Size of fish 31 :498.5
Size of fish 34 :570
Size of fish 35 :525
Size of fish 36 :504.5
Size of fish 39 :12
Total Detected Fish :13
```

Figure 8.4 Sample Output 4



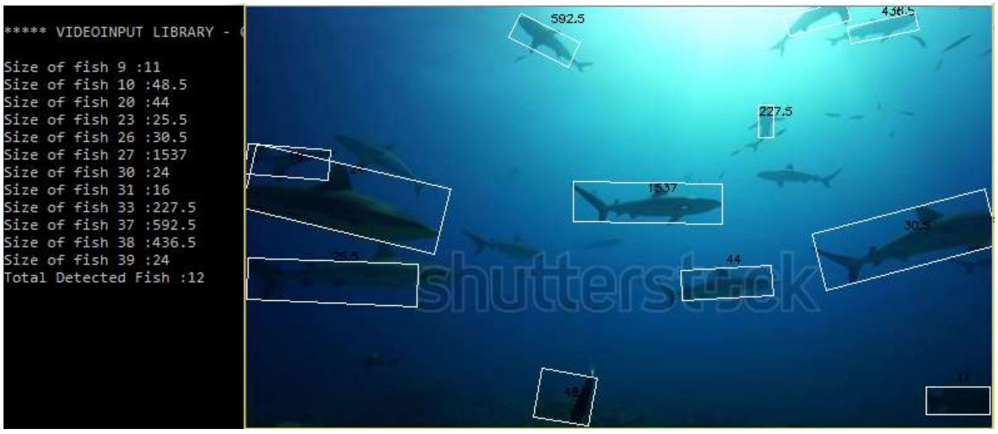
```
***** VIDEOINPUT LIBRARY

Size of fish 3 :23
Size of fish 6 :3254.5
Size of fish 31 :927
Size of fish 32 :2
Size of fish 33 :13
Size of fish 34 :848
Size of fish 37 :783
Size of fish 38 :13.5
Size of fish 40 :4720.5
Total Detected Fish :9
```

Figure 8.5 Sample Output 5



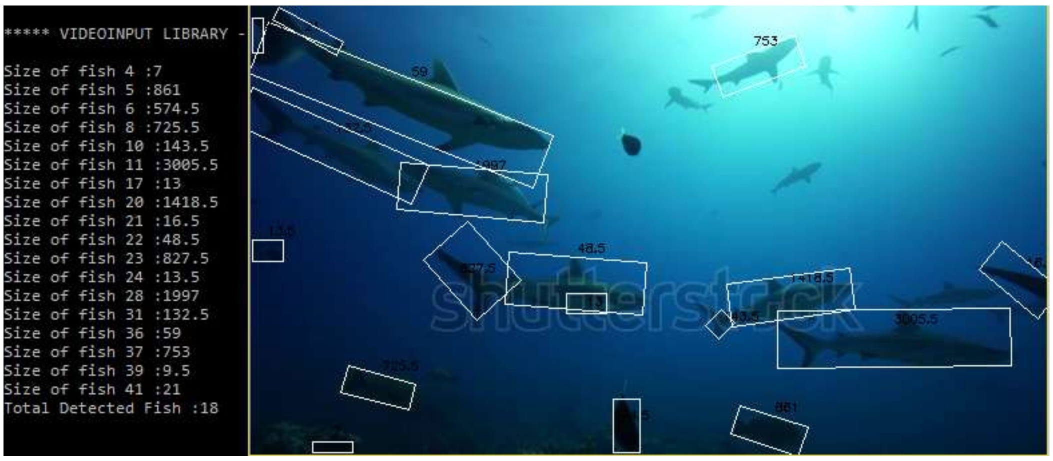
```
***** VIDEOINPUT LIBRARY -

Size of fish 1 :10
Size of fish 2 :598
Size of fish 3 :930.5
Size of fish 6 :499.5
Size of fish 30 :14.5
Size of fish 34 :48.5
Size of fish 37 :664
Size of fish 40 :732
Size of fish 41 :17
Size of fish 45 :552
Total Detected Fish :10
```

Figure 8.6 Sample Output 6



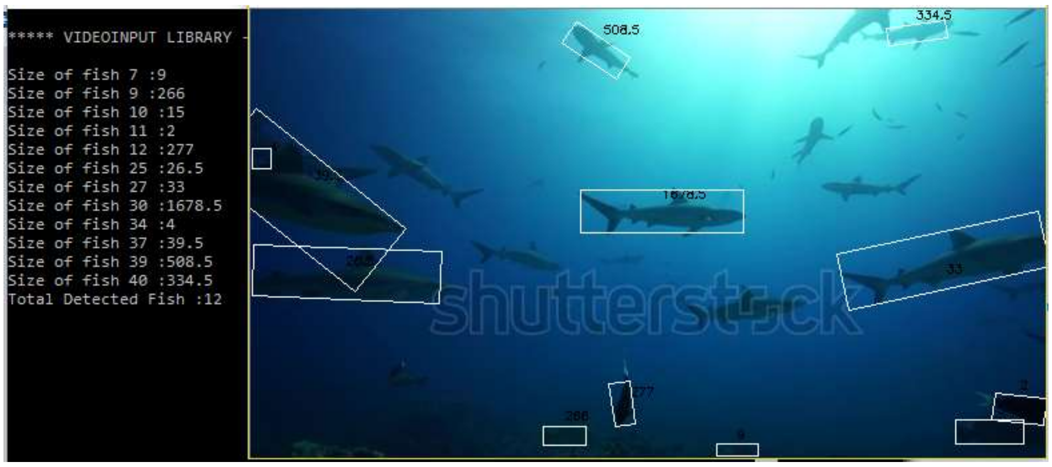
```
***** VIDEOINPUT LIBRARY - 

Size of fish 9 :11
Size of fish 10 :48.5
Size of fish 20 :44
Size of fish 23 :25.5
Size of fish 26 :30.5
Size of fish 27 :1537
Size of fish 30 :24
Size of fish 31 :16
Size of fish 33 :227.5
Size of fish 37 :592.5
Size of fish 38 :436.5
Size of fish 39 :24
Total Detected Fish :12
```

Figure 8.7 Sample Output 7



```
***** VIDEOINPUT LIBRARY -

Size of fish 4 :7
Size of fish 5 :861
Size of fish 6 :574.5
Size of fish 8 :725.5
Size of fish 10 :143.5
Size of fish 11 :3005.5
Size of fish 17 :13
Size of fish 20 :1418.5
Size of fish 21 :16.5
Size of fish 22 :48.5
Size of fish 23 :827.5
Size of fish 24 :13.5
Size of fish 28 :1997
Size of fish 31 :132.5
Size of fish 36 :59
Size of fish 37 :753
Size of fish 39 :9.5
Size of fish 41 :21
Total Detected Fish :18
```

Figure 8.8 Sample Output 8



```
***** VIDEOINPUT LIBRARY -

Size of fish 7 :9
Size of fish 9 :266
Size of fish 10 :15
Size of fish 11 :2
Size of fish 12 :277
Size of fish 25 :26.5
Size of fish 27 :33
Size of fish 30 :1678.5
Size of fish 34 :4
Size of fish 37 :39.5
Size of fish 39 :508.5
Size of fish 40 :334.5
Total Detected Fish :12
```

Figure 8.9 Sample Output 9

Below figure shows the evaluation on the sample output and accuracy percentage have been calculated. Manual human counting of the fishes have been done and the results are compared with the system count. False negative, which are fishes that are not detected, and false positive, which are falsely detected regions, are manually counted as well.

Figure	**Human Count (HC)**	**System Count (SC)**	**False Negative (FN)**	**False Positive (FP)**	**Accuracy ((HC-FN-FP)/HC) ×100%**
8.1	**10**	**10**	**1**	**1**	**80**
8.2	**11**	**13**	**2**	**1**	**72**
8.3	**8**	**10**	**0**	**2**	**75**
8.4	**14**	**13**	**0**	**1**	**92**
8.5	**11**	**9**	**0**	**2**	**81**
8.6	**10**	**10**	**3**	**3**	**70**
8.7	**15**	**13**	**3**	**1**	**70**
8.8	**16**	**14**	**3**	**1**	**30**
8.9	**17**	**16**	**2**	**3**	**71**
					Average:71.22%

Figure 8.10 Evaluation Results

The average accuracy percentage for the tested footage of the system is 71.22%. There are still some inaccuracies.

One type of error is failure to distinguish multiple fishes when they are intersecting each, the system would count the two or more overlapping fish as one big fish. This problem stems from the contour drawing part, since the contours are based on an edge based drawing algorithm, when the fishes are intersecting, the contour would only draw the outermost edges and ignore the intersecting points, forming one big contour. Since the rest of the visual feature matching is reliant on the contours, error occurs. This can be seen in Figure 8.11. Similar issue can be seen in Figure 8.2, Sample Output 2, fishes on the top left corner.

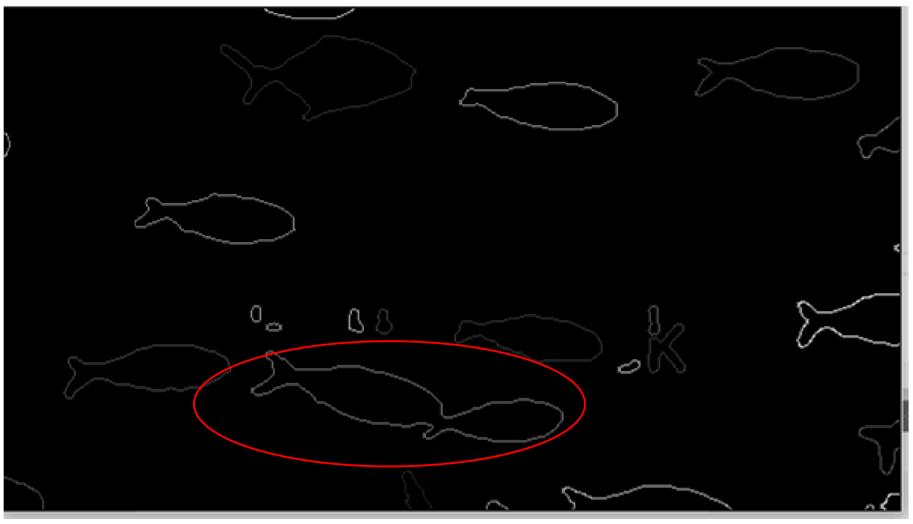


Figure 8.11 Intersecting Fishes

There are also some obvious inaccuracies in the estimation of contour area, as can be seen in the Figure 8.12, the fishes which are close to the sides of the frame, have unrealistic sizes. For example, it is obvious that the fish circled in green is larger than the fish detected circled in red, however the contour output states that the fish circled in green has a smaller area than the red one, the same thing applies to the fish circled in white, it is obviously bigger than the fish circled in red, however the output does not states so. One explanation is because the contours formed at very sides of the frame are not completely closed contours, as can be seen in Figure 8.11, unlike the other contours, they are not closed and were cut off by the frames, making them an incomplete contour with undefined closed area, that is why there are errors in estimating the size of the contours.

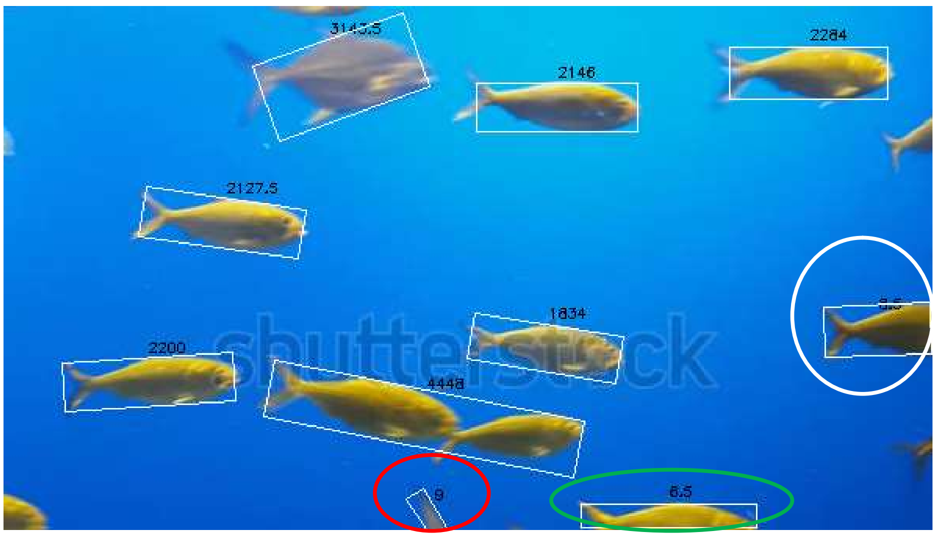


Figure 8.12 Contour Area Inaccuracies

Another error can be seen in Sample Output 2, Figure 8.2, where there is a tinier bounding box within a bigger bounding box. This is caused by imperfect morphological operations where the tiny bits of noise were not merged or eliminated properly, resulting in the contour edge detection algorithm to detect the noise, and by coincidence the contour visual features matched at least three out of five of the visual feature proposed, therefore the system falsely considered it to be a fish. The same issue can be observed in Figure 8.9, Sample Output9, and Figure 8.8, Sample Output 8.

In Figure 8.6, Sample Output 6, there are some fishes that are falsely not detected. Further observation, despite that the contours described them correctly, it is realised they did not pass at least three of the visual features proposed, specifically, elongated structure, upward curvature and downward curvature. The same thing can be observed from Sample Output 4 to Sample Output 9, where the contours representing the specific fishes failed to match at least three out of five visual features, or they have been eliminated immediately for not having enough size to even go through the visual feature matching mechanism.

The last type of error that would be discussed is false positive, where the system detected a fish where there is none. False positive error occurs mainly due to imperfect and incorrect elimination of noises during the thresholding process to convert the mask subtracted image into binary image. As can be seen in Figure 8.13, there is an imperfect separation of background (white pixels) and foreground objects (black pixels), which are the fishes. These imperfect separations can be found relatively close to the side of the frame. With this, as can be seen in Figure 8.14, the contour extraction algorithm would draw out these falsely detected foreground objects (black pixels), which would then go through the feature matching mechanism. Usually, these noises which appear at the sides of the frame would not match three out of five visual feature proposed, however, there are some instances where these noises would get through the feature matching process, leading the system to output false results. Such issues can be seen repeatedly from Sample Output 7 to Sample Output 9.

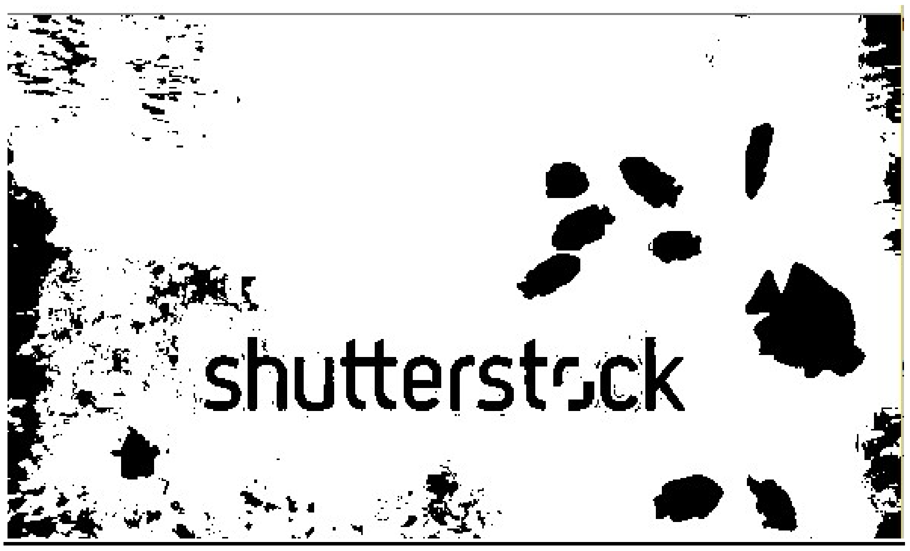


Figure 8.13 Imperfect Separation of Background and Fishes

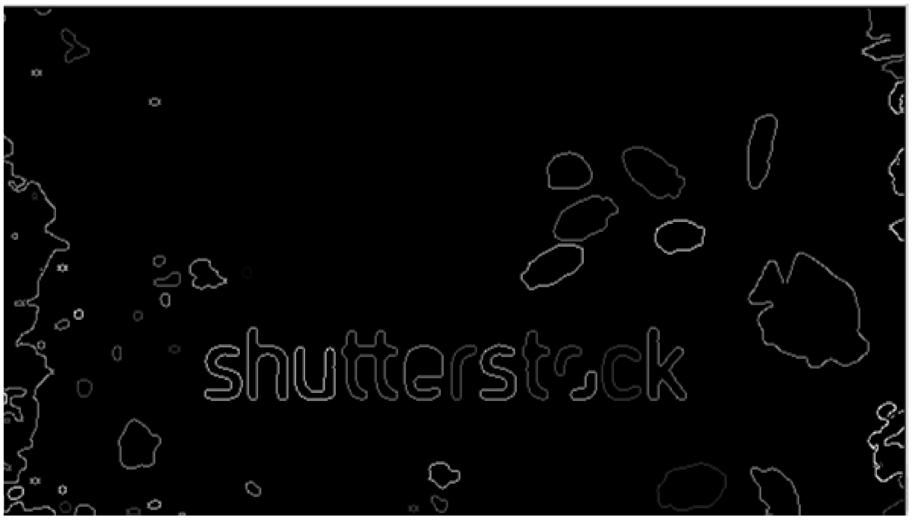


Figure 8.14 Contour Falsely Drawn

From the analysis made regarding errors that the system made, it is found out that there are a lot of instances where contours which does not corresponds to fishes still were falsely labelled as a fish due to coincidentally matching three out of five visual features. Therefore, more visual features that can only be found in fishes can be made to eliminate the contours which do not correspond to fishes.

For the issue where the system fails to differentiate intersecting fishes, a tracking algorithm can be implemented to solve such an issue. This is because the tracking algorithm is able to track individual fish and therefore even if part of the fish is being blocked by another fish, the tracking algorithm would know where the fishes are and will count those fishes as individual fishes instead of labelling them as one big fish.

It can be seen that a lot of errors made by the system usually occur very closely to the side of the frames. Such as the false positive error due to the noises in Figure 8.13, and also the inaccurate size estimation of fishes due to incompletely closed contours due to the limit of the frame size. One way to reduce all these inaccuracies is to only regard the fishes in the middle part of the frame, how large the part of the frame would have to be manually determined. However, this method would require sacrificing some information due to the need to reduce observation size by only regarding whatever is in the smaller cropped out frame.

CHAPTER 9 CONCLUSION

It is no longer sustainable to depend on natural offshore fishing resources to supply for the globally increasing demands for fishes. Therefore, fish farms, is a sustainable solution for this predicament. With a large amount of fishes to handle within a fish farm, Monitoring growth and the condition of fishes in an aquaculture environment is quite important since it gives information as to when the fish is ready to be harvested and when to transfer the fish to another environment.The traditional method of manual fish monitoring is very inefficient, therefore a proposed system is made to increase fish farm's efficiency that would have beneficial effects for the whole world.

This project have proposed a method to monitor fishes underwater under varying environments especially environment where illumination is non-even across the frame. The system detects the fishes, monitors the number of fishes and the estimate each counted fish's size, all automatically. With this, fish farms would only need to analyze output generated by the system to gain needed information.

The inspiration for the techniques used to develop this system have been founded while reviewing other's work on similar fields, and have extracted desirable techniques of many different sources, combining at the same time modifying it to create this system.

Some achievements have been made developing this fish monitoring system. A method for evening out the inconsistent lighting found in underwater footages so as to not interfere with further processes have been implemented. Ways to detect different visual features within an image have been achieved by mainly using features that can be found in contours of object, while another visual feature is by using colour information.

Some issues have been encountered while developing the system, one of the major one is the portability of the system. This means that developing the system using only one sample footage may accidentally over-cater to only that particular footage, which would cause the system to produce poor results when using footage of slightly different environment. Therefore, multiple test footages of varying environment is needed for the pre-developmental analysis for every part of the system to prevent such issue.

There are some improvements that can be made for future work, such as improving the robustness and accuracy of the system, which can be done by implementing more visual feature or refining the existing visual features. Not to mention correcting the intersecting fishes issue.

The overall research experience has been really informative and eye widening. There are so many different ingenious techniques and algorithms found while reviewing other people’s work. It has been a positive learning experience.

References

BMVA The British Machine Vision Association and Society for Pattern Recognition*,* *What is computer vision?,* n.d, Available from:< http://www.bmva.org/visionoverview> [22 February 2018]

Burger, W, Burge, MJ 2010, Principles of Digital Image Processing Core Algorithms, pp. 110–111. Available from: Springer Science & Business Media. [8 April 2018]

Chow, L 2016, *Global Fish Stocks Depleted to 'Alarming' Levels,* EcoWatch, viewed 22 February 2018,< https://www.ecowatch.com/one-third-of-commercial-fish-stocks-fished-at-unsustainable-levels-1910593830.html>

Chu, W 2017, *Higher Marine Food Demand Places Big Fish On 'At Risk' List, Study Finds,* Food Navigator, viewed 4 August, < http://www.foodnavigator.com/Policy/Higher-marine-food-demand-places-big-fish-on-at-risk-list-study-finds >

Davies, ER 2005. 'Machine Vision: Theory, Algorithms, Practicalities'. *Morgan Kaufmann*

Efford, N 2000, *Digital Image Processing: A Practical Introduction Using Java^{TM}*. Available from: Pearson Education [8 April 2018]

Fabic,JN, Turla,IE, Capacillo,JA, David,LT, Naval,PC, & Jr 2013, *Fish Population Estimation and Species Classification from Underwater Video Sequences using Blob Counting and Shape Analysis*, Department of Computer Science, Marine Science Institute, University of the Philipines, Diliman, Quezon City, Philipines

Kale, A n.d., *World Oceans Day:Think Blue To Go Green,* Research Matters, viewed 4 August <https://researchmatters.in/article/world-oceans-day-think-blue-go-green>

Mallick, S 2015, *Blob Detection Using OpenCV (Python, C++),* Learn OpenCV, viewed 22 February 2018, < http://www.learnopencv.com/blob-detection-using-opencv-python-c/>

Moeslund,T (n.d) *Canny Edge Detection*, Denmark: Laboratory of Computer Vision and Media Technology, Aalborg University, Viewed 4 August <http://www.cvmt.dk/education/teaching/f09/VGIS8/AIP/canny_09gr820.pdf>

Otsu n.d. A *threshold selection method from gray-level histograms*. IEEE Trans Systems, Man and Cybernetics

Poynton, Charles, A, 1998, *Rehabilitation of gamma*, Photonics West'98 Electronic Imaging, International Society for Optics and Photonics

Rudolph, C n.d., *Aquaculture: A local, sustainable protein source,* Food@MSU, viewed 22 February 2018, < http://food.msu.edu/articles/aquaculture-local-sustainable-protein-source>

ScienceDaily*, Fish farming*, n.d, Available from: <https://www.sciencedaily.com/terms/fish_farming.htm> [22 February 2018]

Shwab,K 2016, *The Fourth Industrial Revolution: what it means, how to respond*, World Economic Forum, viewed 4 August 2017, <https://www.weforum.org/agenda/2016/01/the-fourth-industrial-revolution-what-it-means-and-how-to-respond/>

Stokes, M, Anderson, M, Chandrasekar, S, & Motta, R n.d., *A Standard Default Color Space for the Internet – sRGB*. Available from: <https://www.w3.org/Graphics/Color/sRGB.html> [8 April 2018]

Szeliski, R 2010, ‘Computer Vision: Algorithms and Applications’ *Springer Science & Business Media, pp. 10–16.*

Toh,YH, Ng,TM, & Liew,BK 2009, *Automated fish counting using image processing*, National University of Singapore, Republic Polytechnic of Singapore.

Xiao, G, Zhang, W, Zhang, YL, Chen, JJ, Huang SS & Zhu LM 2011, ‘Online Monitoring System of Fish Behavior’ *2011* Paper presented at the *11th International Conference on Control, Automation and Systems,* KINTEX, Gyeonggi-do, Korea

Suzuki, S & Abe,K 1985, *Topological structural analysis of digitized binary images by border following,* Volume 30, Issue 1. Available from: Elsevier

Varah, S 2012, *Video: The Next Frontier for Big Data*, Wired, viewed 22 February 2018, https://www.wired.com/insights/2014/09/video-big-data/

Appendix : Project Code

```
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/videoio.hpp"
#include "opencv2/imgcodecs.hpp"
#include <opencv2/video.hpp>
#include <iostream>
#include <sstream>
using namespace std;
using namespace cv;
int keyboard;
int fishycounter = 0, framecounter = 0;
float calculateSD(float data[]); //calculate standard
deviation
VideoCapture capture("fishybusiness2trim.mp4");
VideoWriter video("fishybusinessoutput.avi", -1,
capture.get(CV_CAP_PROP_FPS),
Size(capture.get(CAP_PROP_FRAME_WIDTH),
capture.get(CAP_PROP_FRAME_HEIGHT)));
int uneven = 0;
Mat src; Mat dst; Mat morph;  Mat diff, diff2; int rows = 0;
int cols = 0; RNG rng(12345);
int main(int argc, char** argv)
{
if (!capture.isOpened()){
//error in opening the video input
cerr << "Unable to open video file: " << endl;
exit(EXIT_FAILURE);
}

while ((char)keyboard != 'q' && (char)keyboard != 27){
//read the current frame
if (!capture.read(src)) {
cerr << "Unable to read next frame." << endl;
cerr << "Exiting..." << endl;
exit(EXIT_FAILURE);
}

dst = src.clone();//calculate 0.4 of the total frame size
rows = src.rows*0.4;
cols = src.cols*0.4;

uneven = cols % 2;
if (uneven == 0){
cols = cols + 1;
}
uneven = rows % 2;
```

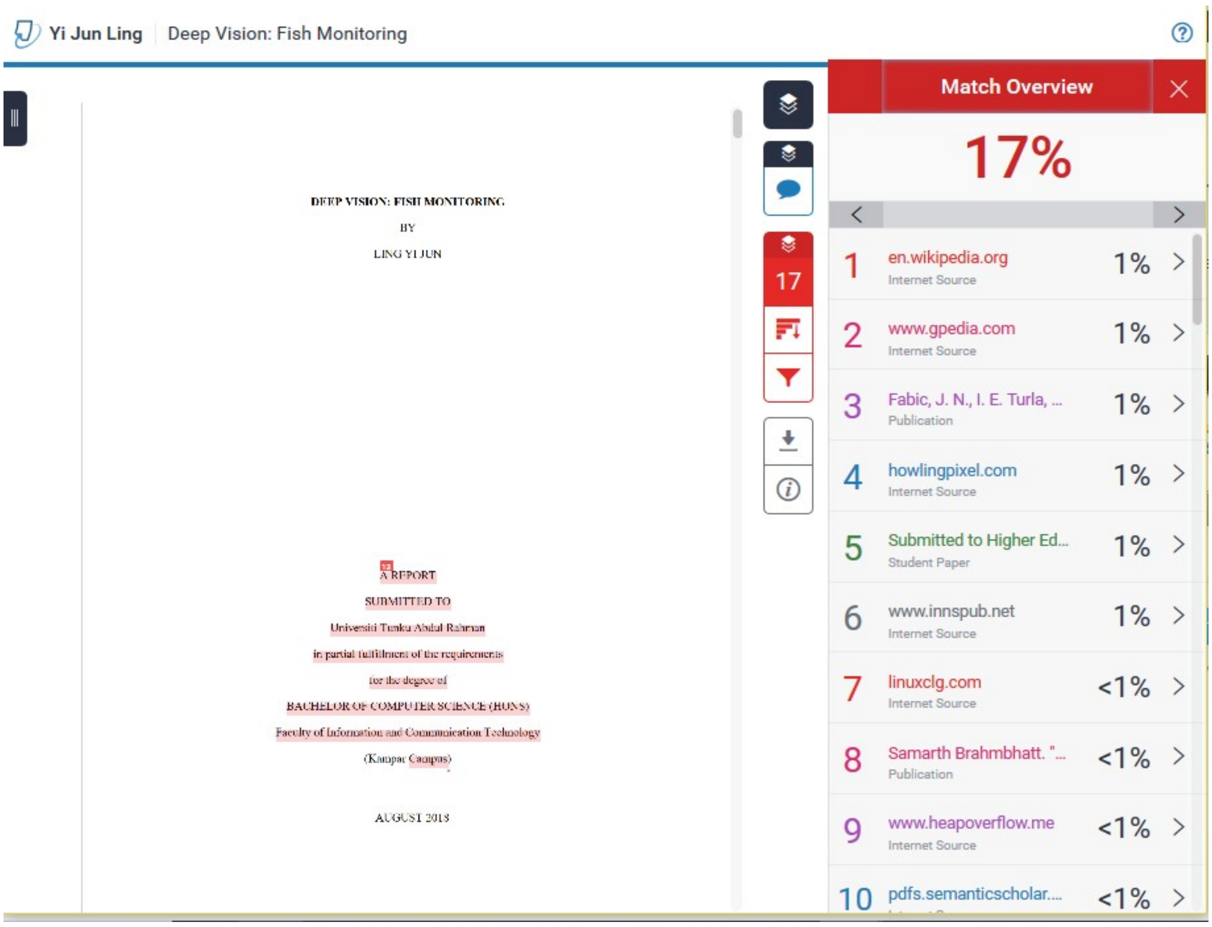
```
if (uneven == 0){
rows = rows + 1;

}

GaussianBlur(src, dst, Size(cols, rows), 0, 0);

diff = dst - src;//subtract image with mask
diff2 = src - dst;

inRange(diff, Scalar(0, 0, 0), Scalar(0, 19, 146), diff);

Mat element = getStructuringElement(2, Size(2 * 2 + 1, 2 * 2 +
1), Point(2, 2));
Mat element2 = getStructuringElement(2, Size(3 * 2 + 3, 2 * 2
+ 1), Point(3, 3));
/// Apply the specified morphology operation
morphologyEx(diff, morph, 3, element);
morphologyEx(morph, morph, 3, element);
morphologyEx(morph, morph, 3, element);

blur(morph, morph, Size(3, 3));
Mat canny_output;
vector<vector<Point> > contours;
vector<Vec4i> hierarchy;
Canny(morph, canny_output, 100, 100 * 2, 3);
findContours(canny_output, contours, hierarchy,
CV_RETR_EXTERNAL, CV_CHAIN_APPROX_NONE, Point(0, 0));
Mat drawing = Mat::zeros(canny_output.size(), CV_8UC3);
int **score = new int*[contours.size()];
for (int i = 0; i < contours.size(); ++i){
score[i] = new int[1];
score[i][0] = 0;
}
Mat hsv;
cvtColor(src, hsv, COLOR_BGR2HSV);
for (int i = 0; i < contours.size(); i++)
{


Point2f points[4];
Scalar color = Scalar(rng.uniform(0, 255), rng.uniform(0, 255),
rng.uniform(0, 255));
drawContours(drawing, contours, i, color, 1, 8, hierarchy, 0,
Point());
RotatedRect rotate_rect;

rotate_rect = minAreaRect(contours[i]);
rotate_rect.points(points);
```

```
//TO DETECT ELONGATED STRUCTURE-------------------------------
--------------------------------------------------------------
--
if (rotate_rect.size.height * 2 <= rotate_rect.size.width |
rotate_rect.size.width * 2 <= rotate_rect.size.height){
//if it is an elongated structure, this will add 1 point to
the contour score
score[i][0] = score[i][0] + 1;

vector< vector< Point> > polylines;
polylines.resize(1);
for (int j = 0; j < 4; ++j)
polylines[0].push_back(points[j]);
cv::polylines(drawing, polylines, true, Scalar(0, 255, 0), 1);

}

//TO DETECT CURVE 1-------------------------------------------
--------------------------------------------------------------
-----
Point ePnt[5];
Point equator1, equator2;
Point cdPnt[5];
double distance1 = norm(points[0] - points[1]);
double distance2 = norm(points[1] - points[2]);
if (distance1<distance2){
equator1 = (points[0] + points[1]) / 2;
equator2 = (points[3] + points[2]) / 2;
}

else if (distance1>distance2){
equator1 = (points[0] + points[3]) / 2;
equator2 = (points[1] + points[2]) / 2;

}
line(drawing, equator1, equator2, 255, 1, 8, 0);
ePnt[2] = (equator1 + equator2) / 2;
ePnt[1] = (equator1 + ePnt[2]) / 2;
ePnt[0] = (ePnt[1] + equator1) / 2;
ePnt[3] = (ePnt[2] + equator2) / 2;
ePnt[4] = (ePnt[3] + equator2) / 2;

for (int j = 0; j < 5; j++){
int counterx = 0;
for (int k = 0; k < contours[i].size(); k++){

Point coordinates = contours[i][k];
float P12 = sqrt(pow(ePnt[j].x - equator1.x, 2) +
pow(ePnt[j].y - equator1.y, 2));
float P13 = sqrt(pow(ePnt[j].x - coordinates.x, 2) +
pow(ePnt[j].y - coordinates.y, 2));
float P23 = sqrt(pow(coordinates.x - equator1.x, 2) +
pow(coordinates.y - equator1.y, 2));
float result = acos((pow(P12, 2) + pow(P13, 2) - pow(P23, 2))
/ (2 * P12*P13))*(180 / (atan(1) * 4));
if (result == 90 && counterx == 0 || (result >= 89 && result
<= 91) && counterx == 0 || (result >= 84 && result <= 95) &&
counterx == 0){
cdPnt[j] = coordinates;
counterx = 1;
}
if ((result >= 84 && result <= 95) && counterx != 0 &&
coordinates.y > cdPnt[j].y){
cdPnt[j] = coordinates;
}
}



}

float data[5];

for (int i = 0; i < 5; i++){
data[i] = norm(ePnt[i] - cdPnt[i]);

}
//cout << "Standard Deviaation: " << calculateSD(data) << endl;

if (calculateSD(data) >= 2 && calculateSD(data) <= 50){

for (int j = 0; j < 5; j++){

line(drawing, ePnt[j], cdPnt[j], 255, 1, 8, 0);
}
score[i][0] = score[i][0] + 1;
}

//TO DETECT ANGLE-----------------------------------------
--------------------------------------------------------------
-----

float P12 = sqrt(pow(cdPnt[2].x - equator2.x, 2) +
pow(cdPnt[2].y - equator2.y, 2));
float P13 = sqrt(pow(cdPnt[2].x - equator1.x, 2) +
pow(cdPnt[2].y - equator1.y, 2));
```

```
float P23 = sqrt(pow(equator1.x - equator2.x, 2) +
pow(equator1.y - equator2.y, 2));


float result = acos((pow(P12, 2) + pow(P13, 2) - pow(P23, 2))
/ (2 * P12*P13))*(180 / (atan(1) * 4));

//since if it is a straight line, the angle would be closer to
180 degrees, for it to not be a straight line, and for it to
be a curve,
//the angle would have to be less than 180, therefore, to make
sure it does not appear to be a straight line
//anything less than 170 is considered a straight line and is
omitted,
//OR
//for the fish body, it has a shape where there is some
content in the middle, and also a downwards curve therefore
the angle should not be less than 170 degrees
if (result <= 165 && result >= 90){
score[i][0] = score[i][0] + 1;


line(drawing, equator1, equator2, 255, 1, 8, 0);
line(drawing, equator1, cdPnt[2], Scalar(0, 0, 255), 1, 8, 0);
line(drawing, equator2, cdPnt[2], Scalar(0, 0, 255), 1, 8, 0);
}



//TO DETECT CURVE 2----------------------------------------
-------------------------------------------------------------
-----
Point cuPnt[5];
for (int j = 0; j < 5; j++){
int counterx = 0;
for (int k = 0; k < contours[i].size(); k++){

Point coordinates = contours[i][k];
float P12 = sqrt(pow(ePnt[j].x - equator1.x, 2) +
pow(ePnt[j].y - equator1.y, 2));
float P13 = sqrt(pow(ePnt[j].x - coordinates.x, 2) +
pow(ePnt[j].y - coordinates.y, 2));
float P23 = sqrt(pow(coordinates.x - equator1.x, 2) +
pow(coordinates.y - equator1.y, 2));
float result = acos((pow(P12, 2) + pow(P13, 2) - pow(P23, 2))
/ (2 * P12*P13))*(180 / (atan(1) * 4));
if (result == 90 && counterx == 0 || (result >= 89 && result
<= 91) && counterx == 0 || (result >= 84 && result <= 95) &&
counterx == 0){
```

```
cuPnt[j] = coordinates;
//cout << "Contour" << i << ", line number " << j << " :" <<
coordinates.x << "  " << coordinates.y << endl;
counterx = 1;
}
if ((result >= 84 && result <= 95) && counterx != 0 &&
coordinates.y < cuPnt[j].y){
cuPnt[j] = coordinates;
}
}

}

for (int i = 0; i < 5; i++){
data[i] = norm(ePnt[i] - cuPnt[i]);

}
//cout << "Standard Deviaation: " << calculateSD(data) << endl;

if (calculateSD(data) >= 2 && calculateSD(data) <= 50){

for (int j = 0; j < 5; j++){

line(drawing, ePnt[j], cuPnt[j], 255, 1, 8, 0);
}
score[i][0] = score[i][0] + 1;
}

//AVERAGE DIFFERENCED PIXEL INTENSITY------------------------
-------------------------------------------------------------
-----------------------------

// matrices we'll use
Mat M, rotated, cropped;
// get angle and size from the bounding box
float angle = rotate_rect.angle;
Size rect_size = rotate_rect.size;
if (rotate_rect.angle < -45.) {
angle += 90.0;
swap(rect_size.width, rect_size.height);
}

// get the rotation matrix
M = getRotationMatrix2D(rotate_rect.center, angle, 1.0);
// perform the affine transformation
warpAffine(hsv, rotated, M, src.size(), INTER_CUBIC);
// crop the resulting image
getRectSubPix(rotated, rect_size, rotate_rect.center, cropped);
Scalar m = mean(cropped);
Scalar s = mean(hsv);

float croppedmean = (m[0] + m[1] + m[2]) / 3;
float srcmean = (s[0] + s[1] + s[2]) / 3;
float differenced = abs(croppedmean - srcmean);


if (differenced >= 20){

score[i][0] = score[i][0] + 1;
}
//SIZE--------------------------------------------------------------
-----
if ((rotate_rect.size.height*rotate_rect.size.width) > 200 &&
(rotate_rect.size.height*rotate_rect.size.width) <20000){

score[i][0] = score[i][0] + 1;
}
//OUTPUT RESULTS----------------------------------------------------
-----

//cout << "Contour" << i << " :" << score[i][0] << endl;

if (score[i][0] >= 4){
vector< vector< Point> > polylines;
polylines.resize(1);
for (int j = 0; j < 4; ++j)
polylines[0].push_back(points[j]);
cv::polylines(drawing, polylines, true, Scalar(255, 255, 255),
1);
cv::polylines(src, polylines, true, Scalar(255, 255, 255), 1);

fishycounter = fishycounter + 1;
if (framecounter % 5 == 0){
cout << "Fish " << fishycounter << " Length & Width :" <<
rotate_rect.size.height << ", " << rotate_rect.size.width <<
endl;
}
}

}

if (framecounter % 5 == 0){
cout << "Number of fish: " << fishycounter << endl;
}
fishycounter = 0;
```

```
imshow("original", src);
//video.write(src);
framecounter = framecounter + 1;
//imwrite("output.jpg", src);

keyboard = waitKey(30);
}
video.release();
capture.release();
}
float calculateSD(float data[])
{
float sum = 0.0, mean, standardDeviation = 0.0;

int i;

for (i = 0; i < 5; ++i)
{
sum += data[i];
}

mean = sum / 5;

for (i = 0; i < 5; ++i){
standardDeviation += pow(data[i] - mean, 2);
}

return sqrt(standardDeviation / 5);
}
```



Yi Jun Ling | Deep Vision: Fish Monitoring

DEEP VISION: FISH MONITORING

BY

LING YI JUN

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONS)

Faculty of Information and Communication Technology

(Kampar Campus)

AUGUST 2018

Match Overview

17%

1	en.wikipedia.org Internet Source	1%
2	www.gpedia.com Internet Source	1%
3	Fabic, J. N., I. E. Turla, ... Publication	1%
4	howlingpixel.com Internet Source	1%
5	Submitted to Higher Ed... Student Paper	1%
6	www.innspub.net Internet Source	1%
7	linuxclg.com Internet Source	<1%
8	Samarth Brahmbhatt. "... Publication	<1%
9	www.heapoverflow.me Internet Source	<1%
10	pdfs.semanticscholar....	<1%

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1of 1

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	
ID Number(s)	
Programme / Course	
Title of Final Year Project	

Similarity	**Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)**
Overall similarity index:_________ % **Similarity by source** Internet Sources:_______________% Publications: _______________ % Student Papers:_______________ %	
Number of individual sources listed of more than 3% similarity: _________	

Parameters of originality required and limits approved by UTAR are as Follows:
(i) **Overall similarity index is 20% and below, and**
(ii) **Matching of individual sources listed must be less than 3% each, and**
(iii) **Matching texts in continuous block must not exceed 8 words**
Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

______________________________	______________________________
Signature of Supervisor	Signature of Co-Supervisor
Name: __________________________	Name: __________________________
Date: ___________________________	Date: ___________________________

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	
Student Name	
Supervisor Name	

TICK (√)	**DOCUMENT ITEMS** Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
	Front Cover
	Signed Report Status Declaration Form
	Title Page
	Signed form of the Declaration of Originality
	Acknowledgement
	Abstract
	Table of Contents
	List of Figures (if applicable)
	List of Tables (if applicable)
	List of Symbols (if applicable)
	List of Abbreviations (if applicable)
	Chapters / Content
	Bibliography (or References)
	All references in bibliography are cited in the thesis, especially in the chapter of literature review
	Appendices (if applicable)
	Poster
	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report. ________________________ (Signature of Student) Date:	Supervisor verification. Report with incorrect format can get 5 mark (1 grade) reduction. ________________________ (Signature of Supervisor) Date: