Investigation into underwater object recognition and tracking for the SAUC-E competition

Jiadi Yao

Bachelor of Science in Computer Science with Honours
The University of Bath
April 2008

Investigation into underwater object recognition and tracking for the SAUC-E competition

Submitted by: Jiadi Yao

COPYRIGHT

Declaration

This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of Batchelor of Science in the Department of Computer Science. No portion of the work in this dissertation has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning. Except where specifcally acknowledged, it is the work of the author.

Signed:

Abstract

In this project, a vision based autonomous underwater vehicle (AUV) navigation system is proposed and its performance demonstrated in a mock-up mission in the SAUC-E competition. The proposed system is used to determine the colour, shape and flash rate of underwater objects and lights. The system tracks them in 2 dimensional space and reports their location. During the evaluation test, the proposed system reached approximately 90% accuracy of the recognised shapes and could track flash beacon accurately when it is off. The proposed system can be used as a solution to the problem of underwater recognition and tracking.

Acknowledgements

I would like to thank my tutor Dr. John Collomosse for all the support and excellent advice throughout the duration of this project. Without his advice and technical insight the completion of this project would not have been possible.

Also thanks must go to all those who helped in gathering the crucial test footage for this project which I could not have done alone. The members of BURST for preparing the AUV, Megill William and Benjamin Williamson for their help in setting out the objects and holding the camera.

Thanks to my girlfriend for the production of my test shapes. Finally I would like to thank my family for helping me get here in the first place.

Contents

Chapter 1

Introduction

Every year, a team of students from the University of Bath enters the Student Autonomous Underwater Challenge - Europe (SAUC-E). SAUC-E is a competition organised by the Defence Science and Technology Laboratory (DSTL) and designed to advance Autonomous Underwater Vehicle (AUV) technology. The competition involves designing and building an AUV that performs a set of missions underwater. It requires a multidisciplinary team to create a fully functioning AUV. These are are the three major tasks for this year:

1. Find a sphere shaped mid-water buoy. The target buoy is of distinct colour. The AUV should make contact with it and avoid a same shaped but differently coloured dummy buoy.

2. Find a circular target situated on the bottom of the tank. It is 1 meter in diameter and has a contrasting 'cross' at its centre. The target is illuminated by several white LEDs flashing at 1Hz to enable the AUV to locate it from a distance. The AUV should drop markers (non-chemical) on to the 'cross', the nearer to the centre; the more credits are awarded.

3. Surface within a designated surfacing zone. The surfacing zone is a 3-meter square pipework located on the surface. There are a pair of co-located items - either orange cones or black tyres - on the floor beneath the surface zone amid are other objects that are distributed randomly in the tank.

Autonomous Underwater Vehicle – a robot, designed to complete specific tasks underwater. Currently, AUVs are used by military for mine hunting, anti submarine; by energy company for deep ocean oil, gas, mineral field survey; by telecommunication company for cable route survey, post lay inspections; and by scientist for oceanographic surveys. Due to the nature of those applications, the need for the close range data are limited. Therefore, sonar, inertial, underwater telemetry constitute their navigation system. To achieve a reasonable accuracy using those sensors for completing the competition, the sensors are needed to be sophisticated and can be very expensive.

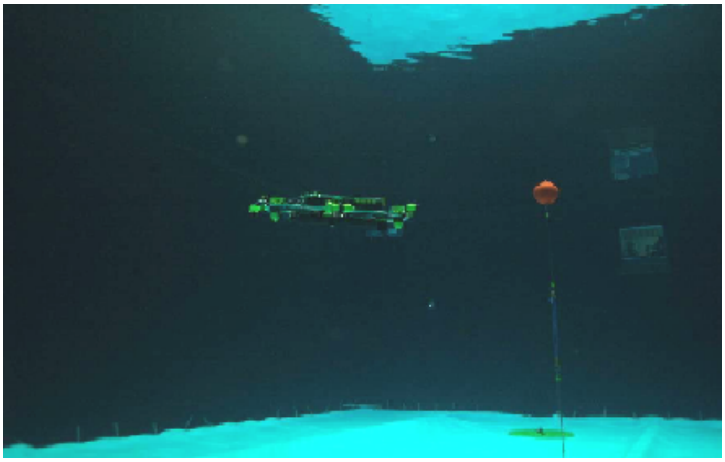


Figure 1.1: AUV approaching the mid-water buoy in the SAUC-E competition

Visual data carries rich information about the environment. With advanced vision algorithms, visual data alone can provide enough navigation information to the AUV. That is why human and most autonomous land and space systems use visual data. Using vision to tackle competition tasks are the common approach for the past SAUC-E entrants. It has demonstrated effectiveness in the past competitions. However, the special effects introduced in underwater environment, breaks the assumptions made by vision processing algorithms. Many algorithms available for land or space system cannot be applied directly for underwater operations.

This project is aiming at developing a computer vision system, which is able to recognise and track the objects observed by a underwater camera in real-time. The existing popular object recognition and object tracking algorithms are implemented then evaluated in underwater environment. The system is delivered to integrate with the Artificial Intelligence (AI) of the AUV to enter the competition in July 2008. Considering the competition is going to carry out in a constrained environment, and there are only a limited amount of objects in the arena, it makes sense to take advantage of those predictable features to assist the vision system, thereby improving the accuracy and speed of the algorithms used by the vision system.

1.1 Problem Statement

To investigate the efficacy and performance of algorithms for object recognition and tracking in an underwater environment. This will involve the evaluation and implementation of algorithms that run in real time and overcome challenges unique to the submarine environment to enable a robot system to carry out unmanned underwater missions.

1.2 Objectives

- Understand computer vision broadly to identify the general problems involved in the real-time underwater computer vision domain.
- Investigate and implement algorithms that are advantageous to the domain. Evaluate implemented algorithms under various conditions. By considering the specific task requirement of the competition, make decision on the set of algorithms best suited for the competition.
- Implement a complete system that can be integrated with the AI component of the AUV, enabling the AUV to complete the competition tasks underwater.

1.3 Structure

The remaining document is organised as follows

- Chapter 2: In order to identify the problems involved, and to recognise useful algorithms or techniques from a huge variety of sources, a literature survey is conducted. Previous entrants' journals provide a start point, the techniques they mentioned and used in their AUV are discussed; the problems they experienced are noted. Many popular algorithms and their suitability in underwater application are discussed too.
- Chapter 3: The outline of the requirement of this vision system for the AUV, the planning of the works for the completion of this project.
- Chapter 4: The algorithms to isolate and track target from the input video stream are discussed. Different challenges arise when locating and tracking different kinds of target. For example, tracking buoy, which always be there; tracking flash beacon, which has on-off cycle. Tracker needs to predict beacon position when it is off.
- Chapter 5: With the target object segmented out using techniques discussed in the previous chapter, the shape classification is discussed in this chapter. This chapter moves from the simple 2D shape (e.g. cross, circle), which can be recognised using classic vision method; to the more difficult 3D object (e.g. cone, tyre), which more recent SURF feature detector are used.
- Chapter 6: Evaluation of different techniques implemented. The free variables are identified. The performance and reliability of each technique are compared and contrasted under the changing of those variables.
- Chapter 7 concludes the dissertation, gives the decision of the techniques included in the system, the strengths, weaknesses and the future work that can be done to improve the system as well as the steps need to take the project towards to the competition in the July.

Chapter 2

Literature Review

Computer Vision is an active research area because there is a such wide range of applications varying from Automatic Visual Inspection System, Biomedical Imaging Techniques, Defence surveillance and of course Robot vision [Acharya and Ray, 2005]. It receives attentions and contributions from the community of academics and professionals. There is vast amount of literatures available.

The purpose of this review is to isolate from the huge variety of techniques, those which might produce the best results for this project and the tasks ahead.

There is a requirement for all entrants to the SAUC-E competition to submit a journal detailing all of the techniques employed in their submission, so there is also a wealth of information about the vision components of previous entrants.

2.1 Introduction

From the tasks described in the section 1, it is necessary to decide the main topics that are relevant to be reviewed. Image pre-processing is to remove noise, correct colours, enhance features. They can make later recognition algorithms more accurate. They are studied in section 2.2. Task 1 mainly tests the AUV's ability to distinguish between colours, recognise shapes and tracking in the underwater environment. Colour space and colour description are studied in section 2.2.2, shape classification are in section 2.3.1 and tracking techniques are in section 2.4 In task 2, apart from the techniques already mentioned, accurately identify the centre of the cross is the key to this task. After the AUV is above the floor target, we could slow down the AUV's movement to a higher degree of accuracy, both improves the movement to fine tune the position before dropping the marker and allows more time for the recognition algorithms to run. Using more sophisticated algorithm and performing multiple recognition algorithm become feasible in this task. Task 3 is about 3D object recognition, to test whether the vision system is capable of recognise underwater 3D objects. Because 3D objects viewed from a different angle may give a completely different image, the more

recent SIFT, SURF feature detector can potentially be used to find similar features from the images, and classify them. In section 2.3.1 SIFT and SURF features are studied.

2.2 Image pre-processing

Image pre-processing is the operations on the images at the lowest level of abstraction, the image is looked at as a matrix of image colour values, there is no semantic operations at all in this process. It is necessary to realise that pre-processing does not increase image information content, and from information entropy point of view, any pre-processing decreases image information [Sonka et al., 1999]. The main purpose of this pre-processing is to suppress information that irrelevant to the specific image processing task, therefore enhances the image features that are important. In my application, I would like to remove noise, while keeping sharp edge information; correct the colour of the image and make the colour easy to extract later.

2.2.1 Noise reduction techniques

This project is focused on the underwater image processing, the images acquired underwater will certainly contain noise. Noise can vary from random floating dirt in the water to camera's sensor generated noise due to low light conditions [Putnam, 1998], these type of noise tend to appear as random high frequency speck in the image, therefore we deal with them using filters. Looking at this particular application, it does not rely on high level of details extracted from an image, and the image itself is generally not that noisy (by watching past video clips captured by the AUV [SONIA, 2007]), noise reduction should not be the focus point nor spend too much computation time on. So I will discuss on some general techniques and popular ones used in the past competitions.

Linear Filters

A linear filter applies a linear operator – a window to each pixel of an image, replacing the centre pixel value with the value calculated from the window operator. They are often used to eliminate unwanted frequencies (high frequency noise in this case) from an input signal, and they tend to be fast. There are a wide range of linear filters. Low-pass (averaging) filter is one of the simplest linear filtering technique. Table 2.1 shows how its window look like. It is taking the average value of the centre pixel and 8 pixel around it, replace the centre

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Table 2.1: 3x3 Averaging filter

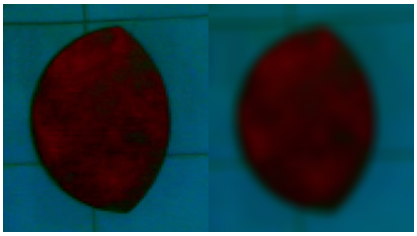


Figure 2.1: Left: the underwater shape; right: the image applied with 3x3 averaging filter. Although the colour is more smooth, the edge of the shape is severely blurred.

1/25	3/25	1/25
3/25	9/25	3/25
1/25	3/25	1/25

Table 2.2: 3x3 Gaussian filter

pixel with the average value calculated. Figure 2.1 is applied with an averaging filter There is nothing more than a simply blur the image by some factor. The biggest criticism is that the noise may have been filtered, but important informations like edges are blurred too because themselves are high frequency components of the image. This can cause problem in the later processing which depend on the strong edge in the image. Similar to the averaging filter discussed above, Gaussian filter is the same process but with a different window weights. Rather than using the average of the values worked out from the window, which does not take into account the diminishing influence of the pixels that are situated in the increasing distance from the central pixel, make the centre pixel contributes more to the final value, and further away the pixel, less influence it has to the final value. Table 2.2 is one example of the Gaussian filter. The advantage this over the simple averaging is, it preserves more edge information while removes the noise, and there are more variables to play with to fine tune the filter for my particular application. Although preserving strong edge information is one of the main point I need to consider, but I also need to think about the trade off between the speed of the filter and the actual affect on the later processing of losing some of the edge information. Since the implementation of these linear filters are simple, it is perfectly feasible to include them in the evaluation of different filtering techniques.

Median Filter

From a complete different angle viewing the problem, median filter is a non-linear filter, based on probability theory[1] [Sonka et al., 1999]. The idea is to replace the current pixel

[1]For a random variable x, the median M is the value for which the probability of the outcome $x < M$ is 0.5 .

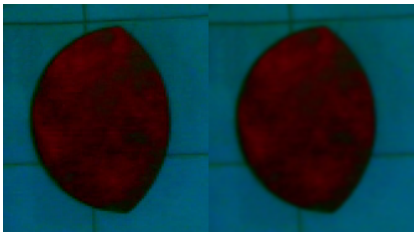


Figure 2.2: Left: underwater image; right: image applied with 3x3 Guassian filter in table 2.2. It preserves the edge information while removes some noise

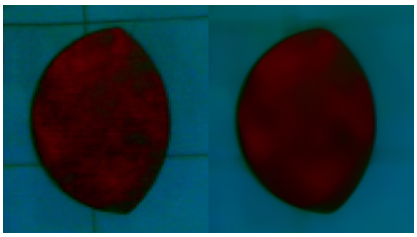


Figure 2.3: Left:underwater image; right: image applied with median filter of size 7. The noise on the shape is mostly remove while the edge of the shape unchanged.

in the image by the median[2] of the value in its neighbourhood (window), because we think the median is more likely to be value of the current pixel. Since the median of the values in its neighbours are not affected by individual noise spikes (lowest and highest values are sorted towards lower or higher end, which will not be picked), so it eliminates noise quite well. Figure 2.3 demonstrated this. But we can equally see the potential problem. When the noise level over half of the source image, even the median value can very likely be the noise and been picked, it can potentially destroy the information in the image. Another drawback is that it wipes out structures that are of the size of the filter window, and this effect causes the texture of a filtered image to be strongly distorted or removed. Fortunately, in this particular project, which acquires image underwater, it is highly unlikely to use the texture of certain object to do the recognition. [Dubel et al., 2005](The winning team of AUVSI in 2005) even uses this property of the filter to smooth out the unwanted texture of the object. One distinctive advantage of the median filter is it preserves high frequency boundary informations very well [Nixon and Aguado, 2002, Sonka et al., 1999]. The computation power needed for the non-linear filter can be expensive, especially if the window chosen is not small [Russ, 1996]. Due to its advantage of keeping the boundary information, and it has been quite popular amount the past SAUC-E entrants [Wallis, 2006], I would definitely compare and evaluate it with other filters described here.

[2]Sort the numbers in increasing order, the middle one is the median

Other Non-linear Filters

Median filter really is just the beginning of the non-linear filters, there are so many other non-linear filters, Vector Median Filter (VMF) [Y., Apr 1990], Arithmetic Mean Filter (AMF) [Plataniotis and Venetsanopoulos, 2001], Hybrid Directional Filter (HDF) [Gabbouj and Cheikh, 1995], Basic Vector Directional Filter (BVDF) [Trahanias and Venetsanopoulos, 1993], Self-Avoiding Walk model (SAW) [Smolka, 2003] are just name a few. They are quite comprehensive and demonstrated in their paper how the method filters a image corrupted with mixed Gaussian ($sigma = 60$) and impulsive noise (10% on each channel) and recovers the image back to an acceptable state [Smolka, 2003]. Unfortunately, these filters require multiple runs (5-10 runs) to achieve such effect and each run takes order of seconds to complete, bearing mind that was just on one single image. This make the real-time application totally infeasible, so it is unlikely they will be employed in this vision system.

2.2.2 Colour and Image Enhancement

Colour has been used as one of the main techniques to extract features from the image among the past SAUC-E entrants [D.Ribas et al., 2006, Assalih et al., 2007], and its use ranges from serving as a small part of the solution such as an initial segmentation technique and extends to being used as a classifier in its own right[Johnston, 2004]. One of this year's task expect the AUV to distinguish between two coloured sphere in the water, and make contact with the target sphere. Since they are both spheres, there is no more information (e.g. shape, pattern) apart from colour difference, so we have to totally rely on colour recognition for this task.

Colour correction of images is a very rich and complicated field, and generally it requires detailed knowledges of the light source and the camera response, which must be obtained with calibration standards [Russ, 1996]. There is a more routine interest in making some adjustments to colour images to permit comparisons between images.

Due to water's limited ability to conduct light [Schechner and Karpel, 2004], from the swimming or diving experience, seeing things underwater gives much limited range compare to in the air. A method of extend visibility range underwater is described by Schechner and Karpel [2004], they proved the visibility degradation is caused by the partial polarisation of the light, and developed an algorithm that inverts the image formation process to recover a good visibility image of the object. Unfortunately, it requires images taken through a polariser at different orientations, which in this project, having an extra mechanical system to turn a polariser in front of the camera, to get the extra little visibility, is too much complication, and quite unlikely.

Colour description and Colour Space

In computer graphics, one of the most generally used colour models is RGB (Red,Green,Blue) colour model. Each colour appears in its primary spectral components of red, green, and blue, a image is then represented by three component images, one for each primary colour. And there are other colour models using different primary colours like CMYK(cyan, magenta, yellow, and black), simple arithmetic formula allows conversion between them. However, this is not suited for *describing* colours in terms that are practical for human interpretation [Gonzalez and Woods, 2002]. For example, we do not think of colour pixels are as being composed of three primary colour that combined to form that single pixel. Instead, we describe its hue, saturation and brightness. Hue is a colour attribute that describes a pure colour (pure yellow, orange, or red), whereas saturation gives a measurement of the degree to which a pure colour is diluted by white light; brightness is the perception of how much light is coming from a source. This model is called HSV (Hue, Saturation, Intensity) colour model, it decouples the intensity component from the colour-carrying information (Hue and saturation) in a colour image. The draw back is, when setting the intensity to maximum, white colour is obtained and setting to lowest intensity, black is obtained regardless of what Hue and Saturation value took, i.e. any Hue and Saturation can give black or white. This is a serious ambiguity of representation, but it is irrelevant in my application as the objective is to ignore the intensity component by either set it to a constant or simply drop it. The HSV model is an appropriate candidate and therefore will be considered in the implementation.

Histogram Processing

Histogram of an image represents the relative frequency of occurrence of the various grey levels in the image. In a poorly contrasted image, a large number of pixels occupy only a small portion of the available range of intensities. Through histogram modification we reassign each pixel with a new intensity value so that the dynamic range of grey levels is increased. A popular technique to stretch the range of intensities include histogram normalisation. The original histogram is stretched, and shifted, to cover all 256 available levels.

$$N_{x,y} = \frac{N_{max} - N_{min}}{O_{max} - O_{min}} * (O_{x,y} - O_{min}) + N_{min} \ \ \forall x, y \in 1, N \tag{2.1}$$

Equation 2.1 is a straight forward process to do normalisation. Another use of histogram is determine the optimal thresholding value, which is discussed in section 2.2.4.

2.2.3 Edge detection

Edge detection is a classic technique for low-level image processing. The high frequency edge information will be highlighted after the process. The whole process is the opposite of the Linear noise filtering, therefore it is possible to pass a different kind of window across the image that would allow removing of the low frequency signal. High frequency signal

in the image can also be recognised as a sudden rise in pixel intensity, taking the first derivative of the rise would give us a peak at the position where it rose, as in Figure 2.4. Popular edge detectors based on single derivative are Robert Cross, Sobel operator, Prewitt

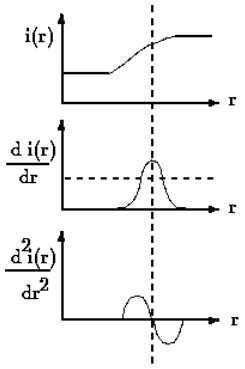


Figure 2.4: Zero, First and Second derivative

operator and Canny operator, they will be discussed below.

Edge detection operator	Window
Roberts Cross	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ $\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$
Prewitt operator	$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$ $\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$
Sobel operator	$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$ $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

Table 2.3: Template windows

The Prewitt edge detection operator uses two 3 by 3 template window Table 2.2.3, which are used to calculate an image of edge strength in the horizontal direction and an image of edge strength in the vertical direction respectively. To produce the edge magnitude from an image, the process of template convolution is used. The value of the centre pixel been put over by the window, is updated by the value result from the window. Both window are used over the entire image to obtain the complete edge information. A Thresholding (discussed in section 2.2.4) can be applied here to identify the strongest edge pixels.

The other significant template based edge detection operators are the Roberts cross operator and the Sobel operator. The template window used for the convolution of these operators are also shown in table 2.4. The Sobel operator has two 3 by 3 template windows, for working out edge magnitude in the horizontal and vertical direction respectively (in the same manor as the Prewitt operator).

The Roberts cross operator has two 2 by 2 template windows, which are applied in an equivalent way to Sobel and Prewitt operators but instead of working out edge magnitude in horizontal and vertical directions, they work out edge magnitude in each of the two respective diagonal directions. The complexity of all the template based approaches to edge detection is very low, as they only involve addition and subtraction. The Roberts cross operator is the lowest complexity of the three as it has the smallest template window but as it uses very few pixels to calculate the gradient it is very sensitive to image noise and only identifies sharp edges. The Prewitt and Sobel operators have larger template window than the Roberts cross operator and are therefore more robust to image noise.

Canny's edge detector was designed to the following three objectives [Canny, 1986]

1. optimal detection with no surprise response;
2. good localisation with minimal distance between detected and true edge position;
3. single response to eliminate multiple responses to a single edge.

The detector includes the Gaussian filter that smooths the image, to achieve it's first goal; the second criterion achieved by a process of non-maxima suppression that only detect true peaks, results a thin lines of edge points, in the correct place; the third constraint concerns location of a single edge point in response to a change in brightness. This is because more than one edge can be denoted to be present.

That was a few choices of different edge detectors, but the techniques we go on to discuss are not deeply affected by which edge operator is used, and speed advantages are unlikely to cause issue when the operator is only used once per frame, as long as the operation is not too involved. Therefore any of the simpler, well known edge detecting methods will suffice.

2.2.4 Segmentation

Thresholding

Thresholding is the simplest segmentation process. For example, target in image may have higher intensity than the background, a binary thresholding on a intensity value that would separate out the target can be used. Region that have higher than the threshold value are assigned to 1, others are assigned to 0. This process is a global thresholding, entire image is threshold on one single threshold value. Only under very unusual circumstances a single threshold for the whole image would success, and it also requires prior knowledge about

the image. Threshold using variable thresholds, in which the threshold value varies over the image depend on local image characteristics. The image is divided in to sub-images, where each threshold value is determined. This is adaptive thresholding [Sonka et al., 1999]. Thresholding can also be a range rather than a point, where if the value within the threshold range, it is picked. Hence multiple thresholding range is also possible, after which the image is reduced to a limited set of grey levels. But the multiple ranged thresholding hardly find any use in this particular project, as no instant during the task is required to segment out more than one target. Thresholding does not limited to the colour, or brightness value of each pixel, it can also be applied on distances between two features which can be used in classification(discussed in detail in section 2.3.2.

Threshold value can also be detected instead of using human vision to assist. An intuitive example would be p-tile thresholding [Sonka et al., 1999], which segments out text from a printed text sheet, because it already knows the rough percentage of the text (black colour) covering the sheet. This allows the thresholding value to be chosen automatically. But unfortunately our AUV will be moving through the water, there is no such defined information.

More complex methods of threshold detection are based on histogram shape analysis. If an image consists of objects of approximately the same grey-level that differs from the grey-level of the background, the resulting histogram is bi-modal [Sonka et al., 1999]. Pixel of objects form one peak, background form another. So the threshold value to separate them is a point between the two peaks. Optimal Threshold is a implementation of this idea, it based on approximation of the histogram of an image using weighted sum of two or more probability densities with normal distribution. The threshold value is set as the closest grey-level corresponding to the minimum probability between the maxima of two or more normal distributions, which results in minimum error segmentation. The algorithm is relatively simple:

1. Choose an initial threshold T, partition the image using T into two regions-background and foreground;
2. Compute the mean value for background μ_b and foreground μ_f;
3. New threshold T is $\frac{\mu_b+\mu_f}{2}$;
4. Repeat 2 and 3 until T do not change, the T is determined threshold value.

That normally takes no more than 10 iterations [Nixon and Aguado, 2002]. The advantage of this optimal threshold is clear – no manual input of the threshold value.

Hierarchical data structure thresholding [Tanimoto and Pavlidis, 1975] is another very interesting idea. It is based on the local threshold method mentioned above. The aim is to detect the presence of a region in a low-resolution image, then by only re-segment the boundary area in lower resolution image, to work out boundary of higher to full resolution image. The lower resolution image is worked out by apply different sized window of low-pass filter described in section 2.2.1. One distinctive advantage this may have in my project

is it extremely tolerance to noise, since segmentation at the lower resolution are based on smoothed image data, in which noise is suppressed. The imprecise borders that result from segmenting smoothed data are corrected by re-segmentation in areas close to borders using one-step-higher-resolution data [Hartley, 1982].

Background Subtraction

Another common method of detecting object is to subtract the background from a image. Although this can be useful for certain tasks, it requires that the object is moving throughout the sequence while the background remains still. Neither of these assumptions are true in my project, the camera will be moving while the targets are static. There is no background as such that can be used to subtracted.

2.3 Object Recognition

In previous section, the focus was on the low level image operations that remove noise, correct colour, segment out edge and region that interest to us. There is no assumption of what is in the image, hence low-level. This section discusses the semantic operation and analysis on the image to extract features that allow us to describe and discriminate between different classes of object, this is feature extraction; plot the description of the object in a high dimensional feature-space, then classify the object by measuring the distance to each class of object in the feature-space that has already been trained or defined, this is object classification. They together forms the infrastructure for object recognition.

2.3.1 Feature extraction

This is the process concerns finding the *features* in a image that allow us distinguish easily between different objects by only comparing those selected features. The selected features should be invariant to a set of conditions that the object could be observed in, so the classifier can always decide an object correctly within the set of invariant properties defined. The features' *invariant properties* are the keys to consider features. For example, use Fourier Descriptor as feature, then the object would be rotation, scale and shift invariant as the descriptor gives a same description when the object is rotated, scaled or wherever the object appears on the image. But it would be projection variant. Below I went through each task, analysed and decided the invariant property that is important for that task, then discussed some potential techniques that meets the requirement, finally decide a few algorithm for implementation.

1. Task 1 requires to find and hit a mid-water buoy, avoid a different coloured decoy buoy. Be able to recognise colour is the key to this task. We not only need to distinguish between target and decoy buoys' colour, but also to recognise the those

colour at different distances. Colour changes a lot when distance varies in underwater environment. They tend to become darker and fade as viewed further away. It has already discussed in section 2.2.2, HSV colour model puts intensity in a separate channel. So it is possible to achieve some degree of intensity invariance by dropping the intensity channel using this model. Colour degrading as distance increase is difficult to measure, it depends on the water the AUV is in and more importantly the ambient lighting condition. To model how different colour fades as distance increase and how ambiant light affect it could be a research on its own. The author have to leave it as an inaccuracy or noise in the measurement and let the classifier to handle it. To make sure the colour blob found is the buoy, it can be useful to verify the blob's shape [Altshuler et al., 2004]. Contour and region-based shape descriptor can be used at this stage.

2. Task 2 is locating a flash beacon, recognise a cross beneath it and drop a marker on the centre of the cross. One simplest approach was offered by Duck University Johnston [2004], they measure the amount of cyan (beacon colour) in every frame, if it meets the threshold, the flash is present otherwise is not. University of Bath's 2006 entry Wallis [2006] used a similar idea, instead of measuring cyan, the intensity is measured and threshold to find the flash in the image. Although the frequency of the beacon is given, from the previous experience of the competition, there is no any other beacon flashing at different rate, therefore it is not worth to detect the exact frequency of the beacon as it show not be able to perform in real-time Wallis [2006]. Once the beacon is discovered, the AUV tracks the beacon (discussed in section 2.4) and navigate to it to find the cross on the floor. Contour and region-based shape descriptor can be used here. To recognise shape, *scale, shift, rotation and projection invariance* are to consider. Depend on how far from AUV to the cross, the scale of the cross changes; the cross could appear at different locations on the observed image and moves around on the image, so shift invariance is needed; depending on the direction the AUV approaches, the cross may rotated; and it is also good to have certain degree of projection invariance as the cross may viewed at an angle.

3. Task 3 is to find two collocated orange cones or black tyres, surface when above them. *Scale, shift, projection invariance* should be considered. The projection invarience needs to be considered more seriously in this task because the cone and the tyre are 3D objects, the observed image can be completely different when viewed from different angle. Finding the similarities between those different views of a same object is the key in this task.

Chain Codes

One simple way to represent an object in a binary image is by Chain Codes. Each boundary pixel of an object must have at least one adjacent neighbouring boundary pixel such that the direction of the next boundary pixel from the current one is specified by a unique number between 0 and 7, each number represents a possible directions as shown in Figure 2.5.

Positional information is omitted from the code itself, making chain codes shift invariant.

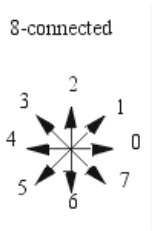


Figure 2.5: Chain Codes

Also rotation invariance can be introduced by using relative description chain code. However while chain codes are simple to create from a segmented image, they are particularly sensitive to noise, as they describing an *exact* shape. Also, scale change would result the chain code length been altered, making comparison more difficult. Although work has been undertaken to overcome these limitations [Li and Zhu, 1988], it is still highly bound to a boundary and the big number of dimensions make it inefficient as a classification method for real-time application.

Fourier Descriptor

Another commonly used shape description method is Fourier descriptor. Most of simple shapes contour could be described as periodic signals generated by a function of distance around the border from a certain point. This signal is then converted into the frequency domain via the Discrete Fourier Transform. Giving a set of N frequency components which represent the signal.

$$X_k = \sum_{t=0}^{N-1} c(t) e^{-itk2\pi/N} \tag{2.2}$$

for k = 0, 1, 2, ...,N - 1, and where c is the function of distance around the border. The coefficients of the Fourier Series can then be used as descriptors for the boundary. The first few Fourier descriptors has enough information to distinguish sphere, cone and cross that has distinctive border differences. The dimensions of the classification space therefore would be relatively low compare to the chain codes. Morse [2000] demonstrated that the Fourier descriptor is shift, rotation, scaling and start point invariant. The Fast Fourier Transform [Cooley and Tukey, 1965] allows the DFT to be computed faster than using a implementation of Equation 2.2, this make the Fourier Descriptors computationally affordable and therefore popular choice of object representation. Furthermore, noise affecting the object border will not be present in the lower frequencies used for shape description, making this technique a prime candidate for implementation during the project.

Region-based shape description

We can use boundary information to describe a region, equally shape can be described from the region itself. An issue with many border-based techniques is that they are sensitive to noise. Region based representations trade this accuracy for robustness, as they tend to rely less on the details of the border but more on the consistent scalar properties of shapes, such as the area. There are quite a few methods available, some are dedicated to deal with complex shapes. In this project, the shapes that going to be distinguished are relatively simple – circle and cross. I would go through some region descriptors that are commonly used.

$$Compactness = \frac{\sqrt{\frac{4}{\pi} \cdot Area}}{MaxDiameter} \tag{2.3}$$

Compactness is the measure of the extent to which a shape fills the convex hull[3] it defines. A filled circle's compactness is 1. Empty space inside shape reduces the compactness.

$$Roundness = \frac{4Area}{\pi MaxDiameter^2} \tag{2.4}$$

Roundness describes how similar a shape to a circle is. A filled circle has roundness equal to its diameter.

$$AspectRatio = \frac{MaxDiameter}{MinDiameter} \tag{2.5}$$

Aspect ratio describes how stretched a shape is.

$$Convexity = \frac{ConvexPerimeter}{Perimeter} \tag{2.6}$$

Convexity is a way to measure how concave a shape is. Using Convex hull Perimeter over shape's perimeter. Space inside the shape is not considered. For any convex shapes, the convexity is 1.

$$Solidity = \frac{Area}{ConvexArea} \tag{2.7}$$

Solidity is a way to measure how empty a shape is compared to its convex hull. A filled convex shape has solidity 1.

$$Formfactor = \frac{4\pi Area}{Preimeter^2} \tag{2.8}$$

Form factor is another useful measure of a shape.

Notice that all of the descriptors are ratios; this is what allow them to be scale invariant, and none of the measurements themselves are dependent on rotation or shift. A selection of these features as feature vector offer benefits over Fourier Descriptor: the feature space relatively low dimension, and the features themselves are more robust to border noise. Sonka et al. [1999] even explained how a complex object can be broken down and represented as graphs of simpler sub-regions, however this is likely to exceed the needs of the project, as well as its real-time constraints, so we assume that any shapes will be simple enough to be representable as a single entity.

[3]Convex Hull: The smallest convex shape which can entirely surround the object.

Hough Transform

All the edge operator discussed in section 2.2.3 do not actually "detect" edges, but rather highlight pixels at discontinuities of intensity in the image for further processing. There are two completely different approaches for finding objects – bottom-up and top-down. Bottom-up method analyses the edge highlights, stitch the edge segments together in order to form a complete shape or system, the system is then identified. This is cognitively more natural for human thinking, but it is very sensitive to noise, gaps between lines or occlusion of the object in the image. A top-down approach on the other hand, requires knowledge and experience about the subject; for example, to find lines in an image, a line model is needed. Then we going through the image, gather evidences of for the existence of a line in the image. When enough evidence is gathered, the line is detected in the image. One of the most used technique implements this method is called Hough Transform. The Hough Transform [Hough, 1959] is a standard tool in image analysis that allows recognition of global patterns in an image. It recognises local patterns and transform them parameter space. It was originally used to find straight lines in the image, but adapted versions can find circles and other general shapes too. When finding lines, it is particularly advantageous when observed line have "holes" in them. The basic idea is the parameters k and b in the line formula $y = kx + b$ can be plotted as a point in a parameter space, where axis are k and b, therefore, a point in the parameter space represents a line in the image space. When we going thought the edge detected image and plot all the edge points into the parameter space, points on the same line in the image would transform to lines that meet at the *same point* in the parameter space. The Hough Transform then simply creates a discretion version of the parameter space called the accumulator array, such that a cell in the accumulator array is incremented for every line that passes through it. Finally, the accumulator array will contain peaks which correspond to probable lines in the original space. Hough Transform is remarkably tolerance to noise due to its evidence gathering approach, and therefore it has been used in many past competition entrants [D.Ribas et al., 2006, Assalih et al., 2007, Albu et al., 2006] and real world underwater applications [Foresti, 2001], and therefore it would certainly appropriate for my project. One of the main use will be the cross centre identification.

Template Matching

Template matching [Nixon and Aguado, 2002] is conceptually a simple process. A target template is given, the best match is found out by putting the template over the image at every possible locations. To achieve scale and rotation invariance, all possible scale and rotation of the template need to be tested too. The comparison process is essentially guessing the parameter depending the location, scale and rotation, so it also named parameter estimation. Its try-every-situation approach makes it grossly inappropriate for this project.

Feature detectors

A group of recent object recognition systems. Feature detector finds a set of "interesting" points in the image, then describe each one with a identifier. Those "interesting" points are selected differently in different detectors, a good detector would choose stable points with high repeatability – a point would always there whatever the rotation, scale, shift, view point or even lighting condition. The classic Harris Corner Operator detectors corners in the image then describes them. But it is variant to scale make it unsuitable to this project. A notable recent development – SIFT (Scale-Invariant Feature) detector [Lowe, 2004] over come the scale variant problem. It is one of the best performed feature detector at present [Bay et al., 2006]. But its high demanding of computation power makes it unsuitable for this real-time project. The SURF (Speed Up Robust Feature) is another recently developed feature detector. It works in a similar way to SIFT, but out performs SIFT in speed. It is a rotation, scale, contrast and illuminant invariant feature detector. SURF works in the following way: firstly, interesting points are selected at distinctive locations of the image, such as corners, blobs and T junctions; then the interesting points and its neighbouring pixels are described by a feature vector; finally, these features are matched between images. Feature detectors tend to slow both at detecting, and matching. SURF feature focused on speed up while not sacrificing the accuracy. It used the Haar-wavelet response and it also used integral images for major speed up. Integral Image is an intermediate representation for the image and contains the sum of grey scale pixel values of image, allows only four addition operations to calculate a sum over any region on the image. The SURF descriptor consists of 64 values for each interesting point. The values are obtained as follows. In order to achieve rotation invariant, SURF split into two stages – orientation assignment and descriptor construction. For further speed up, Upright-SURF does not have the orientation assignment stage, making it rotation variant. But it better suited for applications where camera remains more or less horizontal. This project just falls into this category. In orientation assignment stage, the detector first calculate the Haar-wavelet response in x and y direction. After the wavelet response is weighted with a Gaussian centred in the interesting point, the responses are presented as vectors. The dominant orientation is estimated by calculating the sum of all responses within a sliding orientation window. The longest of the horizontal and vertical response gives the orientation of the interesting point. In descriptor construction stage, a square is created around the interesting point, its orientation is determined in the previous stage. The size of the square is $20s$ where s is the scale of the interesting point determined. This region splits into four sub-squares. For each sub-square, wavelet response d_x and d_y in horizontal and vertical direction are computed using 5x5 equally spaced points. They are summed over for each region, which gives 32 dimension vector. In order to express the intensity change, the sum of the absolute values of the d_x and d_y are calculated, makes the whole descriptor 64 in dimension. The vector is normalised into a unit length to be invariant to contrast, and the wavelet responses themselves are invariant to illumination. There is an extended version – SURF-128, which calculates the sum of d_x and $|d_x|$ separately for $d_y > 0$ and $d_y < 0$ and vice versa for d_y and $|d_y|$, which doubles the descriptor dimensionality to 128. But their experiment only show a slight accuracy gain over the regular SURF with slower

description and matching. SURF-128 would not considered for this project, while Upright SURF would be very suit for this project.

2.3.2 Object Classification

So far, we have discussed how certain feature from an image can be extracted and represented. This is the first half of the object recognition – features extraction. Classification takes those feature measurements, puts them in a *feature space* to classify. Feature space is a space with each individual feature to be its axes. If we use roundness, form factor, convexity and solidity to be the features to describe shapes, roundness, form factor, convexity and solidity would be the feature space's axes. A point in this feature space would define particular shape using these features. If the features are well chosen, different shapes that needs to be distinguished would far apart from each other in the feature space. The feature space can have as many dimension as needed, and the features can be any measurement or observation that helps classification. But a higher dimensional feature space slows down classification and not necessarily increase the correctness of classification. Training is carried out using Ground Truth[4] with different objects in. Clusters of points are plotted in the feature space to represent each object trained. When a test image comes in, its features are measured and plotted as a point in the same feature space. If this point is nearer to one cluster than the other, it is classified as the nearer object. In this project, red, green and blue can be used as colour classification features; the Fourier Descriptor or Shape Descriptor themselves or their hybrid can be used as features to distinguish shapes. Different ways of deciding whether a point is belonging to a cluster is discussed in this section.

K-Nearest Neighbour

A straight forward and fast classification mechanism. K-nearest neighbour classifies by comparing the number of the nearest neighbour the sample point has in the feature space. So for example, let $k = 3$, three nearest feature points are taken, if 2 of them are from class A, the other 1 is from class B, then the sample is classified as class A because $2(A) > 1(B)$. This method is kind of noisy resistant, if in the above example, the one point from class B is actually the nearest neighbour of all 3, it does not matter, the result is not affected by noisy outlier points. This is called smoothing, and it has greater effect for larger values of k [Michie et al., 1994]. The simplicity of the algorithm attracted me to looked into it, but the actual effectiveness in my underwater application is still a concern. Nevertheless, it is planned to be included in the evaluation of this project.

[4]Ground truth images contain known object, used as initial supervised classification training.

Euclidean distance

The simplest distance is the Euclidean distance, which is the "ordinary" distance between two points that one would measure with a ruler. It defines distance between points in any dimension. The Euclidean distance can be calculated by formula 2.9:

$$Euclidean = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}, \tag{2.9}$$

where p_i and q_i are the values for each features.

Using the colour values as example, the distance between two colours $x = [x_R, x_G, x_B]^T$ and $y = [y_R, y_G, y_B]^T$ can be calculated as:

$$|x - y| = \sqrt{(x_R - y_R)^2 + (x_G - y_G)^2 + (x_B - y_B)^2} \tag{2.10}$$

Distance is an important concept in object recognition. It is one of the main ways to discriminate whether the feature vector belongs to a particular object class. With Euclidean distance, it always have difficulty to decide a good place inside the feature clusters to measure the distance from. Mahalanobis distance as I will discuss next, takes into account of how the points in the feature cluster distributes, therefore gives a more accurate "distance".

Mahalanobis distance

A widely preferred alternative to Euclidean distance, it gives more accurate representation of distance from a set. While Euclidean Distance from a cluster defines a hyper-sphere in the feature space with its centre at the mean location of the set, Mahalanobis Distance[Mahalanobis, 1936] defines a hyper-shape which is representative of the directions in which the set varies. For example imagine the training set varies a great deal in dimension x but not at all in dimension y. The Mahalanobis distance for all points with Euclidean distance d from the mean will vary depending on the direction of d, points in the direction x will have a low Mahalanobis distance while variation in dimension y will cause a large Mahalanobis distance, while the Euclidean distance remains the same. The calculation of the Mahalanobis Distance between a test vector $x = [x_1, x_2, x_3, ..., x_p]$ and a feature class C is as follows. The feature class C's mean $\mu = [\mu_1, \mu_2, ...\mu_p]$ and the covariance matrix Σ are calculated, then the Mahalanobis distance is given by

$$D(x, C)^2 = (x - \mu)^T \Sigma^{-}1(x - \mu) \tag{2.11}$$

The trade off between the accuracy and the real-time requirement needs considered. The performance of the distance calculation will be evaluated.

Eigenmodel

A modelling of a data set by fitting Gaussian to it. Eigenmodel converts the discrete points cluster into a continuous Gaussian. For a data set P

$$P = [P_1, P_2...P_n] \tag{2.12}$$

where P_i is a vector of features. The mean μ of the whole set can be calculated:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} P_i \tag{2.13}$$

The deviation of each sample from the mean q is

$$q_i = P_i - \mu \tag{2.14}$$

The covariance matrix of the data set is then

$$C = q\, q^T \tag{2.15}$$

Using Singular Value Decomposition (SVD) [Acharya and Ray, 2005], the eigenvectors V and their corresponding eigenvalues U are calculated from covariance matrix C. The μ, U and V together forms an eigenmodel. The Mahalanobis distance to an eigenmodel can be calculated

$$D(x, C)^2 = (x - \mu)^T U^T V^{-1} U (x - \mu) \tag{2.16}$$

Gaussian Mixture Model

Gaussian Mixture Models refer to the modelling of a data set by fitting multiple Gaussian to it[Moore, 2004]. It can be used to piece-wise model objects. In some case, those feature vectors do not form a cluster in the space, therefore a single eigenmodel could not fit properly over the points, for example a banana shape in 3D space. By fitting multiple Gaussian, the classifiers can then measure the Mahalanobis distance of the feature vector from each one to make a decision.

Support Vector Machines

Two clusters of points are presented in the feature space representing two classes. The goal is to decide which class a testing point belongs to. The idea of the Support Vector Machines [C.J.C.Burges, 1998] is to draw a hyper-plane inside the feature space, the plane divides the space in two, and the distance from the hyper-plane to the nearest members of each class is maximised. The distinctive advantage this has over Mahalanobis distance is speed. The hyperplane is built at the training time. When a test feature vector is known, all the calculation is just to work out which side the vector is in, and the classification is made, the distance to the hyper-plane can be the confidence measure of the result.

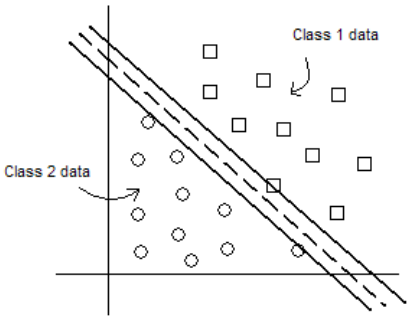


Figure 2.6: Support Vector Machines

The disadvantage is it not capable to work on single class (no where to define the hyperplane) nor natively support more than two classes. A lot more training is involved to adapt it to classify more than two classes. It is similar to binary search. For example to classify 4 classes A,B,C,D, First, divide the class into two group, A B and C D, compare them separately, then compare the results of them, e.g., A with C to find the final classification. Even only with 4 classes, the training has to be done between A and B, C and D, A and C, A and D, B and C, B and D, which it already shows the drawback clearly. This classifier can be applied only in task 1, where two different coloured buoy needs to distinguished.

Neural Networks

Neural Networks have been extensively used in image segmentation and object classification problems [Ripley, 1996, Haykim, 1999]. They are learning networks with large set of interconnected neurons, divided into three layers – input, hidden and output. Figure 2.7. Each neuron know how to process the data come from the previous layer, and pass its result

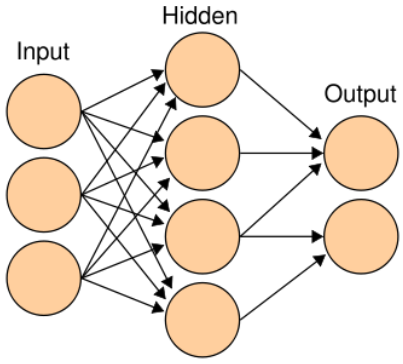


Figure 2.7: Neural Networks

on to next layer. The input neurons carry feature information of the object to be identified, for example, Hue, Saturation and Intensity value of a colour to be classified. The output neurons are the classification results. So one of the output can say: input matches class "red". The hidden layer needs to be trained before it can be used. One way of training is by the Back Propagation Algorithm [Hirose et al., 1991]. A large set of ground truth data is feed into the algorithm, allow the neural to learn and generates coefficient that make the similar input data that outputs same class. At the end of the training, hopefully when we feed the neural the training data, it outputs the correct classification. One problem with this type of classifier is that it is impossible to tell whether the learning was successful. There is no way to tell how good is the classifier by just reading the learnt behaviour of each individual hidden neural. There are many variations of the Neural Network. There is a multi hidden-layered Neural Network, and there are different training algorithms allow supervised or unsupervised training. In addition, the training data itself directly affects the classifier's accuracy [Haykim, 1999]. All these factors make this classification method varies a lot in performance as well as the processing power needed. This classification method would not be the primary one used in this project.

2.4 Object Tracking

Once the object has been identified in the image, it is often useful to estimate where the object will possibly be in the next frame. So when the object is not found in the next frame, the tracking algorithm may still have an estimation of where it is most likely to be. The same technique can also be used to save computing power. D.Ribas et al. [2006],who was the winning Team in SAUC-E 2006, in order to foster the velocity of the algorithm, they defined a window around the identified object, this window is then used in the next iteration to search for the object, effectively reduce the search area. This is essentially the most basic object tracking, with only consider one previous frame. It seems to me is a very sensible start point for implementing the tracking algorithm. With more sophisticated algorithm, it is possible to be used to filter out bad classifications, therefore improves the overall object recognition correctness. For example, a high confidence object tracking output can make the system ignore some unconfident or unexpected output come from classification algorithm.

2.4.1 Kalman filter

Kalman filter is a recursive solution to the discrete data linear filtering problem [Kalman and Rudolph., 1960]. It has been the subject of extensive research and application since recent advances in digital computing, particularly in the area of autonomous [G.Welch and Bishop, 2001], tracking [D.Comaniciu et al., 2000, Collomosse, 2004, Wallis, 2006] or assisted navigation [Sivaraman and Sahay, 1998]. And this technique has been described in M.Isard and A.Blake [1998]'s paper as

> Spatio-temporal estimation, the tracking of shape and position over time, has been dealt with *thoroughly* by Kalman filtering...

Due to the high precision output of the estimation, music industry uses Kalman filter to estimate the loudspeaker's parameter which help them to reduce the harmonic distortions [Ribeiro et al., 2005]. The recursive nature of the filter seems significantly reduced the computing power it is needed. There are applications even on the camera phone, which allows user to use the camera as the sensor to track the features in an image, calculate the motion vectors as user moves the camera and therefore control a computer mouse [Gai et al., 2006].

To use Kalman filter to track motion, the second derivative motion model provided to the filter. The Kalman filter has two phases: *Predict* and *Correct*. In *predict* phase, the current state is predicted using the motion model and previous results, getting an estimated locations and their confidence. When a measurement is made, Kalman filter *corrects* the internal confidence, and hopefully arrive a more accurate state estimation next time.

Kalman filter has been popular in the past SAUC-E and AUVSI competitions for motion tracking [Assalih et al., 2007, D.Ribas et al., 2006], so this is one of the tracking algorithm to be used.

2.4.2 Condensation

A more recent technique come from M.Isard and A.Blake [1998] called Condensation, stands for Conditional Density propagation. It supports multiple hypotheses concurrently as oppose to Kalman filter that only supports one. Instead of saying with Kalman filter, this is the result, and I am this confident about it; it gives you the confidence measure across all the possible result, it is then up to the system how to interpret the data. One example could simply take the result with highest confidence measure to be the predicted object position. The crucial advantage this over Kalman filter is in situation where erratic target movement confuses the Kalman filter and made it focuses on the wrong object; while as condensation won't have this problem as it keeps track of "all" possible target independently. A simple process of Condensation algorithm is described as following: Choose a randomly distributed set of sample vectors V(x,y) to represent the position on the input image, the size of the sample is normally 100 to 200 depending on image size. Define a window size W and H. This is the size used to enlarge the point described by the vector, so that it is possible to evaluate, depending on how many target pixels are there within that window, how good the "guess" was. A set of g corresponding to each sample vector is calculated. With all g normalised to 1, we can plot a 1D diagram (similar to the Gaussian distribution graph), where g is the height of the point on the diagram. From the diagram, one could deduce by the height of the curve, the possible location of the object of interest. Because g is normalised, (all the g add up to 1), a better estimation would be sum up all the multiplications of g and V.

we can see where about is the object of interest from the high curve. to find out the

best estimation of the centre of the object, multiply every V by g, the result is the best estimation of location V'.

A significant property of the condensation algorithm is that it can track objects moving rapidly with highly complicated motion models. It is definitely worth to evaluate if the time allows.

2.5 Conclusion

This literature survey reviewed different techniques to do image pre-processing, object recognition and object tracking. Time also spent on reading the past entrants' journals, which gave me entry point on selecting algorithms. This review provided me an opportunity to broadly learnt the computer vision domain. Many ideas and concepts were learnt from scratch during this review.

In the image pre-processing, many noise reduction filters were discussed and compared, linear filters tend to be fast, but they remove edges too; non-linear filters remove noise but keeping shape edges, although the speed can be an issue. The real affect of the trade off need to be found out during the evaluation. A underwater colour image enhancement technique using polariser was discussed, unfortunately it is not practical in this project. Different ways of describing colour, different colour spaces and colour histogram was investigated, HSV colour space dropping V channel would be a strong candidate for underwater colour classification. A few popular edge detectors were described, but they are not deeply affect the outcome, so any simple and fast one would serve well. Lastly in image pre-processing, segmentation techniques was discussed. Different thresholding methods was extensively discussed as it is one of the most useful techniques.

In object recognition section, I divide it into feature extraction and classification. In feature extraction, focus was on the *invariant properties* for each task. Chain code, Fourier descriptor and some region based extraction method was discussed. From the requirement of the application, Fourier descriptor and a combination of the region based extraction is used as description features; Hough Transform could be used to work out the cross centre in task 2. A few object classification ideas were presented, my opinion was to keep things simple as long as they work. Euclidean, Mahalanobis distance are used. Some feature detectors also discussed. Due to their performance, only the SURF could possibly run in real-time for this project. It is implemented and its feasibility of being used in a real-time application is evaluated.

Finally, in the object tracking. Kalman filter and Condensation trackers are discussed. The decision was made that implement Kalman filter first, if time allows the Condensation could be implemented and evaluated too.

Chapter 3

System Specification

While previous chapter serves the purpose of review many detailed computer vision algorithms for solving the potential problems this project is facing, this chapter takes one step back and consider other aspects as a software development and research work. Software process model, the boundary of this project, the resources needed, requirements and high level design are discussed in this chapter.

3.1 Software Process Model

Incremental development model is going to be used though out this project. According to Sommerville [2007], incremental model is to develop the core system component first, then progressively build upon with more functionality until reaches the final system. This process is suitable for this project. At the beginning, the author faces a steep leaning curve, both in vision subject itself and programming, many decisions made could be invalid or later found to be unimportant. To set out a strict list of requirement at the beginning – as the waterfall model based on – could lead me too far in some minor details. So start programming as early as possible, get a basic program running, set out some test framework and learn the basics of the vision domain is the best route. Later, as domain knowledge accumulate, a more detailed specification could then be formalised.

3.2 System Context

The AUV is a underwater robot used in previous year's competition. The previously used AUV gives me a concrete start point and provides me opportunity to recognise problems encountered. The AUV has three major component, Artificial Intelligent (AI), Control and Vision Figure 3.2. AI is the "brain" that decides actions. It queries the Vision, the Vision answers the question; depending on the answer the Vision replies, it tells the Control what to do; The AI also comprehend which task it is on and what to do in each task. The Control

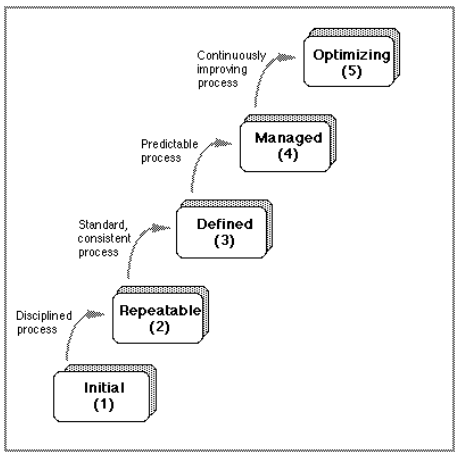


Figure 3.1: Incremental Software Process Model. The core system is developed at the beginning, more functionality is added in the later stage.

is an high level software interface to mechanical control of the movement of the AUV. The Control decides what motors to turn on when instructed to move forward, backward or submerge etc. The Vision is the part this project is based on. It receives commands from AI and respond with useful information; for example, the location of the buoy in the current view, whether a flash beacon can be seen in the current view etc. Due to the high demanding of the computation power from the Vision, it runs on a separate computer from AI and the Control, the communication to AI is via the point to point network. The communication protocol between the AI and the Vision is discussed in section 3.4. For the competition team, this project is to produce a base system for interacting with the AI in the AUV, taking live videos from two underwater cameras, solving the vision problems the AUV will be facing. This project as an academic research, the author will be the "AI" assessing the various output and to evaluate different algorithms. For the fairness of evaluation, videos of underwater scene will be used instead of videos from live camera.

3.3 Resources

3.3.1 Academic Resources

Reading resource in this field is vast, for this competition alone, I have access to all past competition journals from SAUC-E website; Association for Unmanned Vehicles Systems International (AUVSI) organises a similar competition in the US, from where I can access to their journals too. They all specifically target to the problems I may encounter during the development.

Books on digital image processing are very useful, Feature Extraction & Image Processing, The Image Processing Handbook explains many techniques from depth, they all available

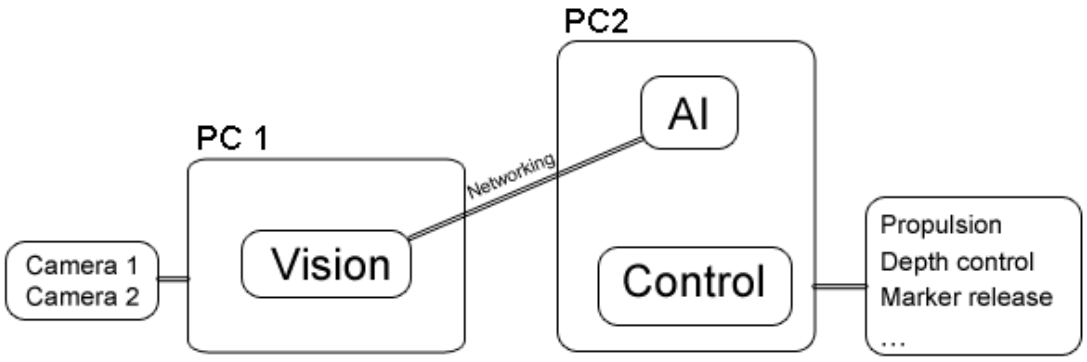


Figure 3.2: AUV high level structure. The Vision runs on a separate PC and communicates with AI via network.

from University library. Journal articles in this field are also useful, mostly available for downloading via library website.

3.3.2 Development Resource

C is chosen to be used as the main development programming language for the following reasons. Open Source Computer Vision library (OpenCV) – a well known computer vision library is implemented in C/C++, which can be used directly for this project. It developed by Intel, providing all the core operations and structures for image processing. Many algorithms, for example, noise reduction filters, Kalman filters, are already implemented in OpenCV. The code tend to be very optimised, some even implemented in assembly for the highest performance possible. Previous student already used C and OpenCV for development, it is always a good idea to build upon previous work. The author has some C experiences, which is also advantageous. Microsoft provides the Visual Studio IDE free of charge for academic use, easing the development and debug cycle. Matlab is a programming language specifically designed for computer graphics and vision. It runs slower compare to C as it is an interpreted language. Due to lack of experience and resources, it was used for main development, but used as a tool for fast testing, data visualisation, figure plotting as well as debugging. Matlab is available from BUCS PC in the library.

3.3.3 Evaluation footage capture equipment

At the beginning of the development, the footage captured previously would be sufficient. In a later stage, as the research direction differ, new underwater footage will be needed. The footage capturing will be carried out in the 25 meter University swimming pool, but if this does not satisfy the requirement (eg. depth), an alternative pool is required. The exact

camera used on the AUV is required since different camera could affect the algorithm and evaluation differently. When possible, using the AUV itself to capture the footage could more accurately emulate the condition.

3.4 Communication Protocol

Mentioned in system context, the vision system is going to run on a separate PC and communicate with AI via network. A communication protocol must be defined between the system. As there is no big flaw found in the previously used protocol, the basics are going to be reused – communicate via custom defined udp packets. Since the tasks have changed, the formation and the interpretation of the message needs re-definition. Figure 3.3 is the standard message format used between Vision and AI, and the meaning of control messages from AI.

0xff	0xfe	0xff	majorOpcode	minor Opcode	unused bytes

Start:	1	0	empty...
Reset:	1	1	empty...
Stop	1	2	empty...
Search for Buoy:	2	0	empty...
Search for Beacon:	3	0	empty...
Locate centre of dropping point:	3	1	empty...
Search for Cone:	4	0	empty...
Search for Type:	4	1	empty...

Figure 3.3: Top: Standard message format used by both Vision and AI; Bottom: Control Messages: AI to Vision. These messages define the instructions available to AI which can be used to control the functioning of the vision system

3.5 Requirements Specification

The requirement of the system going to be developed is largely depend on the tasks that the AUV need to accomplish. The following sections are split based on the tasks.

3.5.1 Functional Requirements

1. Task 1: Find and make contact with a target mid-water buoy, avoid the other different coloured decoy buoy.

1.1. The system shall be able to locate in the input image, distinctive coloured buoys.

1.2. The system shall be able to keep track of the buoys as the AUV moves towards it.

1.3. The system shall correctly distinguish between different coloured buoys underwater.

1.4. The system shall send out coordinate of each buoy.

2. Task 2: Find the flash beacon, drop a marker on the centre of the cross below the beacon.

2.1. The system shall locate a beacon in the image when lights on.

2.2. The system shall be able to tell whether a light is flashing.

2.3. The system shall be able to track the flash beacon, even when times the flash is off as the AUV moves towards it.

2.4. The system shall classify the shapes given appropriate training shapes.

3. Task 3: Identify 3D objects, surface when above them

3.1. The system shall recognise objects regardless its shape, orientation and distance.

3.2. The system shall be able to track the object as the AUV moves towards it.

4. Common

4.1. The system shall have a ready state that start listen for AI's instruction.

4.2. The system shall be able to reset to the ready state when competition task need to be restarted.

4.3. The system shall accept predefined set of instructions from AI and reply with set of defined answers.

4.4. The system shall be able to receive udp packet and send udp packet for communication.

4.5. The system shall switch tasks at run-time.

3.5.2 Non-functional Requirements

1. The system shall process at least 1 frame per second and send information back to AI.

2. The system shall either use video stream as data source for developing and testing or live video stream from the cameras as data source when used on AUV.

3. The system shall be robust enough to follow AI's instruction and prevent hang itself on some unknown instructions, and possibly recover itself if some serious error occurred during operation.

3.5.3 Evaluation

Once the candidate algorithms implemented, evaluation of those algorithms were carried out. Each algorithm is tested against each other under different conditions. These conditions include distance from the camera to the target, the target's orientation, the target's location on the image, target's colour and target's shape. Different algorithms maybe combined to produce a better performance.

Chapter 4

Locating and Tracking Objects of Interest

In computer vision, images are viewed as evidence of objects. Images contain information about the object. The attempt of many computer vision application is to reduce the image to information. The first step in doing so is finding the features. Using one of the competition tasks as an example, an image of a floating red buoy (Figure 4.1) is evidence that there is a reddish coloured blob in the image. Because the colour of the blob matches the colour we expect the buoy to be, and its location in the image is within the range we expect it to be (according to its previous movement trajectory), it can then be classified as the buoy we were looking for. The colour, the location, in this example, are features derived from the image; it is passed to the classification module to output as information. Other features can be derived and used to classify: for example, straight lines and circles can be extracted by Hough transform; shape information of a target's binary mask can be described by Fourier descriptor or shape descriptor.

In the process of reducing image to information, noise in the original image is also transformed to noise in the information. Depending on the type of features the method is used to do the reduction, and the nature of the noise, the resulting information is affected differently. Chain-code, studied in section 2.3.1, a way of describing the contour of the object using a list of direction codes, would very sensitive to noise normally presented in underwater images, resulting in imprecise descriptions of the contour. Whereas region-based methods like Shape Descriptor use features such as area or convex of the object, which is much less affected by the same type of noise, resulting in a more durable description. Noise reduction can be an important first step in many applications. But it is too early to say that noise reduction is beneficial at this stage. It is used in algorithm evaluation to find out how much affect it has in the output of the results, then decide whether it is worth to be used in this project.

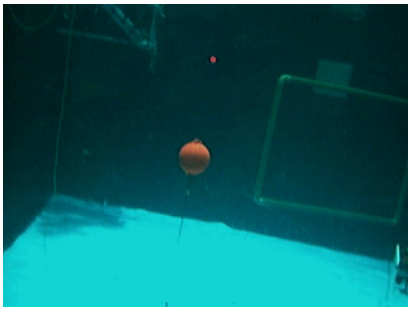


Figure 4.1: The red buoy in 2007 SAUC-E competition

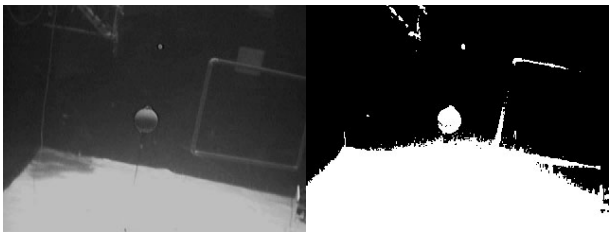


Figure 4.2: Left: Original image grey-scaled; Right: after thresholding. Entire floor is misclassified.

4.1 Target object localisation

4.1.1 Find Coloured buoys

One very effective and widely used step to reduce image is segmentation. Segmentation divides the image into parts that have strong relations to real objects contained in the image. One of the simplest segmentation process is grey-level thresholding. A colour image is first converted to grey-scale. A threshold value is determined to segment the object from the background as they tend to have different brightness. It is fast, and computationally inexpensive, which lends itself to this project's real-time requirement. But thresholding on grey-scale image is not useful in this project as the background could easily have similar brightness objects. Figure 4.2 shows the thresholding on the red buoy.

Throwing away colour information in this colour detection contest sounds like an unwise action. Colour is important. The basics of how a computer stores colour, and several common colour spaces are discussed in the literature review section 2.2.2. Colour threshold on single RGB value was first experimented with. The outcome was not meaningful: not only the real red was segmented out, but also the white and some light colour are segmented out too (Figure 4.3). That is because the white or light background contains high values of all colour components, therefore not only the colour looking red was picked out, all light colours with red channel value above threshold were picked out too. An adapted method was put on test too. It set to pick out the percentage of the red instead of a fixed value.

Figure 4.3: Left: Input image; Right: Thresholding on red channel only. The white buoy is segmented out too

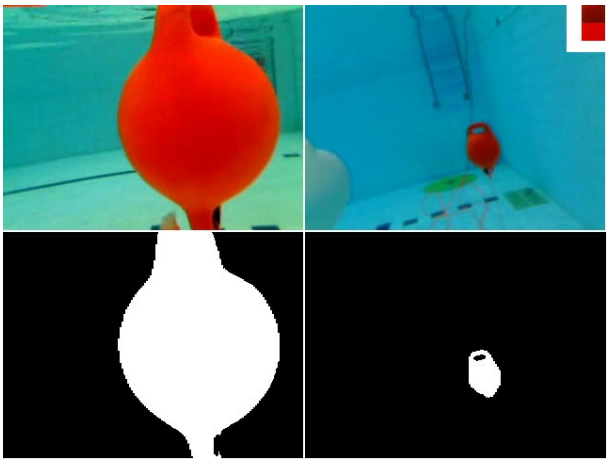


Figure 4.4: Left: near view of the buoy and its ground truth mask; Right: far view of the buoy and its ground truth mask; Top right corner: colour samples taken from both images

Unfortunately, the result was also not as good. Although it picked out red more accurately, it also picked out the dark corner regions. The explanation is dark colour have very small RGB value for each channel, some dark colour may happen to have a small red value, but zero for green and blue channels. In this case, red is actually 100% for this pixel, and it is bound to be segmented out by above method.

While deciding the training sample and test image for developmental testing, it would be less interesting if the algorithm is segmenting out the correct colours from the original image where the sample colours were taken. The application is underwater, colour illumination and other effects changes a lot as viewing position or distance changes. It is also unreasonable to train the system with only one colour, and hope it is able to find the similar colours. I would like an algorithm that is illumination invariant, and trained with relatively small amount of samples. It was set to use two colour training samples taken from two static images – one near view of the buoy and one far view (Figure 4.4). Those two images were also used as testing images.

The colour thresholding problem can also be viewed as a classification problem – given a colour pixel $\vec{c} = [c_r, c_g, c_b]^T$, is it a colour belonging to a class or not? A class of colour s

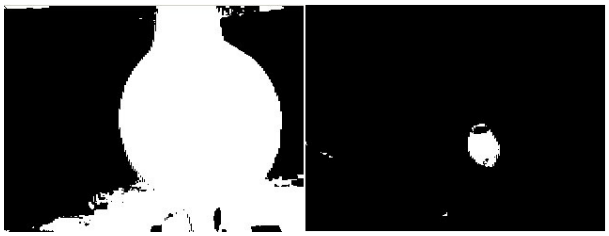


Figure 4.5: Left: Near view of the buoy after RGB Euclidean distance thresholding. Right: Far view of the buoy after RGB Euclidean distance thresholding. Ground in the near view are lit up too

can be constructed by taking n colour samples from the object in the image.

$$s = \begin{pmatrix} r_1 & r_2 & ... & r_n \\ g_1 & g_2 & ... & g_n \\ b_1 & b_2 & ... & b_n \end{pmatrix} \tag{4.1}$$

RGB Euclidean distance classification was first considered. A mean value μ of the training sample was calculated using

$$\vec{\mu} = \frac{1}{n} \sum_{i=1}^{n} \vec{s_i} \tag{4.2}$$

Euclidean distance in RGB space between each testing pixel $\vec{t} = [t_r, t_g, t_b]^T$ in the image and the mean value $\vec{\mu} = [\mu_r, \mu_g, \mu_b]^T$ was calculated by

$$D_E = \sqrt{(t_r - \mu_r)^2 + (t_g - \mu_g)^2 + (t_b - \mu_b)^2} \tag{4.3}$$

This classification approach assumes that smaller the distance of the testing pixel is to the mean value of the sample set; the more likely the pixel to belong to the set. An appropriate threshold on *distance* defines the pixels that belong to the sample colour set. Figure 4.5 demonstrated the result of this method to classify a same underwater buoy at different distances using samples taken from the object at both distances. It is clear that the Euclidean distance thresholding in RGB does not give a satisfactory result even though both sets of red were presented in the sample. There are two reasons why it is so. First, the computer RGB colour space is not the way people describe colour. When we compare the colour of the underwater buoy in distance or nearby, we would say that they are the same colour, just the further one is darker than the nearer one. But RGB colour space would encode those darker and brighter colours with rather different values, making RGB Euclidean distance not classify them as same colour. Second is the assumption made by the RGB Euclidean distance classification: smaller the distance the testing pixel to the mean value of the sample set, the more likely the pixel is belonging to the set. Simply taking the mean to represent the whole set is incomplete. It does not take in to account the set's distribution pattern. A circle and ellipse shaped distribution (Figure 4.6) happen to have a same mean value, but a point inside the circle while outside the ellipse should be classified differently.

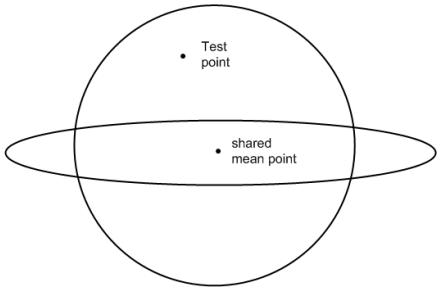


Figure 4.6: Two different distribution pattern with same mean value, a same test point should classified to the set if distribution is circle shaped but not eclipse shaped.

For the first problem, since the colour of the buoy itself has not changed, the observed colour varies mostly in the brightness as distance changes [Tibau et al., 2006]. There is a very good reason to use HSV colour space since it is trying to simulate how people distinguish colour, with brightness in a separate channel. To ignore the colour change due to the brightness change, it is simply classifying without that channel. Figure 4.7 are the

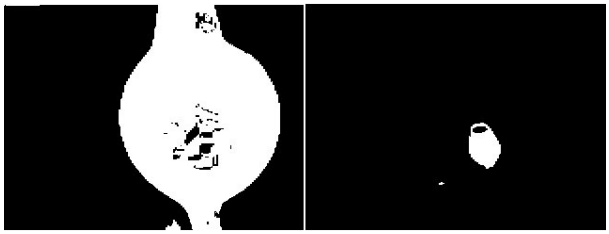


Figure 4.7: Results of thresholding on HS(V) Euclidean distance on both image. Both buoy are correctly segmented out. Only some bits inside the near view buoy has left out.

result of using HS(V) space ignoring V channel. It gives very accurate segmentation of the object across distances, considering the little amount of computation it needed.

Go back to the second reason why RGB Euclidean distance threshold would not work: the probability of a test point belonging to the set is not distributed evenly in all directions. As discussed in the literature review in section 2.3.2, building an Eigenmodel over the entire data set is the most used and an effective method to take advantage of the distribution information to help with the classification. An Eigenmodel consists of mean value μ, eigenvectors U and corresponding eigenvalues V. Eigenvectors describe the direction in which the data varies most; corresponding eigenvalues describe the amount of variation. The same Euclidean distance away, a point in the most varied direction of the sample set should measure shorter than in the least varied direction of the sample set. Mahalanobis distance to an Eigenmodel is the "distance" measured in such a way. The Mahalanobis distance between a test point $\vec{t}$ and an eigenmodel $e = \mu, U, V$ can be calculated using formula 4.4.

$$D_M(\vec{t}, e) = \sqrt{(\vec{t} - \vec{\mu})^T U V^{-1} U^T (\vec{t} - \vec{\mu})} \tag{4.4}$$

Figure 4.8: Results of thresholding on RGB Mahalanobis distance.

Figure 4.8 is the thresholding on Mahalanobis distance to the Eigenmodel. The result observed in the image are having very similar effect as the RGB euclidean – the ground were lit up in the left image, where as there were areas missing in the right image, making the overall results worse than the RGB euclidean. It is rather clear that building an comprehensive eigenmodel would not reduce the RGB colour change bright by brightness change, but actually enhanced it as the images demonstrated. Figure 4.9 is the plot of the training samples in RGB space. The points form two major clusters in the space. Building an eigenmodel over a discontinued set of data would not give a very accurate description of the data distribution due to the uni-model nature of eigenmodel.

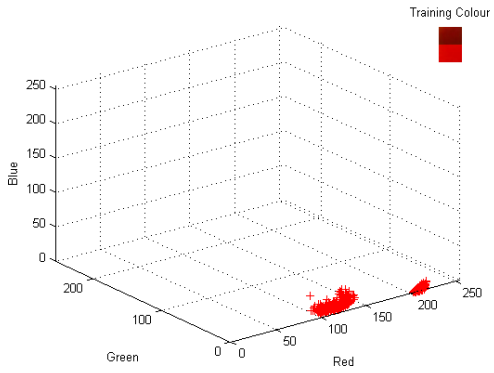


Figure 4.9: Plot of the training colours in RGB space. Two clusters clearly shown.

The initial experiments suggested that the HS(V) was a good approach. Building an eigenmodel using only Hue and Saturation should improve the effectiveness of the approach. Figure 4.10 is the plot of the sample points in HS(V) space, they do not form more than one clusters, eigenmodel would be effective. Figure 4.11 are the result images. Excitingly, it gave the best outcome thus far. Both far and near buoy were segmented out correctly with a same threshold value. More assuring, while the threshold chosen to produce figure 4.11 was 130, the regions that does not belong to the buoy had values vary from 1000 to 4000, far above threshold value. This means a small vary in the image condition would not make big impact to the output result, giving a stable segmentation.

There are going to be two buoys in the water for the buoy task – one target buoy, one decoy

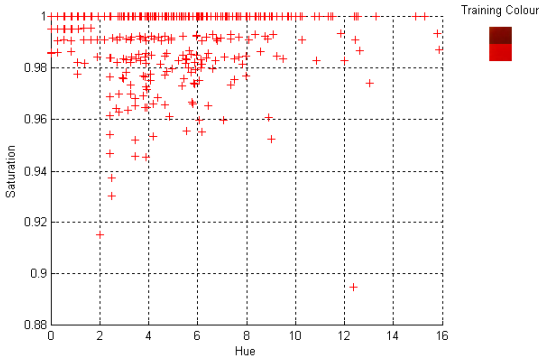


Figure 4.10: Plot of the training colours in H-S space. They do not form more than one clusters

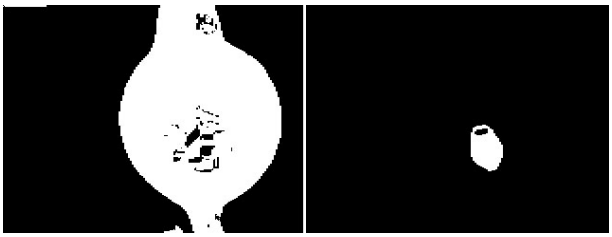


Figure 4.11: Results of thresholding on HS(V) Mahalanobis distance. Similar result as HS(V) Euclidean distance.

buoy. The requirement is to avoid the decoy buoy and hit the target buoy. It is necessary to recognise and track both buoy, so that the AUV can navigate around the decoy while move towards the target buoy. There are several ways, and it also depending on the algorithm used to tracking. Kalman filter and Condensation tracker have the ability to tracking more than one target in the frame. Although in the literature, Condensation was proven more effective in multi-target tracking, while Kalman filter suffers mixing two targets into one when one goes behind the other. Condensation was eventually not implemented due to the necessary detailed understanding of the algorithm, it was not used. The other solution to track two buoys is to double the detection code, giving it training colours for each buoy then tracking them individually by separate tracking module. The downside from the programming point of view is it made the vision system slight harder to maintain (duplication of code), and possibly slower. But it is the fastest and a reliable way to adapt the current system to meet the requirement.

In the competition description, the buoys are of distinct colour – red and green. But one of the buoys available to the author is white. Even worse, the developmental footages were captured in a swimming pool, where entire pool floor and walls are all white – very similar colour to the buoy. This makes segmenting the white buoy from the white wall and floor difficult due to the nature of the colour based segmentation technique. The HS(V) algorithm

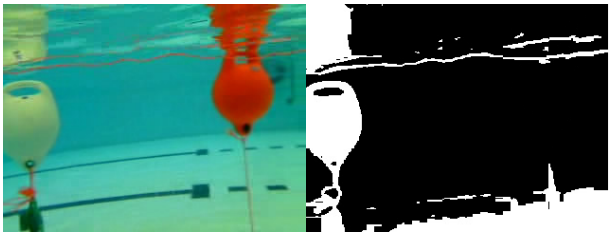


Figure 4.12: Colour segmentation do not perform good when finding a white buoy near white floor.

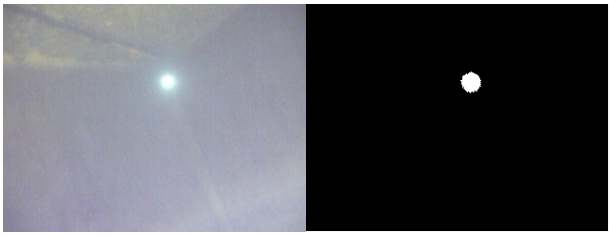


Figure 4.13: Flash beacon using brightness thresholding. It gives an accurate location of the flash with no any mis-classification.

was applied and figure 4.12 is the result. The near floor was lit up and misclassified, but the more further away walls did not confuse the algorithm. In this situation, it is necessary to apply an shape recognition or use other approaches. I would consider this situation one of the worst cases, and would not expect to happen in the competition. The competition pool is very large and deep (50m(l) x 12.5m(w) x 20m(h)), there is no chance that the floating target would be located only 1 meter away from the wall or floor. Even it does, the floor and the wall are made of concrete, which has very different colour from the red and green buoys.

4.1.2 Find flashing beacon

In a dark with no active lighting environment, detecting a flash beacon is an easy task. A flash beacon emits a strong beam of lights, make it easily visible from distance away. The characteristic of a flash on an image is that it appears as an bright circle, and all the surrounding regions have a much lower brightness. Thresholding on the grey-scale image is the most straight forward approach. Figure 4.13 show the brightness thresholding on the official released flash beacon image. It gives a very accurate flash location. Because there is no useful official underwater video footages of the flash beacon can be used for developmental testing, video footages of a similar flash beacon in a shallow swimming pool is used. There are two significant difference between the capturing environment and the possible competition situation. First, the flash available to the author is much weaker than the official one. With this weak flash, the flash spot was visible and segmented accurately

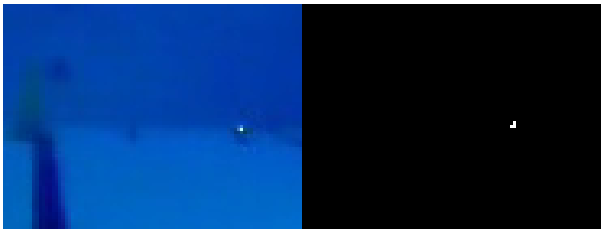


Figure 4.14: A flash at 15 metres away. The brightness thresholding still finds the flash correctly

even from about 15 meters away underwater (Figure 4.14). This length is significant, even in the competition. This factor actually makes the test environment more difficult than the actual competition. If the algorithm works in a more difficult situation, it certainly would work in a simpler one. Second difference is the depth of the pool. In order to allow enough in water distance between camera and the flash, the author had to use the length of the pool instead of the depth of the pool. So the flash beacon would located on the floor of one side of the pool and camera captures from the other side. This introduces a problem. There is a mirror effect on the surface of the water due to the light speed change when leaving water. This "mirror" created an image of the flash beacon above the water when viewed from a distance. The brightness thresholding algorithm alone has no way to tell between the original and the image. The good news is this problem goes away when the flash is situated on the floor of a 10 meters pool, where the AUV would have to "look down" for the flash. Since it was already demonstrated, finding the flash beacon does not pose any difficulty, even the simplest algorithm can work reliably. It is reasonable to say that as long as the beacon remains to be the brightest spot in the captured image, the distance to the flash does not matter.

Previous student also focused on detecting the exact flash rate of the beacon to further confirm the target flash. But I do not believe it adds on much to the vision system in this project simply because there is no other beacons flashing at a different rate.

It is necessary to confirm the detected bright spot is flashing when looking for it, in case some bright reflection has been mis-detected by the thresholding. To detect a flash beacon, a 15-frame (about 1 second) buffer record was created. It filled with results of whether a spot was seen in that particular frame of video. To decide whether a seen spot was a flash beacon, it only need to count the number of frames with flash in. If about half frames says seen and half does not(according to the flash rate), then a flash beacon is detected in the current view; if all 15 frames reported seen spot, then that was not a flash beacon. The AI should stop the AUV for one second before asking the vision to find a flash beacon, otherwise the 15 frames may not showing the same view. Once the flash beacon is confirmed, tracker comes to action. The tracker can have a buffer build-in to tell whether the beacon is "flashing". So the 15-frame method would not needed.

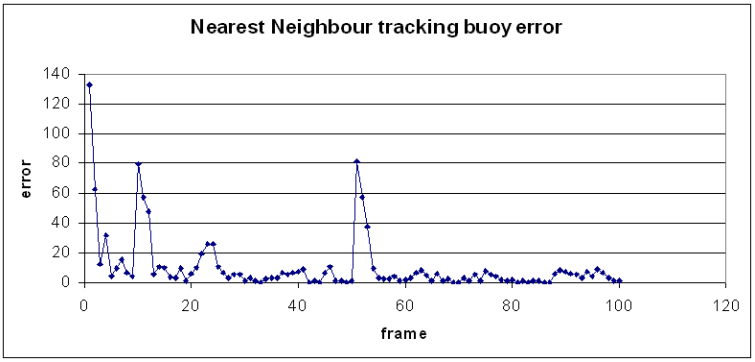


Figure 4.15: Nearest Neighbour tracker's error distance when tracking a underwater buoy. The tracker had times when confused by the error segments came from the colour classification.

4.2 Tracking Object

Once an object is confirmed to be the target, the location of the most likely blob is passed to the tracker. Tracker is used to predict the most possible location of the object in the next time instance without any new measurement. It estimates location when object is occluded or temporary leaves the view. There are two different situations in tracking – moving camera and stationary camera. The best algorithm could be different in each case. Two trackers implemented – Nearest neighbour and Kalman filter. The condensation had to be gave up due to limited time.

4.2.1 Nearest Neighbour

The simplest tracker, based on the nearest neighbour. It assumes the next location of the target is the blob that is nearest to the current location. There was no implementation problem ran into. Figure 4.15 is the error vs frame number plot when tracking a underwater bouy, because there were times that the colour segmentation give more than one target blob, made this tracker confused. Figure 4.16 is the error vs frame plot of it tracking a flashing beacon underwater. From the graph, it gives a merely satisfactory result. When the flash is on, segmentation gives a accurate reading, so the tracker gives a accurate reading; when flash is off, no more measurement come in, the tracking simple reports where the last time see the flash. There were time that the tracker is 'lucky' that flash moves to the tracker. But in general, the tracker slides away when flash is off. This track relies on accurate location measurement.

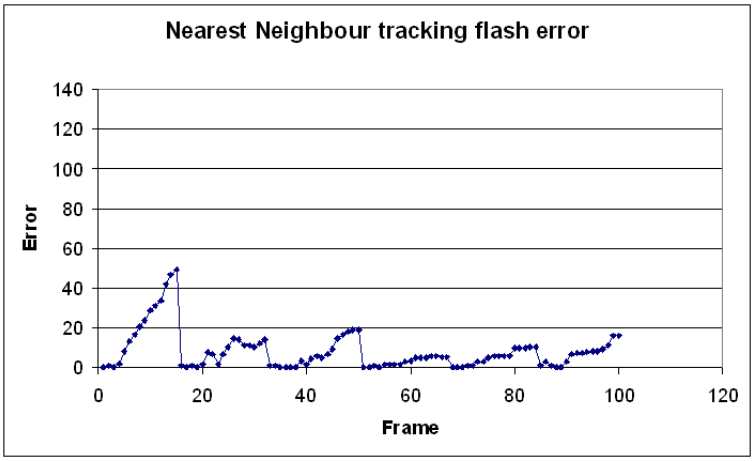


Figure 4.16: Nearest Neighbour tracker's error distance when tracking a underwater flash beacon. The tracker gives a good reading when flash is on, the error increases more and more once the flash is off.

4.2.2 Kalman Filter

One of the most effective motion tracker, Kalman filter has been applied in many motion tracking application. It is a recursive estimator, as already mentioned in literature review. It operates on predict & correct cycles. It was a steep learning curve when attempting to understand the tracker. I am going to use a simpler and less statistical manner to explain how it works.

First of all, Kalman filter is a general estimator, many behaviour that can be modelled by a formula can be estimated by a Kalman filter. In our case, the motion is modelled and the location at specific time is estimated. Kalman filter consists of a state vector, a transition matrix, a process noise matrix, prediction error matrix and measurement noise matrix. A state in Kalman filter in this case is a vector of location, speed and acceleration

$$\vec{x} = [s_x, s_y, v_x, v_y, a_x, a_y]^T \tag{4.5}$$

where s, v, a are location, velocity and acceleration respectively, x indicates in horizontal direction and y in vertical direction. They were initialised to 0. The state transition matrix (formula 4.6) is the motion model written in a matrix format, allowing it to transform a state vector to the next time instance.

$$\begin{pmatrix} 1 & 0 & dt & 0 & \frac{dt^2}{2} & 0 \\ 0 & 1 & 0 & dt & 0 & \frac{dt^2}{2} \\ 0 & 0 & 1 & 0 & dt & 0 \\ 0 & 0 & 0 & 1 & 0 & dt \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.6}$$

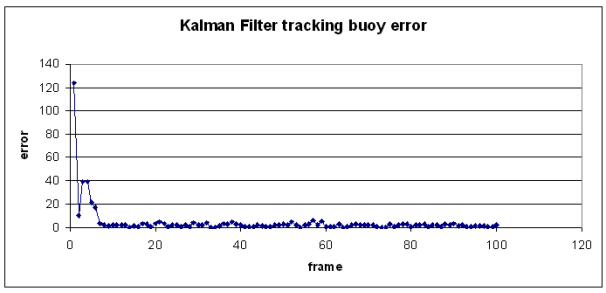


Figure 4.17: Kalman tracker's error vs frame number when tracking an underwater buoy. Although there were times that the classification gives an error reading, the Kalman filter still predicts the location accurately. High error at beginning is because the initialisation location of the tracker.

dt is the time difference between two consecutive images. This matrix does not change through out the whole tracking process. Process noise matrix Q describes error due to the motion inaccurately modelled by the motion formula. It is initialised to identity matrix. Prediction error matrix P stores filter's internal confidence about the current prediction. It is initialised to 0 and changes by the filter through out the process. Finally the measurement error matrix R gives a way to tell how much error are there in the inputting measurement.

The Kalman filter is first given an location measurement at time t and a confidence of that measurement. Internally, Kalman filter advanced itself t by applying the state transition matrix at time $t-1$: $\vec{x}_t = A\vec{x}_{t-1}$, getting its location and confidence at time t. Now there are two sets of locations and confidences. Kalman filter then take into account of both location and its confidence, producing a best estimation at time t. Filter's internal confidence is also updated at this time using the new measurements. This process is repeated until all frames are tracked.

Depending on how the measurement took place, the way to calculate the confidence of a measurement differs. In this project, the target was found by using colour segmentation. The average Mahalanobis distance from the testing colour to the training colour is therefore used as a confidence measure. The smaller the distance, more confident the measurement is.

Initial test shows Kalman filter's strong tracking ability on slow moving buoy (Figure 4.17). Even through there were times that the classification gives an error reading, the Kalman filter handled it with its internal confidence about the location. The initial testing on tracking a flash beacon had similar behaviour as the Nearest Neighbour (Figure 4.18). The author believe that is because the motion of the testing video. The video is captured with a hand-held camera, the movement was deliberate created by shaking the camera. More carefully captured videos need to be used to find out the true performance of the tracker in the evaluation.

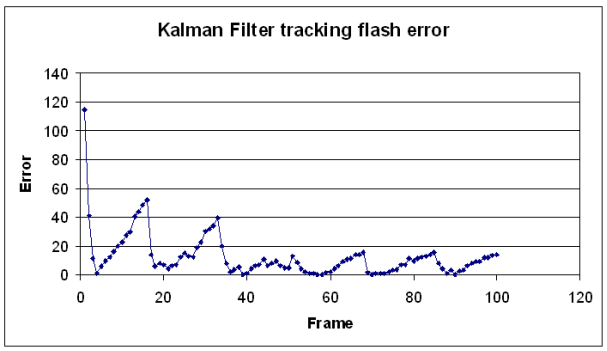


Figure 4.18: Kalman tracker's error vs frame number when tracking a underwater flash beacon. It had very similar performance as Nearest Neighbour tracker above. High error at beginning is because the initialisation location of the tracker.

Chapter 5

Shape Recognition

So far the focus has been on finding certain colours in a image given some colour training samples. The image was segmented into interested region (target) and uninterested region (background) depending on colour. This chapter concentrates on the common next step once the image is segmented – finding its shape detail. Fourier Descriptor, Shape Descriptor is implemented; dimension reduction are discussed with implementation details. The second half of this chapter moves to 3-dimensional object recognition using feature detectors. SURF feature and its implementation is discussed.

5.1 2D Shape classification

Using previous chapter's result, the binary mask of a potential target is the input for the shape recognition. Binary mask is a common way to store shape information because the detail of a shape is in its edge contour and inside region. The texture of an object does not affect the overall shape of that object – cross with mesh texture and cross with horizontal line texture should all be classified as cross. Two techniques implemented in this project for shape recognition work on different basis. The Fourier Descriptor relies on one measurement alone – the distance of the shape's contour to the centroid. It is considered as a sophisticated approach to shape representation. The original shape can be recovered by using this representation. The image of a shape in the underwater environment is rather noisy, and contour is very vulnerable in a noisy image, making this approach weak. The shape descriptor on the other hand, is a crude approximation to the shape. Only a fragment of certain features from the shape are captured by the descriptor. Although regions are more stable than contours, due to lighting change and heavy noise, the segmented region of shape can also have empty spaces inside, making Shape Descriptor less stable.

5.1.1 Fourier Descriptor

As discussed in the literature review in 2.3.1, the Fourier Descriptor is one of the most widely used shape-description algorithms. It has already been demonstrated by previous competition entrants that it has strong descriptive power even underwater.

Given a shape, the Fourier Descriptor works in the following way: it finds the centre of a shape $C = (c_x, c_y)^T$ and calculates the Euclidean distance from the centre to all the boarder points

$$B = \begin{pmatrix} b1_x & b2_x & ... & bN_x \\ b1_y & b2_y & ... & bN_y \end{pmatrix} \tag{5.1}$$

where N is number of points in the boarder. Using Euclidean distance formula 4.3 on page 35, we get a list of distances.

$$D = (\sqrt{(c_x - b1_x)^2 + (c_y - b1_y)^2}...\sqrt{(c_x - bN_x)^2 + (c_y - bN_y)^2}) \tag{5.2}$$

This list of distances are then Discrete-Fourier-Transformed to get a list of frequency components (complex numbers). The first element of the Fourier components (d.c. component) is the average value of the x and y co-ordinate, which is the centre of the boundary expressed in complex form. It is discarded. The magnitude of the rest of the complex numbers are calculated to describe the strength of a particular frequency. These magnitudes are already ordered in lower frequency to higher frequencies, so the descriptor is *rotation invariant*. Scale invariant is achieved by normalising these magnitudes by dividing the area under the magnitude graph. The whole list itself is the most accurate descriptor of the input shape. But the longer the descriptor, the higher the dimension of the feature space, which slows down matching, and makes it more vulnerable to noise. But too little descriptor gives a loose description of the shape, which also leads to mis-classification even through it is robust to noise and runs faster. First a few low frequency components are taken to represent the shape instead of the entire list. Sixteen was chosen by plotting the curve on Figure 5.1.

In this implementation, the centre of a given binary mask is the centre of the mask's bounding box[1]. This centre point may not be even inside the shape itself for certain non-convex shape. It creates confusion between the slender Moon and a round moon where the distance from the centre of the bounding box would be very similar in both cases (Figure 5.3). This could easily solved by finding the true centre of the shape. Because the shapes we are attempting to distinguish are relatively simple, and for ease of programming, the bounding box's centre was used. There were two ways of calculating the border signal. One way is walking along the contour and, for every point on the contour, calculate the distance to the centre point. The other one uses angle. For every degree(1° to 360°), emit a ray from the centroid, calculate the distance the ray travelled before hitting the contour. From implementation point of view, none of the methods pose difficulty. The first method was used because the latter fixed the resolution of the contour, and hence affects the descriptive power. For big shapes on the image, 1° is too far apart when the ray hits the contour.

[1] the minimum upright rectangle that encloses the entire shape

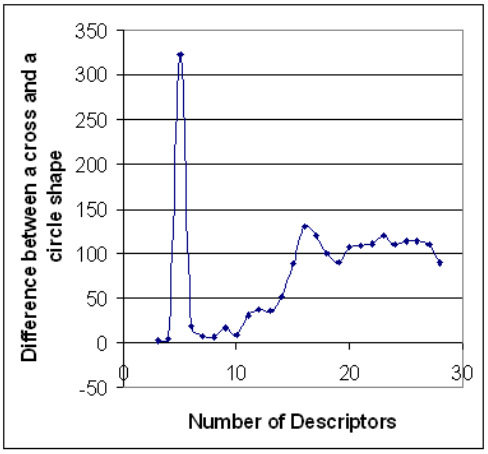


Figure 5.1: The number of descriptor vs difference. Two shapes, cross and circle was described by different number of Fourier Descriptors. The high point at 3 was a coincidence due to inaccurate description; the best number found was 16, where enough descriptors for describing the shape while high frequency noise have not introduced as much.

It could easily miss important high frequency corners. While for small shapes, 1° could give many repeated readings. The former method avoids such problems because it captures every single possible variation the contour has – although this could also mean that the noise is kept too. But it does not matter since the Discrete Fourier Transform and the selection of descriptor handles the noise problem more appropriately later. Once this list of distances are obtained, they were treated as frequency signals and passed into an Discrete Fourier Transform function. This function returns a list of complex numbers split into real and imaginary parts. The first DC component was dropped as explained before, the magnitude of these complex numbers were calculated by $|Complex| = \sqrt{real^2 + imaginary^2}$. The first 16 magnitudes calculated were taken and normalised. this is the descriptor of the input shape (Figure 5.2). Once the descriptor vector is obtained, the classification process can be carried out.

5.1.2 Shape Descriptor

Shape Descriptor is another candidate for real-time shape description. Instead of focusing on the features allowing recreation of the shape, Shape Descriptor captures fragments that enables distinguish between various types of shape. Six features were chosen as the descriptors:

$$Formfactor = \frac{4\pi Area}{Preimeter^2} \tag{5.3}$$

$$Roundness = \frac{4 \cdot Area}{\pi MaxDiameter^2} \tag{5.4}$$

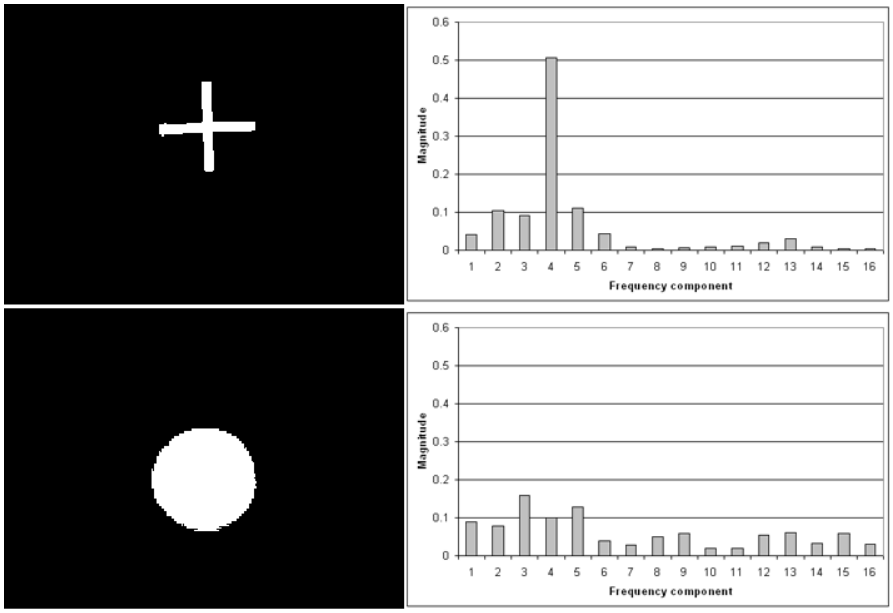


Figure 5.2: Top: Cross's binary mask and its 16 Fourier Descriptor; bottom: Circle's binary mask and its 16 Fourier Descriptor. There is a clear distinction between two descriptors

$$AspectRatio = \frac{MaxDiameter}{MinDiameter} \tag{5.5}$$

$$Convexity = \frac{ConvexPerimeter}{Perimeter} \tag{5.6}$$

$$Solidity = \frac{Area}{ConvexArea} \tag{5.7}$$

$$Compactness = \frac{\sqrt{\frac{4}{\pi} \cdot Area}}{MaxDiameter} \tag{5.8}$$

The implementation of calculating each feature from an binary mask was reused from a previous student's work, preventing repeated effort on the same piece of code. Figure 5.4 is the same slender moon and moon described by the Shape Descriptors. They have a clear distinction this time.

5.1.3 Dimension Reduction & Classification

In previous sections, the concentration was on the formation of a descriptor. Given a shape, Fourier Descriptor or Shape Descriptor can return a vector identifying the shape using their own measurements. Classification then uses the descriptors to make decisions. Classification was already used in section 4.1.1 for colour detection. Recalling the process, each colour in RGB has three features – Red, Green and Blue. (Hue Saturation and

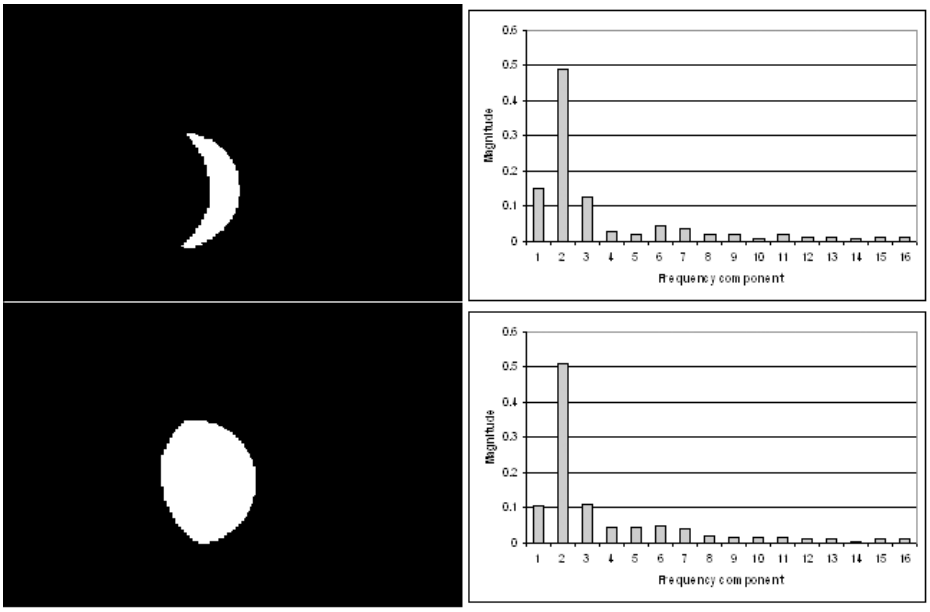


Figure 5.3: Top: Slender moon's binary mask and its 16 Fourier Descriptor; bottom: Moon's binary mask and its 16 Fourier Descriptor. There is no clear distinction between their descriptors, they could get mixed up when classifying.

Intensity in HSV case) One such vector represents one and only one colour in RGB space, thereby appearing as a single point in the RGB space. After plotting all the training colours in the space, an eigenmodel was built covering the training data in order to take into account the distribution pattern. To classify whether a testing colour is belonging to the training set, Mahalanobis distance was calculated and a threshold was applied. With shape classification, the situation is only slight differ. A Fourier Descriptor for example, is a 16-dimension vector instead of 3, but it is still possible to plot a point in a 16 dimensional space to represent that vector. Then the rest of classification process is the same.

High dimensionality is a serious problem in terms of classification accuracy and computational complexity. With same amount of the training data, it is much easier to find the distribution pattern in 1 dimension compared to say 3 dimensions. This because in lower dimension, there is significantly fewer possible locations (values) the points could have, hence producing a clearer trend compared to that in a the higher dimension. The performance impact comes from the need for more training data as well as the huge search space created by the high dimension. In practice, quite often the features that were chosen are correlated with each other. Karhaunen-Loeve Transform or Principal Component Analysis [Acharya and Ray, 2005] is an effective way to eliminate the repeated information in the features. Using the basic matrix transformation: rotation and translation of points can also be represented by rotating and translating the base vector. To transfer points from a lower dimension to a higher dimension, it only needs to find the correct vectors in the high dimension that covers the lower dimension. These vectors are concatenated to form

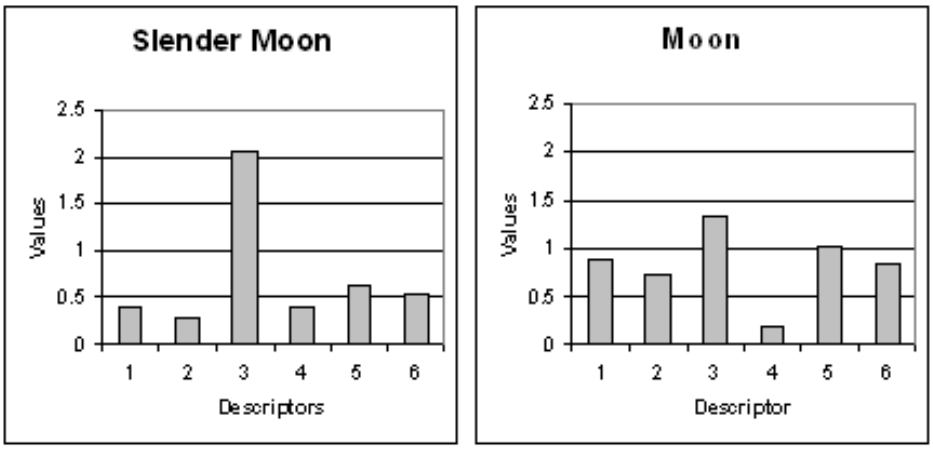


Figure 5.4: Same slender moon and moon described by the Shape Descriptor. They are clearly different.

the projection matrix M in formula 5.9.

$$P_m = MP_n, \tag{5.9}$$

where P_m and P_n is the vector in m and n dimension respectively and $m > n$, M is the projection matrix of n wide and m high.

From formula 5.9, it is easy to flip the equation to find the projection matrix transforming the other way 5.10.

$$P_n = M^{-1}P_m \tag{5.10}$$

The Moore-Penrose pseudo-inverse operation was used to invert the non-square matrix M, it was discussed in Moore [1920]. The most important property of the pseudo-inverse we used is

$$MM^{-1} = 1 \tag{5.11}$$

which allows the deduction from formula 5.9 to formula 5.10 by multiplying M^{-1} on each side.

For a given set of high dimensional data, eigenvectors describe the most varied direction of the data set, and they often do not full-fill the entire dimension the data were originally in. The eigenvectors with non-zero eigenvalue were used as new axes for the reduced-dimension data.

Five shapes of 50 samples each was used as the initial testing data. Their descriptors were calculated by apply the Fourier Descriptor (or Shape Descriptor). The eigenvectors and eigenvalues were worked out for this list of 250 observations containing 5 shapes. Since the function already deflated and arranged the eigenvectors in descending order, first a few eigenvectors with non-zero eigenvalues were going to be used as the new base for the entire data set. Those eigenvectors were concatenated into one matrix, and its inverse was computed using the pseudo-inverse function. The resulting matrix is the project matrix. This

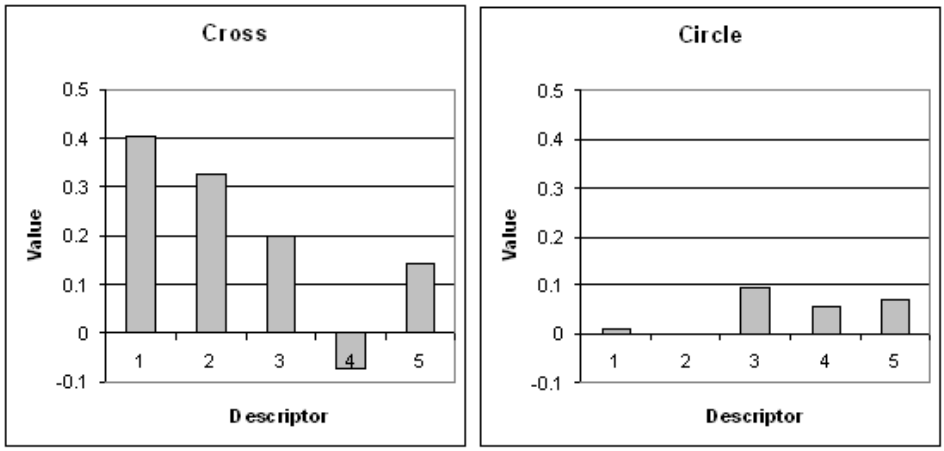


Figure 5.5: After dimension reduction, Fourier Descriptor can distinguish between cross and circle more clearly using fewer descriptors.

entire process was implemented in one function, taking a matrix of observations, returning a projection matrix. It enables a simple and general use of the dimension reduction when attempting to classify. In the quick test, the 16-dimension Fourier Descriptor was reduced to 5 dimensions. This was a dramatic reduction that must improve the computation speed in later stages. But it also means there was a considerably amount of redundant information when using the original Fourier Descriptor to distinguish between those 5 shapes. Figure 5.5 shows the Fourier Descriptor after dimension reduction to distinguish circle and cross using only 5 descriptors.

Once the projection matrix is obtained, the original high dimensional feature-vectors can be projected to a lower dimension while keeping the key variations. Using the lower dimensional feature-vectors, each shape class can build its own eigenmodel in a lower dimension and calculate the Mahalanobis distance in the lower dimension. There is an extra step now when attempting to classify – projecting the test shape's feature-vector down – but that's no more than one matrix multiplication.

In section 4.1.1 on page 33, we interested in deciding whether a testing pixel is belonging to a training colour or not. A threshold value was used for the optimal answers based on best precision and recall. Now the problem became deciding which shape classes the testing shape is most likely to belong to. The test shapes' Mahalanobis distance to *each* shape class were computed. After an ascending sort operation, the first shape class of this sorted list (shortest distance) is returned as the shape of the test input.

The Fourier Descriptor is contour based, the Shape Descriptor is region based. It is interesting to find out how their hybrid performs, especially with dimension reduction function on hand, the high dimension could be reduced to a reasonable size. Detailed evaluation of these two classifier and their combination are in discussed in chapter 6 on page 54.

5.2 3D Objects

In the later tasks, such as cone- or tyre- matching, using only shape could be inefficient. The cone and tyre are 3 dimensional objects, their perceived shape (projection) would change as viewing angle changes; for example, the upright tyre can look like a "ring" from the side, and it can look like a rectangle when viewed from the front, it can also appear as a series of different shapes when viewed from an angle. A feature detector is used. They work ground up, directly analyse the original colour image instead of the binary mask. They find any "interesting" point and describe it using a feature vector, which later used to deduce similarities between images. But these feature detectors tend to computationally expensive, many existing applications are not in real-time. The SURF feature detector focused on improving the speed while not reduce the accuracy. It has the best balance on accuracy and speed, so the author decided to evaluate it in this project to find out the feasibility of such technology been used in real-time application.

5.2.1 SURF

Their is a SURF implementation available to the author. Given an image, the SURF finds all the interesting points, and each described by a 64-dimension feature vector. When matching one image against another, every interesting point in one image must compare to every interesting point in the other. Images reported match if there are a fair amount of matched interesting points.

To use the SURF in this project for target recognition, a database of images containing target is loaded at the beginning. Then for every query frame, the SURF feature is computed and compared to all the images in the database. A matched target is reported in the current frame for the best matched database image. Except loading the database image at the beginning, all steps should run in real-time.

There are two processes that potentially slow down SURF. The computation of features of an image, and the matching between images. The former has direct affect on the latter because if more features found, more comparisons has to be made when matching. There are two direct ways that can limit the number of features detected – lower the sensitivity of the detector and lower the resolution of the image. It is clear that both sacrifices the accuracy. Lower sensitivity means only strong features are detected, those strong feature may not necessarily on the object; lower resolution of image kills the details straight away. There are other ways the author also investigated into. When detecting features in query frame, make it only detect the feature actually on the object itself instead of the entire image. For instance, segment the target before using feature detector. But this could lead to dependency on the accuracy of the segmentation process. The extra step and the new problem arise may compensate the benefit. Through, this option is open. A balance between the accuracy and the speed must be found for using in this project.

How to quickly and reliably match hundreds of high dimensional vectors against another is

an active research area. One method suggested by Bay et al. [2006] is this. An interesting point in the query image is compared to an interesting point in the database image by calculating the Euclidean distance between their descriptor vectors. A matching pair is detected if its distance is closer than 0.7 times the distance of the second nearest neighbour. Another Method uses angle between vectors instead of distance between high dimensional points. The dot product is calculated between a interesting points in query image and every interesting points in database image. The angle between vectors are then calculated using acos function. An interesting point pair is considered match if its best matched angle is smaller than 0.6 times the second smallest angle. Both methods are all based on the nearest neighbour ratio match strategy [Baumberg, 2000], so significant accuracy difference is not expected from between them. These two matching algorithms are all implemented to test if one runs faster than the other.

In the second matching algorithm, there were some optimisation took place. Acos function is expensive to compute, costs add up when it is used in a double loop. A lookup table was used to replace the acos function.

Chapter 6

Evaluation

Images, test results produced in chapters 4 and 5 are mostly developmental test results. They only serves the purpose of finding out whether the code written was correct and give the author initial impressions of how the algorithm might perform. In this chapter, the aim is to access all the strengths and weaknesses of each implemented algorithms via systematic testing. Instead of just conclude simple good or bad distinction, the good algorithms are put in extreme condition to find their limits. The focus are remain on the task on hand, only the conditions that are potentially resides underwater are considered.

6.1 Test Aim

- To find out how each algorithm could perform in the simulated competition environment.
- To find out the limitations of algorithms as situation gets worse.
- To conclude the best set of algorithm or hybrid of algorithms that are going to be used towards the competition.

6.2 Variables

When attempting to find out how certain condition affects the output of a system, we tend to fix all other conditions and vary only that condition. This is the principal of systematic testing. The conditions we would like to find out are: how distance affects colour segmentation underwater; how shapes, rotation, scale, projection affects shape recognition as well as how these factors make difference in 3D object recognition. Other factors, for example, water clarity, water depth, ambient lighting and camera sensitivity are also want to be find out. But due to the limited time and resources for testing, they have to be left for further investigation. The ambient light condition could not be controlled or measured

in any meaningful way without a proper tools. It can be assumed to be constant as most of the videos are took in the same pool in one single session.

The variables are:

- Distance between the camera and the target
- Orientation of the target
- Projection of the target
- Shape of the target
- Colour of the target
- Motion of the target

6.3 Shape design

Shape design, see figure 6.1

6.4 Test plan

Shapes for testing shape recognition algorithm are described in figure 6.1 with explanation of why choosing that shape. Those shapes are made into different colour, they can also be used in testing the colour recognition. The test-video capture plan is in the appendix.

6.4.1 Test data gathering

As listed in the test plan (Table 6.1), the footage that enables the testing for each objective will be in 5 categories too, with some of them been reused. For detailed test-video capture plan, see A.1 in appendix.

6.4.2 Training data gathering

Training data is captured separately from the testing data or specially selected from the testing data. In colour recognition, colour-training samples are taken from the testing videos at different distances. The amount of samples at different distance are kept same. No training data is needed in flash beacon recognition and tracking. In shape classification, ground-truth binary-masks of the shape are used for training. No projection of shapes are included in the training images, as it is expected to be handled by the algorithm themselves. For 3D recognition, training images are taken from the testing video.

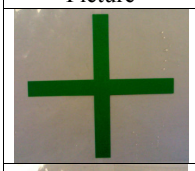
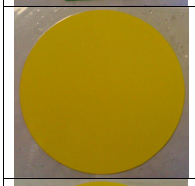
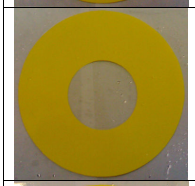
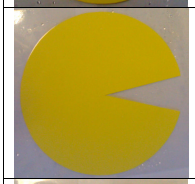
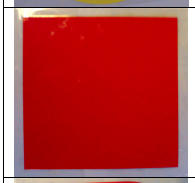
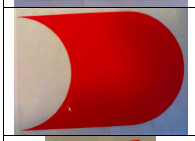
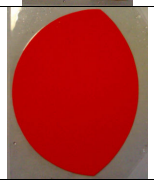
Picture	Name	Colour	Description
	Thin Cross	Green	Whether the algorithm can recognise cross as it will be in the competition
	Fat Cross	Green	It made from same sized square as the thin cross. Testing how fatness affects of being classified as cross. Cross may have a different fatness in competition compare to the cross at hand.
	Circle	Yellow	Whether the algorithm can recognise circle
	Ring	Yellow	Whether the algorithm can distinguish between circle and ring
	Pac Man	Yellow	Whether the algorithm can distinguish between circle and Pac man
	Square	Red	Whether the algorithm can recognise square
	Modified Square	Red	This shape has exactly same area as the square. Testing whether algorithm can distinguish it from square
	Slender Moon	Orange	Whether the algorithm can recognise more interesting shape
	Moon	Orange	Whether the algorithm can distinguish between Slender Moon and this one

Figure 6.1: Shape Design

To test	Algorithms	Targets	Variables
(1) Colour based target recognition	HS(V) Euclidean HS(V) Mahalanobis RGB Mahalanobis	Shapes made with red, orange, yellow and green colour	Distance
(2) Flash beacon recognition	Brightness threshold	Flash beacon Always on LED No light	Distance
(3) Tracking	Nearest Neighbour Kalman Filter	Buoy Flash beacon	Distance Target motion
(4) Shape recognition	Fourier Descriptor Shape Descriptor Hybrid	Fat/Thin cross Circle, Ring, Pac-man Square, Modified square Slender Moon, Round Moon	Distance Shift Orientation Projection
(5) 3D object recognition	SURF	Cone Tube	

Table 6.1: Test plan

6.5 Colour based target recognition

Colour based segmentation is first put on test. Video footages of all 9 shapes made are captured underwater. The camera was placed at 1 meter away, the shapes were rotated, tilted and moved around in the camera view; the camera then slowly moves back to about 3 meters, same action were applied to the shape. No automatic colour corrections or exposures of the camera were turned on, thereby limiting the possible bias from the camera's automatic adjustment. Each algorithm were given same training image, a grey-scale distance mapping is produced. Darker the pixel colour, more likely the original colour is belong to the training colour set under that algorithm. In distance map, if the target is significantly darker than the background, a threshold would easily segment the target out from the background.

6.5.1 Colour & Distance

Colour and distance need to be discussed together. Changing colour alone do not give a meaningful conclusion. Different colour has been affected differently when viewed underwater as distance changes. Red colour is affected most. The red became dark grey from only 3 meter away (Figure 6.2), while as yellow hardly changed the observed colour (Figure 6.3).

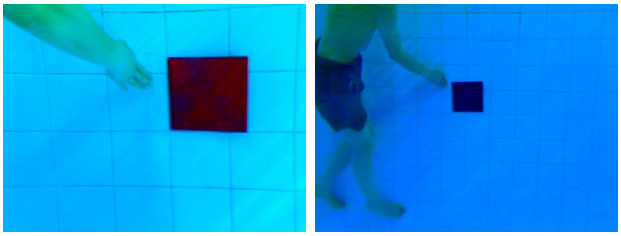


Figure 6.2: Red square appears as dark grey in distance

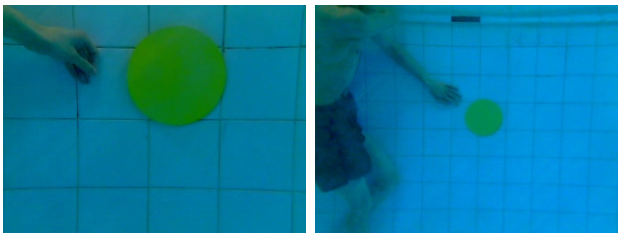


Figure 6.3: Yellow did not change significantly as distance increase

When colour changed significantly, for example, red became dark grey, RGB Mahalanobis distance worked better than HS(V) distance (Figure B.1-B.6 in appendix); When colour only had slight change, for example, green became darker green, HS(V) had a clear advantage. This can be explained by the nature of the selected features. In HS(V), the intensity channel is dropped to make the change from green to dark green unnoticeable. But since the red has lost its colour when moved far way instead of became darker, so dropping the intensity channel would *lose information* for describing the new colour. While RGB kept all the information, although it misclassified the person as "red" too, it did not miss the important areas inside the red square. The other two colours had same result, orange like red – lost colour when far away; yellow like green – became darker when far away.

HS(V) Mahalanobis distance always gives a better contrast between the darker one and lighter ones. So if the classification is correct, for example green on figure B.5, Mahalanobis distance gives a more stable result (bigger difference between target and background) than Euclidean distance.

Unfortunately, the precision of both methods are quite low. The person and the hand in the most of the images would also be segmented out once thresholding is applied. This confirms to the author that an extra stage of shape classification is necessary before deciding a target. Colour alone is not enough.

6.6 Brightness Based Flash Beacon Recognition

While the colour segmentation performed not as good as expected, the brightness thresholding gives a satisfactory result.

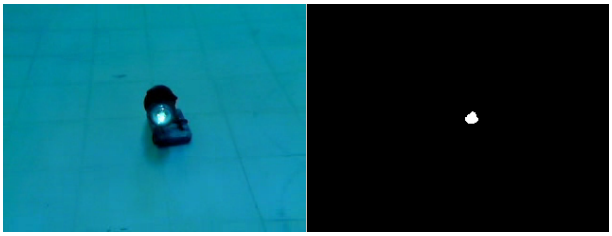


Figure 6.4: The flash beacon segmented out correctly when it is near using global static threshold value of 0.4

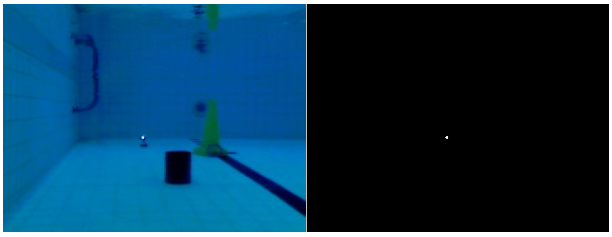


Figure 6.5: The flash beacon segmented out correctly when it is far away using thresh value of 0.4

6.6.1 Distance

All thresholding was set on a fixed value of 0.4, the flash beacon at different distances are all correctly segmented out with one static value as shown in figure 6.5 and figure 6.4. No misclassification was detected when the flash is off (Figure 6.6). But this method could not distinguish between bright spot if there are multiple one in the view. The brightness value alone would be insufficient. That is where other features of the flash beacon needed to be considered; for example, the flash rate or template matching of the entire device instead of just the flash spot. For this project its application, this technique is sufficient.

6.7 Tracking

6.7.1 Target Motion

The camera would installed on the AUV and follow AUV's movement, when capturing testing video, the camera was moved to simulate AUV's movement. In order to investigate how different motion affects tracker's performance, constant speed, acceleration and rotation were simulated by moving the camera. Figure 6.7 shows the error vs frame when Nearest Neighbour tracker and Kalman filter tracks a constant speed moving underwater buoy. The Kalman filter showed it strong ability to smooth down an error when measurement went wrong, while the Nearest Neighbour tracker suffered measurement error seriously. The biggest error the Kalman filter made was 19-pixel while as the Nearest

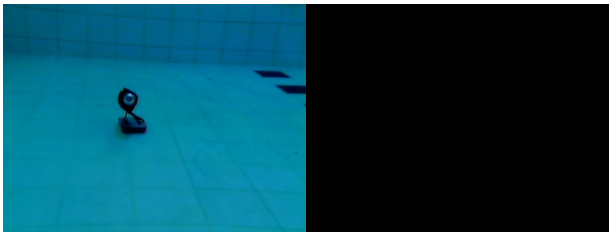


Figure 6.6: Nothing is segmented out when flash is off using the same thresh value of 0.4

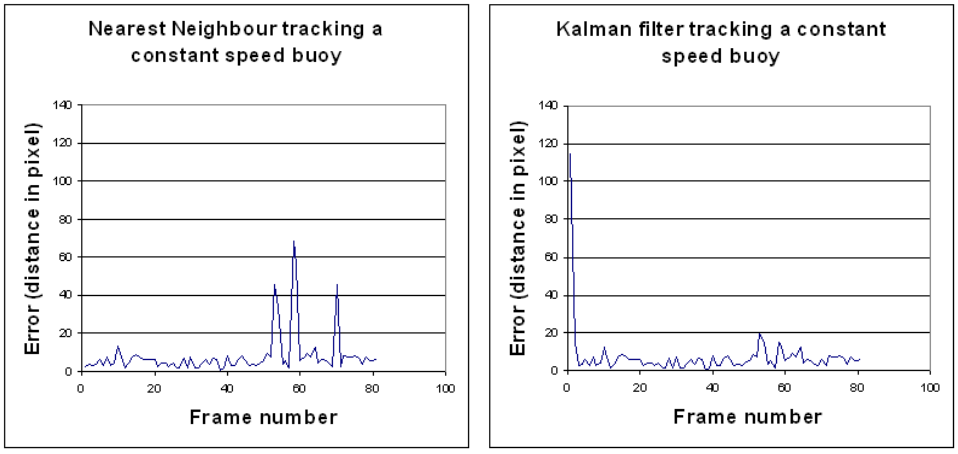


Figure 6.7: Left: Nearest Neighbour tracker's error distance to track a constant-speed moving mid-water buoy, average error distance 8-pixel. Right: Kalman tracker's error distance to track the same buoy (The first high error was due to initialisation) average error distance 5-pixel. There are several error peak in the Nearest Neighbour tracker due to classification error at segmentation stage (worst case 68-pixel). Kalman tracker has those peak too but at a much lower level (19-pixel). That's because the Kalman's internal prediction smoothed the error out.

Neighbour tracker missed 68-pixel away from the true target location.

Both tracker also tracked a constant speed moving flash beacon. Figure 6.8 show their error distance. When the flash is on, both tracker tracks the bright spot reliably, within 10-pixel distance. But when flash is off, Nearest Neighbour tracker stopped at where it last seen the flash, therefore the error distance increases as time passes; the Kalman tracker modelled the motion using the previous frames, tried to follow the unlit flash. The error distance is significantly lower, 40-pixel in Kalman compared to 81-pixel with Nearest Neighbour tracker. The author have to accept that the "hand controlled" constant speed was not exactly "constant", there were sudden moves when capturing the video underwater, made the Kalman tracker missed direction the flash was moving when the light was on. The Kalman tracker could performed better if the motion was more stable.

As the Nearest Neighbour tracker's behaviour is now understood fully, only Kalman filter

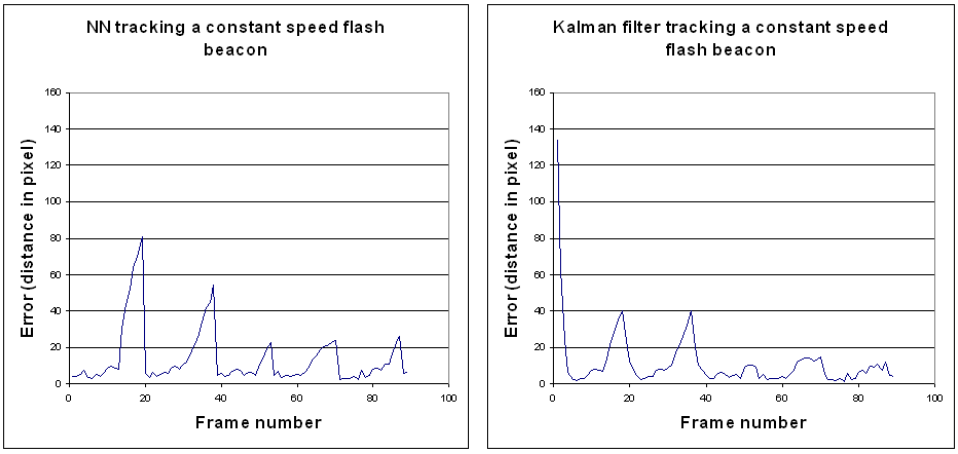


Figure 6.8: Left: Nearest Neighbour tracker's error distance to track a constant-speed moving flash beacon underwater, average error distance 14-pixel. Right: Kalman tracker's error distance to track the same flash beacon, average error distance 9-pixel. Both tracker gives big error reading when flash is off, but the Nearest Neighbour tracker suffers more – worse case was 81-pixel away – when flash is off; the Kalman filter was trying to predict the motion, so worse case is lower, 40-pixel away. The first high error in Kalman was due to initialisation

is put through to track following more complex motion.

Due to the limited opportunity to capture underwater videos, the author had to capture other motions on the bench. It also gave a more accurate control over the motion actually captured. An acceleration and a circular motion of a flash beacon was captured and tested. Figure 6.9 (left) is the Kalman filter's error distance when tracking an accelerating flash beacon. The Kalman filter gave a better result compared to track underwater "constant speed" moving flash beacon, considering acceleration would be a more difficult motion to track. The worst case the tracker missed was 31-pixel compare to 40-pixel, and the average error was just above 5-pixel compared to 11-pixel. Figure 6.9 (right) showed the error distance of Kalman filter tracking a circular motion flash beacon. It gives even better result with worse case 14-pixel and average just below 4-pixel. The author conclude the high tracking error in the earlier underwater footages was because the footage captured was not actually simulating the constant moving, but rather a hand shaking complex motion.

Kalman filter is definitely out performs the Nearest Neighbour when tracking different motions.

6.7.2 Distance

In the way this program uses tracking, the distance would not have direct impact on the tracking itself. An input frame was first applied various segmentation and classification

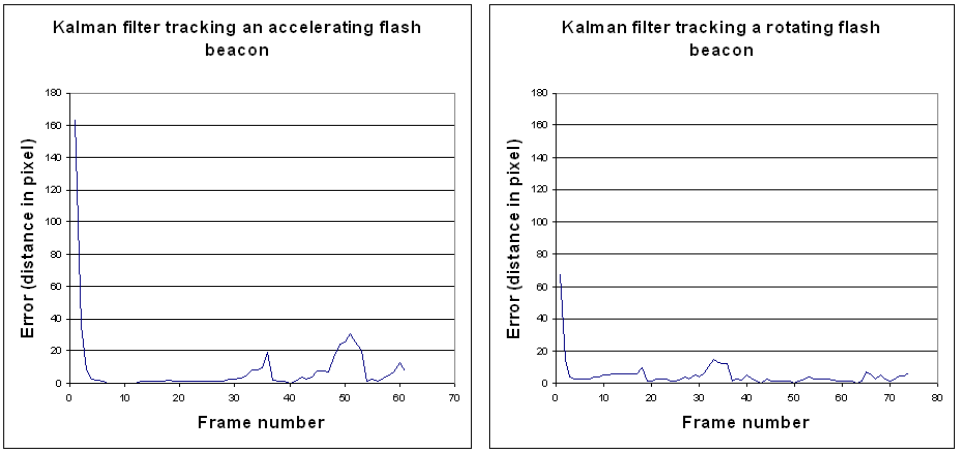


Figure 6.9: Left: Kalman filter tracking an accelerating flash beacon, the average error distance is 5-pixel. Right: Kalman filter tracking an rotating flash beacon, the average distance is just below 4 pixel. Both cases have a lower error rate compared to previous ones.

algorithm to find the most likely target, then the target's location passed to the tracker. When distance increases, as long as the target recognition do not fail, then the tracking is not affected. Figure 6.10 is the error vs frame for tracking the underwater flash beacon at 3 metre away. Both algorithm did not show affect come from the distance increase.

6.8 Shape Recognition

6.8.1 On the bench

Shape recognition first evaluated out of water, in the same environment as the training videos for each shape were filmed. These bench tests will give a benchmark for the algorithms, representing the best performance they are capable of, and will serve as a good comparison to any results gained underwater. All three testing algorithms are trained with same set of training video, the testing video used was the training video itself to find the best possible classification result each algorithm can deliver.

Table 6.2 to 6.4 are the confusion matrix for Fourier Descriptor, Shape Descriptor and their combination. Fourier descriptor, as expected, the circle and the ring are mixed up. Because Fourier Descriptor is contour based the circle and the ring have a same contour, therefore Fourier Descriptor could not distinguish between them. It in fact gave a better result than expected. The author thought both the circle and the ring would all be classified as either circle or ring. The Fourier Descriptor must have noticed some slight difference between their contour. An supervising result from the Fourier Descriptor, the slender moon was not confused with the moon, which the author specially designed the pair to trick the Fourier

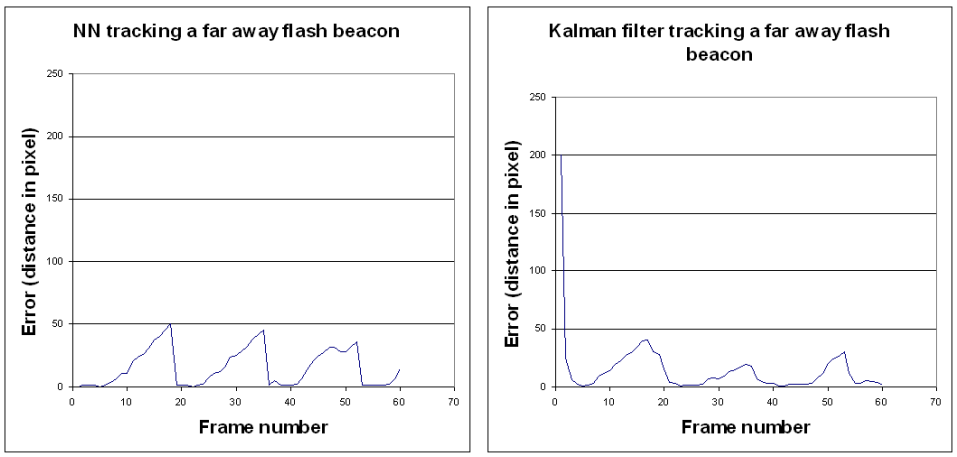


Figure 6.10: Left: Nearest Neighbour tracking a distant flash beacon, average error distance 15-pixel. Right: Kalman filter tracking a distant flash beacon, average error distance 11-pixel. Both result did not demonstrate *distance* affecting the tracking accuracy.

Descriptor. Figure 6.11 is the visualisation of the descriptor. They only differ slightly, which means they may get mixed up when conditions get worse. The overall classification correctness was 88%.

Shape Descriptor performed very well except one – circle have been misclassified to Pac-man many times when the segmentation left minor spaces inside the circle, 14% of circle has been classified as Pac-man. This showed the Shape Descriptor's weakness – do not record the exact spaces occurred on the shape. But it had a very high overall classification correctness – 97.8%.

Their hybrid, Fourier Descriptor plus Shape Descriptor did not performed as good as hoped for. Instead of combining the strength, it actually showed both algorithm's weakness. When classifying circle, 88% of them are classified as ring – the type of mistake would expected from Fourier Descriptor, 8% classified as Pac-man – Shape Descriptor's error, and lastly only 4% was classified as circle. While classifying the ring, 12% went to fat cross, which never happened in the individual algorithm. But I must say, except circle, ring and Pac-man, which more or less belongs to a same group, this hybrid performed better than any individual algorithm alone.

6.8.2 Underwater

The same evaluation carried out using footages captured underwater. The shapes were place at 1 meter away from the camera, which is slight further than the shapes were trained. Unfortunately, the square and m-square shape's footage was so bad that could not be used to produce meaningful result. Table 6.5 is the result for Fourier Descriptor. It demonstrated its weakness in describing shapes that constantly have broken contours. The

Classified as	Ground truth shapes								
	TC	FC	C	R	P	S	MS	SM	M
Thin Cross	80%	0	0	0	0	2%	0	0	0
Fat Cross	0	96%	0	0	0	0	0	0	0
Circle	0	2%	62%	38%	4%	0	0	0	0
Ring	0	2%	38%	62%	0	0	0	0	0
Pacman	0	0	0	0	96%	0	0	0	0
Square	2%	0	0	0	0	98%	0	0	0
M-Square	0	0	0	0	0	0	100%	0	0
Slender Moon	0	0	0	0	0	0	0	100%	2%
Moon	0	0	0	0	0	0	0	0	98%

Table 6.2: Fourier Descriptor's confusion matrix when classifying shapes which trained the classifier.

Classified as	Ground truth shapes								
	TC	FC	C	R	P	S	MS	SM	M
Thin Cross	100%	0	0	0	0	0	0	0	0
Fat Cross	0	96%	0	0	0	0	0	0	0
Circle	0	0	86%	0	0	0	0	0	0
Ring	0	0	0	100%	0	0	0	0	0
Pacman	0	0	14%	0	100%	0	0	0	0
Square	0	0	0	0	0	100%	0	0	2%
M-Square	0	4%	0	0	0	0	100%	0	0
Slender Moon	0	0	0	0	0	0	0	100%	0
Moon	0	0	0	0	0	0	0	0	98%

Table 6.3: Shape Descriptor's confusion matrix when classifying shapes which trained the classifier.

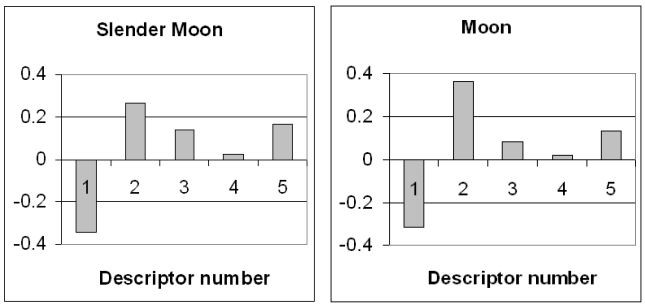


Figure 6.11: Left: The slender moon shape's Fourier Descriptor (after Dimension Reduction); Right: The moon shape's Fourier Descriptor (after Dimension Reduction). They look similar to each other, this means when noise introduced, they may confused with each other.

Classified as	Ground truth shapes								
	TC	FC	C	R	P	S	MS	SM	M
Thin Cross	100%	0	0	0	0	0	0	0	0
Fat Cross	0	96%	0	12%	0	0	0	0	0
Circle	0	0	4%	0	0	0	0	0	0
Ring	0	0	88%	86%	90%	0	0	0	0
Pacman	0	0	8%	0	10%	0	0	0	0
Square	0	4%	0	2%	0	100%	0	0	0
M-Square	0	0	0	0	0	0	100%	0	0
Slender Moon	0	0	0	0	0	0	0	100%	0
Moon	0	0	6%	0	0	0	0	0	100%

Table 6.4: Combination of Fourier Descriptor and Shape Descriptor's confusion matrix when classifying shapes which trained the classifier.

correctness fall below 0.5 happened on most of the shapes except the pac-man. While Shape Descriptor (Table 6.6) showed its region-based advantage. The thin cross and the slender moon were classified with 100% correctness. Although it experienced misclassification from the fat cross to the square, that is understandable when the fat cross under serious noise. One supervising result is no ring was classified as ring. The majority of the rings were classified as circles, which is very different from the benchmark of 100% correctness. The combination algorithm gives the best result of all. With the thin cross and the slender moon classified with 100% and most of the other shapes with more than 70% correctness. Since the fat cross never had been classified well in either of the individual algorithm, it only had 43% correctness, slightly lower than the Fourier Descriptor's correctness of 49%.

Classified as	Ground truth shapes						
	TC	FC	C	R	P	SM	M
Thin Cross	30%	0	0	0	0	0	0
Fat Cross	0	49%	0	0	0	0	0
Circle	0	3%	22%	46%	7%	0	4%
Ring	1%	7%	74%	54%	2%	60%	4%
Pacman	0	0	0	0	91%	0	0
Square	69%	41%	0	0	0	0	0
M-Square	0	0	0	0	0	0	0
Slender Moon	0	0	0	0	0	19%	31%
Moon	0	0	4%	0	0	21%	61%

Table 6.5: Fourier Descriptor's confusion matrix when classifying underwater shapes at 1 metre away.

Classified as	Ground truth shapes						
	TC	FC	C	R	P	SM	M
Thin Cross	100%	14%	0	0	0	0	0
Fat Cross	0	5%	0	7%	0	0	0
Circle	0	0	80%	0	0	0	0
Ring	0	0	0	93%	0	0	0
Pacman	0	0	4%	0	99%	0	0
Square	0	80%	0	0	0	0	0
M-Square	0	0	0	0	0	0	1%
Slender Moon	0	1%	4%	0	1%	100%	16%
Moon	0	0	12%	0	0	0	83%

Table 6.6: Shape Descriptor's confusion matrix when classifying underwater shapes at 1 metre away.

Classified as	Ground truth shapes						
	TC	FC	C	R	P	SM	M
Thin Cross	100%	0	0	0	0	0	0
Fat Cross	0	43%	0	0	0	0	0
Circle	0	0	71%	0	0	0	0
Ring	0	1%	23%	98%	9%	0	0
Pacman	0	0	0	2%	90%	0	0
Square	0	55%	6%	0	0	0	0
M-Square	0	0	0	0	0	0	0
Slender Moon	0	1%	0	0	1%	100%	17%
Moon	0	0	0	0	0	0	83%

Table 6.7: Combination of Fourier Descriptor and Shape Descriptor's confusion matrix when classifying underwater shapes at 1 metre away.

6.8.3 Noise Filter

Since algorithms like Fourier Descriptor already experience serious misclassification, continue making the condition worse would not lead to meaningful patterns. Noise filter is applied to the underwater footages used above, hope to bring back some correctness of each algorithm.

Table 6.8 is the result the Fourier Descriptor applied with an extra stage of median noise filter of window size 15. It is clear that the it had improvement. It doubled the correctness on classifying the thin crosses, reached 64%; the moon had 20% correctness increase compared to without the noise filter. But the pac-man shape experienced decrease in correctness, from 91% down to 64%. This can be explained by comparing the binary mask of the pac-man before and after the median filter in Figure 6.12. The sharp corners at pac-man's mouth area was rounded out, leaded to misclassification by the Fourier Descriptor. Gaussian noise filter of size 15 was also applied, the Fourier Descriptor had some improvement too, but not as good as the median filter in the correctness sense. Through the pac-man was classified more accurately under Gaussian filter than under median filter, with correctness of 76%, this is because the Gaussian filter did not *alter* the shapes' corner as strongly as the median filter. The other two algorithms are all had certain correctness increases too. One thing must be note is that after the noise filter was applied, the colour segmentation was able to perform more accurately and more stalely with the background noise removed, which lead to this general classification correctness increase.

6.8.4 Distance

The underwater shapes then placed at 3 metres away, same experiment was repeated. The result can be seen from table 6.9 to table 6.11. The thin cross shape was actually very hard

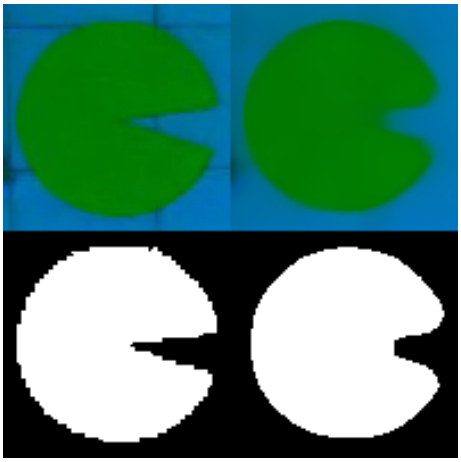


Figure 6.12: Median filter's effect. Left: image without median filter and its segmented mask; Right: same image applied with median filter of window size 15 and its segmented mask. It is clear that the pac-man's mouth area has rounded up, the original high frequency corner is smoothed out.

to see from the videos even by the author, the colour segmentation was unable to find it from the video or give a meaningful binary mask of the cross. It could be because the shape has a small visible area, and the visibility underwater is not good. It was excluded from the testing shapes. All three algorithms suffered serious impact. None of the algorithms could even identify the fat cross or the pac-man in distance. The Fourier Descriptor's 6% on the fat cross and 29% on the pac-man was the best of all three algorithms. Most of the misclassification of the fat cross went to the circle, the square and the moon. This is understandable as when the fat cross got further, its corners blurred made it classified as a rounded shape. The pac-man just had exactly the same problem. Except those mentioned two, the combination-algorithm performed correctly on the rest of the testing shapes. The noise filter applied had a much smaller window size of 3, big window size would distort the shape straight away. The noise filter helped a lot in segmenting the target. Without the noise filter, the author could not even get meaningful shapes out of the video simply because there were too much noise.

6.8.5 Projection

When the AUV approaching the cross and circle on the floor, their projection would be observed first. Certain degree of projection invariance is beneficial, but not essential.This experiment would find out which algorithm handles the projection best. Projections of the shapes was filmed underwater at 1 metre. The camera was hold still, the shapes was tilted in front of the camera. From table 6.12 to table 6.14 are the experiment result. It is quite clear that their ability to handle projection is limited, and the combination algorithm did not have any dramatic advantage. There is a common pattern, the circle, the ring, the fat

Classified as	Ground truth shapes						
	TC	FC	C	R	P	SM	M
Thin Cross	64%	0	0	0	0	0	0
Fat Cross	0	87%	0	0	0	0	0
Circle	0	0	17%	23%	0	0	0
Ring	0	0	68%	70%	36%	5%	1%
Pacman	0	0	1%	0	64%	0	0
Square	36%	13%	0	0	0	0	0
M-Square	0	0	0	0	0	0	0
Slender Moon	0	0	0	0	0	23%	19%
Moon	0	0	14%	7%	0	72%	80%

Table 6.8: Fourier Descriptor's confusion matrix with median filter 15 applied. The thin cross and the moon had big correctness improvement, while pac-man decreased in correctness compared to table 6.5

Classified as	Ground truth shapes					
	FC	C	R	P	SM	M
Thin Cross	0	0	0	0	0	0
Fat Cross	6%	0	0	0	0	0
Circle	39%	24%	26%	65%	0	4%
Ring	1%	75%	72%	5%	36%	63%
Pacman	0	1%	0	29%	0	0
Square	0	0	2%	0	0	0
M-Square	0	0	0	0	0	0
Slender Moon	0	0	0	0	10%	0
Moon	2%	0	0	0	54%	33%

Table 6.9: Fourier Descriptor's confusion matrix with underwater shapes placed at 3 metres away.

Classified as	Ground truth shapes					
	FC	C	R	P	SM	M
Thin Cross	0	0	0	0	0	0
Fat Cross	1%	0	0	23%	0	0
Circle	1%	98%	0	6%	0	0
Ring	0	0	41%	0	0	0
Pacman	0	0	0	1%	0	0
Square	0	0	0	0	0	0
M-Square	0	0	0	0	0	0
Slender Moon	1%	0	0	0	100%	0
Moon	97%	2%	59%	70%	0	100%

Table 6.10: Shape Descriptor's confusion matrix with underwater shapes placed at 3 metres away.

Classified as	Ground truth shapes					
	FC	C	R	P	SM	M
Thin Cross	0	0	0	0	0	0
Fat Cross	0%	0	0	0	0	0
Circle	92%	100%	0	0	18%	8%
Ring	8%	0	99%	90%	0	0
Pacman	0	0	0	0	0	0
Square	0	0	0	0	0	17%
M-Square	0	0	0	0	0	0
Slender Moon	0	0	0	0	82%	0
Moon	0	0	1%	10%	0	75%

Table 6.11: Combination of Fourier Descriptor and Shape Descriptor's confusion matrix with underwater shapes placed at 3 metres away.

Classified as	Ground truth shapes						
	TC	FC	C	R	P	SM	M
Thin Cross	11%	0	0	0	0	0	0
Fat Cross	1%	0	0	0	0	0	0
Circle	21%	39%	1%	6%	2%	0	10%
Ring	0	57%	6%	7%	62%	0	13%
Pacman	1%	0	0	0	4%	0	3%
Square	25%	2%	0	0	0	0	0
M-Square	7%	0	0	0	0	0	0
Slender Moon	3%	0	18%	37%	0	85%	10%
Moon	32%	2%	74%	50%	32%	15%	64%

Table 6.12: Fourier Descriptor's confusion matrix of classifying projection of the shapes underwater.

cross and the pac-man were all mostly classified as slender moon or moon. That is more or less what those shapes look like when they tilted. The author notice that when the thin cross was tilted, the colour segmentation failed many times, leave the cross looked like a bar. It is unfair to give the algorithm a bar shape and expect it to classify it as a cross.

6.9 3D object recognition

The main found out is that the SURF feature detector was unable to perform in real-time for object classification on the current AUV. The average time to compute SURF feature on a 320x240 sized image was about 2 seconds on 1.6GHz PC. To match this image's feature to a database of 8 images takes no less than 10 seconds. A lot of speed up has done, for example reduce the input image size and reduce the SURF's sensitivity to lower the number of features detected. But to get a reasonable speed (2 seconds to calculate features and finish matching), the SURF feature would lost its ability to distinguish objects. Other means of classifying the cone and the tyre has to be used.

6.10 Summary

Colour based target recognition has affected seriously by the new find out – colours for example red, when ambient light is low, the camera could not see any colour except dark grey. This problem breaks the initial assumption about observed colour underwater – colour gets darker (not lost colour) when distance increase. Once this problem present, the classification process could not achieve distance invariant by dropping the intensity channel because distance is no longer connected to intensity alone. The RGB Mahalanobis distance is going to be used to ensure the stability of the colour recognition.

Classified as	Ground truth shapes						
	TC	FC	C	R	P	SM	M
Thin Cross	57%	0	0	0	0	0	0
Fat Cross	0	0	0	0	1%	0	0
Circle	0	0	0	0	0	0	13%
Ring	0	0	0	1%	0	0	0
Pacman	0	0	0	0	1%	0	
Square	0	0	0	0	12%	0	0
M-Square	0	76%	2%	0	2%	0	0
Slender Moon	42%	0	8%	21%	14%	100%	5%
Moon	1%	24%	90%	77%	71%	0	82%

Table 6.13: Shape Descriptor's confusion matrix of classifying projection of the shapes underwater.

Classified as	Ground truth shapes						
	TC	FC	C	R	P	SM	M
Thin Cross	43 %	0	0	0	0	0	0
Fat Cross	0	0	0	0	0	0	0
Circle	1%	0	21%	6%	24%	0	26%
Ring	0	0	0	19%	4%	0	0
Pacman	0	0	0	0	1%	0	
Square	21%	100%	3%	1%	12%	0	0
M-Square	0	0	0	0	0	0	0
Slender Moon	26%	0	5%	19%	0	100%	5%
Moon	0	0	72%	55%	60%	0	69%

Table 6.14: Combination of Fourier Descriptor and Shape Descriptor's confusion matrix of classifying projection of the shapes underwater.

Brightness based flash beacon recognition found to be effective. It has tested over distance range of 10 metres underwater with one static threshold value, provided accurate flash beacon location.

Two tracking algorithms were tested on tracking buoy, constant moving, accelerating and rotating flash beacon both on the bench and underwater. The Kalman filter was consistently performed over the Nearest Neighbour tracker, it is put forward.

Fourier Descriptor, Shape Descriptor and their combination was tested against each other in various condition. The combination demonstrated best performance by combining the best of both. While Fourier Descriptor unable to distinguish between circle and ring, the combination easily reached 100% and 99% respectively in classifying circle and ring in the water 3 metres away. When classifying the fat cross underwater 1 metre away, the Shape Descriptor was only able to correctly classify 5%, the combination reached 43% correctness. And in general, the combination produces better result as seen from the experiment results. Taking the advantage of dimension reduction, the combination descriptor only had 5 dimensions. Considering the 5-dimension Fourier Descriptor, the 5 dimension of the combination gives a better description to distinguish the available shapes. Noise filter was found to be beneficial, especially when objects are far away. But they equally need to be used with care. As shown in the testing, median filter could potentially alter the shape, round up the sharp corners the shape has, which could lead to misclassification for some algorithm.

Chapter 7

Conclusions

This project aimed at implementing various computer vision algorithms and evaluating them in the underwater environment. Paying attention to the specific task will be in the SAUC-E 2008, to conclude a set of algorithms that are both sufficient in accuracy and are able to run in real-time. The recent feature detector, such as SURF, was also implemented.

This project provided the author a great opportunity to learn about computer vision domain from practice. General approaches used in vision applications are identified and learnt during this project. Various algorithms for different purposes are also successfully implemented and most of them work reasonably well in real-time without further optimisation.

The shape classification using combination of Fourier Descriptor and Shape Descriptor was suggested in the previous research. It has been implemented, and put in to evaluation along side with Fourier Descriptor and Shape Descriptor, achieved satisfiable result. Evaluation show that the joined descriptor combines the advantage of both algorithms, and produced best overall result in classifying shapes. The Shape Descriptor demonstrated better classification result underwater compared to Fourier Descriptor, which means description using contour was disrupted more seriously than using region in underwater environment. None of the implemented shape classifiers have real advantage in projection invariance, although none of them were implemented with projection invariance in mind. Projection invariant must dealt with specifically.

The implemented Principal Component Analysis routine reduced feature vector from dimension 22 to 5 in testing, saved several folds of computation power, which contributes to classification in real-time. It also demonstrated the amount of redundant features in the original feature space, providing a way of assessing the usefulness of the chosen feature in a set of data.

One of the implemented tracker – Kalman filter demonstrated strong ability to track objects that is not always detectable (flash beacon). Constant motion, acceleration and circular motion are all tracked with low error rate.

This project also attempted SURF feature detector, produced a basic working system for

future improvement.

This project developed a complete system implemented from scratch, that will enable the BURST team to re-enter this year's competition.

7.1 Critical Analysis

The evaluation and its planning should took place earlier. Underwater video footages should be attempt to be captured by myself in every opportunities had. Using the footages found on the internet and captured by previous student made the author was not able to realise the colour distortion was so serious. Therefore not explored in this project. This also leaded to delay the systematic evaluation.

While implementing different colour algorithms, algorithms were tested on very limited amount of video footages. When results went slightly unexpected, the author only blamed the coding bug in the implementation but never thought that it was the limitation of the algorithm. This is also due to the limited resource author had at that time while not attempted to capture more. There were times that the library function were not working properly in specific condition, made the author really frustrated when could not find bug in my own code.

Because author's interest, there were times the attention had been shifted to hardware and the operating system the vision code was to run on. Although these work was beneficial to the competition team, it is less relevant to the subject of this project. That time could be used to push this project further.

7.2 Future work

Colour recognition and classification experienced serious problems. That is because colour distortion and colour lost was more seriously than originally thought would be. Particularly, red totally lost its colour in only 3 metres away underwater. Some advanced colour correction could be applied. The author discovered the Automatic Colour Equalisation (ACE) [Rizzi et al., 2003] algorithm for digital image unsupervised enhancement, it is based on Human Vision System and demonstrated effectiveness in underwater and artificial lighting conditions. Although no time has left to include this technique within this report, the author is planning to include the algorithm in the final system delivered to the team.

The tracking of an object can be enhanced in two ways. Currently the system is tracking the target after the object is classified as target, removing the contribution the tracker prediction could bring to the classification. The tracker-predicted location can be used together with other features for the classification of a target. The other improvement is instead of just tracking the centre of the target, the contour can be tracked, providing more comprehensive location information.

The feature detector is a not well researched area with a lot of potential for future investigation.

Bibliography

T. Acharya and A. K. Ray. *Image Processing Principles and Applications*. WILEY, 2005.

D. Albu, A. Birk, W. Chonnaparamutt, P. Dobrev, A. Giurgiu, S.-C. Mihut, B. Minzu, R. Pascanu, S. Schwertfeger, A. Stan, and S. Videv. Sauc-e iub team. *SAUC-E Competition Journal Archive 2007*, 2006.

R. C. Altshuler, J. F. Apgar, J. S. Edelson, D. L. Greenspan, D. E. Horng, A. Khripin, A. N. Knaian, S. D. Lovell, S. O. Newburg, J. J. McRae, and M. B. Shieh. Orca-vii: An autonomous underwater vehicle. *AUVSI AUV Competition Archive, 2004*, 2004.

H. Assalih, J. Cartwright, V. Chalencon, B. Davis, A. Durrant, E. Faraud, F. Figiel, D. Harber, M. Jiminez, N. Johnson, A. Lees, P. Long, J. M. Perez, Z. Qiang, J. Sawas, C. C. Sotzing, and Y. Petillot. Nessie ii autonomous underwater vehicle. *SAUC-E Competition Journal Archive 2007*, 2007.

A. Baumberg. Reliable feature matching across widely separated views. *PROC IEEE COMPUT SOC CONF COMPUT VISION PATTERN RECOGNIT*, 1:774–781, 2000.

H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. *European Conference on Computer Vision*, 1:404–417, 2006.

J. Canny. A computational approach to edge detection. *IEEE Transaction on Pattern Analysis and machine Intelligence*, pages 679–698, 1986.

C.J.C.Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

J. P. Collomosse. *Higher Level Techniques for the Artistic Rendering of Images and Video*. PhD thesis, University of Bath, 2004.

J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, pages 19:297–301, 1965.

D.Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. *Computer Vision and Pattern Recognition*, pages 142–149, 2000.

D.Ribas, N.Palomeras., X. Ribas, G. de Marina, E.Hernandez, F.Chung, N.Hurtos, J.Massich, A.Almohaya, and J.Vila. Ictineu takes the challenge. *SAUC-E Competition Journal Archive 2006*, pages 5–28, 2006.

W. Dubel, J. Greco, A. Chinault, C. Francis, A. Barnett, K. Claycomb, A. Melling, E. M. Schwartz, and A. A. Arroyo. Subjugator 2005. *SAUC-E Competition Journal Archive 2005*, 2005.

G. L. Foresti. Visual inspection of sea bottom structures by an autonomous underwater vehicle. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, pages 31:691–705, 2001.

M. Gabbouj and F. Cheikh. *Vector Median-vector Directional Hybrid Filter for Color Image Restoration.* Tampere University of Technology, 1995.

Y. Gai, H. Wang, and K. Wang. A virtual mouse system for mobile device. *Chinese academy of science*, pages 127–131, 2006.

R. C. Gonzalez and R. E. Woods. *Digital Image Processing.* Prentice Hall, 2002.

G.Welch and G. Bishop. An Introduction to the Kalman Filter. *ACM SIGGRAPH 2001 Course Notes*, 2001.

R. L. Hartley. Segmentation of images flir – a comparative study. *IEEE Transactions on Systems*, pages 553–566, 1982.

S. Haykim. *Neural Networks: A comprehensive foundation.* Prentice Hall, 1999.

Hirose, K. Yamashita, and S. Hijiya. Back-propagation algorithm which varies the number of hidden units. *Neural Network*, 4(1):61–66, 1991. ISSN 0893-6080. doi: http://dx.doi.org/10.1016/0893-6080(91)90032-Z.

P. V. C. Hough. Machine analysis of bubble chamber pictures. *International Conference on High Energy Accelerators and Instrumentation*, 1959.

Johnston. Duke university robotics teams autonomous underwater vehicle:charybdis. *AUVSI AUV Competition Archive*, 2004.

E. Kalman and Rudolph. A new approach to linear filtering and prediction problems. *Transactions of the ASMECJournal of Basic Engineering, 82(Series D):*, page 35C45, 1960.

X. Li and Z. Zhu. Group direction difference chain codes for the representation of the border. *WA: Society for Photo-Optical Instrumentation Engineers*, page 372, 1988.

D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

P. C. Mahalanobis. On the generalized distance in statistics. *Proceedings of the National Institute of Science of India*, pages 49–55, 1936.

D. Michie, D. Spiegelhalter, and C. T. (eds). *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, 1994.

M.Isard and A.Blake. CONDENSATION-Conditional Density Propagation for Visual Tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

A. Moore. Clustering with gaussian mixtures, 2004. URL `http://www.autonlab.org/tutorials/gmm14.pdf [Accessed 9-12-2007]`.

E. H. Moore. On the reciprocal of the general algebraic matrix. *Bulletin of the American Mathematical Society*, pages 394–395, 1920.

B. S. Morse. Shape description, 2000. URL `http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_ COPIES/ MORSE/ boundary-rep-desc.pdf [Accessed 8-12-2007]`.

M. Nixon and A. Aguado. *Feature Extraction & Image Processing*. Newnes, 2002.

K. Plataniotis and A. Venetsanopoulos. Color Image Processing and Applications. *Measurement Science and Technology*, 12:222, 2001.

G. Putnam. understanding sensitivity, 1998. URL `http:// oemagazine.com/fromTheMagazine/ jan02/ testtalk.html [Accessed 10-Nov-2007]`.

R. A. Ribeiro, A. J. Serralheiro, and M. S. Piedade. Application of kalman and rls adaptive algorithms to non-linear loudspeaker controler parameter estimation: A case study. *IEEE International Conference*, 2005.

B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

A. Rizzi, C. Gatta, and D. Marini. A new algorithm for unsupervised global and local color correction. *Pattern Recognition Letters*, 24(11):1663–1677, 2003.

J. C. Russ. *The Image Processing Handbook*. CRC, 1996.

Y. Schechner and N. Karpel. Clear underwater vision. *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, 1:I–536–I–543 Vol.1, 2004. ISSN 1063-6919. doi: 10.1109/CVPR.2004.1315078.

M. R. Sivaraman and R. R. Sahay. Applications of kalman filter in waas, 1998. URL `http://www.gisdevelopment.net /technology/gps/ techgp0033.htm [Accessed 27-11-2007]`.

B. Smolka. On the new robust algorithm of noise reduction in color images. *Computers & Graphics Volume 27, Issue 4*, pages 503–513, 2003.

I. Sommerville. *Software Engineering*. Addison-Wesley, 2007.

SONIA. Sonia auv 2007 final run on board camera capture, 2007. URL `http://www.youtube.com/watch?v=PWLfFMLaguE [Accessed 9-12-2007]`.

M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis, and Machine Vision*. PWS, 1999.

S. Tanimoto and T. Pavlidis. A hierarchical data structure for picture processing. *Computer Graphics and Image Processing*, 4(2):104–119, 1975.

A. Tibau, B. Privat, G. Voleau, F. Costela, N. Thaniotis, and J. Rodriguez. Nessie: an autonomous underwater vehicle. *SAUC-E Competition Journal Archive 2006*, 2006.

P. Trahanias and A. Venetsanopoulos. Vector directional filters-a new class of multichannel image processing filters. *Image Processing, IEEE Transactions on*, 2(4):528–534, 1993.

C. L. B. Wallis. An investigation into computer vision techniques for underwater object recognition. *Dept. of Computer Science, University of Bath*, 2006.

A. J. H. P. N. Y. Vector median filters. *Proceedings of the IEEE*, 78(4):678–689, Apr 1990. ISSN 0018-9219. doi: 10.1109/5.54807.

Appendix A

Underwater footage capture plan

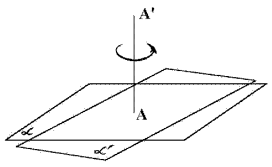
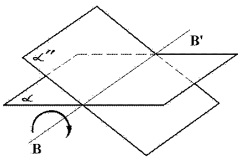
Target	Action
Red buoy White buoy	1. Film from near to far
	2. Film 2D Rotation 360°
	3. Film 3D Rotation 180°
Flash beacon Constant light Light off	1. Approach target near to far at constant speed
	2. Approach target near to far at a accelerated speed
	3. At constant speed film target near to far until out of sight
For each of the 9 shapes	1. Film from near to far
	2. Film 2D Rotation 360°
	3. Film 3D Rotation 180°
Cone Tube	1. Film from near to far
	2. Film 2D Rotation 360°
	3. Film 3D Rotation 180°
	4. Film target from top → approach from far to near
Demonstration:	
2D rotation 	3D rotation 

Figure A.1: Underwater video capture plan

Appendix B

Colour based target recognition test result images

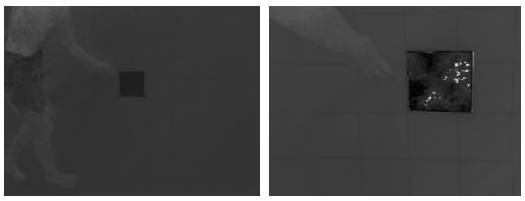


Figure B.1: HS(V) Euclidean distance mapping on red

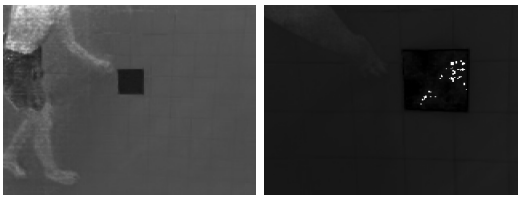


Figure B.2: HS(V) Mahalanobis distance mapping on red

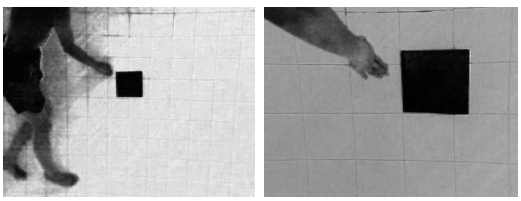


Figure B.3: RGB Mahalanobis distance mapping on red

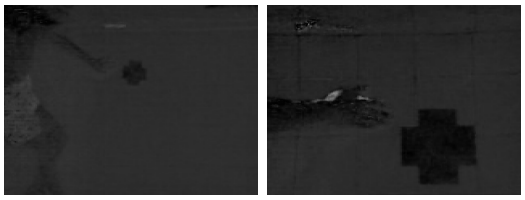


Figure B.4: HS(V) Euclidean distance mapping on green

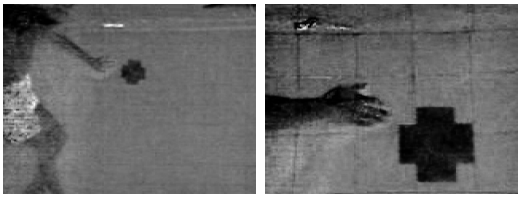


Figure B.5: HS(V) Mahalanobis distance mapping on green

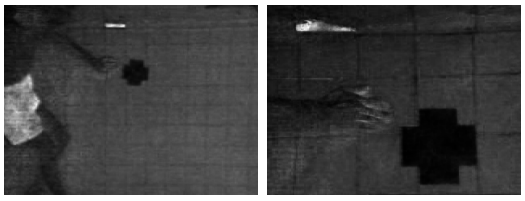


Figure B.6: RGB Mahalanobis distance mapping on green

Appendix C

Gantt chart

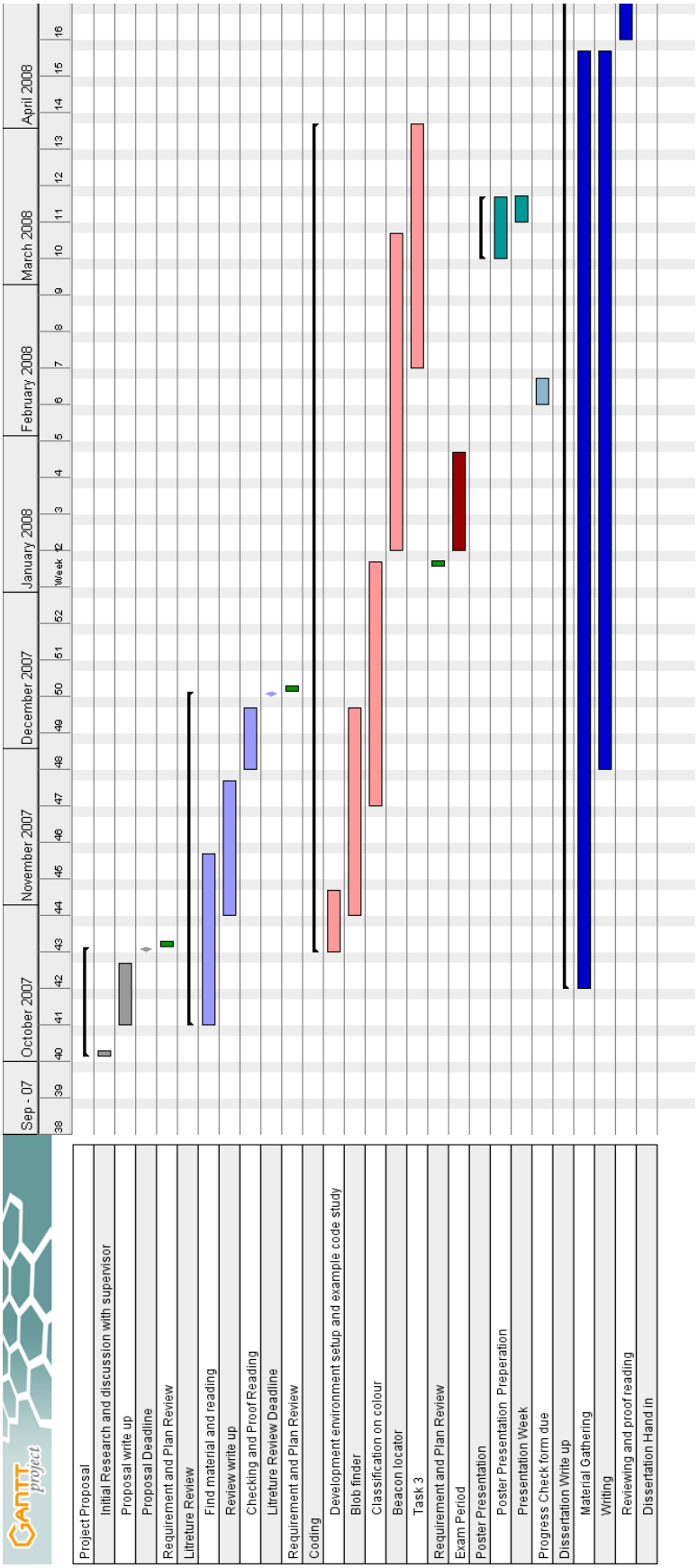


Figure C.1: Gantt chart first version

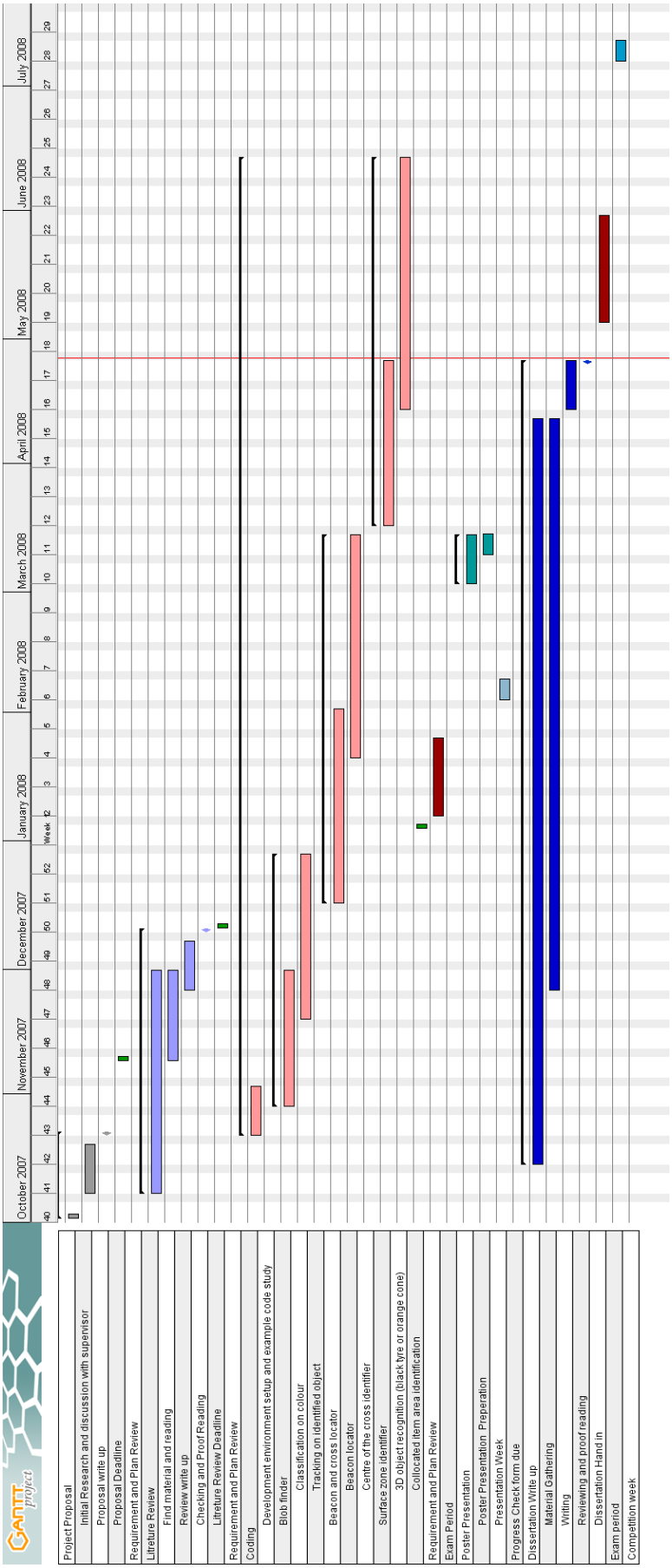


Figure C.2: Gantt chart first review

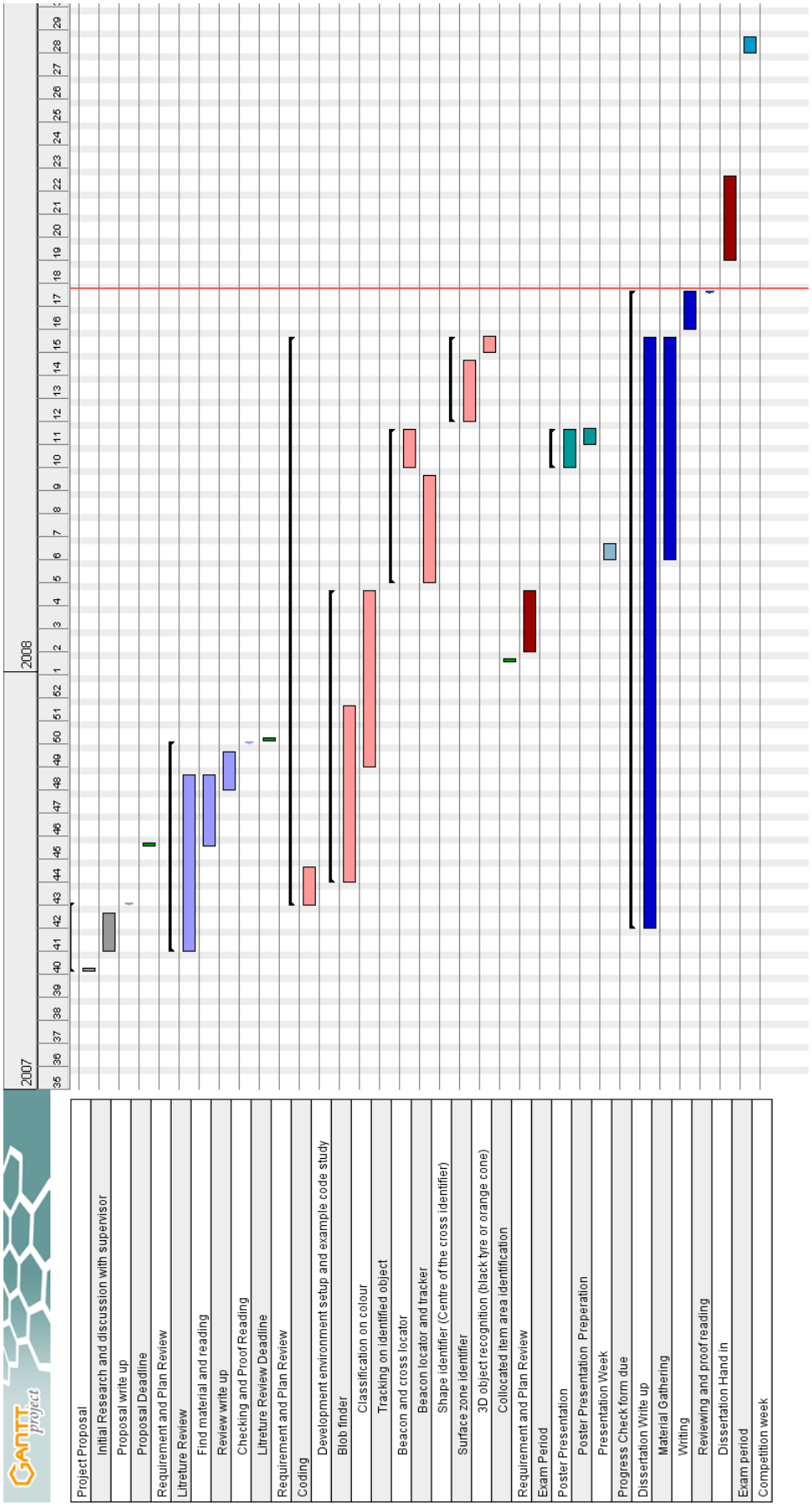


Figure C.3: Gantt chart second review

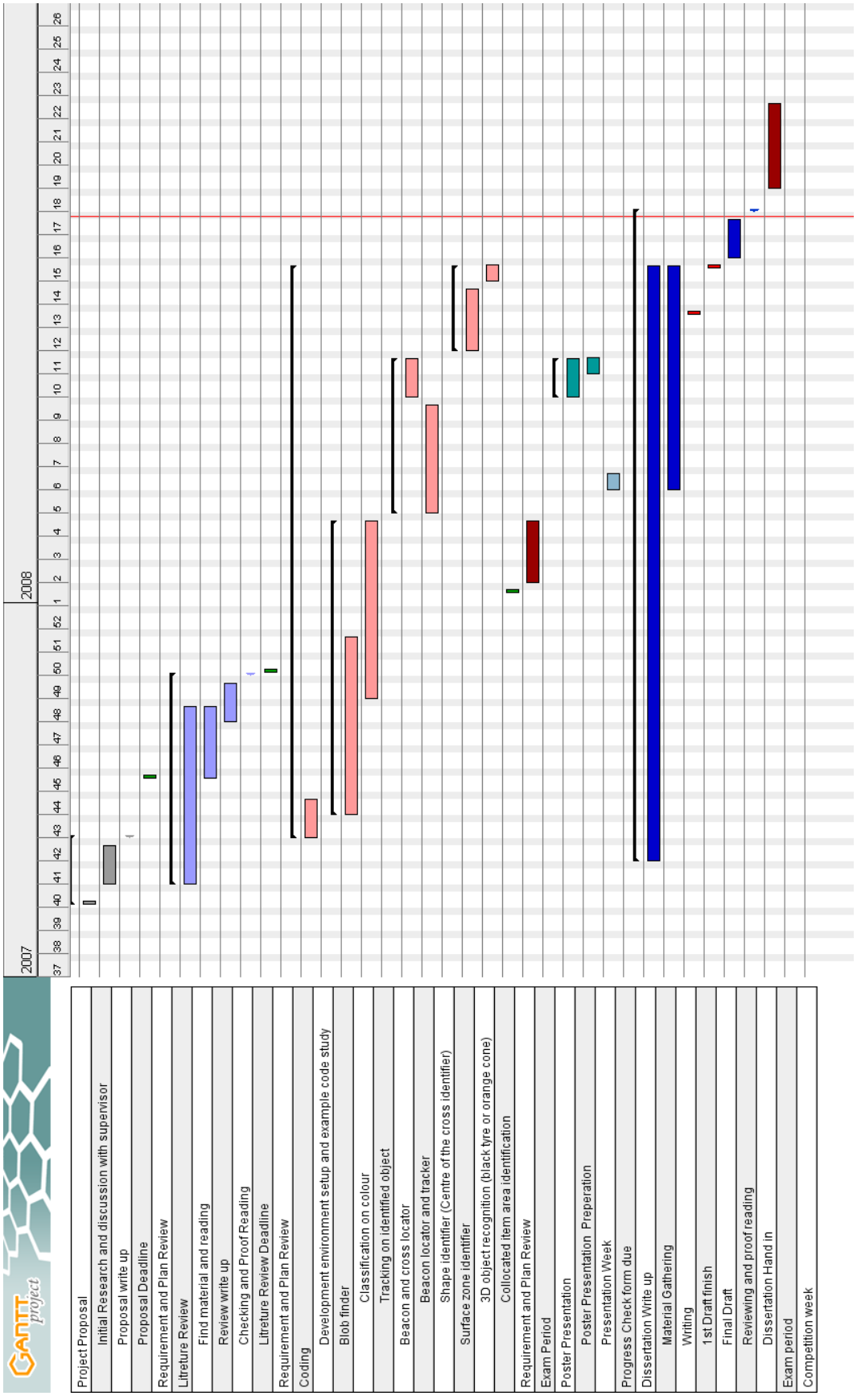


Figure C.4: Gantt chart third review

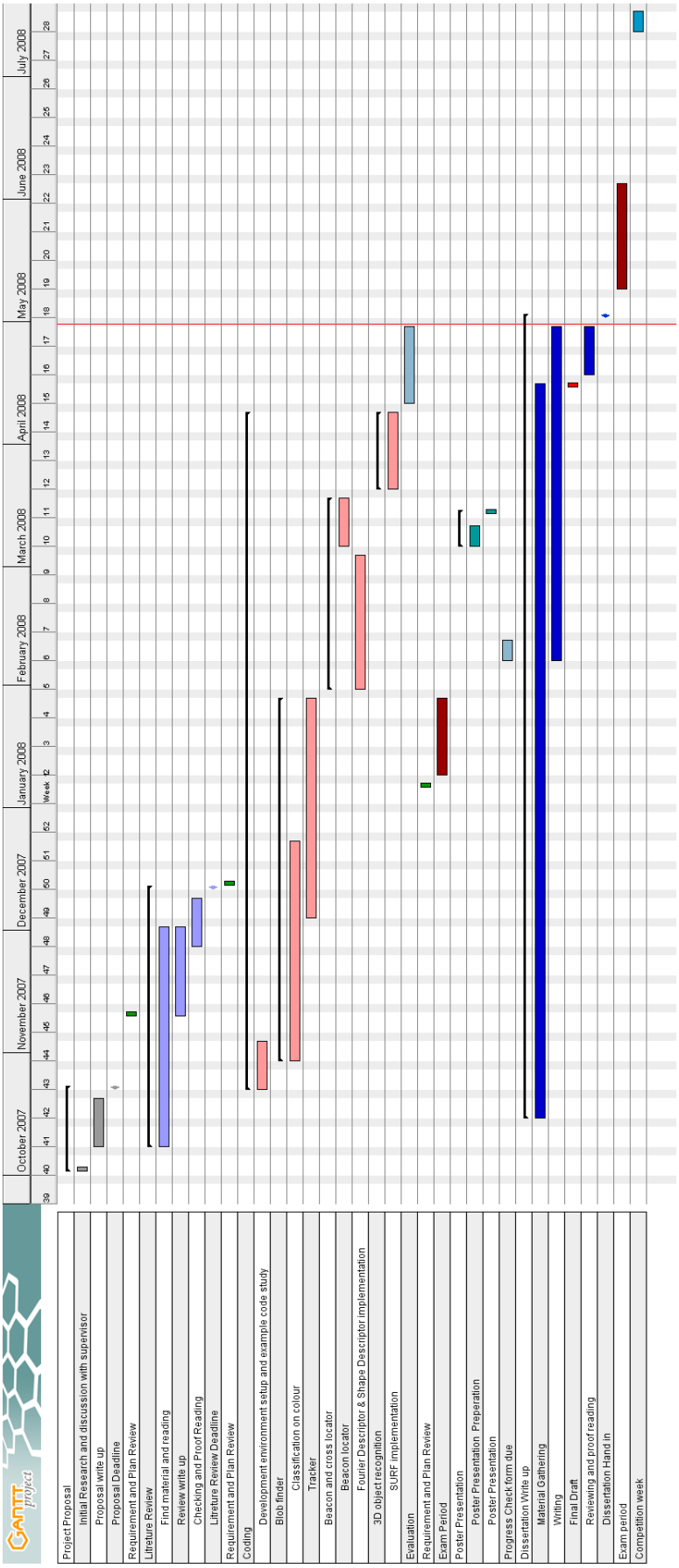


Figure C.5: Gantt chart final version

Appendix D

Included Disk