An Investigation into Computer Vision Techniques for Underwater Object Recognition

Christopher Louis B Wallis
BSc (Hons) Computer Science

2006

An Investigation into Computer Vision Techniques for Underwater Object Recognition

submitted by Christopher Wallis

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 specifically acknowledged, it is the work of the author.

Signed .....

This thesis may be made available for consultation within the University Library and may be photocopied or lent to other libraries for the purposes of consultation.

Signed .....

Abstract

We investigate techniques relevant for autonomous computer vision in an underwater environment, in particular techniques enabling students from the University of Bath to create a fully functioning Autonomous Underwater Vehicle for DSTL's first Student Autonomous Underwater Challenge. A variety of segmentation techniques are implemented, in particular taking advantage of domain assumptions of large areas of single colour, and building on the work of previous AUV vision systems. We also implement a selection of techniques for shape representation and classification, notably Fourier Descriptors and Shape Descriptors, and evaluate them in combination with our segmentation techniques with respect to the underwater domain.

Acknowledgements

Many thanks to my supervisor Dr. John Collomosse for his expert guidance, and for always giving me more time and patience than he was obliged to. 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 and Steve Allen for prepping the AUV, Adrian Sureshkumar for his role in the earlier videos, Robert Pentelow for the production of my test shapes, and the courier who brought them to Bath. Finally thanks go to my parents for supporting me over the years.

For B.

Contents

Chapter 1

Introduction

Each year in the US, a competition is run by the Association for Unmanned Vehicles Systems International[1], in order to attract fresh students, and along with them fresh ideas into unmanned vehicle research. Each year the competition organisers release a 'mission', which will involve each Autonomous Underwater Vehicle (AUV) completing a set of tasks. Last year for example they were required to navigate through a submerged set of goalposts, find a beacon, and follow a line on the pool floor until a target bin was located. While this may seem like quite an artificial situation, the tasks are based on very real world needs. Autonomous vehicles are being used more and more frequently because of the cost and utility benefits they offer in comparison to the human controlled Remotely Operated Vehicles (ROVs) or other devices which have been used in the past. AUVs are currently being used by oil and network companies for finding breaks in pipes and cabling laid up to 7,000 feet underwater (the motivation behind following the line) and by the military for defusing underwater mines, or carrying out reconnaisance (finding bins), and even more commonly, by oceanographers for surveying the sea bed. The University of Bath this year hopes to enter its own AUV into a competition in the UK, the DSTL Student Autonomous Underwater Challenge - Europe (SAUC-E)[2], the first ever of which will run from the 3rd to 6th August 2006 and will be based on the AUVSI version.

This project will aim to investigate and evaluate Computer Vision techniques relevant to this field, in order to provide the most robust and effective vision component for our robot. The competition is not simply a computer vision contest however, the tasks require a multidisciplinary team to create a fully functioning AUV. As well as being able to see, the AUV needs to control its movement, and navigate the competition arena by making decisions based on what it is seeing. This project will not address the artificial intelligence, nor the control of the AUV's motors or other control mechanisms, as we focus purely on its vision requirements. In doing so however, we must be conscious of the requirements of the other components, and not forget that the system is being created for a specific set of tasks, and is not a general purpose underwater vision toolkit.

[1] AUVSI: http://www.auvsi.org

[2] DSTL's SAUC-E: http://www.dstl.gov.uk/news_events/sauce/

1.1 Aim

To find optimal solutions for underwater object recognition for unmanned submersible vehicles and implement some effectively for use in aiding navigation and task completion for the University of Bath's own SAUC-E entry.

1.2 Objectives

The main objectives of the project are;

- Identify appropriate or applicable existing techniques specific or related to this domain.
- Investigate and evaluate the application of these various techniques in the domain, taking advantage where possible of the specific task at hand, the recognition of known objects.
- Enable conclusions for the best techniques to implement for the upcoming SAUC-E competition.
- Deliver the developed algorithms to our SAUC-E entry team, and integrate them with the rest of the AUV's software components.

1.3 Applications

Despite having quite a specific purpose, we hope that the project will also be of use for applications outside its original context.

- A lot of the techniques discussed are in terms of autonomous real-time application, therefore it will provide an additional resource to anyone aiming to implement or assess similar techniques for an autonomous robot.
- More specifically we deal with techniques in a challenging but predictable environment, and investigate the techniques than can take advantage of such conditions. It could therefore be of potential use to others investigating possible techniques for other atmospherically challenged computer vision applications, such as autonomous vehicle navigation in light fog, smoke or heavy rain, where fog in particular might cause the same kind of effects we experience underwater.
- The journals of previous entrants will doubtless be an invaluable resource to this project, and so in turn this dissertation should provide a comprehensive starting point for future entrants to either the DSTL or AUVSI competition.

1.4 Structure

The remainder of this document is organised as follows.

- Chapter 2: In order to identify potential techniques, and for the author to gain an understanding of the problems which are to be solved we conduct a literature review. The efforts of past entrants are discussed, as well as more general techniques for computer vision assessed, and some techniques and approaches are chosen for investigation.

- Chapter 3: Refines and expands upon the objectives given here, and discusses the planning and methodology involved in the execution of the project. A specification of the protocol to be used for communication between the system components is given.

- Chapter 4: We focus on the initial task of locating objects of interest in an underwater environment, discussing the problems that are encountered when trying to isolate targets in the water from the background, and the techniques which are employed to combat them.

- Chapter 5: We detail the algorithms implemented for the measurements of objects identified with the techniques implemented in Chapter 4, and explain the choices made during their implementation. We also discuss the implementation of the techniques identified during the Literature Review, and describe how they are used to identify and distinguish between targets and erroneous shapes.

- Chapter 6: We give a critical evaluation of the techniques implemented, detailing the tests carried out, and the results they yielded.

- Chapter 7: We provide closure for the dissertation, reflecting on its strengths, weaknesses, successes and failures, as well as discussing the steps that will need to be taken before the SAUC-E competition in August.

Chapter 2

Literature Review

2.1 Introduction

Computer Vision literature is vast, as an area of wide application it receives attention and contributions from a huge community of academics and professionals. There is therefore no shortage of techniques for object recognition, and the purpose of this review is to attempt to isolate from this huge source of techniques those which might produce the best results for our task set and underwater environment.

There is a requirement for all entrants to the US competition to submit a journal detailing all of the techniques employed in their submission, so there is a wealth of information about the vision components of previous entrants. On top of this, there is a good base of literature from academics who have previously delved into the field of underwater vision, as well as for the fields of autonomous robot vision. These sources will serve as a good entry point into the literature, providing an understanding of what common problems and solutions exist (see section 2.2).

As well as looking at previous excursions into this domain, we also review some of the popular techniques useful in many computer vision applications. In sections 2.3 and 2.4 we study techniques for Image Processing, which concerns the manipulation and measurement of images and their features such that all kinds of information can be extracted from the scene, as well as repairing image defects such as noise for which underwater vision is well known to contain. In addition, section 2.6 is a review of some techniques from the field of Pattern Recognition, which allow us to use the measurements obtained during image processing to serve as a basis for identification of a particular type or class of object. This is a crucial step if the AUV intends to discriminate between the shapes it is looking for, as well as to identify them in the first place. Finally, the survey is concluded with a summary of the most relevant points from the body of the review, and the foundations are laid for the rest of the project.

2.2 Previous Work

Each year the entrants to the AUVSI competition have been required to submit a journal along with their AUV, describing all aspects of their submarine. The vision

component usually receives considerable attention and so a discussion of previous attempts is the subject of this section. To reflect the overall structure of the dissertation we first describe the image processing undertaken by the teams, and then the pattern recognition.

The journals demonstrate many different methods of performing the same tasks, yet at the same time the majority of them share very common themes. The importance of colour in achieving the tasks and recognising the objects is unanimous, 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.

The biggest use of colour has been in locating beacons and coloured LEDs. Many of the teams have simply used Euclidean distance from a target colour in RGB space to binarise their images into marked/non-marked pixels. Large clusters of the colour are deemed to be the beacon and progress is made accordingly. While this may be satisfactory in the highly controlled competition environment, it must be said that this kind of processing will tolerate very little in terms of like-coloured clutter. Duke University do not even require that the colour is in a local cluster (Johnston, 2004), they measure the amount of cyan (beacon colour) in every frame. If it meets a threshold then the beacon is present otherwise it is not. This forces them into an undesirable situation where their method certainly will be resistant to noise if the threshold is high enough, but a high threshold will prevent the system from being able to recognise a beacon with low light, or one dimmed by the distance.

A more involved approach has been offered by MIT (Apgar, 2005), who use colour as a starting point for their object recognition. Pixels of target colour are treated as 'seeds' for a flood-fill style algorithm which uses a tolerant range of colours around the target colour, outside which pixels are deemed not to belong to the target. The connected 'blob of colour' approach has distinct advantages over the Duke method. Not only does it allow the system to distinguish between like coloured objects in the same frame, but it prevents like coloured objects from contributing to the same measure such as the one implemented by Duke, which seems particularly vulnerable, considering that the beacon's location was then judged to be the centre of gravity of all cyan pixels, meaning that two completely seperate blobs of cyan, perhaps at opposite corners of the screen, would cause the algorithm to deduce that the beacon was centred in its sights. Even though a threshold is offered to combat potential noise, there is no reason why noise should be allowed to contribute in the first place, which highlights another of the common themes amongst the papers; the simple rejection of small items. There seems to be a consensus that small particles of colour are to be discarded with no further processing. The advantages to this are obvious, it both narrows the possibilities down to a more likely few, and saves the vision system valuable time in attempting to classify unlikely candidates.

Colour has also been put to other uses in previous competitions. Interestingly, Amador Valley High School choose to create their edge images from the green channel of the input image alone (Droher et al., 2005). Whether or not this made much difference to their final frame processing rate is unknown, but seems good practice nonetheless. If the target edges they are searching for occur as much, or more on the green channel as on the others, there is indeed no need to process all of the channels. Using domain knowledge to enhance algorithms is part of what computer vision is about (Sonka et al., 1999). This is especially true with respect to our need for a real-time system, where cutting corners like this could save valuable processing time,

and will be applied where possible during this project. This minimalist strategy is also seconded by MIT's 2004 entry (Altshuler et al., 2004), which transforms images into a lower resolution in order to obtain 'a more manageable number of pixels'.

One of the most sophisticated uses of colour is the University of Florida's 2005 entry (Dubel et al., 2005); SubjuGator takes a 'photograph' of the water at the beginning of every round, and uses this image as the basis for ongoing detection of non-water colours. In more detail, the initial image is used to calculate the mean and variance of what is assumed to be water. Pixels of colour outside 95 percent of this confidence range are subsequently classified as LEDs. This is a novel approach, and seems like an appealing solution, however there do seem to be potential pitfalls with this strategy. Firstly, it is completely dependent on what the AUV is facing when it first awakens, and whilst it may be possible to aim the AUV before switching it on, this hardly seems a robust academic solution. Secondly, as the AUV moves into new territory, perhaps shallower water, or near to a cliff face, the colours in the scene could change dramatically, rendering this initial measurement worthless. It does however bring our attention to the fact that in general much of an underwater scene will contain large patches of a single colour, namely what children might describe as the colour of water. There is generally little variation in colour aside areas of interest. This idea is worth investigating further, and in section 4.1.2 we give attention to some similar yet more robust techniques for this style of approach, known in computer vision as 'background removal'.

While the use of such image processing techniques are well documented in the journal papers, pattern recognition seems to receive far less attention. This is mainly due to the fact that the tasks and targets themselves are not complex, and pattern recognition is not always neccessary to provide a working solution. Many teams seem to get by on very little verification, such as Duke's previously mentioned beacon finding technique, some go one step further and provide quality metrics but very rarely are advanced pattern recognition techniques used.

Any pattern recognition which *has* been produced by previous competition entrants has obviously been influenced by the specific mission targets for each year. The mission has never warranted algorithms which can distinguish between different objects of similar colour or nature, and as such algorithms have tended to be of a go/no-go nature. The majority have been based upon the Hough Transform (see section 2.5.2) identifying lines in the image, which are then tested against certain criteria before they are given the go ahead or rejection. Amador Valley recognise underwater bins by a simple set of eliminations (Droher et al., 2005). Their algorithm: starting with a group of lines organised into pairs, eliminate those with insufficient intersections with other lines or parallel neighbours. Surviving lines are then separated into groups based on the angular differences between them. If enough lines remain in a group, then it represents the box. This technique would seem to work under ideal conditions, however, there is no provision for objects which may be occluded or off-screen slightly - in which case not only would edges belonging to partially occluded bins be rejected as the bin, but a good indication of bin presence is ignored. MIT (Apgar, 2005) provide a more tolerant algorithm by calculating a confidence measure based on the lines. The confidence is calculated from a number of expected properties of the bin. Among the properties are; aspect ratio - the ratio of width to height, the interior consistency of the bin, and the polarity of the opposite sides'

edges in the original edge image. Also, by measuring and checking multiple features, they increase the robustness of their solution.

Modularity and Communication As one of the project objectives is to integrate the resulting system with the rest of our AUV's systems, notably its Artificial Intelligence component, it is also pertinent at this point to comment on the architectures used for communication in past competitions.

There are few options, it is generally and rightly recognised in these competition journals that modularity is a great benefit to design, easing the processes of development, testing, maintenance, and problem finding to name but a few. Once the components of the system are separated, their communication must rely on some external system. In 2001 Amador Valley High School chose to implement a system based on shared memory (Bryant et al., 2001). While they state that this separation into modules and reliance on shared memory should add robustness to the system in case of failure, the reverse could also be said. Having numerous processes polling and writing the same area of memory seems an equally convenient way to *reduce* portability of the system or its modules, to another operating system for example or even to a separate computer. There are also synchronisation issues such as, how to know when it is safe to overwrite data in the shared memory, as well as the unavoidable protocol issues such as where to write the data, in what order and what format. In 2004 Cornell University opted for a dual-computer setup (Borland et al., 2004), recognising the processor intensive needs of the vision component they allowed it its own computer. This was then linked to the AI via an Ethernet switch, and a state synchronisation protocol established for the purposes of communication. This seems to be the most advantageous and robust solution, separation of the processor intensive components allows vision processing and control decisions to take place concurrently, allowing the control to always be in control, while allowing the vision the processor time it needs. The networked solution is also more portable, the interfaces for TCP/IP are available for a vast amount of programming languages and control systems and are suitably standardised in comparison to the implementation required to create a shared memory on a particular operating system.

Objectivity After discussing and critiquing the efforts of previous AUVSI entrants it is important to note that the sources of information, the competition journals, are themselves judged as part of the competition entry. Presumably because of this there is often a final paragraph stating how confident the teams are of performing well in the competition, however in reality many of the entries do not even manage to reach the validation gate. These journals must therefore be taken as subjective descriptions of the entry rather than critical academic evaluations of it. Mention is never given to any of the downsides of any of the approaches implemented, and no kind of test results are included - or asked for. Furthermore, as a summary of their entire work rather than specifically the vision segment, the finer details of the vision algorithms are often omitted. Consequently the conclusions drawn here are based solely on the author's consideration of the techniques, as there is unfortunately no more substantial basis from which to form an argument.

2.3 Image Processing

Much of the previous section describes operations that previous teams have performed on their images to attempt to extract some kind of information. The techniques mentioned so far have mainly been higher-level Image Processing, as the lower level operations are not given much time in the journal. The lower level operations are the more standard, one-size-fits-all approaches such as passing a template/kernel over an image in order to shift or change the information within the image. The higher level processing takes more of a semantic approach, techniques such as counting the amount of cyan in the image and finding its centre of mass, and usually aim to take information *out* of the image. These two groups are often considered as distinct, the lower level being known simply as Image Processing while the higher level is known as Feature Extraction. This section discusses the lower level techniques, often neglected in the journals probably due to their simplicity and the fact that they are so essential that they are commonplace.

2.3.1 Colour Spaces

Colour is commonly represented in computers in what is known as RGB space, a mix of red green and blue intensities that combine to produce other colours. Colour can also be represented in alternate spaces such as HSL (or HSV), with components Hue Saturation and Light combining to produce the same colours. The reason this is worth discussion is that by representing colours in HSV we can modify the light component independently of the other information that makes up a colour. This is useful because while humans barely notice a lighting change over an object, it can make a huge difference to a computer algorithm. By setting the brightness to a constant value in the image, an image can be obtained in which objects which originally had a lighting change across them have instead a simple matte hue. This is especially appropriate for the underwater domain considering the challenges the water poses; 'crinkle patterns' caused by surface ripples refracting light unevenly into the water, and the attenuation of light over relatively short distances. Unfortunately this technique can not be extended to correct the colour loss that is caused by attenuation of some wavelengths of light more than others, and the problems this causes will have to be addressed elsewhere.

This is obviously not a useful technique if the task requires the vision to distinguish between similar shades of the same colour, however the competition will likely mirror reality in that the objects will be chosen for their contrast to the surroundings. If an object was put underwater with the intention of an AUV finding it, it would probably be painted orange first.

2.3.2 Contrast Enhancement

Contrast enhancement is the process of altering the intensities in an image in an effort to make objects stand out from each other, therefore improving the contrast between them. Contrast enhancement could be used in our context to improve the contrast between a distant dim object and its background. This would be useful in a situation where the AUV is known to be surrounded by only either water or targets. However this is not our situation, our environment includes some foreground to which targets

will generally already be well contrasted, while those in the background will not. Any successful contrast enhancement would rely on the vision system already having an understanding of what the image contained in terms of where the background was and so on. Once such understanding was already available however, the problem would already have essentially been solved and any number of these techniques could then be used to extract the correct information. Contrast enhancement does not seem like an appropriate choice, given the inconsistency of our images.

2.3.3 Noise Reduction

As mentioned previously, images acquired underwater are particularly prone to contain noise. This noise can have a significant effect on later information extracted from the image and consequently can affect the success of an algorithm. Plenty of techniques exist to reduce noise in images, but always represent a tradeoff. To understand why, the noise can be considered as high-frequency speckles in the image; pixels whose values deviate sharply and uncharacteristically from their neighbours. The purpose of noise reduction filters then is that they reduce this high frequency information in the image and thus they are low-pass filters. Fortunately our application does not require a high level of detail as other applications like iris recognition for example, so this loss of detail should not be critical, we merely require objects to retain their original colour and to be reasonably well defined from their surroundings.

Simple Low-Pass Filter The fastest and most elementary noise filter is the average, or simple blur. It operates by passing a small grid (or 'kernel') over every pixel in the image, replacing its centre pixel by the average of all the pixels inside the grid. This has exactly the same effect (and is the same underlying mathematical operation) as a low-pass filter for digital signal processing, thus removing the high frequencies in the image. The biggest criticism of the blur filter is that as well as removing noise, significant blurring occurs over boundaries in the image, clearly because they themselves are high frequency components of the image. This can cause problems for later image processing which depends on the existence of strong borders in the image, but as it is unknown whether or not these will be required the simple averaging filter could well be appropriate simply due to its speed advantage over the rest of the filters discussed here, considering that all processing must be completed under real-time constraints.

Median Filter A very popular choice in computer vision literature, the median filter is a local-averaging operator which selects the median value from a region around the pixel being evaluated, usually for small regions such as 3x3 or 5x5. The median filter is more popular than the average filter because it preserves the high frequencies which are created by object boundaries, whilst attenuating noise comparably to the average filter.

While we will certainly wish to sacrifice high frequency image detail in order to have an image which is easier to segment, achievable with both filters discussed so far, the need for boundary preservation will depend on the remaining techniques. For techniques invariant to scale and working on the border alone we might assume that the loss of a few pixels around the image will pose no significant problem. Conversely,

any techniques which depend on well defined borders or accurate borders (perhaps high frequency ones like on a saw edge) *will* require such a property from the filter. Therefore the median filter will be used selectively, in cases where it is appropriate.

Peak and Valley Filter Similar in nature to the median filter, the Peak and Valley Filter (Windyga, 2001) is claimed to offer similar noise reduction performance to the median filter, whilst being faster to compute. The algorithm aims to eliminate noise by replacing local maximum and minimum intensities by the value of their most conservative neighbour. Its operation is better explained through example. Consider the values 15,13,1,16,17, using a five sample grid, the Peak and Valley Filter would consider the 1 as a valley amongst its neighbours. When centred on the 1, it would declare its centre as a local minimum, and replace it with the most conservative neighbour, the 13. Similarly in the sequence 15,13,45,16,17, the centre is a local maximum and will be replaced by 17, the most conservative neighbour. The main advantage over the median filter is speed, since no sort of the data is required at each step. Results shown in the demonstrating article indicate that only a fraction of extra noise is permitted through this filter compared to the median filter. It may be the case that this is perfectly adequate for our needs, and represents a balance between the speed of the blur filter and the border preservation of the median filter.

As these techniques are all kernel based operations there is a question over what size kernel is appropriate. The size of the kernel in each filter corresponds to the band of frequencies attenuated. In terms of images this is understood by a 3x3 grid removing frequencies which repeat inside the kernel itself, so any signals which repeat in less than three pixels will be removed, as will objects smaller than 5 pixels - but this is of no concern as we cannot classify a sub 5 pixel shape. Larger kernels will consume a greater portion of the high frequencies, as well as taking more time to compute. Analysis of the best size is difficult to continue at this stage, and so will be heuristically evaluated during the implementation.

Used by last year's winning team (Dubel et al., 2005) in the US competition, boundary preserving iterative scale-space techniques described by Perona and Malik (1990) appear to offer high quality noise reduction. The technique is presented as a precursor to edge detection, as the authors claim the technique is particularly apt at preserving borders while removing noise. The process involves iterating over multiple scale-spaces of the same image, and although it was put to use by a previous entrant in the US competition we will not consider it further, as it is geared mainly towards providing a good basis for edge detection rather than for noise removal.

All of these filters operate on the assumption that noise values are of a 'salt and pepper' nature, that they are isolated deviations from the correct value. This is a form of spatial noise, deviations from the correct value from one pixel to the next. Noise can also occur temporally, for example the fluctuations in value of a particular pixel from one image or frame to the next, probably originating from camera imperfections or quantisation errors. Other noise removal techniques focus on removing this form of noise, such as frame averaging where by keeping a running average of all pixels in the frame over a few frames a better approximation of the correct value can be obtained (Russ, 1995). Similarly as with spatial noise the estimation could be found by selecting the median value etc, but the distinction is simply that we are operating on the same pixel over time, rather than neighbouring

pixels across the image.

A similar project has very successfully used temporal noise reduction techniques (Warren, 2002), in which miniscule signals from aeroplanes can be detected in the distance, by removing other such miniscule noise through temporal filtering. There is a crucial difference between our applications however since the camera in the other application was still. Despite moving slowly, the spatial difference between consecutive frames taken from the AUV's camera would be noticeable, meaning that the value of a pixel in one frame might not correspond to its value in the following frame. The spatial noise filters discussed however are highly suitable, as they are fast, effective, and can be put to use on individual frames.

2.3.4 Edge Detection

A classic technique in computer vision, edge detection gives rise to a multitude of measurable features detailed below. The underlying technique is to examine neighbouring pixels in the image for sharp differences in values (colours) which might represent the border between two objects in the image. This is essentially the opposite to noise reduction, as the aim is to accentuate high frequency components in the image. Like with noise reduction there are a number of popular kernels to achieve this, such as the popular Sobel Operator (Sobel, 1970) which has been used previously in the AUVSI competition.

Again there are strengths and weaknesses between the operators. The Roberts Cross Operator (Roberts, 1963) is another popular choice, known for its simplicity and its speed of calculation. The Sobel operator although commonly used always causes blurring between edges and as deduced in (Porteus and Collomosse, 2005) would cause problems if accuracy of edge location was important. Although Heath et al. (1997) state that academics agree recent edge detectors are more advanced than classics like the Sobel, we find no particular advantage of this for our task set. 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.

2.4 Feature Extraction

As discussed above there are two areas of Image Processing, the first which we have just discussed concerns low-level operations on the image. This section discusses the more semantic operations which are executed on images in order to produce information for the classifier. This information is often in the form of feature vectors, a group of measurements which define a point in the feature space which can then be used for classification and tracking. This will be explained in more detail in section 2.6, but for the moment we will simply describe what kind of techniques can produce these feature measurements from an image.

Before these features are extracted however, we need to have an understanding of how the image itself is represented.

2.4.1 Image Representation

Digital cameras and computers commonly store images as matrices (known as rasters), where each element in the matrix represents the brightness of a pixel. There are other ways to represent the same information, but since our camera will deliver images in raster form, the time spent converting to alternate representations would obviously have to be sacrificed in order to obtain some benefit elsewhere.

One alternative which could be advantageous in our domain is run length encoding. As noted in (Bruce et al., 2000) many environments designed for autonomous robots are rich in large areas of single colour, in order to simplify the task of object recognition. The same is particularly true in our underwater environment, where conceivably even an entire viewed scene could be of a single colour, ie, if nothing is in sight but water, and in general little clutter will be present. Bruce et al. (2000) take advantage of this in the formation and analysis of their connected components[1]. Every pixel in the original image is classified into a certain category of colour before the image is converted into run length encoding and vertically neighbouring runs are linked to each other in a tree structure. The important part, and most relevant for this section is the feature extraction which takes place. During the scan for connected components the tree structure for each component is updated with simple measures; area, bounding box dimensions, and centroid location. The efficiency gained here is that these simple measures can be computed faster from a run-length image (RLI) than a conventional image (as the header of each run contains information on the length of each run) and subsequent operations on regions can work run by run rather than pixel by pixel. These simple measures then are sufficient for the remaining needs of their application. For this project however we are mainly concerned with shape. Using an RLI literally means encoding this shape into a less natural representation, so that subsequent operations require some form of 'decoding' of the representation. Imagine trying to find the next pixel along the contour of a shape. In a matrix the 8 neighbouring pixels can quickly be tested, however in an RLI it becomes less straightforward, requiring the decoding of any number of runs just to check whether or not it contains a neighbouring pixel at all. This is just one of potentially many measurements of shape which would be more complicated to calculate from an RLI than a matrix image. Therefore although successfully used in the 1999 RoboCup, a similar domain, this approach is not appropriate for this project, especially considering that it costs processing time to convert to in the first place.

2.4.2 Object Representation

Objects in the image are plain for a human to see, but in order for a computer to reason with them they need to be converted to some form of manageable numerical representation. This is the feature vector previously mentioned, the representation of an object by a distinct number of measurements of the object in the original image. The aim is to convert the source image, a group of pixel values in a matrix, into some more concrete and understandable representation of shape.

Important properties of different representations are their invariances; the mathe-

[1] Connected Component: A group of connected pixels in the image of the same colour or deemed to belong to the same object

matical distortions by which they are unaffected. For example measuring the diameter of a circle, scaling the circle and then measuring the diameter again will cause the diameter to change, hence this representation of shapes is not scale invariant. Measuring the ratio of a circle's circumference to its diameter however will not be affected by the scale of the circle, and thus the circumference/diameter ratio is scale invariant. It is not projection invariant however as under projection the ratio will change. This is of fundamental importance to our domain, since our objects can undergo translation (as they move across the screen), projection (as we approach from different angles), scaling (as the distance varies), and rotation (as the AUV rolls, or the target is oriented differently).

In addition to all these desirable invariances we require that the representations are not invariant to shape, since it is the shape itself that we are trying to capture in the representation, and so some representations such as using simply the length of the object border and the previously mentioned diameter/area ratio are not sufficient.

There are 3 established schools of object representation, as defined by (Sonka et al., 1999).

- Contour Based Representations
- Region Based Representations
- Shape Classes

Contour Based Representations

The unifying idea behind countour based representations is that they are 1D representations of 2D closed curves. There are a number of ways to represent these borders, ranging from using the boundary length alone to frequency analysis of the boundary signal. This section discusses a few of the most prominent categories of contour representations:

- Chain Codes
- Simple Geometric Border Representation
- Fourier Descriptors

Chain Codes Chain codes are a simple and elegant way of storing boundary information. Borders are represented as a sequence of numbers, where each number represents the direction of the next pixel in the border. The traditional reference grid known as Freeman's Code (Freeman, 1961) could therefore represent the border of a square with unit sized edges with the code: 1,3,5,7. Positional information is as such omitted from the code itself, making chain codes translation invariant. However while chain codes are simple to create from a segmented image, they are particularly sensitive to noise, as they work on an exact shape. Also, arbitrary changes in rotation and scale clearly alter the resulting chain code dramatically. Although work has been undertaken to improve the performance of this technique in the presence of noise (Li and Zhu, 1988) it is still highly bound to a boundary, and using it as the basis for classification is not feasible as the number of dimensions would be huge, and would

be dependent on the length of the border itself. It does however provide a stepping stone to further representations, and other feature extraction mechanisms and so is worth understanding at this stage.

Simple Geometric Border Representations Instead of measuring the direction of the border at each pixel, geometric properties of the border can be measured such as boundary length and boundary curvature. While boundary length might lack the ability to operate well in a complex and scale varying scene due to its overly simplistic nature, curvature might offer better recognition performance. The curvature of a shape can be understood as the rate of change of slope, for example along a straight line there will be low curvature and along the border of an 'S' there would be mostly high curvature. This is similar to chain codes in the way that it could be used to re-draw a shape, and effectively encodes the border. Therefore the same problems as found with chain codes resurface. The number of measurements will depend on the length of the border, meaning the resulting description is dependent on scale. Sampling the border a fixed number of times might allow a scale invariant description to be produced, however one problem that remains is that the signal depends on the point at which sampling begins - making it rotationally variant.

Further geometric representations are discussed in Sonka et al. (1999) but we begin to see the need for representations which do not depend on the border so rigidly, most importantly that the length or sampling locations on the border do not have an effect on what can be used as a consistent set of descriptors for a shape regardless of its scale or orientation.

Fourier Descriptors Fourier Descriptors (Kindratenko, 1997) provide a solution to the problems encountered thus far. Firstly some signal of the border is required. Potentially any of the boundary representations discussed so far could be interpreted 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.

$$X_k = \sum_{t=0}^{N-1} c(t) e^{-itk2\pi/N}, \tag{2.1}$$

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. Krzyzak et al. (1989) demonstrates that the magnitudes of each of these frequency components are

> invariant under rotations, ..., changes in size, mirror reflections, and shifts in the starting point.

and the signal itself will be translation invariant as long as the border representation is so, meaning in turn that the Fourier descriptors will be.

The Fast Fourier Transform (Cooley and Tukey, 1965) allows the DFT to be computed faster than using a naive implementation of Equation 2.1, making Fourier Descriptors a fast and powerful, and therefore popular choice of object representation. Another significant advantage is that the key descriptive information is contained within the first few Fourier descriptors, meaning that the dimensions of the classification space are relatively low compared to the representations above. 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.

One final point on Fourier descriptors is that the more descriptors that are used the more accurately the original shape is represented, however very few are needed to describe a good approximation of the shape (Kauppinen et al., 1995). The advantage for the real-time requirement of this project is that the neccessary level of accuracy can be judged during testing, so that the amount of computation can be reduced to the minimal amount.

Despite being a good solution to many of the problems found so far, Fourier descriptors do not provide a solution for projection invariance. This is a reasonably important property since the AUV can move in 3D space around its targets and thus could receive a projected image of them.

Shape Invariants Shape invariant representations are concerned with finding properties of shapes which are invariant under affine transformations, in particular invariant to projection. This means objects can be recognised from wherever the camera is in 3D, or whatever pose the object has in relation to the AUV (within limits). This is of great utility for our application as our camera will be roaming around in 3D space. A classic shape invariant is the Cross Ratio, which works on the mathematical certainty that straight lines will map to straight lines under projection. It has been shown that the Cross Ratio is stable with respect to spatial quantization noise and remains functioning for objects rotated up to 60 degrees (Forsyth et al., 1991). The Cross Ratio is a function of four distinct points, meaning that the technique is reliant on finding at least four 'landmarks' on the candidate object to be recognised. These landmark features can often be points of high curvature, however when searching for man-made shapes underwater there may not be any such features to use, on the contrary the fact that man-made objects will often be void of distinct features has been used to positively *identify* man-made objects in underwater environments (Olmos and Trucco, 2002). Even so, Forsyth et al. (1991) describe a method for using steady curves as features for shape invariant recognition, however the method consists of five non-trivial steps. Edge Detection, Curve Extraction, Conic Fitting, Computing Shape Descriptions, and Model Matching. The extent of computation during this cycle effectively prohibits its use in our teal-time application, and such will likely be the case for other projection invariant methods. We continue then to look for any techniques which might provide an alternative to the use of Fourier descriptors.

Semantic Nets Mentioned for completeness, semantic nets offer the vision system more power in terms of knowledge of relative positioning between objects, such as in face recognition where the eyes will always be in a position relative to the nose.

This however finds little application in our domain as each object will generally be suspended alone in water or on the floor and have not enough complex distinct parts to merit this advanced level of representation more commonly associated with face recognition.

2.4.3 Region Based Representations

Besides simple representations of object borders, information extracted from the *contents* of the borders can similarly be used as features. An issue with many border-based techniques is that they are sensitive to noise, for example adding noise to a straight line would cause significant change to a chain code. Region based representations trade this accuracy for robustness, as they tend to rely less on a fragile border and more on the more consistent scalar properties of shapes, such as the area - for which adding Gaussian noise to the border would have practically no effect. Both Sonka et al. (1999) and Russ (1995) dedicate a large section of their respective shape description chapters to such measurements, a sample of which are described here to give the reader a general understanding.

Eccentricity The ratio of the longest chord[2] to the longest chord perpendicular to it.

Elongatedness The ratio of an object's length to its width.

Compactness The measure of the extent to which a shape fills the convex hull it defines[3]. Defined differently between the authors, but essentially a ratio of area to perimeter length.

Convexity The ratio of a shape's convex perimiter length to its regular perimeter length.

Notice that all of the descriptors are ratios; this is what allows them to be scale invariant, and note also that none of the measurements themselves are dependent on rotation or translation.

The use of a selection of such features as a feature vector offers similar benefits as Fourier descriptors; the feature space is of a relatively low dimension, and the features themselves are more robust to border noise than the other representations surveyed. Once again the projection of shapes will cause issue for the descriptors, and in fact it is predicted that the quality of classification based on such measures will deteriorate faster under projection than Fourier descriptors since properties like area will change faster during projection than will the magnitude of frequencies required to create the border. However this region-based method does represent a nice complement to the border-based Fourier descriptors and so will provide an interesting comparison as the second technique implemented for this project.

Sonka et al. (1999) explain how complex objects 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.

[2]Chord: A line between any two points on the border of a shape.

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

2.4.4 Shape Classes

Point Distribution Models

Similar to Shape Invariants in that they offer a solution to projection, and are based on locating landmarks on an object, Point Distribution Models (PDMs) offer a far more flexible approach to shape representation.

The technique involves identifying the landmarks of an object as points along its border which are then used for classifier training in which many instances of the object are added to the model. What distinguishes PDMs from the rest of the techniques mentioned here is that it is particularly suited to enable recognition of objects which have undergone some form of deformation. The training stage calculates the principle components of variation in the landmark points, enabling the classifier to recognise new instances of the object which have undergone similar deformations as the objects in the training set. This is applicable for projection since the training set could be a sequence of the target shape under various projections, training the classifier to accept the shape as its landmarks deform under projection. While the techniques above all have problematic issues arising from projective transforms the PDM simply takes this variation into account as part of the landmark variation encountered during training.

A lot of examples seen in the literature demonstrate the use of PDMs in applications where static cameras recognise objects undergoing projection deformations, and so for the same reasons it is equally suited to modelling the kind of projection deformations which static shapes will undergo when looked at by a roaming camera.

PDMs have recently been successfully implemented with near real-time performance rates (Mathes and Piater, 2005), however the level of understanding required and time likely to be spent on implementation would more than likely prohibit any other work being undertaken, and so the technique is left for future work.

2.5 General Feature Extraction Techniques

2.5.1 Template Matching

Probably the most intuitive technique presented in this review, template matching attempts to find objects by exhaustively comparing a template of the object with every possible location of it in the image. The template can be literally an image of the target shape, as the comparison is based on the difference between the template and the location it is placed upon. It will not be explained in detail since it lacks many of the important properties we require, namely scale and rotation invariance, for which multiple templates need to be created for each target shape and each tested exhaustively on the image. This need to define every-possible-situation templates make it grossly inappropriate for this project.

2.5.2 The Hough Transform

Mentioned already in this Literature Review due to its popularity amongst AUVSI entrants, the Hough Transform (Hough, 1962) is a method for extracting straight lines from an image. More specifically, it works on top of an edge-image as produced by the edge detection operators discussed above to produce an estimation of lines present in the image. The technique is based on the fact that a line in two-dimensional space can be defined by two variables, the gradient and intersect with the y-axis. These two variables could be plotted in another two-dimensional space, and would uniquely represent this line, hence this second space is known as the 'parameter space'. Conversely a point in the original space defines a line in the parameter space, representing all original space lines which it could belong to. The Hough Transform then simply creates a discretised version of the parameter space called the 'accumulator array', and draws lines in the parameter space according to every point in the original space (our edge detected image), 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.

The reason the Hough Transform is highly appropriate for use with underwater images is its remarkable tolerance to noise and occlusion of lines. Due to its 'evidence gathering' approach, as long as there is sufficient evidence for a line then it is detected, regardless of breaks in the line or noisy edges surrounding it. For this reason it has been used in a variety of real-world underwater applications (Foresti, 2001) in which marine growth commonly causes occlusion, and will certainly be considered for any tasks requiring detection of lines.

Although not designed for shape recognition, the Hough Transform has been generalised to detect more complicated shapes (Ballard, 1981), extending its remarkable resilience to occlusion and noise over arbitrary shape detection. Unfortunately the extensive computation required to perform the transform for such complex shapes makes it unfeasible for a real-time application given the regular hardware of the AUV.

2.5.3 Active Contours

Active Contours, also known as snakes (Kass et al., 1988) operate on a principle of 'energy minimisation', attempting to iteratively close a set of points around the target object in order to gain a precise representation of it. However each iteration is a demand on processor time, of which there can be many, which makes snakes not particularly suited to real-time applications.

While snakes may be too slow for use as a general feature extraction method, the precision they offer makes them a suitable candidate for the more precision demanding tasks in the competition such as preparing to drop the marker on a clear target, in which case the AUV should be moving very slowly - allowing more time for vision processing. Potentially then, this technique may be implemented before the actual competition, but its inclusion in the dissertation is unlikely.

2.5.4 Background Reduction

A fine line between Image Processing and Feature Extraction, background reduction encompasses techniques which aim to delete the background from an image in order to be left with potential targets. As previously mentioned, Dubel et al. (2005) used an initial measure of the scene to classify pixels in subsequent scenes.

Another more common method is to deduce the background from a sequence of images by averaging their contents in an array, and then subtracting this average from each image. Although this can be useful for some tasks, it requires that the object is moving throughout the sequence while the background remains still. Neither of these assumptions are true in our domain, the camera will be moving while the targets are static, meaning that either our static targets will be removed along with the rest of the background, or everything will appear as foreground because the camera is moving.

However we remain optimistic that more domain specific techniques can be developed as demonstrated by Dubel et al. (2005), since background subtraction would be an ideal way to reduce computation time as all that remains afterwards are potential target objects, with their shapes already well defined.

2.6 Classification Schemes

So far we have discussed all the techniques neccessary to reduce an input image into some numeric form, often what we have referred to as a feature vector; a group of potentially unrelated measurements of the shape in the image. The classification of this feature vector will be the final step in our shape recognition algorithm, methods for which are reviewed here.

2.6.1 Classifier Theory

Once the n features[4] of objects are chosen to be used in identification (for example taking the first n Fourier descriptors of an object), they define what is known as a *feature vector* $x = [x_1, x_2, x_3, ..., x_n]$ which is contained in a *feature space* of n dimensions. Objects in the image can then be reduced to this feature vector as described above, and therefore can be represented by a point in the feature space. The goal is to pick a set of features such that the vectors of each class of object create points which cluster distinctly in this feature space, enabling discrimination between the feature vectors belonging to particular object classes, and hence classifying.

Linear Discrimination Functions

Simple and widely used, Linear Discrimination functions (Fisher, 1936, 1937) take a feature vector and apply weighted coefficients to it, outputting a class 'token' representing which class of object the feature belongs to. A single class Linear Classifier is a function:

[4] A Feature: The measurement produced by one of the feature extraction methods already discussed

$$y = f(\vec{w} \cdot \vec{x}) = f(\sum_j w_j x_j), \tag{2.2}$$

where $\vec{w}$ is the vector of weighting coefficients, created through some analysis of multiple training feature vectors, and y is the resulting class token; simply yes or no in the case of a linear classifier, as the function f performs a threshold on the real value returned by the dot product.

The primary benefit of using linear classifiers is the speed of classification, as it costs only one addition and one multiplication per dimension, and as such they are often used when the feature space is of extremely high dimensionality. As both of the descriptors chosen to represent shape are reasonably low in dimension, there is room for a little more computation in order to achieve more accurate classifications.

Minimum Distance Principle

The minimum distance principle is a little more intuitive than the previous approach. An exemplar feature vector (or, a set of exemplars) is specified in the feature space and represents all objects of a class. A Euclidean or Mahalanobis (described below) distance of test vectors are calculated from this training data and classified as belonging to the class if they lie within a certain distance of the exemplar.

Once this distance is found, as with the linear discrimination function a limit needs to be set as to how close a feature vector has to be to the exemplar or exemplary set before it can be classified as a class member, this process is known as thresholding. Thresholding, and equally classification, does not necessarily come right at the end of the vision process, it is a generally applicable technique - the acceptance of only results above or below a defined cutoff, and can find equal use during the feature extraction stage, where pixels of a certain colour can be thresholded respective to their proximity to an exemplar colour, leaving only pixels which are thus classified as a member of this colour. This illustrates well the concept discussed in the introduction to classifier systems given in Duda et al. (2000), which states that the recognition process is not strictly bottom up, the top layers (classification of a pixel colour) can also feed information back down to the lower layers (feature extraction via Hough Transform).

Where it is less approprate to use a fixed cutoff for thresholding, techniques for adaptive thresholding can be used (Chow and Kaneko, 1972), which take into account local image features such as local general pixel intensities, brightness values etc, which has great potential in our environment in which the camera will sometimes pick up a bright pool floor against a contrasting target and sometimes will barely be able to distinguish between background and the colour-duped version of the target.

2.6.2 Mahalanobis Distance

Mahalanobis distance (Mahalanobis, 1936) is a widely preferred alternative to Euclidean distance in the field of pattern recognition as it gives a more accurate representation of distance from a set. A specific Euclidean Distance from a cluster defines a hyper-sphere in the feature space, with its centre at the mean location of the set.

A fixed Mahalanobis Distance though will define 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 $\bar{d}$ from the mean will vary depending on the direction of $\vec{d}$, 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.

This is accomplished through some initial Principal Component Analysis on the training set in order to create the covariance matrix Σ of the training set with mean $\mu = [\mu_1, \mu_2, \mu_3, ..., \mu_p]$.

The Mahalanobis distance from this set of a vector $x = [x_1, x_2, x_3, ..., x_p]$ is given by,

$$D_M = \sqrt{(x-\mu)^T \Sigma^{-1} (x-\mu)} \tag{2.3}$$

The technique essentially warps the point according to the principal components of the training set, such that the final distance is measured in standard deviations of the set, making it clearly more representative of variation of a set.

Reading through this section should give an appreciation of the reasons why features must be carefully chosen. We have already discussed the relative merits of the accuracy of each feature measurement technique, however an additional concern is how well the training data is able to cluster in the feature space. This is where we truly understand why classification cannot be performed on some feature vectors such as boundary length alone, since this one dimensional space would mix points for any shape depending on its size rather than its shape. We also see that using a single region-based feature would cause certain distinct shapes to be reduced to the same point in the feature space, making the classifier unable to discriminate between them. It is known from the literature that Fourier descriptors are sufficient for this purpose, and it is suggested by Russ (1995) that a combination of Shape descriptors are able to do the same.

As a final point we note explicitly that feature vectors do not have to contain similar features, they can be any synthesis of similar or dis-similar measurements, for example a perfectly valid vector could be made from area, the average red intensity, compactness, and the x-coordinate. In a domain where so many of the previous entrants have relied on the use of colour, some kind of hybrid feature vector may well be useful at some point.

2.6.3 Gaussian Mixture Models

Gaussian Mixture Models refer to the modelling of a data set by fitting multiple Gaussians to it. They have multiple potential uses for this project. Firstly they enable the unsupervised classification of a data set through a process called Expectation Maximisation. If the number of point-producing classes is known then Gaussians can be iteratively fitted to the data via Expectation Maximisation to maximise the likelihood that the Gaussians fit around the true clusters. This kind of unsupervised learning could clearly be advantageous for our autonomous robot, meaning it could potentially segment an image without any prior indication of what

it was looking for, and perform shape analysis on each segment. The EM algorithm however as stated is iterative, and the time taken to fit Gaussians to every frame would not allow the AUV to operate in real-time.

Gaussian Mixture Models can also be used to piece-wise model objects (in our case shapes) which do not form a cluster in the feature space which can be approximated by a single eigenmodel, for example they might form a banana shape in 3D space. By using the above mentioned Expectation Maximisation, a set of Gaussians can be iteratively tightened around the banana, and classifiers can then measure the Mahalanobis distance of the feature vector from each one to decide membership. This could well come in useful if our feature vectors do not form a significant cluster, since the EM can take place at the training stage.

2.7 Trackers

Once the object has been identified in the image, it is often useful to estimate its motion so that if the object is not found in the following frame, the algorithm may still have an estimation of where it is most likely to be. The same technique can also be used to filter out bad classifications, where high confidence in the location of an object will mean a few stray classifications will not affect the system's judgement.

Presented here are two trackers popular in the Literature, which have properties appropriate for our real-time constraints.

The Kalman Filter

The Kalman Filter (Kalman, 1960) is a general error tracking and linear function approximation algorithm, which applied to our task would allow the modelling and prediction of object motion. The algorithm cooperates with the vision system in a correct-predict cycle, allowing the filter to predict locations when the vision is unsure, and allowing the vision to correct the filter when it *is* sure. At each time interval, if the object position is known the filter is told its location along with a confidence measure of the correctness of the measurement, and thus updates its state and confidence in its own correctness. At each time interval also the Kalman Filter updates its internal prediction of the state of target motion, encompassing 2nd order derivatives such that estimates of location, velocity and acceleration are all maintained. Therefore when the vision has no location on the object, it can increment the filter by a time increment and use the prediction returned.

Widely used in autonomous applications, including the AUVSI competitions, due to its real-time capabilities as well as its relevance in estimating motion, the Kalman Filter is perfectly adequate for the needs of this project.

The Condensation Algorithm

Designed to combat scene 'clutter' and camouflage, the condensation algorithm (Isard and Blake, 1998) claims to perform better than the Kalman Filter on the grounds that it can support multiple hypotheses concurrently, whereas the Kalman Filter supports only one. This gives it a crucial advantage in situations where camouflage

and erratic target movement can throw the Kalman Filter off track, allowing it to focus on the wrong object, when in some cases the Kalman Filter can become so confident in its own prediction that its incorrect state becomes unrecoverable. The finer details of the algorithm are less relevant to this project which focuses mainly on shape classification, so we will simply dicuss its merits.

Sample videos available on the Internet demonstrate this clearly very powerful algorithm tracking a leaf on a bush blowing in the wind. However remarkable this is, it is of questionable relevance to our application domain. One of the main features of the algorithm, that the object can be tracked against a cluttered background finds little application, as backgrounds in a swimming pool are somewhat trivial; either they will be bluey distance, or they will be the pool walls or floor, neither of which can be considered particularly cluttered.

Another significant property of the condensation algorithm is that it can track objects moving rapidly with highly complicated motion models. Again this is not particularly advantageous in our application as nothing in our swimming pool will be moving very rapidly; the AUV in fact moves very slowly and the targets themselves will be stationary. These though are examples of where this algorithm outperforms the needs of the project rather than where it underperforms. There are in fact no significant drawbacks with the use of the Condensation Algorithm, apart from potentially more complexity of our algorithms.

The choice then is superficial, and as both are perfectly adequate we opt for the simpler one, in order that more project time may be focused on the matters of shape analysis.

2.8 Conclusion

Past entrants demonstrated that colour is an invaluable tool in recognising areas of interest, and Florida provided a unique and interesting method for background removal by finding colours in the image which are remarkably different from the mean and variance of the whole image, although a few potential drawbacks were noted. This idea will be investigated during the project, and hopefully the drawbacks to their solution will be overcome.

The modularity and communication of previous entrants software and hardware architechture was also assessed, concluding that the ideal and more popular setup has been to allow the vision system its own computer and to use network sockets to communicate results to other system components. Since our system must communicate with external systems in the same manner, this method seems to be the most advantageous and will likely feature in our system.

A number of low level image processing techniques were considered, of these the illumination invariance offered by the HSV representation was considered to have significant advantages in this domain. A few noise reduction filters were also discussed, all of which might be useful for the project depending on the circumstances. A distinction between temporal and spatial noise reduction techniques was made, and the latter chosen as the more appropriate method for this environment.

Methods for representing shape numerically were discussed, and two sets of features

were chosen for their satisfaction of key invariant properties required in our domain; Fourier Descriptors and Shape Descriptors. It was suggested that both should allow strong distinctions between shapes despite the difficulties of noise in the environment, and that of the two, the performance of the Shape Descriptors would likely deteriorate faster under projection.

A short section on Classifier Theory explained the issues which need to be understood to create a high quality classifier, as well as the use of classifiers in intermediate stages of the classification cycle, and noting the qualities and particular relevance of Mahalanobis Distance in classifications of membership to a class defined by a set of training data. The role of thresholding was briefly explained in relation to such distance measurements, along with a discussion of the potential advantages for adaptive thresholding in our environment.

Finally the relevance of motion estimation models to this project was explained and two powerful linear approximation filters described, where no significant disadvantages were found with either, and the Kalman Filter was chosen for this application purely for its relative simplicity.

Chapter 3

Project Planning & Requirements Analysis

As the chapter before served to reduce the vast amounts of available literature to a select few techniques, this chapter aims to explicitly define the project, its requirements and its boundaries, as well as documenting the high level project management decisions and processes which play a key part in any dissertation.

3.1 Software Process Model

Early on in the project an incremental approach to software development was deemed appropriate. The distinguishing feature of the incremental approach is that core components of the system are developed first, which are progressively built upon to expand and refine the project.

The incremental approach is particularly suitable for this project for a number of reasons. Firstly, time is very limited in all dissertations, so the early development of key components can serve as a form of risk limitation, major problems will be encountered and overcome as early as possible meaning that the negative impact of a critical bug or lack of foresight is well reduced.

The style of the dissertation can also mean that subject learning continues throughout the project time-frame. This is especially true for projects requiring the understanding of a lot of mathematical content such as computer vision projects. Again the incremental approach offers advantages over more linear development cycles here, in that what is well understood can be implemented early on allowing the author more flexible and thus better use of his time to continue digesting some aspects of the subject matter while development is already underway, which according to an authoritative source (Sommerville, 2001) is the true essence of incremental development. Equally as this learning is likely to continue throughout the project's short lifetime the project itself may be subject to minor changes, in developing the truly core components first we limit the probability that time is wasted developing fringe refinements which are later dropped.

Thought was given to other potential software development methods, but none offered such significant advantages as the incremental approach. This is mainly

because the project concerns primarily the development of algorithms. Prototype-based processes are popular with those experimenting with HCI applications or those in software houses developing the next version of their product, however they are of limited use here as the algorithms developed are unlikely to undergo the kind of protoype- review cycle which makes these process models useful.

3.2 System Context

The vision system produced in this project is not intended to exist in a void, it will be one component of a larger system. The University of Bath's AUV will contain three key software components; Motor Control, Vision, and critically, Artificial Intelligence. Artificial Intelligence will be the central component of the AUV, and will therefore need to communicate with Vision. The vision system clearly must be able to pass its results or general information to the AI module, however it may also be advantageous to allow the vision component to *receive* instructions from the AI. Allowing the vision component to receive instructions would mean that it could focus on one task at a time, effectively speeding it by a factor of how many tasks it has in total.

Also as discussed in the Literature Review, computer vision involves a huge amount of processing compared to for example AI, and as such it will be placed on its own computer.

The fact that the AUV is autonomous makes defining the system boundaries very simple indeed. For the competition, the vision system must take images from two cameras, and be able to converse with the Artificial Intelligence, exchanging results and control instructions. No other actors will be involved in a competition run. However this is also an academic investigation, and so during the evaluation phase the author could be considered as an actor, needing to see various information output from the internals of the program. Also, while discussing system context, it is made clear here that for the purposes of this project (as an academic piece of work) the system will not run from a pair of live cameras. Instead it will run on videos intended to be the inputs of these cameras. This is for practicality as well as fairness of evaluation, so the fully working system (input from two live cameras) will have to be implemented after the dissertation is completed.

3.3 Risk Evaluation

Final Year Projects are known to be high-risk, they demand the completion of a significant amount of work within a strictly defined deadline, for which the consequences of missing are very serious. For this reason thorough risk analysis was undertaken at the beginning of the project and reviewed as the project continued. The risk analysis is presented here as a discussion, identifying only the most serious and likely risks associated with the project and the strategies which will be used to reduce the damage they could cause, or to reduce their chances of occuring.

Coding Delayed: The risk that the coding is delayed due to bugs, misunderstandings, and lack of experience with the programming language and or vision libraries,

is relatively high. The author has very limited experience with both the language and tools to be used, furthermore one of the common criticisms of the C programming language is that it is easy to create bugs, hence the International Obfuscated C Code Contest, and bugs cause delays. However the adoption of incremental development as discussed ought to minimise the effect of this risk by allowing the starting of development as early as possible, enabling working the most critical bugs out of the system at early stages.

AUV Unavailable For Footage: One of the major resources for this project, the University of Bath's own AUV will be used to gather footage on which the algorithms will be developed and eventually tested. The AUV however is under constant development and used by a variety of University teaching staff as well as students. The AUV could be taken to Mexico for research or taken to pieces for improvements to be made. Such a scenario could have a serious effect on the project and there is a high probability of such scenarios taking place. As a contingency, reasonably good quality footage can be gained by placing the camera inside a small tube with a clear end and submerging the waterproof end. Such footage will be used until the AUV is available to go underwater itself.

Pool Unavailable For Tests: The University swimming pool is to be used as a source of footage, but its timetable must be shared between various groups. While the pool is not available at all times, there are always free slots in a given week, short delays can be expected and taken into account with early planning and good organisation.

Human Resources Unavailable For Test Footage: As a computer scientist, the author does not have the skills or knowledge required to ready the AUV for a pool test. Any footage gathering sessions dependent on the AUV will also be dependent on the availability of key Mechanical Engineering students and staff, who have working knowledge of the submarine. The testing of this project will rely on having gathered sufficient underwater footage of a decent quality with the submerged AUV, however the testing period will also coincide with the Easter break. In order to mitigate against problems occurring here, firstly, the test data should be gathered as soon as possible to allow most time possible for delays. Secondly, the holiday plans of these key individuals should be ascertained, so that in the event that footage is delayed until Easter there will be no critical delays beyond this point.

Underwater Footage Becomes an Impossibility: In the worst case scenario that unforeseen problems prevent the AUV from being ready to capture footage before the project deadline, simulations will have to suffice. The effect of water could be synthesised in many ways, for example to study the effect of noise, synthetic noise could be added to an open air video. To study the effect of colour loss, a function could be passed over all objects in the image duping them towards the colour of the background, and finally to study the effect of distance and backscatter clouding objects in the image, object borders could be passed through an eroding function to randomly erode the border.

3.4 Resources

As reflected in the Risk Analysis this project is dependent on a variety of external resources, they are presented here categorised into the area they will be of use in the project.

Academic Resources

There should be no problem in finding information on computer vision principles or techniques, entire journals exist which are dedicated merely to sub-fields under the umbrella term computer vision, IEEE Transactions on Image Processing and the International Journal of Pattern Recognition and Artificial Intelligence are to name but a few. Of course there are umbrella journals too, such as the International Journal of Computer Vision, and Computer Vision and Image Understanding.

As such a popular topic there are also a multitude of books available on the subject, such as The Image Processing Handbook, or the more recent Image Processing Analysis and Machine Vision.

For project advice and subject clarification a computer vision researcher is also available for one hour a week.

The Internet will also be of use in this project, from The Computer Vision Homepage to the OpenCV forums, both code examples or solutions to sub-problems may be available on the internet.

Development Resources

The project will make use of Intel's Open Source Computer Vision Library[1]. OpenCV has been used by AUVSI entrants in the past and is well suited and in fact aimed towards real-time applications, sporting heavily optimised code and common computer vision functions. Its code was originally created inside Intel, and optimised by hand for Intel architecture chips, some functions are even implemented in assembly for the highest performance possible. OpenCV is implemented in C++ but compatible with C, thus dictating the use of one of these languages for the rest of the project. The only other serious contender for implementation language choice would have been Matlab. Commonly used by academics and professionals for Computer Vision applications, Matlab provides support for a host of mathematical operations useful in this domain, as well as a simple semantics for representing matrices, images, and operations on them. However one criticism of Matlab is that it is difficult to create modular code. Also, the higher level control of the vision component will require reception of, and control flow decisions based on, messages from the AI component, as well as the possible need for user-defined structures, which would have made C the preferable choice regardless of library language. The author is more acquainted with C than C++ and, since our algorithms will unlikely require anything more object oriented than the simple structures available in C, C seems an appropriate choice.

Given this decsion, a C IDE will be useful to reduce development time, specifically

[1] Intel's OpenCV: http://www.intel.com/technology/computing/opencv/

easing bug finding and memory management. Fortunately the University have a partnership with Microsoft's Developer Network, and a copy of Visual C++ can be obtained freely through it for the purposes of this project.

C enables the developer to directly control the allocation of memory for the program which gives it speed advantages over other languages, but this also allows the developer to create memory leaks in programs. A memory leak in software intended to run for up to an hour is unacceptable (see section 3.5). Sustained memory leaks cause the operating system to make intensive use of its virtual memory, vastly slowing down the processor until the virtual memory runs out and the operating system stops the process. In order to avoid this eventuality an extension for Microsoft's Visual C++ will be used to monitor memory usage[2], this tool outputs warnings during debugging sessions, highlighting code which is leaking memory. The extra code required for monitoring the memory does not compile into release versions of the code so that our algorithm speed will remain unaffected.

Footage Gathering Resources

The AUV, in combination with the University's 25m swimming pool will be necessary to recreate the conditions encountered by AUVs in the competition. The AUV is already the owner of a standard Logitech webcam, which will be used to capture the videos. Either a laptop will be required to remotely network with the AUV from the poolside to initiate filming, or a program will be written to automatically capture video sequences whenever the AUV is switched on, in case the network cable is too short for comfort. In addition to this, waterproof targets need to be found or created to provide the targets for object recognition. The beacon will be provided by DSTL in late March, and preceeded by delivery of a video of it functioning underwater. A validation gate can be created from a scrapped school desk, while specific target shapes will be cut from aluminium and tin to meet test specifications (see section 3.6). All targets (save the beacon) can be spray painted to meet colour requirements. There is a lifeguard present at all times in the pool, courtesy of the University, so poolside safety is already taken care of. Finally the arrangement of targets in the pool, guidance and monitoring of the AUV can not all be done by one person so extra human resources will be required in the form of members of the BURST team[3].

[2]Visual Leak Detector by Dan Moulding - http://www.codeproject.com/tools/visualleakdetector.asp
[3]BURST: Bath University Racing Submarine Team, the team responsible for creating the AUV

3.5 Requirements Elicitation

Traditional Requirements Elicitation techniques are often concerned with developing large systems to be used by many people or groups, referred to as users. Large software systems often have many users, and such formalisms as Use Cases are useful to aid thorough analysis of the situation and required system. The system to be implemented in this project is very small, and has only one conceivable user - the AI component. Use cases and other such formalisms were clearly not developed with completely autonomous systems in mind, such as the one being developed, and therefore offer no significant advantages over the thoughtful and systematic consideration of potential requirements. There are also very few stakeholders involved in this project, therefore methods such as Viewpoint-Oriented Analysis were deemed unneccessary for the successful elicitation of requirements.

3.5.1 Sources of Requirements

The requirements of the system are derived from three main influences; the tasks to be performed, the users of the system, and the stakeholders of the system. We describe these sources here in order to justify the requirements drawn in sections 3.5.2 and 3.5.3.

Mission Statement

Highlighted here are those parts of the mission statement which will influence the decisions made during this project, while interested readers may find the entire mission statement and competition rules on DSTL's website [4].

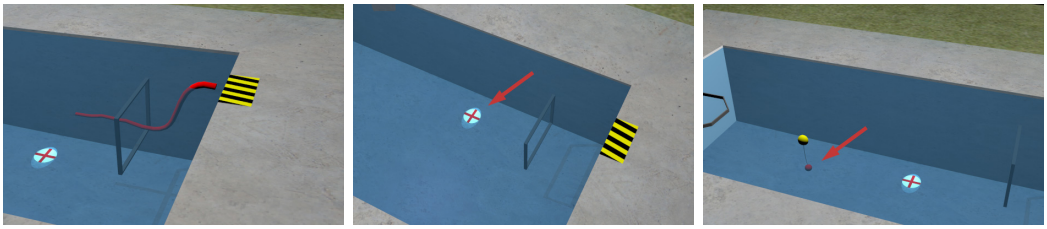
According to the competition rules, this year's entrants must be able to visually recognise up to five items. The first is a 'validation gate', through which the AUV must pass before attempting any other task, and for which the penalty of missing is the termination of the run. The second task described is to drop a marker on a target area on the floor of the pool. The target which we will refer to from now on as 'the sunken target' consists of a circle of minimum 1 metre diameter, which will have a contrasting cross at its centre, and will be illuminated by a flashing beacon. Finally the AUV must make physical contact with a bright orange midwater target. Figure 3.1 shows these tasks which make up this year's mission, as interpreted and illustrated by the University of Girona [5].

Users and Stakeholders

To explicitly establish the boundaries of the system to be developed, and analyse potential users, the inter-dependencies of the AUV's architecture are shown in Figure 3.2. Vision is shown to be directly involved in the mission tasks, and to have only inter-dependencies with the AI component. As such, the primary and solitary 'user'

[4] The current mission statement resides at: http://www.dstl.gov.uk/news_events/sauce/mission-rules-06.pdf.

[5] The University of Girona are also competing in the competition. Their team website is at: http://eia.udg.es/sauce/missio_EN.htm



(a) The AUV must first navigate through a 'Validation Gate'

(b) The AUV must locate a sunken target, illuminated by a flashing beacon.

(c) The AUV must then make contact with a midwater target.

Figure 3.1: The mission for DSTL's Student Autonomous Underwater Challenge - Europe 2006.

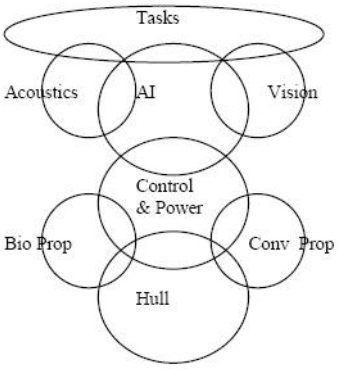


Figure 3.2: The responsibilities for the AUV different hardware and software components are divided between members of the team. Vision plays a direct role in the mission tasks and must collaborate with the AI.

of this system will be considered as the AI component. The main stakeholders of the system are the members of the BURST team, who want to see the AUV succeed, and primarily the developer and maintainer who will be responsible for adapting and applying the system developed during the weekend of the actual competition.

3.5.2 Functional Requirements

1. Target Recognition

   1.1. The system must be able to recognise three distinct kinds of object. Shapes, structures, and objects of modulating visibility.

      1.1.1. The system must be able to recognise the presence of a flashing beacon in the image. The frequency of its modulation will be made available in March.

      1.1.2. The system must be able to recognise the presence of the Validation Gate in an input image. The gate will be a rectangle of pre-defined dimensions and colour. The gate recognition should attempt to recognise gates that are only partially on-screen, where reasonable.

      1.1.3. Since the midwater target is as yet undefined, the system must be able to distinguish between 'classes' of shapes. Specifically, it must

be able to classify some representation of a given input shape as one of the shapes it is trained upon. The system must also be able to classify a shape as 'unclassified' in the event that it does not match any given shape with enough confidence.

1.2. Invariances

1.2.1. Depending on the angle of approach of the AUV, the ground targets will appear at different rotations. The system must be able to recognise the shapes despite this rotation, and thus must be rotation invariant.

1.2.2. Although very distant shapes will be made invisible due to light attenuation there is much room for targets to appear in different scales. Therefore every technique must be independent of scale, within reason.

1.2.3. In our environment scale invariance does not translate to distance invariance, as the effect of the colour loss through water and backscatter must be taken into account. The system should therefore make every effort to recognise objects which are in the distance, in order to save the AUV time in searching the competition area.

1.2.4. For the competition it seems that our AUV may be able to accomplish the tasks without having to look at a projected shape, the floor targets will be surveyed from a downward facing camera, and the midwater targets from a horizontal one, and the validation gate will be pointed at the AUV to begin the run. Despite not being critical, this project should attempt to measure the effect of projection on the algorithms, as it provides further insight into the qualities of the chosen techniques.

1.3. Each target recognition system should support some kind of confidence measure for internal utility during the competition as well as for algorithm evaluation purposes. Results must only be reported to the AI if the target is matched with a condifence measure greater than some defined threshold.

2. In order to satisfy the academic requirements of the project, the system must be designed in a manner that facilitates the training of multiple shapes. This is to say that it must accept some form of shape description as training data, and do so for multiple shapes.

3. The system must be able to both communicate mission information to the AI component, as well as receive instructions from it. Communications must follow a predefined protocol agreed between vision and AI.

4. Running Time.

4.1. The system should be able to run without human intervention for up to an hour (the AUV's battery life, and also the time allotted for each team attempt) and must be able to run for an absolute minumum of 15 minutes (the time allowed for a single mission run in the competition).

4.2. The competition rules state that the run may be restarted by the team at any point during the fifteen minutes. To this end the vision system

may support some kind of 'reset' message from the AI. This will depend on the AI's own support of such functionality.

5. The tasks must be able to be performed in an arbitrary order, instructions for which will be received from the AI module at run-time.

6. The competition rules state that marks will be awarded for AUVs which are economic. Though the vision system may not use as much power as the fins there is definitely scope for economy. The system may contain a number of features designed to save power, such as not using cameras unless neccessary, and switching off when no longer needed.

7. The vision system may search for all items concurrently. And report relevant information to AI at any time. This feature was requested by the AI designer so that the AI has more knowledge and is better poised to make control decisions.

3.5.3 Non-Functional Requirements

There are a number of system requirements which are less measurable and quantifiable, and which fall under the non-functional category.

Real-Time Framerate The system must be able to process images at an acceptable rate. Processing a frame is defined as the time taken from the moment the system receives an input image from camera or video, until it is ready to receive the next frame. An acceptable rate will be judged by the designer and may be different for each task.

Accuracy Arguably the most important requirement of all, if the system is not accurate then there is no point in it doing anything at all. The system must be as accurate as possible, however it will likely need to comprimise to entertain other high priority requirements such as speed and testability.

Robustness In order to meet the Running Time requirement, the system must be robust. Unexpected inputs or messages from the AI must be handled without incident. Equally, the system must be prepared for internal errors, and handle them with no consequences.

Reliability The system should be engineered in the best manner such that only pre-defined messages are sent to the AI, and in an understandable manner. The system should not send undefined or non-sensical messages to the AI.

Versatility The system must be as adaptable as possible to new situations and requirements. Previous AUVSI competitions have purposely kept information back from the competitors in order that entries be of the most versatile nature. The DSTL challenge is to be no different, and as yet the midwater target has not been properly defined. It should be possible to re-train the shape recognition system as quickly and easily as possible, with the minimal amount of target footage. Ideally this will include a blob-selection mode where the target is picked out by an expert user from a sequence of video frames.

Testability In order to satisfy both the academic requirements of the project, and produce the best possible system for the competition, and not least to aid the

debugging process, the code must have built-in feedback mechanisms. The feedback may be of statistical form such as framerate or success rate, or a visual nature, such as the lines detected in an image.

Maintainability Future generations of Bath students will undoubtedly be following in the footsteps of this year's team, and improving on the work undertaken here. To this end, the code should be well documented, commented, and easily readable.

A couple of potential conflicts arose within these requirements; the first is that of testability and the need for a real-time framerate. Outputting feedback on the operation of the code, and of results can take significant processor cycles, which are entirely unneccesary in the context of a run in the competition. To this end a solution had to be found to accommodate both requirements. The solution in fact was both trivial and effective, by using C's `#define` statments we were able to insert feedback code which could be hidden from the compiler with speed and ease. The second conflict is the desire for maintainability, which again has issue with the desire for real-time processing speeds. Many speed optimizations can impact the readibility of code, making it less glamourous or its purpose less evident. This is a difficult conflict and has no trivial solution. The higher priority of speed must be allowed to take precedence over code readibility, however wherever readability sacrifices are made useful comments can be inserted to mostly make up for the loss.

3.6 Testing

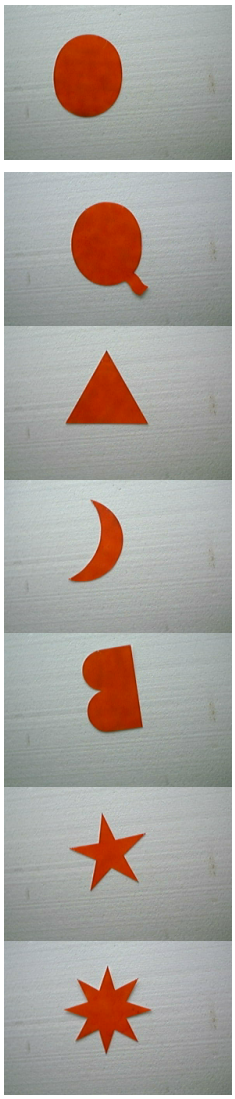
This project implements algorithms to identify three different classes of objects (see competition mission in section 3.5.1), those recognisable by shape, those recognisable by structure, and those recognisable by the frequency of their modulation. As described briefly above, in order to develop and eventually evaluate these algorithms a suite of testing videos will be necessary. The videos will attempt to test the limits of each algorithm by varying the variables which may affect their performance. The beacon video as stated will be received in March, and a mock validation gate was created from an old school desk, however the shape testing required an extra level of detail.

3.6.1 Shape Recognition

In order to be able to observe the effects of shape on our algorithms a set of test shapes were created, with the intention of stretching each technique and representing as many shapes as possible in the testing.

The shapes were painted white on one side and fluorescent orange on the other, as orange has featured in past AUVSI competitions, and will do so in this year's DSTL version. The reason it has been chosen there, and is chosen here, was well put by Philip Brown from DSTL's SAUC-E steering committee. "If you are going to put something underwater with the intention of finding it afterwards, you don't paint it black". White was chosen for its neutrality, and because it is also a bright colour.

Figure 3.3 shows and describes the shapes created for evaluation, all of which were created at a uniform scale, such that size of the shape would not have an influence on results. The requirements though state that our algorithms must also be scale invariant, so further shapes were needed of constant shape and varying scale, and are shown in Figure 3.4.



O: The base case for both algorithms, the O is supposed to be easily recognisable up to the most difficult circumstances and should provide an interesting comparison to recognition of 'harder' shapes.

Q: Designed to test the algorithms' attention to detail and ability to distinguish between very similar shapes. The Q is the same size and shape as the O, with an added detail.

A: Another simple shape for similar purposes as the O, yet this time composed of straight lines rather than curves.

Moon: The moon is supposed to push the algorithms in terms of their ability to recognise slender, elongated shapes.

B: Intended to represent shapes which are composed of both curves and sharp angles as well as straight lines and corners.

Star: A deviation from the low-curvature shapes already shown, the star tests the algorithms' abilities to determine slender shapes with high frequency borders.

Barnacle: The hardest case, intended to measure the accuracy of classifications between this and the Star, a high-frequency version of the Q and the O.

Figure 3.3: The shapes designed for the purposes of evaluating our shape recognition algorithms.

3.7 Schedule

As an undergraduate dissertation, and also as part of a team effort, this project must meet some important time constraints. Firstly, dissertations are due in on the 8th May 2006, before which all academic portions of the project must have been finished. Some of the requirements though may be allowed to slide beyond the dissertation deadline, as they are more oriented towards a successful competition entry than

Figure 3.4: An E was chosen for its large edge area such that it might be more affected by environment variables (distance/noise) than more compact shapes such as the O. The original E was recreated at 50% and then 150% of its size for the algorithms to be tested against scale-invariance.

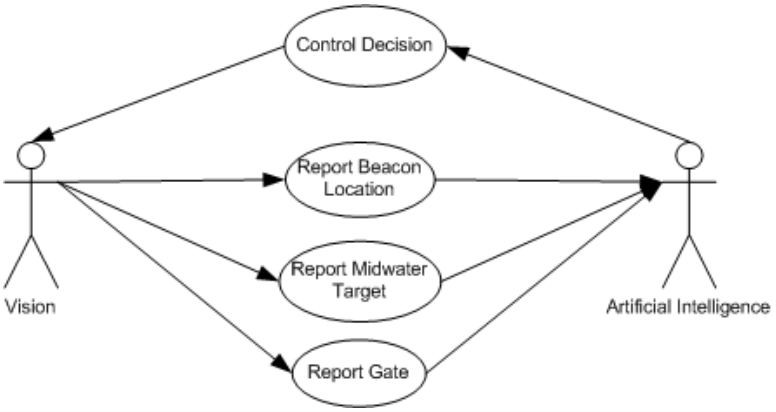


Figure 3.5: Use case model for the Vision Component

as part of this academic investigation. For example the assisted-learning feature described, or the fully implemented protocol. In addition to the academic deadlines there are a number of team deadlines which due before the 8th May. There will be a design freeze in early February, by which time the vision component should be well enough defined that informed decisions on the submarine's design can be finalised. This is followed by code delivery at the end of March, ready for integration testing.

3.8 System Analysis

While we have already discussed the reasons for not formally analysing the system, Models of Use as explained by Sommerville (2001) are well suited to provide a formal analysis of the communication aspects of the system, as it is precisely interactions between actors that they model. Use-cases identified can be expanded into use-case descriptions, detailing both the data involved in the interaction as well as what is needed to initiate it. This could not be more ideal for designing a protocol.

The main use-cases are shown in Figure 3.5, for which use case descriptions were developed and presented below.

In the following use case desriptions, the System is always assumed to be the Vision component, while the actors are always Vision and AI.

Use Case	Control Decision
Data	The AI sends a message to Vision indicating either a control decision, Start or Stop, or a flag indicating which target it would like to receive information about. The data is the control instruction itself, each message will contain an instruction.
Stimulus	The AI makes a control decision requiring the change in operation of the Vision Module.
Response	The AI sends a message to inform Vision of it's decision.

Use Case	Report Beacon Location
Data	Vision sends a message to AI informing it of the location of the Beacon relative to the image it was located in. The data sent is simply an x and a y coordinate, with origin at the bottom left of the image. The same applies for reporting information on the Circle and the Cross.
Stimulus	Vision realises that it has identified a target in the frame.
Response	Vision sends AI a message informing it of the target.

Use Case	Report Gate Information
Data	Vision sends AI a message listing the x-y coordinates of all four corners of the gate, relative to the screen's origin in the bottom left corner.
Stimulus	Vision realises that it has recognised the validation gate in the frame.
Response	Vision sends AI the information.

Use Case	Report Midwater Target
Data	The Vision sends AI either information on the distance of the midwater target, or sends confirmation that the midwater target has been touched.
Stimulus	Vision has recognised the object in frame and knows the distance it is from the AUV.
Response	Vision sends AI the appropriate message.

Admittedly this may not everybody's idea of the correct use of use cases, however borrowing the structure of use cases analysis in this way *does* allow us to break down the requirements into formal chunks and eases analysis while reducing the probability of requirements being missed, so whether it is 'correct' or otherwise is not such a great concern for this project.

3.8.1 Protocol Design

The use case descriptions can essentially be used as a design specification for the protocol. The only choices which remain are how to implement it, these are described here as they flow comfortably from the analysis.

As described in the Literature Review, communication will be performed over TCP/IP, primarily because of it's wide availability. The only real networking choice was over which transport layer to use. UDP is a better candidate than TCP for a number of reasons. The first is simplicity of implementation, for which we have a need. The developer of the AI component is a Mechanical Engineering student, so any added complexity such as establishing a connection would have to provide an advantage elsewhere. The reliability (packet re-sending) offered by TCP is hardly valuable given that packets are only to cross a single cable, and certainly will never cross a router where the majority of transmission errors could be expected. UDP also tops TCP by offering multicasting, the ability to send the same packet to multiple destinations. This would be advantageous for debugging the communication or other interactions between the components of the AUV, by allowing a poolside spectator computer to be included into the same loop, and to receive copies of the same packets.

Lastly we must design the message format for our custom protocol. As mentioned above, the use-case descriptions more or less provide a specification for each set of messages. We simply add a header to all such messages to identify them as messages from Vision, and distinguish them from another system component or noise on the wire.

Again to keep the implementation as simple as possible every message has the same length, while format always depends on the first two fields in the message. The standard message format is shown in Figure 3.6.

Every field is a single byte in length. The first three bytes are the Vision Signature, to identify the message as valid. The next two are the opcodes, and determine the

0xff	0xfe	0xff	majorOpcode	minorOpcode	...

Vision Signature — 13 remaining bytes

Figure 3.6: The standard message format to be adhered to by both Vision and AI.

Start*:	1	0	empty...
Reset*:	1	1	empty...
Stop*:	1	2	empty...
Search for the Gate*:	2	0	empty...
Search for the Beacon*:	3	0	empty...
Search for the Circle*:	3	1	empty...
Search for the Cross*:	3	2	empty...
Search for the Midwater Target*:	4	0	empty...

Figure 3.7: 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.

format for the rest of the message. Each use case has it's own major opcode, while minor opcodes cater to the different information requirements of each different type of message relating to each object. There is no particular reason for this major/minor structure apart from that it is simple to understand and leaves plenty of room for future developments of the protocol, a whole byte per opcode leaves a lot of options. The contents of the message after the two opcodes are dependent on the opcodes themselves, and are documented in Figures 3.7 and 3.8.1.

In the following description of the protocol, 'empty...' means: fill the remaining fields with 0's. These 0's are not to be checked, they are supposed to avoid any situation where the message signature is sent out of context by accident or mistake and causes an error.

Navigation Information

The message from Vision to AI with major opcode 0 was purposefully left unassigned in case in the future the Vision system is enhanced such that it plays a more active part in AUV navigation, and makes navigation decisions such as to stop (to avoid collisions etc). It makes sense that important instructions such as stop, which could be used in an emergency, are given priority in the protocol.

The coordinates passed relating to the gate are of the its corners with precedence left to right, top to bottom. This is the current version of the protocol, but as explained later, the system was mostly developed under a different assumption while the AI requirements were being formulated.

*To address the issue of unreliability, key control messages in the protocol specification are marked with a star, and must be sent twice when they are sent. This may appear careless, but it is the same strategy that is employed by the Internet Control Message Protocol (ICMP), one of the core components of the Internet. If ICMP is robust enough to play such an important role in a worldwide network depended on

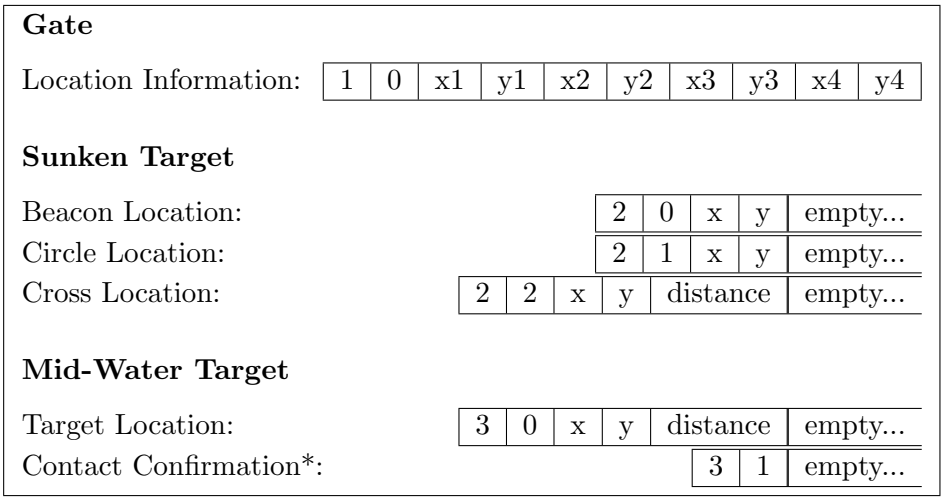


Figure 3.8: Information Messages: Vision to AI. The messages which vision will use to communicate results and information to the AI component.

by millions of people on a daily basis, then it is more than adequately robust for our dual-host application. It also allows the system to remain simple, and avoids the need for any confirmation of receipt messages.

Chapter 4

Locating Objects of Interest

As explained in the Literature Review, a fundamental early step of a computer vision application is to extract some form of information from a given input image. At its simplest this can be no more than counting the amount of pixels in a certain colour range, as demonstrated by Johnston (2004). Entire books have been written on the subject (Nixon and Aguado, 2002), explaining techniques ranging from the Hough Transform for extracting lines, to Active Contour Models which tighten around target features. All these methods however share the same goal, to take an image, and turn it into *information* which can be used by the classifier, be it a pixel count, mathematically defined lines, or any other information. This chapter discusses the techniques implemented for this purpose in this project.

This informational view of things enables us to understand the need for a common operation in computer vision; noise reduction. When transforming an image into information, noise in the image is understandably transformed into noise in the information. During the Literature Review a small selection of noise reduction techniques were chosen for further investigation due to their potential utility in our domain.

Two of the three techniques identified, simple averaging and the median filter, are already implemented in the OpenCV libraries. Not expecting to better their implementations these ready made functions were put to use in the project. The peak and valley filter required an implementation of our own, however the description of it's functioning in the literature review is sufficiently detailed such that the implementation poses no interesting questions or problems.

Figure B.1 (presented in Appendix B due to its scale) serves to demonstrate the properties of each filter, however any analysis of the results is meaningless unless it is considered in terms of the next operation. We therefore leave further mention of noise reduction until section 6.3.3, where it is evaluated with respect to the effect it has on the algorithms.

4.1 Region Localisation

When attempting to classify shapes in an image we must at the very least provide some way to inform the algorithm where one shape ends and another begins. In

computer vision this process is known as segmentation, the result of which is an image in which every pixel has been assigned a 'label' representing the class it has been placed in. In some applications, such as for robots in the RoboCup[1], these classes are pre-defined, and may contain a label for each target object; goal, ball, teammates, etc. Images are segmented so that the algorithms can identify objects in the scene, and analysis continues. In our case however we choose to define simply two classes of pixel for this stage; interesting and uninteresting. The intention being that all pixels considered as background such as the water or sea bed will be given the uninteresting label, while pixels which may belong to our targets are marked interesting. Connected groups of many such pixels (commonly termed 'Blobs') are special since they represent probable targets, and can be passed on for shape analysis.

This approach is appropriate for two reasons. The reason it is possible is that shape analysis should need only a shape as input. Shape is not influenced by colour or other measures available in the image, a square does not have to be green. The reason that this is desirable is because the underwater environment makes it difficult to make segmentations based on colour. The bias towards bluey-green light means that the same object *will* appear a different colour as its distance from the AUV changes, unlike the RoboCup environment in which colours are defined, static, and dependable. We must also remember the real-time requirement imposed upon the system, the minimal amount of processing should be undertaken at all stages. This binary segmentation is in effect the least amount of segmentation we can get away with.

4.1.1 Colour Based Model

Despite the difficulties already explained in dealing with segmentations based on colour, the first technique attempted just that. However rather than the static range of colour as used effectively in an algorithm for the RoboCup (Bruce et al., 2000) this technique is based on pixels beating a threshold of colour-space distance of subject pixel colour to a cluster of given target colours.

The reason that a cluster of target colours was chosen to be used as a sample is that it then becomes easy to model the variation in colour visible across the surface of underwater objects, by providing a sample image of the object when it is submerged. Then the Mahalanobis Distance of the subject pixel to the eigenmodel of this set is forgiving for pixels varying in a similar dimension to this underwater variation. The Mahalanobis Distance then should respond well (small distance) to colours similar to the sample data, and badly to those fundamentally different.

Given our sample set of target colours $\vec{p}$, containing n pixels, where

$$\vec{p} = \begin{bmatrix} r_1 & r_2 & ... & r_n \\ g_1 & g_2 & ... & g_n \\ b_1 & b_2 & ... & b_n \end{bmatrix}, \tag{4.1}$$

and with mean

$$\vec{\mu} = \frac{1}{n} \sum_{i=1}^{n} \vec{p}_i. \tag{4.2}$$

[1] http://www.robocup.org

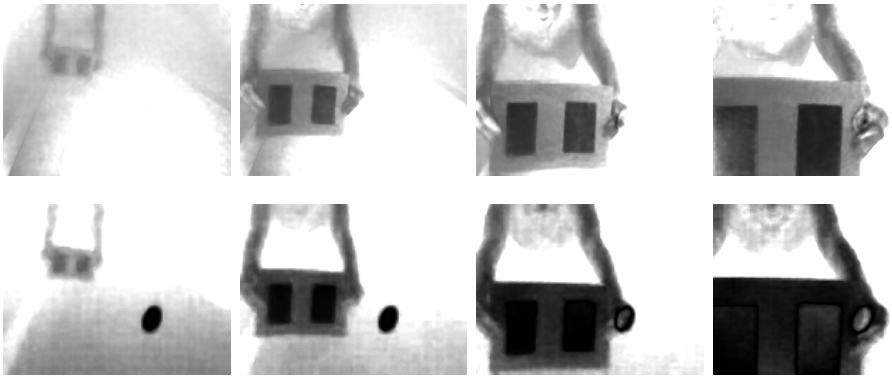


Figure 4.1: The difference caused by using different sample images to extract the target. Top: Using openair sample. Bottom: Using mid-distance sample.

The covariance matrix of the set $C = \left[(\vec{p} - \vec{\mu})(\vec{p} - \vec{\mu})^T\right]$, can be reduced into its eigenvectors U and eigenvalues V by eigen analysis, which then represent the principal components of the training set (the vectors in colour space in which the sample varies the most) and make up our eigenmodel e.

These principal components are then used in the Mahalanobis Distance calculation, already shown in section 2.6.2 but repeated here to demonstrate the role of the eigenmodel with respect to a coloured pixel. For a point (in our colour space) $\vec{p} = [r, g, b]$, the Mahalanobis Distance from the eigenmodel is given by;

$$D_M(\vec{p}, e) = \sqrt{(\vec{p} - \vec{\mu})^T U V^{-1} U^T (\vec{p} - \vec{\mu})} \tag{4.3}$$

When using this method, there is obviously a question over what set of colours to use for creating the eigenmodel. There are a few options available. Using a cutout of the object from an underwater image will be the perfect sample for that same image, but when the distance to the target changes, as it gets both closer and further away, less and more of the colour reaches the camera, and the sample becomes less accurate. Most unlikely to be useful then, would be using an image obtained out of water, since it represents a quality of colour which will only be found underwater if the object is extremely close. This argument can be appreciated in Figure 4.1 in which the targets are two pink rectangles inside a larger yellow rectangle. The intensity of the pixels in the Figure represent the Mahalanobis Distance from the pixels in the original image to the sample set. The darker the pixel the closer the match to the target. The open air sample improves visibly the closer the object becomes, but performs very weakly for anything which is even slightly distant. The performance of the mid-distance target is particularly good near the distance it was acquired at (frame 2), however it worsens quickly as the object draws nearer, where the outer rectangle becomes much darker (supposedly a better match) while the target itself begins to lighten, meaning the perceived colour of both has changed quite significantly.

This causes a problem however, as setting a static threshold will mean that the object will only be extracted when it is within a certain range of distance defined by the sample colour and threshold tolerance. The project requirements state that the algorithms should be tolerant to changes in target distance, meaning that a

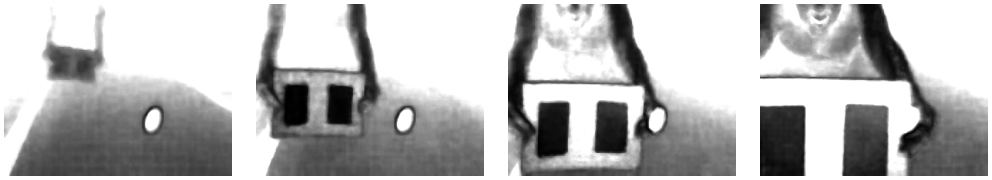


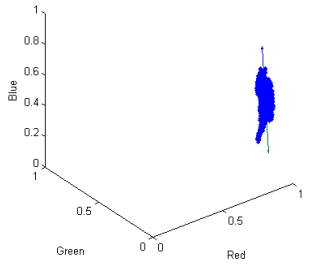
Figure 4.2: The effect of using samples which are mixtures of the colour at varying underwater distances, shown here is the result of a half-half sample of the target at approximately one and three metres.

solution needs to be found to this problem. A potential solution is to use an image which is a mix of 'snapshots' of the object at varying distances. This means that the training data will vary greatly in the colour 'directions' that water most affects, effectively digging a 'tunnel' through RGB space, oriented from the openair target colour point to the background colour point. This is semi-visualised for the reader in Figure 4.3, in which the principal components of each colour set are plotted inside the RGB cube, showing the difference that a mixed sample makes to the eigenmodel. As such the Mahalanobis Distance will react favourably to any colour which is in or near this tunnel (an ellipsoid defined by the eigenmodel), which could be described as the principal component of water's effect on this colour. The obvious downside to this approach is that a greater proportion of the RGB cube is being carved out by this eigenmodel, increasing the amount of colours with a low distance to it, which is noticeable as the larger proportion of darker areas which appear in the demonstrations of mixed samples compared to single distance samples. Still, it is clear that this image would provide a better basis for a static threshold, as the target's Mahalanobis Distance is consistenly low despite its distance from the camera.

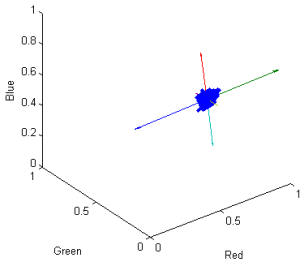
So we have a few choices over which sample colour to use. At this stage it seems that using the open air sample is out of the question, as it only gives low distances for very close targets. The mixture of samples seems to be the optimal choice, as it picks out the target at a distance equally as well as the mid-distance sample does, whilst being more robust to changes in distance of the actual target.

However there is also a question over whether using a different thresholding technique might yield better results than the static one assumed thus far. Explained in the Literature Review, dynamic thresholding allows a threshold to vary on a per-pixel basis, the advantage being obvious in our case that distant targets will only create a small signal because their colour is weak, meaning a low threshold is required. A low threshold however means that any closer objects which are similar in colour will also be highlighted, meaning that the entire near pool floor/sea bed could be lit, effectively obscuring any target which lay there to begin with. The effect of adaptive thresholding is shown in Figures 4.4 and 4.5, for which the open air target and the mixture samples were chosen. Adaptive thresholding works on local contrast, and these two have demonstrated the best contrast from their surroundings so far.

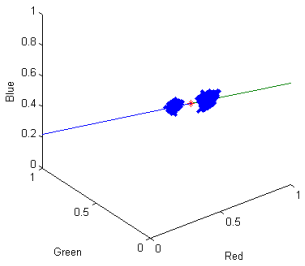
It is both encouraging and surprising to see that adaptive thresholding allows the algorithm to pick the target out of the distance as well as a human might, and does so the best by using the original open-air sample. This is because the training set has effectively been made less tolerant of the water, but the thresholder has been instructed to highlight signals if they stand out enough from their local area. It



(a) Open air pink.



(b) Underwater pink.



(c) Underwater pink mixed with distant underwater pink. Shows the 'tunnel' effect.

Figure 4.3: The progression of a colour as it moves further away underwater, its colours are diluted towards the colour apparent from the water itself. The sets of colours are the same as used for creating Figure 4.1 and 4.2, and the eigenmodel is plotted to show the principal components of each set.

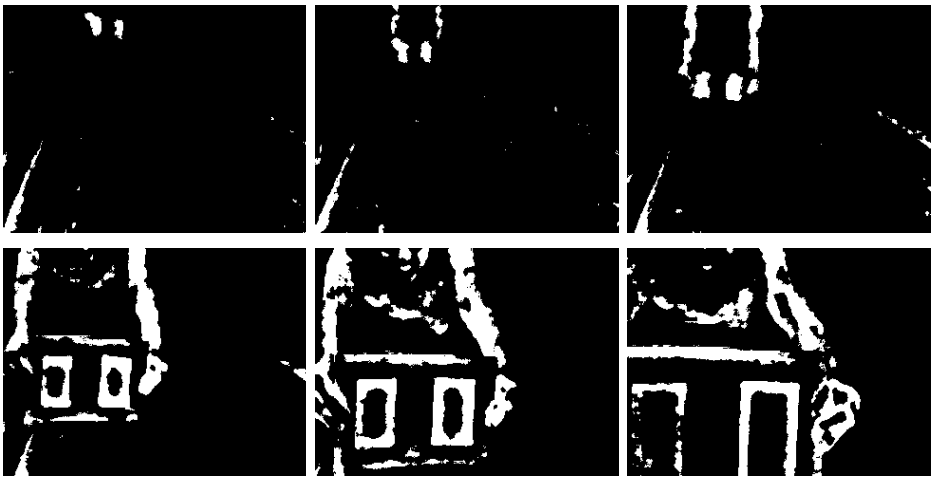


Figure 4.4: The open-air sample colour sequence shown in Figure 4.1 after being fed through an adaptive threshold.

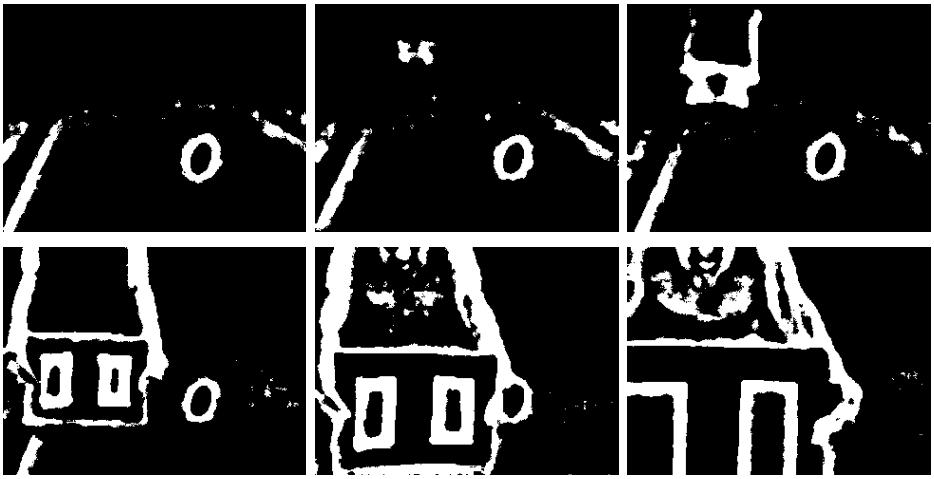


Figure 4.5: The mixed-sample shown in Figure 4.2 after being fed through an adaptive threshold.

is important at this point however to be critical and evaluate this result for our application. The image obtained from the adaptive threshold in the very first frame correctly identifies the target in the distance, but we must ask whether it produces a shape of good enough quality for us to work from in later stages. The targets are two rectangles, but neither of the most distant blobs would be recognised by humans as rectangles, and so ought not to be by any algorithm. Also, due to the neccessity of adaptive thresholds to work on a local basis, the insides of each target become hollowed out as their scale increases, because the thresholder finds no contrast within the object itself. This could have severe repurcussions depending on the next stage of analysis. Techniques relying purely on the border, such as our Fourier Descriptors (see Chapter 5) should be unfazed by this hollowed shape, but region based techniques like our Shape Descriptors would certainly struggle, since measures like area are heavily distorted. We must then consider the next step before declaring adaptive thresholding a success; extracting useless information is no better than extracting no information at all, and so both techniques are evaluated with respect to the tasks in Chapter 6.

4.1.2 Rarity Based Model

Our second technique for segmentation bears resemblance to that attempted by the University of Florida in their past AUVSI entry. By calculating the mean and variance colour values of what they assumed to be 'background' at the beginning of the run, Florida based all subsequent segmentation with this initial sample as a guide. The pitfalls of such a technique were discussed in the Literature Review. Here we propose a similar yet more robust technique, rarity-based thresholding.

With the intention of creating a technique more versatile to new situations than Florida's 'first frame is always background' method, we decide to use every frame to form an approximation of what is background and what is interesting. As an extension of the colour-based technique we then build an eigenmodel using all pixels in the frame, following which the Mahalanobis Distance between the colour of every pixel and this eigenmodel is computed.

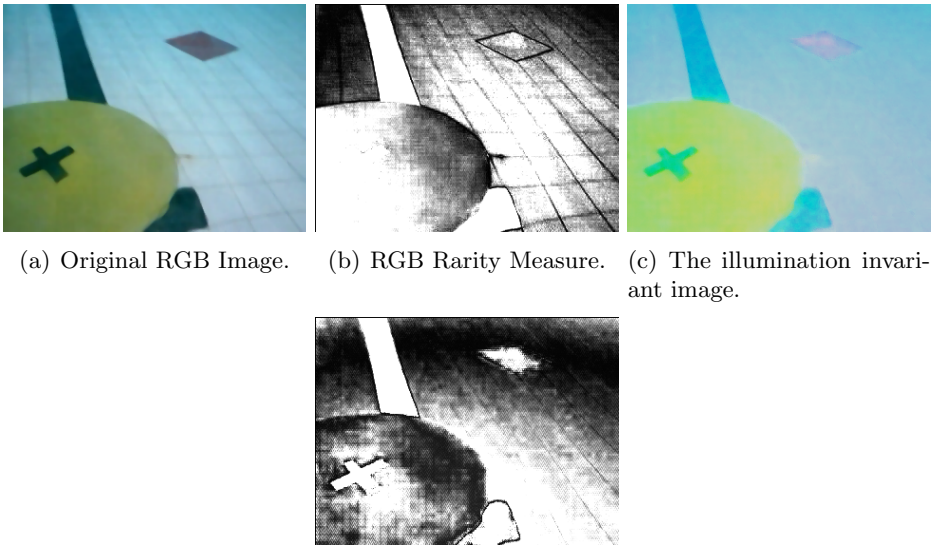
At this point two key questions arise. How great a portion of the image should be allowed to classify as rare to begin with, and how rare should a pixel be to classify as interesting. To understand the difference between these similar concepts, imagine the two following situations; the first, where we have nothing but background in the scene, if we dictate that the rarest 3 percent of the image should always be rare then 3 percent of this background will wrongly be classified as interesting. This is where the question of how rare a pixel must be is important. On the other hand imagine the scene is divided between two different backgrounds, sea and sea bed; many pixels may exceed a small rarity threshold resulting in large portions of the image qualifying as rare, but we do not wish to have an image in which 50 percent is classified as rare simply because the sample data underlying the eigenmodel is not of a compact enough nature. Both of these scenarios are potentially common in our environment, therefore this problem needs to be addressed.

Some time was given to investigating a 'second-rarest' approach to solve the problem of two contrasting backgrounds in the image. By taking the interesting pixels from the first frame and using them to mask the interesting part of the original image, a kind of 'second-order' rarity image could be produced, in which ideally the dominant rare portions such as sea bed will be removed, allowing a stronger eigenmodel to be produced. Both frames could then be passed on for shape analysis. However very little success was encountered with this technique. Sometimes part of the target object would be caught in the first threshold, then the other part in the second pass. It was decided that attention would be better spent on improving the first measure than continuing on unstable ground.

Other efforts to stabilise this rarity approach could have been scale-space approach, in which the rarest parts of the image would vanish, and lead to a stronger eigenmodel. Similarly a heavy averaging filter could have produced the same effect. Again adaptive thresholding techniques have an advantage in this specific situation, as the low gradient of rarity between the distance and the pool floor would mean little of it would satisfy the threshold, while the higher gradient created at the border of a small target would qualify. This approach would fail though in the situation where the target was located in an already rare part of the image, and so a better solution is needed.

Potentially some form of automatic classification could be implemented, attempting to deduce the existence of more than one significant cluster in the image and then perhaps creating an eigenmodel for each significant background. However this begins to cause problems for our real-time requirement, as performing this step on each frame would be expensive.

While we have made advances from Florida's original algorithm, this style of approach continues to contain serious flaws. The main problem is that the contents of each scene themselves have an effect on the algorithm which is trying to segment them. One effect of this is that as soon as an object takes up more than a certain percentage of the image by definition it ceases to be rare and will be classified as uninteresting, whether it is background or not. A simple remedy was found to solve this problem and is discussed shortly. Further problems are encountered in situations where the background colour is not sufficiently consistent, resulting in large portions of the image being classified as interesting. This has a doubly negative effect as it highlights the only weakness of our binary segmentation scheme. Interesting objects which are adjacent to one another, ie touching or overlapping, will -as the technique



(a) Original RGB Image. (b) RGB Rarity Measure. (c) The illumination invariant image.

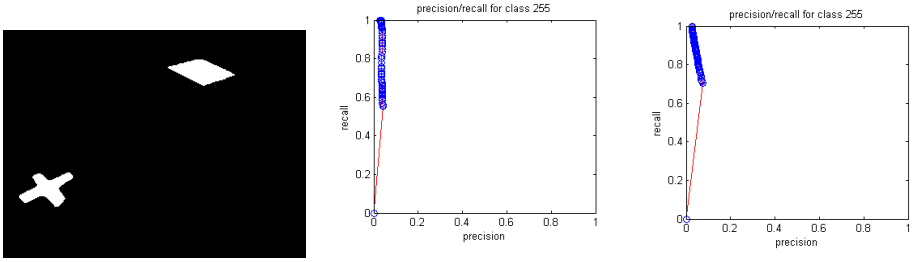
(d) HS(V) Rarity Measure.

Figure 4.6: Better results were obtained by performing rarity-based thresholding on HSV images with their luminance set to a fixed value of 0.8.

stands- end up both being part of the same blob. This is not unavoidable, previous and current dissertations here at the University of Bath have explored techniques for judging whether two adjacent objects actually form the same object or belong next to each other by designing fitness metrics which could for example evaluate the curvature of the border of the shape and deduce that it should be split in two. The application of such fitness metrics might would likely be out of the question for our real-time system however, and as the time allowed for final year projects is limited this is not an avenue which can be pursued.

Illumination Invariance One significant difficulty encountered during development of both of these techniques, as already identified in section 2.3.1, was the effect of light on the resulting segmentation. The most dramatic effect was observed during rarity based thresholding, and can be seen in Figure 4.6. As light travels with less ease through water than it does air, we encounter more difficulties than might be expected using similar techiques in other environments. The amount and colour of light reaching the camera changes dramatically over a very short distance. Figure 4.6 shows images in which pixels have been converted to their Mahalanobis distances from the frame's eigenmodel, meaning a high intensity represents a rare colour. It is clear to see that the image with a fixed luminance value causes the rarity measure to highlight the objects that a human might regard as rare in the scene. The RGB segmentation is highly affected by the change in lighting across the center of the pool floor, while the HSV version more understandably classes the closest area of the floor as rare. The distant target also has a stronger signal in the HSV image. The most remarkable difference is probably over the circular target, the segmentation of the RGB image decides half of it to be rare and not the other half, while the constant luminance image performs much more consistently.

In order to objectively assess the quality of this approach, we marked the same image by hand, highlighting the rare targets visible, and plotted precision/recall curves for



(a) Ground Truth Image (b) RGB Rarity Precision/Recall (c) HS(V) Rarity Precision/Recall

Figure 4.7: Precision/Recall curves for both rarity measures shown in Figure 4.6.

both the RGB rarity measured image and then the normalised HSV image. The ground truth and results are shown in Figure 4.7, where it can be seen that though both images had very high recall, the precision of the RGB image was unavoidably poor while the HS(V) image could be thresholded more precisely at only a slight loss to the recall quality.

Our Two Stage Tracking Process Given that the rarity based thresholding is based on rarity, it is destined to break down as the AUV closes in on this rare object, as the object's colour becomes dominant in the image. This is handled by a two-stage tracker. Any rare blob which the system deems likely to be one of our targets is tracked. As the AUV approaches, and the blob's area nears the boundary of what will still be classed as rare, a 'photograph' is taken of the object, using the blob as a mask over the input image, and sampling it to initiate target-colour based static thresholding as described first in this chapter.

Rarity-based thresholding though can be temperamental, as explained it can be affected by the content of the scene. In order to compensate for this slight unpredictability we impose that a blob large enough for photographing must exist for a minimum of 15 frames before it is sampled and tracked based on its colour. This is simply implemented by using a running average frame, where input frames are scaled by 1/15, and kept in a circular buffer until it is their turn to be subtracted from the running average. Any blobs in this running average frame are considered temporally consistent and can be analysed further.

A fixed number of pixel samples are always taken from the blob, but the number of samples is far fewer than the area of the blob. The sampling method implemented is simple, regular samples over the blob at a calculated interval such that the sample covers the entire area of the blob.

Even though we have switched to colour tracking the object may get closer still, thus potentially altering its perceived colour. Although not implemented for this project it is suggested that if this is the case, a simple resampling of the blob at not too distant intervals will suffice to prevent the object disappearing under the threshold. This interval should be engineered to resample the blob before the leeway given by the threshold is exceeded; this could feasibly be judged by the area of the blob, if it grows enough to exceed a certain scale factor threshold, then assume it has become nearer and resample it.

Figure 4.8: Every pixel is converted to its Mahalanobis Distance from the set of target colours, and a threshold applied.

In order to summarise the work and aim of this part of the project, and explain the transition to the next chapter, Figure 4.8 shows how an input image can be converted into a blob ready for shape analysis.

4.2 Beacon Recognition

Probably the least demanding of the tasks, the beacon poses no major problem in being located as it gives off a huge aura of rarity in any image. Contrary to the previous merits of forcing a constant luminance on the image we choose to leave the luminance as it is, as the brightness is the most distinguishing feature of the beacon.

To locate the lit beacon is trivial; we first perform a rarity-based threshold, then any remaining blob which has an average intensity greater than a high threshold, and exceeds a minimum area threshold is positively identified as the beacon. As no test video has as yet to provide a non-beacon blob which meets the set criteria, no provision has been made to select between competing blobs at this stage.

Strangely, the application of our usual noise reduction has an adverse effect in this scenario, which is caused by two factors. The first is that our basic noise reduction techniques work on the basis that noise is fluctuations of intensity in a local area (in each colour channel). By averaging or taking the median of this local area, we lose the highest peaks of intensity, and intensity valleys approach the mean. This is counter productive considering that we are conducting a search based on pixel intensity.

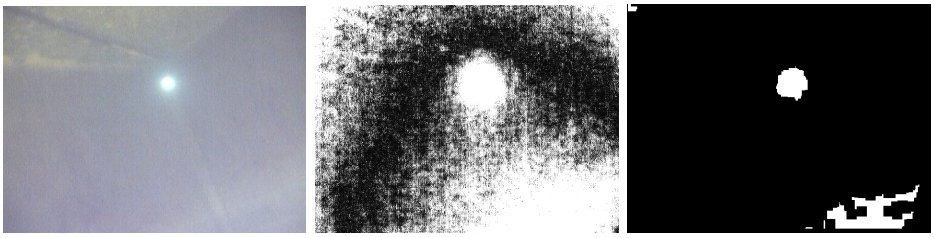


Figure 4.9: Every pixel is converted to its Mahalanobis Distance from the set of all colours in the frame, the results of this rarity measure are then thresholded to segment rare image components. The third image is the thresholded rarity image after 10 erosions.

Figure 4.10: Samples of the video of the competition beacon provided by DSTL, each image is 25ms on from the last.

4.2.1 Signal Detection

Thus armed with a method to determine the presence of the beacon in the frame, we wish to verify that it is our target beacon with a specified modulation of 250ms on, 750ms off. To accomplish this, upon every identification of the beacon in the image, its measured intensity is placed at the head of a circular buffer 100 slots long. The reasoning behind the length of this buffer is exactly the same as behind the requirement of temporal persistence of rare blobs, we wish to identify the beacon only if it has persisted in the image for a few cycles at least. This history of image intensities is then treated as a signal and transformed into the set of its frequency components via the Discrete Fourier Transform, whose implementation was borrowed from Paul Bourke[2] of Swinburne University.

The frequencies contained within the signal are then available for identification as the frequency of the beacon or otherwise. This is a source of difficulty, since the actual frequency represented in the signal history will depend on the sampling rate, the rate at which the algorithm can process a single frame. The competition beacon is estimated to operate at a frequency of 1Hz. Assuming that we were able to process ten frames per second, and given that we have chosen to keep a history of 100 frames, this would mean that the whole signal represented in the buffer would be the tenth harmonic of the original signal, phase shifted somewhere between zero and one tenth the period, and therefore the 10Hz frequency bin would have a magnitude vastly greater than the 50 remaining frequency bins returned by the DFT (the other 50 returned are redundant since we are dealing in real signals). The magnitude itself would in this perfect scenario be approximately half the measurements of intensity from the image and so cannot be calculated precisely in advance, however it should be distinct enough that if an empirically defined threshold is met then the beacon can be declared as present in the signal.

Well known to practitioners of Digital Signal Processing however is that such faultless maths is unlikely to occur in real world signals. While in theory our perfectly sampled 10Hz signal should fall only into the tenth frequency component, in practice due to measurement errors the strength of this frequency will be spread across a few bins in that area, as the exact frequency falls in-between the 'gaps' of the discrete frequencies; an artifact known as 'spectral smearing'. The effect this smearing will have is to reduce the magnitude of our tenth component and share it between neighbouring frequencies in smaller amounts, meaning that our threshold can not simply be just below half the expected intensity of the beacon, and must be considered in terms of this smearing effect. We continue this discussion in Chapter 6 where the effects can be seen in actual algorithm results.

[2] http://astronomy.swin.edu.au/ pbourke/other/dft/

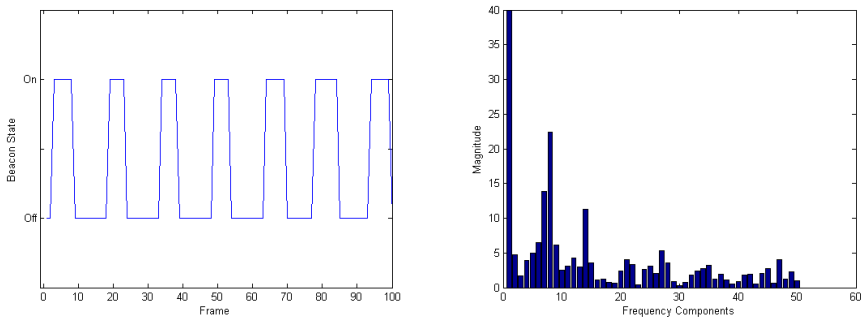


Figure 4.11: The signal generated by the beacon is effectively a square-wave, and as such will contain a number of high frequencies. Fortunately they do not interfere with the low level fundamental frequency of the signal and so do not need to be addressed. Also noticeable is the spectral smearing caused by mathematical imperfections in the original signal.

Since only one frequency component should be needed, there is no need to calculate them all as was the initial implementation, used to test the concept. Single-bin DFT calculations are the obvious choice, and were implemented at this stage of the project, however this requires recalculation over the entire signal for every frame. The Sliding Goertzel Transform (Jacobsen and Lyons, 2003) has been shown to provide

> computational advantages over the traditional DFT or FFT for many applications requiring successive output calculations, especially when only a subset of the DFT output bins are required.

It is clear that this is precisely what is required of this implementation, and the reason why it has been put to the same use previously in the AUVSI competition (Apgar, 2005). Sliding transformations recognise and take advantage of the fact that the signal is being effectively viewed through a 'sliding' window over the whole signal. They are therefore implemented as Finite Impulse Response digital filters, meaning that once a frequency coefficient (magnitude) is obtained, the frequency coefficient of the same signal viewed through a window advanced by one sample can be calculated using one shift operation, one addition and one subtraction.

Another concern comes from the fact that our signal is effectively a square wave, shown in Figure 4.11. Because of these sudden changes in amplitude the frequency components will also contain some high frequency components, not associated with the fundamental frequency of the signal but required to generate the sharp impulse like transitions as the beacon switches state. This should not concern us too much however, as the beacon itself is of a low frequency the higher frequencies should not interfere with our identification.

Figure 4.11 also demonstrates the effect of spectral smear. As the signal imperfections could vary over time, and cause the resulting target frequency bin to be of unpredictable magnitudes due to this smear, we must consider attempting to reduce it. Spectral smear can be reduced by 'windowing' the input signal, where windowing implies that a template is convoluted with the input signal to alter its properties, making it more accessible to DFT analysis. We are in fact already windowing the

signal, however the rectangular window we are using (abruptly starting and stopping) is known to be produce bad spectral smearing. The Hann Window (Blackman and Turkey, 1959) is chosen by Jacobsen and Lyons (2003) since it can be applied in just a three point convolution at each step with the frequency domain representation of the signal (although this implies also calculating the two bins neighbouring the frequency of interest).

The consequence of all of this is that after our initial 100 frames, successive frequency bins can be calculated in 3 multiplications and 4 additions, as opposed to the $2N$ multiplications and $2N$ additions used in recalculating single-bin DFTs. However for the moment, the most significant processing expense is the rarity measurement of each pixel, therefore before the Sliding Goertzel algorithm is introduced the rarity measure will have to be optimised.

The stumbling point for the implementation of this algorithm is our real-time requirement, rather, the fact that the algorithm is monitoring a live external process. While the code functions well on a test video, in real-time two problems can occur. Firstly if our algorithm is not running at a fast enough frame rate to remain under the signal's Nyquist Limit then it can not accurately identify the signal. The other problem is that our sampling of the process may be not regular enough, perhaps the code varies from one frame to the next, perhaps due to other processes running on the machine or code which is not sufficiently predictable. Whilst one might assume that this is is not a problem, that code will always run at the same speed, it does in fact turn out to be an issue. The mechanism used to separate all the connected components in an image is understandably affected by how many blobs exist in the image to begin with. If the thresholded image contains a lot of noise the code is slowed considerably, despite the fact that tiny blobs are thrown away. This conflicts with our need not to de-noise the image, but a solution was found in the form of erosion, the process of 'chipping away' pixels at the edge of blobs in the image such that small blobs disappear completely.

Eroding the image is a fantastic way to reduce this noise, and in fact works *better* when no smoothing has been applied, as the noise (of which there is a lot) will generally remain unconnected in local areas. By chance it also solves another significant drawback to our technique.

The beacon casts a great aura of rarity around it, at times causing the resulting blob to be of double or triple its true diameter. By eroding this aura-blob we obtain a mask which fits closer around the actual beacon, rather than its glare. This is invaluable considering that the method used to decide on the beacon's presence is the average intensity of it's detected area. By reducing the aura's size, a greater proportion of the sample comes from the beacon's center, the brightest part. This technique is essential if the beacon is found in a very dark environment - as the rarity threshold is likely to include the entire aura, due to the lack of any other rare colours. The mask then applied will yield a relatively dull intensity due to the inclusion of surrounding semi-dark pixels. Finally there is a the question of how many times to apply erosion, it is important to get a good intensity yet also important not to erode the entire beacon if it is in the distance or if the environment contains a lot of rare colours, so a fixed number of erosions is not the optimal solution. There are a few options for dynamic calculation. The figure could be proportional to the area of

the potential beacon blob, or depend on the average intensity in the image, aiming to erode a lot if much of the image is dark, and less if the image is bright. Both techniques could be tested during the evaluation stage.

4.2.2 Optimisations

As mentioned above this technique will rely on the AUVs ability to process frames faster than the beacon's Nyquist Frequency. The initial implementation operated nowhere near fast enough and required significant optimisations in order to get it performing quicker.

To speed up rarity thresholding the same eigenmodel is used for 15 frames in a row, before calculating a new one. This is possible since it has been identified that the AUV will move particularly slowly, the content of a frame should not change significantly enough over 15 frames to cause many problems for the remaining computations. The speedup won from this optimisation however was not enough and more needed to be made.

Allocation of memory and other such structure initialisations also incur a time-cost when they are performed on a frame-by-frame basis, and so where possible any such code was moved into an initialisation function to be called at the beginning of program execution rather than every frame.

After these two improvements were made, the biggest loss of time became the Mahalanobis calculations of pixel colour from the eigenmodel. A Euclidean measurement would be less involved and if performed without using floating point calculations a significant speedup could be achieved. The question though, is whether it will perform reliably enough to be worth using, it is not something investigated in this project, but should be tested at some point if optimisations can not be found elsewhere. Another alternative might be to reduce the number of pixels in the computation as described during our discussion of previous competition entrants, thus easing the load on the Mahalanobis calculations. Although brief attempts at this resulted in a loss of average beacon intensity due to the desire to use OpenCV's fast image scaling method, the potential is still there.

4.2.3 Tracking

Once we can safely recognise both a beacon present in the current frame, as well as its presence in the last 100 frames, we finally need some method to track its location when it is off, or passes undetected. This tracking can also be used as an extra stage of verification, that what we think is the beacon truly is the beacon, rather than a spurious piece of the image masquerading as the beacon for one frame.

To this end we employ the Kalman Filter, as described in the literature review. The filter is implemented by OpenCV and all that is required is to decide how many degrees of motion we wish to track. The motion of the AUV means that image contents are capable of being static, moving steadily, as well as acceleration and deceleration, but are offered no more degrees of freedom. Our decision is made for us then, and the filter is implemented to be of the 2nd-order.

The other parameters available to the Kalman Filter concern how confident state

updates are, as well as how fast the filter should converge to an answer. The filter should never converge as our objects are always free to change their state of acceleration, as the AUV may collide with an obstacle or be pushed in the right direction by a competition aid. Also as a final consideration, even when the beacon is not detected we must still update the filter's state, and so we simply feed back its own prediction.

4.3 Summary

Two methods for extracting target information from images were implemented, and we experimented with a number of combinations of thresholding and sample data, some of which show promising early results. It was noted that the next stage must be considered when trying to judge the quality of each solution, the beacon finder for example would not function if adaptive thresholding was used, as the brightest part of the light might be hollowed out. Adaptive thresholding showed good potential for extracting distant targets, and allows us to use a sample obtained in open air, prior to any footage being taken underwater at all which could potentially be an enourmous advantage in a real-world situation where the autonomous sub must go and find a newly sunken target. For this project though, since what follows from here is mainly an academic comparison of two techniques it would be unfair to use adaptive thresholding, given that the Shape Descriptors rely on such measures as the area of a shape, which adaptive thresholding subverts. Therefore static thresholding will be employed until the Evaluation chapter, using as a sample whatever creates the fairest information on an image-by-image basis.

Chapter 5

Shape Identification and Analysis

The previous chapter explained how we move from receiving an image, and how potential targets can be extracted from it. This simple extraction sufficed for finding the beacon, once an object passes the extraction constraints, namely intensity restriction, then we believe it to be the beacon. For recognising shapes however no such simple measure will suffice, as stated before the only thing which determines a shape is its shape. The Literature Review explored various techniques for computer representations of shape and concluded that of these techniques the most promising were the Fourier Descriptors and the Shape Descriptors found in Russ (1995) and Sonka et al. (1999). The first half of this chapter discusses the implementation of these two techniques for shape recognition, while the second half concentrates on our other functional requirement which can not be solved by simple feature extraction, the recognition of a validation gate.

5.1 Object Classification

The two techniques for shape recognition that were implemented work on two fairly different bases, the Fourier Descriptor method relies on the strength of one measurement alone, the signal of some function of the shape's border, and could be considered quite a sophisticated approach to shape representation. Shape Descriptors conversely could be conceived as quite a crude approximation to a shape. Used individually they would be extremely weak at distinguishing between even fundamentally different shapes, but the power lies in their combination. It was predicted that the combination of a well chosen few Shape Descriptors would lead to an acceptably strong distinction between possible shapes. This section will mainly discuss the decisions made during their implementation, while their evaluation takes place in the next chapter.

5.1.1 Fourier Descriptors

The function chosen for signal generation was the distance of the border, $d = [x(t), y(t)]$, from the centroid of the blob, $[x_c, y_c]$, which we shall refer to as the

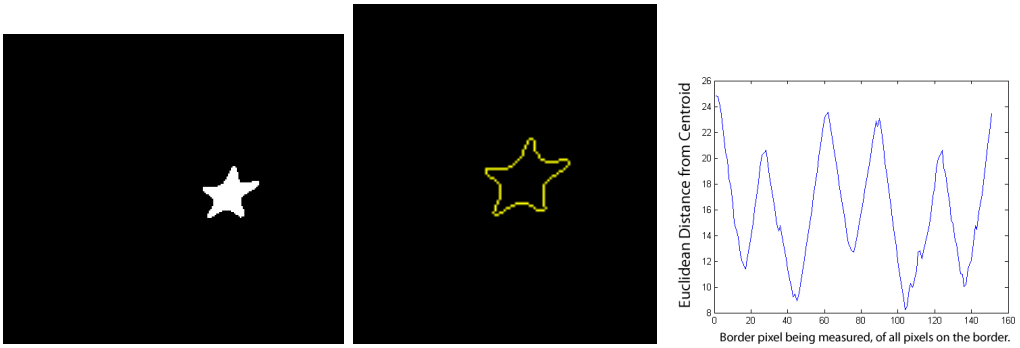


Figure 5.1: A blob is converted into a border representation, from which its radius signal is generated.

Radius Signal;

$$r(t) = \sqrt{(x(t) - x_c)^2 + (y(t) - y_c)^2} \tag{5.1}$$

The radius signal was chosen after considering that any function acting directly on the border alone, such as curvature (measuring the rate of change in direction from one border pixel to the next), would be more susceptible to noise, especially for smaller scales. For reasonably large shapes the difference that a two or three pixel difference would make to the resulting radius signal is very small compared to the overall length of the centroid to the border, and therefore has a higher and preferable signal to noise ratio. Border curvature on the other hand would suffer greatly in the presence of such border noise (assuming it is calculated over neighbouring border elements, since object scale can change) and would likely generate a signal with many high frequencies. Although these high frequencies would effectively be filtered out later on by using only the few low frequency descriptors that are necessary to describe the shape, it seems only sensible to choose the method with a lower signal to noise ratio in the first place. This decision is supported by Zhang and Lu (2001), who suggest that the centroid distance signature of shapes is superior to that of other common signatures used for this purpose; curvature, complex coordinates, and cumulative angle function. However their results were based on a precision/recall measure, meaning that for classification using a few low frequency descriptors there may be in truth little difference. Regardless, we have captured a relatively noiseless signature of the shape, as can be seen in Figure 5.1.

In terms of the program, our method for measuring the Fourier Descriptors takes as input a shape in the form of a 'Blob'. A structure created for the purpose of containing a single shape the Blob holds a binary image in which there is only one white connected component; the shape to be analysed. The previously described techniques for isolating areas of rare or sample colour are used to create these Blobs, therefore they shall all be of a reasonable size. Incoming blobs are passed through the OpenCV function findContours, which returns the list of every pixel along the shape's border.

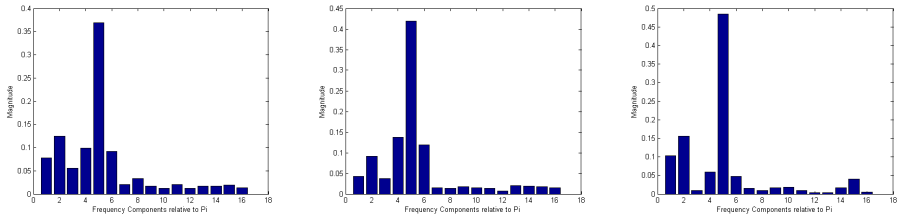


Figure 5.2: The Fourier Descriptors calculated from three consecutive images of the underwater Star shown above.

The blob's centroid is taken to be the average location of all the pixels within it. For non-convex shapes this could easily not be actually a pixel internal to the shape at all, however for the purposes of this algorithm it will suffice, as all we require is that a consistent signal is produced under any rotation of the shape. The location of this average pixel will always be consistently placed in relation to the shape regardless of it's orientation, and therefore the radius signal generated will also be consistent.

The radius signal can then be generated and passed to the DFT function, again courtesy of Paul Bourke. The requirement of scale invariance is achieved by discarding the first frequency bin (the DC component), and scaling the rest of the descriptors by the area under the descriptors histogram. As stated in the literature review, rotation invariance is achieved when we take the magnitudes of the Real and Imaginary frequency components to make the Fourier Descriptors themselves. Figure 5.2 shows three examples of the resulting frequency components of the Star's Radius Signal as shown above.

Due to a shortage of time, no specific technique was implemented to address the Projection Invariance requirement. We may find however that to a certain degree the algorithm will hold up to projection anyway, it is just a question of how much.

The feature vector for this algorithm finally, is made of the first N (currently 16) frequency components (excluding the DC), and will be used in training and classification, described later on in this chapter.

5.1.2 Shape Descriptors

Discussed in the Literature Review, Shape Descriptors are fundamentally different from the Fourier Descriptor approach in that they do not fully describe the shape. While a frequency representation of a shape can be used to recreate the original shape to a desired degree of accuracy, Shape Descriptors only capture small fragments of an object's shape. However by combining the individual features we obtain a relatively strong feature vector which although could not be used to recreate the shape, is quite effective at distinguishing it from others.

The features chosen for inclusion into the feature vector each have individual strengths, compared in a figure by Russ (1995) in which each metric is shown reacting differently to different shapes.

The descriptors chosen were;

$$Formfactor = \frac{4\pi.Area}{Perimeter^2} \tag{5.2}$$

$$Roundness = \frac{4.Area}{\pi.MaxDiameter^2} \tag{5.3}$$

$$AspectRatio = \frac{MaxDiameter}{MinDiameter} \tag{5.4}$$

$$Convexity = \frac{ConvexPerimeter}{Perimeter} \tag{5.5}$$

$$Solidity = \frac{Area}{ConvexArea} \tag{5.6}$$

$$Compactness = \frac{\sqrt{(\frac{4}{\pi})Area}}{MaxDiameter} \tag{5.7}$$

The reasoning behind the choice of descriptors was to maximise the number of descriptors while minimising the amount of measurements or computation that needed to occur. With effectively three extra feature measurements (we take advantage of already knowing the area of the blob) we can calculate six descriptors for a feature vector. Extra features may be added as desired, but early testing suggested that these six performed sufficiently well. Other possibilities were described by Russ (1995), such as including measurements of fibre length, which better represents elongated and twisted shapes, however as the competition targets are a circle and a cross we will concern ourselves with other such non-intertwining shapes, for which the chosen descriptors are better suited to describe than fibre measurements.

While the area found during the blob extraction phase was re-used, it was necessary to re-calculate the length of the perimeter. The problem here, as described by Russ (1995) is that the border is represented by pixels, where diagonal pixels cover a greater distance than non-diagonal ones. This would mean that as a shape of fixed size is rotated, the number of pixels in its border changes, making a significant difference to border length for shapes with long straight edges, like a cross for example. To solve this problem, the findContours function is run again, this time returning the border in Freeman's chain code (as described in the literature review). Summing the number of diagonal and non-diagonal members (even and odd codes) then allows the border length to be calculated in two operations as:

$$PerimeterLength = odd + \sqrt{2}\,(even) \tag{5.8}$$

The technique used to find the maximum and minimum diameter was also taken from Russ (1995). By rotating the shape 90 degrees at increments of our desired accuracy, the maximum caliper dimensions can easily be measured by a single run over the pixels (per rotation) to find the minimums and maximums in both x and y directions. This avoids the otherwise necessary exhaustive search of every border pixel combination to find these measurements, and decreases the algorithm complexity from $O(n^2)$ to $O(n)$.

5.1.3 Training and Classification

The implementations of both the Shape and Fourier descriptors use a similar training and classification scheme. Unfortunately the code could not be shared due to small differences in the required techniques.

Both techniques use an initial training period in which a video of 100-150 frames of pre-segmented video of each shape are fed to the training algorithm. For each frame, the trainer simply uses the 'getDescriptors' method common to each implementation to obtain the descriptors for this frame. Every frame's descriptors are added to the training set, which is finally used as the basis for an eigenmodel.

The difference in implementations is that the Fourier Descriptors create a high-dimensional space, in which 150 points may not form a significant cluster despite originating from the same object. In order to better analyse the clustering we compute an eigenmodel from the training data for all the shapes, for which the eigenvectors U represent axes of significant variation, and null space has been dropped. Taking the inverse of the eigenvectors U^{-1} allows us to project all our training data into this new space where there are no null dimensions. Now computing an eigenmodel for each shape in this significant space, clustering should be more apparent, and allows better models to be computed. The only consequence being that new feature vectors must also be projected into this new space before being classified. Classification is currently implemented as finding whichever eigenmodel the new feature vector has the lowest Mahalanobis distance to, of all the eigenmodels created at the training stage. This serves for the purposes of evaluation, but will need to be expanded upon for the competition.

The process for the Shape Descriptors is the same, but without need for this lower dimensional representation.

5.2 Gate Location

Of the three tasks defined in the requirements only one remains, the location of the validation gate. Like in any software development scenario, we must ask ourselves whether we can reuse what we have already created. The validation gate made for development footage was black, and it could potentially be black in the arena, although simply the fact that it will not emit light should rule out the beacon location technique to begin with. The reuse of the shape analysis code is certainly more feasible, the gate after all is just a hollow rectangle. A simple semantic net could encode the need to have a rectangle within a rectangle, but this immediately raises the question of how we recognise the inner rectangle, because it is the empty part of a greater shape it's contents could feasibly be any part of the background at all, but if we could recognise the background this well we wouldn't have needed the rarity measures previously discussed.

This is perhaps a small concern that could be overcome, such as by simply using the inner border of the shape if one exists. A better reason that the previous code should not be reused is that both Fourier Descriptors and Shape Descriptors would perform very poorly in the face of clutter and occlusion by other objects or the screen's edges. This is likely since the gate could be very thin, a very small amount

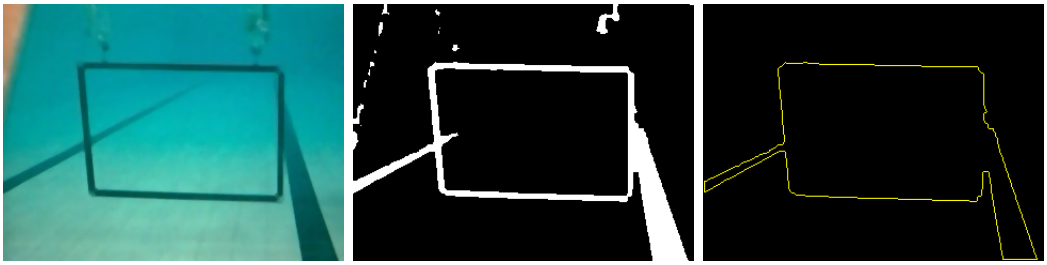


Figure 5.3: The size and 'transparency' of the gate means that re-using one of the techniques for describing shapes is inadvisable.

of noise or occlusion could break the line, altering the nature or appearance of the shape dramatically. Similarly, objects behind the gate may be visible through its centre doubling the possibility for this kind of distraction which is demonstrated in Figure 5.3.

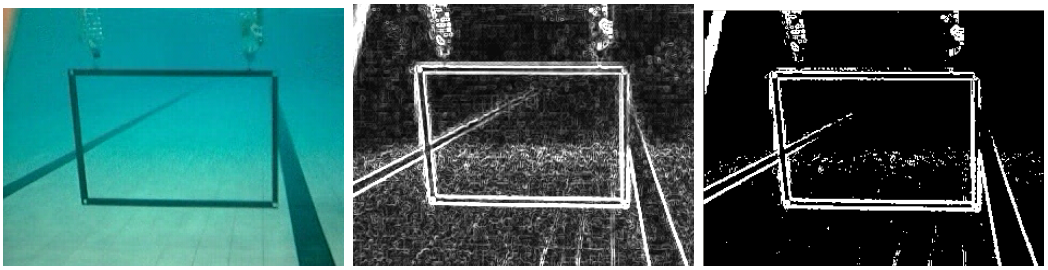
The most convincing argument for not reusing the previous techniques however, is that a vastly superior one is available. The Hough Transform is the perfect contender for finding lines in images, and is championed throughout Computer Vision literature, including many previous competition journals.

5.2.1 The Hough Transform

The mechanics of the Hough Transform are described with sufficient detail in the Literature Review (see section 2.5.2), so here we will discuss the details for its sucessful implementation in our domain. The literal implementation is dealt with by OpenCV, but the parameters for it's operation are left for application tuning, so here we simply discuss the consequences of its use on the implementation of the gate finding algorithm.

Initially we encountered problems with the first implication of using the Hough Transform, that it must operate on an edge image (ie, the image in which white pixels represent the edge of an object in the original image). Early assumptions that previous work with rarity-based thresholding was equally applicable to this task were quickly proved wrong. Firstly our black gate was being converted to a blue gate in the constant luminance HSV image required for a good rarity segmentation, assumed to be due to the lack of hue information in the black source pixels, or the slight amount of blue from backscatter in the water being normalised to be much brighter. Either way the effect was that the gate had no presence in the rarity image. Despite knowing the problems with using an RGB image for rarity based thresholding, that is what was attempted next.

Contrary to the expectation that the resulting edge image would contain only interesting edges, the image highlighted edges which were not even present in the original image. This in the most part was due to the rarity measure picking out areas of the pool floor etc, and is visible in images in the previous chapter. Solutions for this are possible; passing a first order sobel operator over the original image gives us an initial map of where the gradient is changing and where it is not. This, thresholded and binary ANDed with the rarity-thresholded image might only extract edges that were also rare, however this is a slight overkill for the task, and there is a simpler



(a) Original Image (b) Edge Detection (c) Thresholded Edge Image

Figure 5.4: Initial steps in the gate finding algorithms.

and faster solution. We find that the best image to pass to our edge detector is the basic RGB image with no rarity preference at all.

There is a question at this stage over whether or not to denoise the image prior to edge detection. The Hough Transform is well known for its resistance to noise and so in theory noise reduction is not completely necessary. In practice though the median filter is a cheap operation, and in images where the gate is small and a large amount of noise is present, the Hough Transform might well find as much evidence for lines in the noise as lines on the gate. As it is a fast operation to include we shall include it, since any extra lines generated by the Hough Transform at this point could lead to extra computations in the searching stage so we ought to try to limit it somehow.

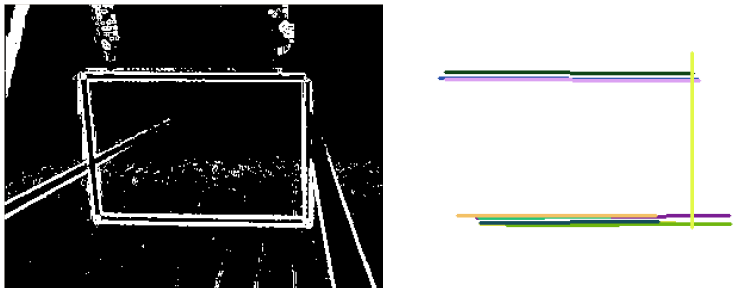
We comment at this point on the corner-cutting approach taken by Droher et al. (2005), who used only one colour channel in their edge generation for the AUVSI competition. This same tactic worked sufficiently on our test footage, however the speed gains are relatively small. Given the loss of robustness from making such assumptions, and imagining a scenario in which the background will not always be so predictably plain as in our test video, we decide to use all three colour channels in creating edges.

5.2.2 Filtering Candidate Lines

So far we have simply discussed on what basis the Hough Transform is working, in order to identify a gate in the image we must understand what the Hough Transform produces, and how this product is evaluated.

OpenCV allow three forms of the Hough Transform, the regular method which simply returns all mathematically defined lines in the image satisfying the accumulator threshold, a probabalistic method returning line segments rather than whole lines, as well as a multi-scale implementation. As our gate is made out of four line segments, and it will be important for us to know where each line begins and ends for quality analysis, it seems appropriate to opt for the second method. Thus used on an edge image, this function returns a list of all qualifying line segments in the image, in terms of their start and end points. This list is then used as the starting point in our search for the gate.

Since we know that we are about to begin some kind of search of these lines to find within them the optimal combination we will sort the lines into two sets. The set



(a) Thresholded Edge Image (b) Probabilistic Hough Transform Results

Figure 5.5: The production of lines from the edge image.

of horizontal lines in the image will be represented by $\mathcal{H}$, while $\mathcal{V}$ will contain the vertical lines. This division will increase the efficiency of our search algorithm as it will not waste iterations trying to make a gate from four horizontal lines. The only drawback with this optimisation is if the gate is rotated at some unpredictable angle, meaning it's lines do not fall neatly into these categories. In all reasonable circumstances however the validation gate will be rotationally predictable, to be specific the gate should always be level, while the algorithm should compensate for a certain degree of roll of the AUV. The criteria for these groups were left fairly open during development, between 0 and 45 degrees is horizontal, between 45 and 90 is vertical. It was considered however whether increasing the elitism of these groups would improve the search results; and could be further explored during the evaluation phase.

With these two set of lines, we may start our first search algorithm.

5.2.3 The Elementary Approach

The idea behind the elementary approach was to perform an exhaustive search over all possible combinations of gates within the lines from the two groups. The complexity of this algorithm would be highly undesirable if it needed to operate on large scale numbers, however the number of lines we receive should always be relatively low for our application, and it was hoped that adopting a 'gate building' strategy would allow many of these possible choices to be eliminated in one comparison, thus making the practical running time much more acceptable.

Gate building refers to the process of moving through the list of every horizontal line and attempting to build a gate around it with any of the vertical or remaining horizontal lines, where at each step the next line considered must meet some specified quality criteria, for which pseudo-code is given in Algorithm 1.

The survival criteria are based on having minimum distances between the parallel lines, and the suitability of the angle between them to form a rectangle. The quality score is produced by a set of quality functions which are the topic of the next section.

Problems were soon discovered with this method. As predicted the running time and speed is acceptable despite the apparent complexity, but the method struggles to construct any valid gates at all. This in fact is no fault of the algorithm, were there four decent gate-defining lines in the image it would certainly find them. The

Algorithm 1 The Elementary Algorithm. The lines are searched exhaustively, skipping those wich fail initial criteria. Gates surviving the criteria are scored until finally the best candidate is returned.

```
while H ≠ ∅ do
  h1 ← next(H)
  remainingH ← H/h1
  while remainingH ≠ ∅ do
    h2 ← next(remainingH)
    if criteria(h1, h2) > acceptable then
      while V ≠ ∅ do
        v1 ← next(V)
        if criteria(h1, h2, v1) > acceptable then
          remainingV ← V/v1
          while remainingV ≠ ∅ do
            v2 ← next(remainingV)
            if criteria(h1, h2, v1, v2) > acceptable then
              score ← quality(h1, h2, v1, v2)
              if score > bestScore then
                bestGate ← ((h1, h2, v1, v2), score)
              end if
            end if
          end while
        end if
      end while
    end if
  end while
end while
return bestGate
```

problem is that there are rarely enough good quality lines in the image to reconstruct the entire gate, visible even from a seemingly easy image in Figure 5.5. The lack of decent lines can be a product of many variables, the hiding of an edge by the camera's field of view limits, image noise or background preventing a decent edge from occuring over the length of the line by smoothing the gradient, or displacement of the edge due to gate occlusion. Given that so many things can disrupt this process, the aim of building an entire gate had to be reconsidered, the conclusion of which was to relax the strategy to need only three sides. This causes fundamental problems for the implemented algorithm though, as it becomes unclear what kind of structure we are trying to build. A quick fix could have been added so that whichever group contained the most lines would serve as the 'foundation stone' for the rest, however this seems neither intelligent nor robust and it is clear that another method not handicapped in such a way is desirable.

5.2.4 The RANSAC Approach

The attraction of the Random Sample Consensus approach (Fischler and Bolles, 1981) is that an exhaustive search of the data is unneccessary. The general method is to randomly pick candidates (uniform point sampling) from the data set and assess

Algorithm 2 The RANSAC implementation. Random lines are picked from the two sets and evaluated against a speedily evaluated set of criteria. Surviving gates are scored until an iteration limit N is reached, when the best gate is returned. The algorithm is run twice, once where $\mathcal{S} = \mathcal{H}$ and $\mathcal{P} = \mathcal{V}$, and then the opposite.

```
for i = 0 to N do
  a ← rand(S)
  x ← rand(P)
  y ← rand(P)
  if criteria(a, x, y) > acceptable then
    score ← quality(a, x, y)
    if score > bestScore then
      bestGate ← ((a, x, y), score)
    end if
  end if
end for
```

their suitability as a solution. This is repeated an appropriate number of times for the task at hand, with respect to the probability of a random point chosen belonging to the model. In general, the number of good quality points might be expected to reach a threshold before the solution can be declared found, however we prefer to simply take the underlying sampling method for its probability of selecting a good model from the data and forego this final accumulator variable. Our solution will simply be the best scoring group of selected points, where we understand points in the data set to be lines in the image.

This randomised approach is particularly suited to this problem as it performs *better* when assessing the minimal amount of data points, as it increases the probability that all points picked will be inliers to the correct model. Turning our need to search for only three edges to our advantage. Also the randomised approach allows us the freedom of not having to specify which gate edge might be missing, occluded, off screen etc, making it a far more flexible solution. We continue sorting the lines into two sets, however from now on we use the sets $\mathcal{S}$ and $\mathcal{P}$ to indicate which lines are being referred to; $\mathcal{S}$ is the set from which only a single has been taken, while $\mathcal{P}$ is the set of lines from which a pair have been taken. This abstraction simply allows us to talk about the algorithm in general terms.

Our RANSAC algorithm, described in Algorithm 2 is run twice, once searching for gates made from two horizontal lines, and then for gates made from two vertical lines. This is necessary since our quality functions must know which type of gate they are evaluating, whether the lines from $\mathcal{P}$ are from $\mathcal{V}$ or $\mathcal{H}$. This is because the measures we describe in the next section are not independent of the configuration of the edges as they might be for a square, aspect ratio for example. During implementation it was found that 300 random picks of each of the two combinations of line choice were perfectly adequate, but further testing may reveal acceptable results with far less.

5.2.5 Quality Metrics

Two kinds of metrics are described in the above algorithms; elimination critera, and quality metrics. The reasons for this separation are threefold. Firstly, with both

algorithms described we acknowledge that there may be multiple survivors of the elimination stage, if a choice must be made between them then the quality metric can be used as a way to select the best candidate. Secondly, the quality metrics generally are more advanced than used in the elimination stage as there are less gates to evaluate so the processing is more wisely spent. Crucially though, the elimination thresholds define an important range which can be used by the quality function, without which it would be impossible to give fixed range scores, as there would be no upper bound in most of the measurements. The theoretical advantages then are clear, however it must be said that the distinction is less obvious between the two stages in the code. Some parts of the quality metrics' calculations have to be worked out at the elimination stage, so you could also consider the eliminations as a method of terminating the quality measurement early because the lines do not form a gate.

This section discusses all metrics under the same banner, as they all serve the same end purpose. As a quick reminder; $\vec{a} \in \mathcal{S}$ while $\vec{x}, \vec{y} \in \mathcal{P}$.

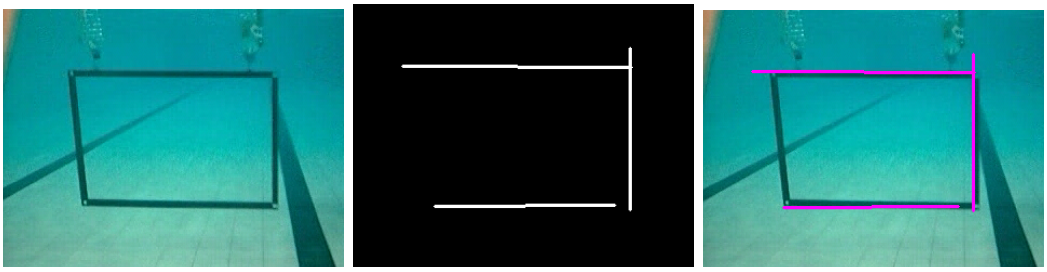
Vertical Link: $V(\vec{a}, \vec{x}, \vec{y})$ The vertical link check consists of creating a line between the mid-points of x and y, then taking the magnitude of its projection against a. Gates whose vertical links do not satisfy ($V(\vec{a}, \vec{x}, \vec{y}) >= \frac{1}{2}|\vec{a}|$) are eliminated. This metric does not take part in the quality score, partly because the maximum range is difficult to define, and partly because heuristic evaluations revealed it to be less useful than the other metrics.

Overlap: $O(\vec{a}, \vec{x}, \vec{y})$ The three lines used per gate allow the calculation of four 'intersection overlaps'. ($\vec{a}$ with $\vec{x}$), ($\vec{a}$ with $\vec{y}$), ($\vec{x}$ with $\vec{a}$), and ($\vec{y}$ with $\vec{a}$). An intersection overlap is defined as the amount by which one line 'overshoots' the end of another, at the point they intersect. Overlaps are defined in terms of a proportion of the intersected line's length. Overlaps greater than one fifth are not tolerated, and intersections within this fifth are given scores in linear proportion between 0 and 1. The four overlap scores are finally averaged to form the $Ol(\vec{a}, \vec{x}, \vec{y})$ measurement.

Parallelism: $P(\vec{x}, \vec{y})$ Being that $\vec{x}$ and $\vec{y}$ should represent opposite sides of the gate, they should also be almost parallel. This measure is not used to disqualify gates, in recognition that it might react badly to projected versions of the gate, however it is allowed to contribute to the score of a surviving gate.

Orthogonality: $Or(\vec{a}, \vec{x}, \vec{y})$ Similar to the parallelism check, $\vec{x}$ and $\vec{y}$ should be more or less at right angles to $\vec{a}$. This check may seem similar but adds an extra utility over the parallelism alone, as it allows the lines from $\mathcal{P}$ to be in a style of 'V' formation (potential effect of projection) and score badly in the parallelism round, but to score well here because they are both relatively orthogonal.

Length: $L(\vec{a}, \vec{x}, \vec{y})$ Length of lines has been used in previous underwater vision work to distinguish man-made objects from underwater clutter such as algae and other naturally forming articles. Its inclusion is of debateable worth here, as we are in a pool containing nothing but man made objects. It is thus included as a small fraction of the quality score, making it a gentle preference towards larger gates, and also used to disqualify very small gates for similar



(a) Original Image. (b) The winning gate with 91% confidence. (c) The result.

Figure 5.6: A summary of the gate finding process.

reasons as the disregarding of small blobs in earlier sections. The preference towards larger gates is not a violation of the scale invariance requirement, as it is weighted so that the stronger metrics make the majority of influence. This means it should only encourage longer segments of lines from the best scoring cluster of lines.

Aspect Ratio: $A(\vec{a}, \vec{x}, \vec{y})$ Arguably the most important metric, since the dimensions of our target gate are known scores can be given according to the dimensions of each potential gate. As before elimination criteria define the range, to which the score is proportional.

The final classification takes the form of a single class linear classifier; where the feature vector is made from the metrics just described, and for which the weighting coefficients were empirically analysed during development. In terms of Equation 2.2;

$$\vec{x} = [A(\vec{a}, \vec{x}, \vec{y}), Or(\vec{a}, \vec{x}, \vec{y}), Ol(\vec{a}, \vec{x}, \vec{y}), P(\vec{x}, \vec{y}), L(\vec{a}, \vec{x}, \vec{y})] \tag{5.9}$$

$$\vec{w} = [\frac{2}{6}, \frac{2}{6}, \frac{1}{6}, \frac{1}{6}, \epsilon], \tag{5.10}$$

$$Gate? = f(\vec{w} \cdot \vec{x}), \tag{5.11}$$

where f is a simple threshold.

In terms of implementation, the *Gate Quality* of the highest scoring gate is returned from each of the gate finding techniques to the main control loop, where the gate is accepted if its quality measure is above a fixed threshold and rejected otherwise.

There was slight uncertainty at the development stage over the consistency of the metrics implemented. While it is known that well defined gates get high scores, as shown in Figure 5.6, it is not known how well they will score rectangles of different degradations, such as projected gates or those with poorly fitting lines, or how gracefully the metrics will collapse under degradation. This is another dimension to be explored in the Evaluation chapter.

5.3 Some Refinements

5.3.1 Efficiency

In order to achieve the real-time speeds that were necessary for this project, a few refinements were made to speed the code's execution. Some common low level optimization tricks were implemented, such as taking repeated calculations outside the body of a loop where possible, and dealing without floating-point arithmetic wherever possible. The only serious decision however was to setup the list of blobs prior to a blobfinding run, as it represented a compromise. The allocation and deallocation of memory takes time, if it is not needed to be run inside the main loop then it shouldn't be. This decision though has consequences on the rest of the program. First, an initialisation function must always be called before the blobfinding function. Second, it prevents the possibility that two frame segmentations can be used concurrently within the program, as there is only one list of blobs. The first implication is less serious, however the second may mean that the code is less versatile, which is one of the project requirements. The vital processing time saved however outweighs this concern, especially as it is hard to imagine a situation in which two frames might need segmenting at once.

5.3.2 Accuracy

The implementation of the gate finder was, due to a lack of requests or progress from the AI designer, originally intended to pass the location of the center of the gate to AI, relative to the screen so that AI could steer through it with dead-reckoning. The centre location was calculated using a ratio of the detected edge lengths, and using it to scale a vector orthogonal from the center of the edge from $\mathcal{S}$. However after the majority of the program was developed, and as can almost be expected during such a communal effort project as the AUV, it emerged that the relative size of the vertical lines would be necessary for the more accurate navigation of the vehicle, and that vision would be required to provide locations of the four corners. This made the accuracy of the line segments an important priority, as well as the stability of the edges.

The stability is now more of a concern because the measured corner locations will be more susceptible to temporal process noise (the difference in produced lines from one frame to the next) than the centre estimation would have been. The gate centroid is relatively easy to estimate accurately due to the availability of three of its sides to use as a reference frame. The gate corners are a different matter, as we need to find all four corners with just three lines for information, meaning the location of two edges will have to be estimated from the aspect ratio. The difficulty lies in keeping the lines generated by the Probabilistic Hough Transform as accurate in length as possible, while also merging lines with breaks in - one of the main reasons for the use of the Hough Transform to begin with.

To obtain the stability of detected corners there are two things which can be done, the first is to maximise the quality of lines generated by the Hough Transform, and the second is to use kalman filters for each corner location, in exactly the same way as they are used for the beacon. While the implementation of the entire corner finding algorithm will have to be delayed until after this disseration is submitted, the

accuracy of the Hough Transform was improved by increasing the angle resolution and further restricting the distance OpenCV allows between two lines before joining them. This will mean a longer time spent processing each frame, but speed in this case must be a lower requirement than accuracy, otherwise the University of Bath's AUV will be the fastest entrant to fail the mission.

Chapter 6

Evaluation

Until this point, conclusions have been drawn mainly on the basis of limited sets of footage captured to aid developmental testing. While success has been encountered at each crucial incremental stage of development, it is important to subject these algorithms to a wider and more representative range of situations which the AUV is likely to encounter, so that the true strengths and weaknesses of each technique can be ascertained. The distinction between success and failure is not clear cut in computer vision, as any algorithm will fail given the right conditions (total darkness for example), therefore much of the testing tries to focus on finding the limits of each technique.

As well as focusing on the general aspects of evaluating any computer vision system, particular attention must be paid to the specific task at hand, as this project's aim is to evaluate algorithms for an underwater environment, in particular for an AUV. The main concern should be how the algorithms perform as we manipulate the variables which are relevant to the underwater domain, and the movement of the AUV. We also hope to use this stage to validate the non-functional requirements of the project, notably accuracy and performance. Most importantly for the reader, this chapter aims to provide closure on all the points of discussion that have been raised throughout this work.

6.1 Testing Aims

- To evaluate the aptitude which the implemented techniques demonstrate in the task domain.
- To study the degradation in performance of each algorithm when faced with increasingly challenging images.
- To compare and contrast the qualities and drawbacks of methods discussed in Chapters 4 and 5.
- To enable a confident conclusion of the best techniques to employ in the forthcoming SAUC-E challenge.

6.1.1 Environment Variables

In order to make this testing as rigourous as possible it was important to identify the free variables within the environment, and manipulate them independently such that their effects on the algorithms could be monitored.

These variables were identified as;

- Target Distance
- Target Size
- Target Shape
- Target Orientation
- Target Colour
- Image Resolution
- Light
- Water Clarity

Of these, two posed difficult problems. The clarity of the water could not be controlled directly due to the limitations of our testing resources. Its effect could have been simulated by manually introducing synthetic noise into the videos, however time for evaluation is limited and we choose to focus our attentions on the qualities of the methods discussed in previous chapters. The amount of light in the scene is also difficult to manipulate, and due to the desire to monitor the effect of the water itself it would not be scientific to use image editing software to synthesise the lighting change. Since all evaluation videos were gathered in the same pool session, these two variables will at least remain constant throughout the evaluation, making sound any conclusions based on controlling other variables.

Pictures of the shapes manufactured are given in section 3.6.1 along with the reasons for their choosing. These test shapes which we named B, A, E, SmallE, BigE, Star, Barnacle, Moon, Q, and O, took care of the variables of size and shape, while the target distance, orientation and image resolution were left to control during the test.

6.2 Test Plans

After early attempts to gather footage in the pool for incremental testing it became obvious that further such opportunities would be rare, and therefore that every effort should be made to maximise the gain from a single session. There was also a need to ensure that no environment variables went untested and that the experiment had been thoroughly thought out. To satisfy these needs a detailed experiment plan was created and is included as Appendix A.

Ground Truth	Classification							
	B	A	E	B.le	Star	Moon	Q	O
B	1	0	0	0	0	0	0	0
A	0	1	0	0	0	0	0	0
E	0	0	1	0	0	0	0	0
Barnacle	0	0	0	1	0	0	0	0
Star	0	0	0	0	1	0	0	0
Moon	0	0	0	0	0	0.969	0	0.0313
Q	0	0	0	0	0	0	1	0
O	0	0	0	0	0	0	0	1

Table 6.1: Benchmark Fourier Descriptor results: Fourier Descriptors used in an ideal environment. Each shape was classified approximately 30 times.

Ground Truth	Classification							
	B	A	E	B.cle	Star	Moon	Q	O
B	1	0	0	0	0	0	0	0
A	1	0	0	0	0	0	0	0
E	0	0	1	0	0	0	0	0
Barnacle	0	0	0	1	0	0	0	0
Star	0	0	0	0.456	0.544	0	0	0
Moon	0	0	0	0	0	1	0	0
Q	0.955	0	0	0	0	0	0	0.0455
O	0	0	0	0	0	0	0	1

Table 6.2: Benchmark Shape Descriptor results: As throughout these results, both sets of results were generated concurrently from the same Blob object to eliminate any unfairness occurring.

6.3 Shape Recognition

6.3.1 On The Bench

The main focus of this project, the recognition of shapes is considered first. The first evaluation took place out of water, in exactly the same environment as the training videos for each shape were filmed. Separate footage for which was taken for each shape, since we aim to test the recognition of shapes in as yet unseen images. These 'bench' tests will forge 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. It also allows us to validate that our algorithms are working as expected.

The results of the bench tests are shown in Tables 6.1 and 6.2, in which a stark difference can be seen between the two techniques. Generally the Fourier Descriptors performed more accurately and consistently, having a similar degree of success and confidence recognising each shape. The shape descriptors understandably being not as general a technique as the Fourier Descriptors display signs of favouritism between the shapes. Interestingly it outperforms the Fourier Descriptors on both the B and

Ground Truth	Classification							
	B	A	E	B.le	Star	Moon	Q	O
Barnacle	0	0	0	1	0	0	0	0
Star	0	0	0	0	1	0	0	0
Moon	0	0	0	0	0	0.875	0	0.125
Q	0.05	0	0	0	0	0.05	0.85	0.05
O	0	0	0	0	0	0	0	1

Table 6.3: Fourier Descriptors performing underwater at a distance of approximately 2 metres. The transition has had only a minor effect.

the Barnacle, recognising them with a significantly higher degree of accuracy. On the other hand however the A and the Q are receive significantly worse treatment, not receiving a single correct classification.

Unintuitively the majority of the Shape Descriptors' mis-classifications are attributed to the B, rather than the author's expected confusion between the O and Q. In order to try and explain this behaviour, we note that these three shapes are three of the four most convex shapes in the set, therefore reducing the power of two of the six features, and immediately restricting the dimensionality of the space in which these shapes may distinguish themselves from each other. At the same time we note that the Shape Descriptors perform best on shapes which have greater concave areas, such as the Barnacle and the E, another indication of the influence had by the choice of features.

Finally, attention is drawn to the high quality recognition of the Barnacle being common to both techniques, both giving consistently low Mahalanobis Distances to it, making it one of the key shapes of interest for further evaluation.

6.3.2 Underwater

Unfortunately, despite the extensive planning put into the experiment, a number of external factors meant that time in the pool was cut short. Also, due to manufacturing imperfections in the porthole of the AUV, many of the videos were corrupted. Badly affected frames were removed from the test footage, however these two problems resulted in the availability of only a fraction of the planned footage. Due to this setback, evaluation continues on a reduced set of shapes.

We now evaluate results obtained from the University's pool, with shapes at a slightly further but comparable distance to the benchmark tests. Static thresholding is used in combination with a target colour, as in the bench tests.

As can be seen in Table 6.3 the Fourier Descriptors have been visibly affected by the transition underwater, the mean Mahalanobis Distance of each shape from its training set has risen in every case, sometimes significantly, and small classification errors have been introduced. Overall degradation is obvious, however the majority of all classifications remain well placed.

The Shape Descriptors on the other hand have suffered dramatically from the change (see Table 6.4), the O being the only shape not to be affected while the rest of the classifier is rendered effectively useless. Interestingly, the Shape Descriptors outper-

Ground Truth	Classification							
	B	A	E	B.le	Star	Moon	Q	O
Barnacle	1	0	0	0	0	0	0	0
Star	1	0	0	0	0	0	0	0
Moon	0.917	0	0	0	0	0.0833	0	0
Q	0.05	0	0	0	0	0	0	0.95
O	0	0	0	0	0	0	0	1

Table 6.4: Shape Descriptors performing underwater at approximately 2 metres. The results display a significant deviation from the benchmark results.

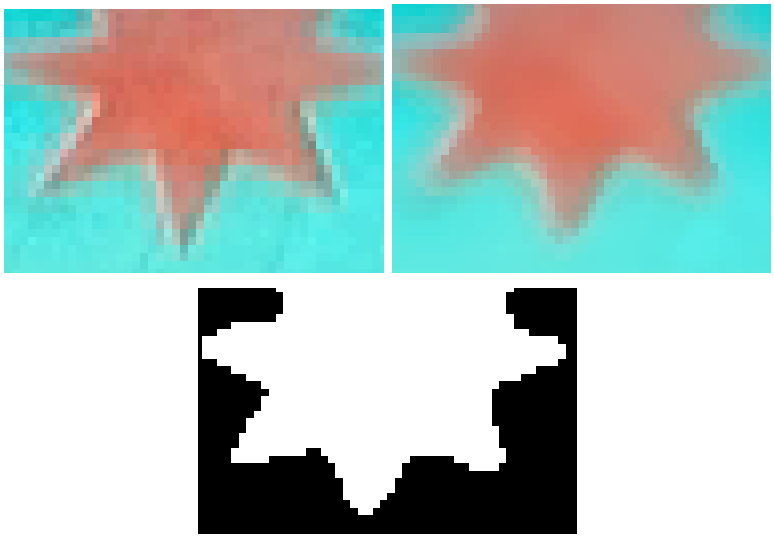


Figure 6.1: The sharp points and caves of the border have been rounded by median filtering.

form the Fourier Descriptors in classifying the O more accurately and consistently, while every other shape aside from the Moon goes without a single recognition.

These results were achieved using exactly the same algorithm parameters as used in the bench tests, with the exception of changing the sample colour to a hybrid underwater sample as described in Chapter 4, however when examining the Blobs that were being made available for the classifiers the reason for such a dramatic change is clear.

Looking at the mean Mahalanobis Distance for each correct classification, in particular the Barnacle, it is clear that the change has caused the descriptors to alter dramatically, meaning that whatever was previously distinguishing the Barnacle so well has been lost. As already noted, the Shape Descriptors appear to perform the best on shapes with large concave areas, and so the poor performance could be justified by Figure 6.1.

A little consideration reveals that the median filter, although generally noted for its quality in preserving borders, is inappropriate for preserving borders which are outnumbered by the background inside the filter kernel. This is obvious, since the median filter is using the median value of all those in the pixels, fringe values are removed.

Ground Truth	Classification							
	B	A	E	B.le	Star	Moon	Q	O
Barnacle	0.413	0	0	0.587	0	0	0	0
Star	0	0	0.0833	0.5	0.417	0	0	0
Moon	0	0	0	0	0	1	0	0
Q	1	0	0	0	0	0	0	0
O	0.2	0	0	0	0	0.4	0	0.4

Table 6.5: Shape Descriptors at close range with a 3x3 Median filter.

Ground Truth	Classification							
	B	A	E	B.le	Star	Moon	Q	O
Barnacle	0	0	0	1	0	0	0	0
Star	0	0	0	0	1	0	0	0
Moon	0	0	0	0	0	1	0	0
Q	0.05	0	0	0	0	0.05	0.8	0.1
O	0	0.2	0	0	0	0	0.2	0.6

Table 6.6: Fourier Descriptors operating on 3x3 Median filtered images of close shapes.

6.3.3 Noise Reduction

Before concluding that the Shape Descriptors have been badly chosen we attempt to improve their performance by reducing the size of the median filter to 3x3.

Table 6.5 shows a clear improvement over the previous results for Shape Descriptors, with the Moon being classified without fail and the rest of the shapes having taken steps towards returning to their benchmarks. We also note that an improvement has been made to Fourier Descriptor classifications, which have nearly returned to benchmark quality, shown in Table 6.6.

Having seen only an improvement by reducing the filter size, we continue to use no median filtering at all. Unfortunately this time there is no similar increase in performance. The Barnacle which was classified so well on the bench is still misclassified despite itself having a mean Mahalanobis Distance from its training set similar to that of the Fourier Descriptors the classification is being under-cut by other shapes, revealing our Shape Descriptors weakness in the handling of a variable not related to noise reduction.

Ground Truth	Classification							
	B	A	E	B.le	Star	Moon	Q	O
Barnacle	0.457	0	0	0.478	0.0652	0	0	0
Star	0.0909	0	0	0.455	0.455	0	0	0
Moon	0	0	0	0	0	1	0	0

Table 6.7: Shape Descriptors on close shapes with no median filtering.

Ground Truth	Classification							
	B	A	E	B.le	Star	Moon	Q	O
Barnacle	0.0588	0	0	0.353	0	0	0.176	0.412
Star	0	0	0	0.154	0	0	0.538	0.308
Moon	0	0	0	0	0	0.0714	0	0.929
Q	0	0	0	0.0645	0	0.0323	0.258	0.645
O	0	0	0	0	0	0	0	1

Table 6.8: Fourier Descriptors performing on shapes 4 metres away, with no median filtering and static thresholding.

As for the reasons for their failure, there are two candidates; projection and quantisation. After witnessing the effects that corruption of the sharpest points has on the Shape Descriptors, we can speculate that similar problems will be caused by quantisation when the shape moves further away, and less pixels are available to represent sharp features. Secondly, as discussed later in section 6.6, many of the videos contained shapes having undergone minor projections, which could be accountable for some of the poor performance.

These results allow us to conclude simple averaging an inappropriate noise filter, as it would have a similar effect of rounding off sharp points surrounded by another colour. Also the 'most conservative neighbour' approach of the Peak and Valley filter will cause exactly the same problem.

It seems then that the Shape Descriptors have already reached their limit, despite performing well on the Moon, the fragility they have already displayed means that they will unlikely be suited to this environment, which is full of possibilities for loss of detail. The difficulty they display in distinguishing between even fundamentally different shapes even on the bench is mostly a reflection of the Descriptors chosen, but highlights the difficulty in choosing appropriate measures, and means it is not a general purpose technique. The good results observed with recognition of the Moon however demonstrate that if used in the right circumstances this technique can be as powerful and reliable as our alternative technique, and suggest that used appropriately, a few shape descriptors could well come in useful for *verifying* the location of the cross once it is near.

6.3.4 Distance

To veryify our requirements of Distance Invariance, we move the shapes approximately 4 metres from the camera, and repeat the experiment, for which results can be seen in Table 6.8.

These results are a marked contrast to the quality previously achieved with Fourier Descriptors. In particular the Star has become unrecognisable to the algorithm. Knowing that the algorithm itself is capable of recognising a well defined Star, the previous stage must be performing ineffectively, which is shown in Figure 6.2.

Median filtering this time enhanced the borders slightly, at the cost of the rounding effect already discussed, and did make slight improvements to the classification of each shape however the improvements are not significant enough to show here. This

Figure 6.2: Colour bleeding across the edges of the star cause Static thresholding to miss its finer points, causing mis-classifications.

Ground Truth	Classification							
	B	A	E	B.le	Star	Moon	Q	O
Barnacle-NoMedian	0	0	0	1	0	0	0	0
Barnacle-Median	0	0	0	0.889	0	0	0.0556	0.0556
Star-NoMedian	0	0	0	0.846	0.0769	0	0.0769	0
Star-Median	0	0	0	0.538	0.231	0	0.231	0

Table 6.9: A comparison of Fourier Descriptor results using a 7x7 kernel adaptive threshold when median filtered and when not median filtered. This time the results are less conclusive.

problem however was expected, and as discussed in Chapter 4 adaptive thresholding offers a potential solution.

6.3.5 Adaptive Thresholding

We begin by assessing the utility of median filtering with our new thresholding technique. We do not perform as thorough an analysis as before, but wish not to simply assume that the conclusions hold for both thresholding techniques. Table 6.9 shows the two 'sharp' edged shapes which fared the worst under median filtering are not both adversely affected by it. For the sake of keeping this evaluation to a reasonable size we continue to use no median filtering as the rest of the variables change, as we primarily use the Barnacle for further testing as its recognition seems so far the most robust.

Realised while executing the evaluation, and visible in the results is that one of the assumptions made early on in the project was revealed to be not so valid. We

Ground Truth	Classification							
	B	A	E	B.le	Star	Moon	Q	O
Barnacle	0	0	0	1	0	0	0	0
Star	0	0	0	0.846	0.0769	0	0.0769	0
Moon	0.0159	0	0	0	0	0.524	0.0635	0.397
Q	0.0323	0	0	0.0968	0	0.161	0.355	0.355
O	0.265	0	0	0	0	0.0294	0.118	0.588

Table 6.10: Fourier Descriptor results using adaptive thresholding (7x7 kernel) from a target colour measurement, with a 3x3 median filter, at 4 metres.

Ground Truth	Classification							
	B	A	E	B.le	Star	Moon	Q	O
BarnacleClose	0	0	0	1	0	0	0	0
BarnacleMiddle	0	0	0	1	0	0	0	0
BarnacleFar	0	0	0	0.75	0	0	0.107	0.143

Table 6.11: Adaptive thresholding performing on the Barnacle at three Distances.

stated that the position of the centroid of the shape should not be of significance since it should be consistent for all the shapes. While it is true that it is consistent, the reason for the confusion between the Moon and the O at this stage is because their signals look very similar when the Moon is very slender, since observed from outside the Radius Signal will be almost indistinguishable from a shape which is symmetric, and alike to the moon on both sides, as is the O. In this case a fine grained curvature measure would have been vastly more appropriate, reflecting the huge change in gradient at the Moon's peaks versus the relatively constant curvature of the O. It is possible that using such a signal in conjunction with the low frequency Fourier Coefficients might have had an advantage over the radius signal, however this must now be left for future investigation.

Since the Barnacle has performed so well at 4 metres, we test it at 6 metres to see if it breaks down, and also at the closest distance (2 metres) in order to verify that the method is not simply suited to objects at these distances. Table 6.11 shows the Barnacle being correctly identified at all three test distances.

We also note here that adaptive thresholding offers an alternative advantage over static thresholding, as the 'hollowing out' of blobs means that one rare object existing inside another rare object can often avoid becoming part of the same blob, since the rarity measure changes at both their borders and is caught by the adaptive kernel.

6.3.6 Size

Three shapes were designed for this test (refer to section 3.6.1), unfortunately only the larger two of them made the test footage due to the problems already discussed.

As briefly mentioned before, the effects of quantisation could potentially have a large effect on the algorithms. To this end, we measure the quality of the Fourier Descriptors when faced with the same shape in two scales and at two distances; hoping to isolate the differences caused by changes in scale and changes in distance. As already shown, static thresholding is not capable of resolving shapes in the distance so the same parameters as so far used sucessfully for adaptive thresholding are continued but with one exception. The borders created by our previous 7x7 are sometimes very thin, and the border is broken in places, ruining the Radius Signal. The size of the adaptive kernel was increased to 11x11, in order to create thicker edges so that the best analysis could be made of the few frames available.

Table 6.12 shows that the size of the target makes a significant difference to our algorithms. In fact, during the creation of the bench test for the E, in which both BigE and SmallE were treated as the same for the purposes of the test, the BigE generally received higher distances from the training set, which consists purely of

Ground Truth	Classification							
	B	A	E	B.le	Star	Moon	Q	O
SmallEMiddle	0	0	0.357	0	0	0	0.143	0.5
BigEMiddle	0	0	0.813	0	0	0	0	0.188
SmallEFar	0	0	0	0	0	0	0.0909	0.909
BigEFar	0	0	0.629	0	0	0	0	0.371

Table 6.12: Adaptive thresholding performing on the two different sized E's at 4 and 6 metres.

the SmallE, and caused a high standard deviation for Mahalanobis distances for this shape. This is unfortunate as it means our algorithms are not as scale-invariant as was intended, and it means we are unable to draw decisive conclusions over this section, for which there is no more time for rooting out bugs and mistakes.

6.3.7 Rarity-Based Thresholding

Having performed so well with the target-colour based thresholding, the Barnacle is tested again here with respect to rarity-based thresholding. Having assumed that adaptive thresholding would again be the dominant choice, we were surprised to find that despite the 11x11 kernel, the borders being created were broken in every frame. This is because the rarity measure provides less of a contrast around our target colour than the measure of distance from the actual target colour. We revert then to static thresholding, noticing that the colour chosen always has a high rarity in the image and setting the threshold to only accept pixels with 99-100% rarity measurement.

The results of the rarity-based experiments are shown in Tables 6.13 and 6.14. We see that the two techniques are highly comparable at the four metre mark, both having difficulties with the same shapes, as well as offering similar rates of accurate classifications. The rarity-based model is performing consistently slightly less accurately than the target-colour based model, however since it operates with no knowledge of the target's colour this can certainly be viewed positively.

Ground Truth	Classification							
	B	A	E	B.le	Star	Moon	Q	O
BarnacleClose	0	0	0	1	0	0	0	0
BarnacleMiddle	0	0	0	0.833	0	0	0.0556	0.111
BarnacleFar	0	0	0	0.536	0	0	0.179	0.286

Table 6.13: Rarity-based static thresholding used on the barnacle.

6.3.8 Summary

The Shape Descriptors chosen had difficulties even distinguishing between our test shapes in the environment most similar to the training videos. This is a direct reflection on the descriptors chosen, as better performance is witnessed over shapes

Ground Truth	Classification							
	B	A	E	B.le	Star	Moon	Q	O
Barnacle	0	0	0	0.833	0	0	0.0556	0.111
Star	0	0	0	0.538	0	0	0.385	0.0769
Moon	0	0	0	0	0	0.5	0	0.5
Q	0.0323	0	0	0.0968	0	0.0323	0.323	0.516
O	0.265	0	0	0	0	0.0294	0.118	0.588

Table 6.14: Rarity-based static thresholding used on the 4 metre videos.

which have greater convex areas. Despite this initial setback, the Shape Descriptors then showed themselves to be particularly fragile under degradations of the border, not withstanding the slightest variations, and making them unattractive as a main option for the competition. Conversely, some aspects of the Shape Descriptors were encouraging, performing better on certain shapes than the Fourier Descriptors, and indicating that if well chosen measures were used then good results could be achieved.

The Fourier Descriptors performed well all-round after the transition underwater, yet were hampered by poorly defined input shapes from the segmentation stage as the distance to the shapes progressed, revealing static thresholding from a target colour to be another fragile technique. Both adaptive thresholding from a target colour and rarity-based static thresholding showed themselves to be solutions of significant promise, each effective over a good range of distance, and with only slightly more accurate results from the target-colour based model.

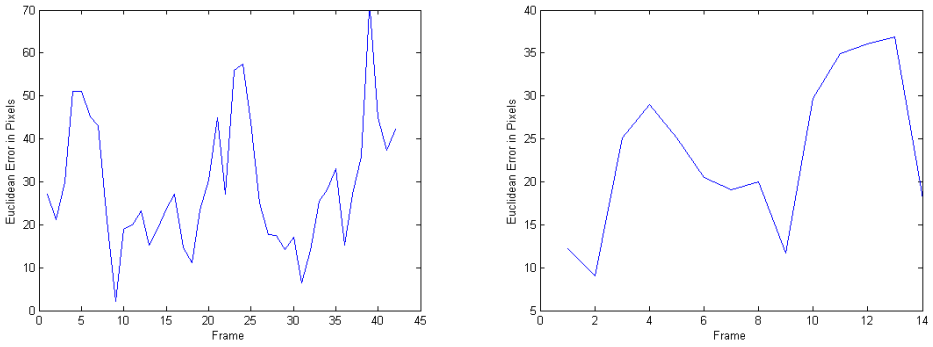
6.4 Gate Finding

As stated previously, the initial intention to provide AI with details of the centre of the gate was made redundant as the AI designer later requested the locations of the four corners of the gate to be exchanged. However this came too late in the process to be accommodated, and so for now we evaluate with respect to the original requirement of passing the centre of the gate, using this as a guide to how well the code can perform, and as an indicator of how well it will find the corners.

Again many of the videos taken for this purpose were quite corrupted by the imperfections of the AUV's camera window, however this is less of a problem in this scenario, as the corruption generally occurs over a single edge of the gate, meaning the three required for our algorithm are still available. Despite this, problems were caused by the OpenCV implementation of the Hough Transform, which seems temperamental regarding extraction of vertical lines (and is a supposedly fixed bug in the library). This will have to be resolved before the competition, however we attempt to evaluate the solution quality regardless.

6.4.1 The Effects of Distance

The gate was filmed at three distances, the center of the gate marked up by a human 'expert' in each frame, and the algorithms tested against this ground-truth, for which the measurement error is shown in Figure 6.3.



(a) Gate at Close Range (b) Gate at Medium Distance

Figure 6.3: The error between the algorithm's estimation of the gate centroid and the centre marked by hand in a sequence of images.

The distances used were the same as for the shape recognition evaluation, 2, 4, and 6 metres, from which nothing was found of the gate at the furthest distance hence its graph is not shown here. However this is the video in which the gate is most seriously corrupted by the AUV's dome, and so unfortunately no firm conclusions can be drawn as to any algorithm deficiencies at this range. It is quite feasible that the gate could be recognised at such a distance, more so than the shapes because the Hough Transform does not require detail to recognise a line, it simply needs enough evidence for its accumulator meaning that finding a line is simply a matter of lowering the accumulator threshold.

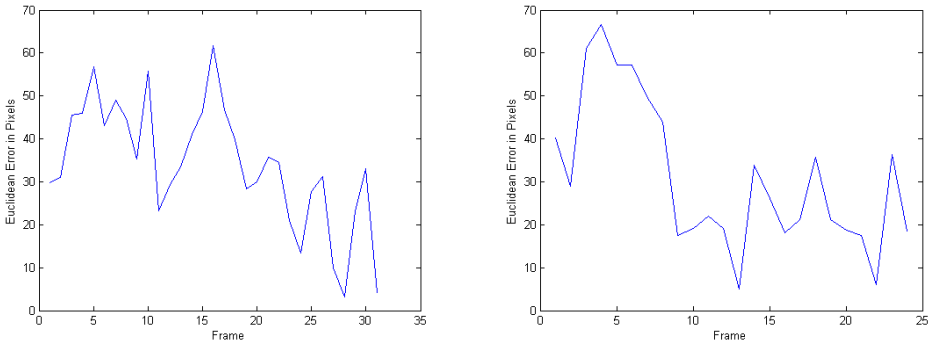
The nearest gate, although having the highest average error, had the highest success rate at finding the gate in the image, missing it in only one frame of 43. The middle distance however, despite having a lower average error only found 14 gates in 83 frames. In fact it is not quite fair to say that the error is lower, the distance to the gate has been doubled, greatly reducing the lengths of its sides and so the error has simply been scaled down, since the measurement deduced is relative to the size of the gate.

A number of extra videos were used to test the algorithm against the other environment variables of projection and rotation, however due to the lack of time in the pool, instead of the planned purpose-made videos they had to be cut from what video was available.

6.4.2 Rotation and Clutter

The cluttered video was handled particularly well, only one false-positive identification was made, however an ideal testing scenario would have involved the controlled addition of other straight edges in the pool behind and around the gate. Our clutter was of a lower grade and consists of students walking behind the gate, who have few straight lines on them.

Rotation was also handled without issue, as with the cluttered video the error made a similar degree of error when it *did* recognise the gate, and recognised it only ten times fewer than in the non-rotated video.



(a) Measurement error caused by gate rotation.

(b) Measurement error caused by background clutter.

Figure 6.4: The error between the algorithm's estimation of the gate centroid and the centre marked by hand in a sequence of images.

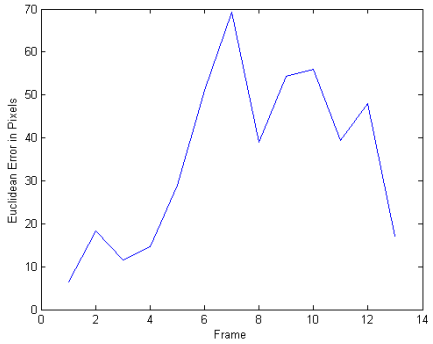


Figure 6.5: The error between the algorithm's estimation of the gate centroid and the centre marked by hand in a sequence of images in which the gate is at approximately 30 degrees to the AUV.

6.4.3 Projection

In general projection was handled well, although the lines were not fully defined and the algorithm generally preferred to pick straighter short lines than it did to use the longer but less parallel and orthogonal lines. This is caused by the weighting coefficients of the classifier and represents a balance between wanting to recognise projected gates, and wanting to reject spurious lines masquerading as projected gates. The aspect ratio in combination with a preference for line length will come in the most useful in finding a good balance for this situation, as a good aspect ratio might make the gate more acceptable despite poor orthogonality and parallel measurements.

These results confirm the correctness of the decision to calculate the center of the gate using the non-parallel edge as an authoritative source of information, since its validity is supported by the two lines stemming from it. Had our requirements remained only to report the centre of the gate this would have been a clear success. However since we must now report the corners of the gate, it becomes much more important to fit lines to the gate as closely as possible, and be careful not to estimate the corners based on lines which do not represent the projection, thus feeding false

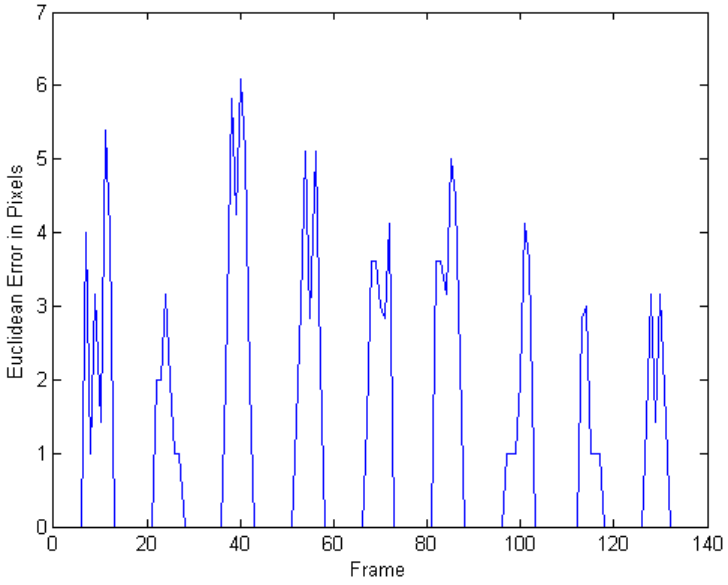


Figure 6.6: The error between the algorithm's determination of the location of the beacon's centre when compared to that of a human, in the lightest of the three videos.

information to the AI component.

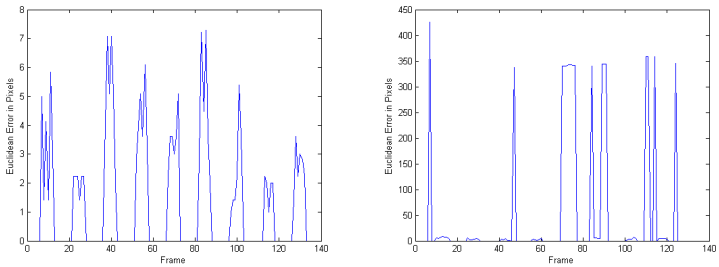
6.5 Beacon Finding

6.5.1 Lighting

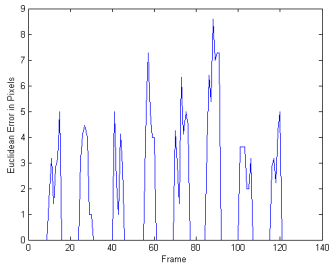
As the video of the beacon provided by DSTL features three levels of lighting, it seems prudent to test the location of the beacon as the lighting changes. The video was segmented into the three light levels, each of approximately 130 frames and each one was marked up by hand such that a matrix was obtained of ground truth beacon locations for that image. This was then compared to the marks made by the algorithm, without using the Kalman Filter, since it would be unfair on the human when there is no light by which to see the off-beacon. The Euclidean Distance (error) between the ground truth and algorithm results are shown in Figure 6.6, and are quite satisfactory. The AUV merely has to use the beacon as a guide to find the sunken target, so a maximum error of 7 pixels is more than adequate.

Also evident from the Figure is that the algorithm did not locate a beacon where there was none, nor did it fail to locate a beacon where there was one. Both human and algorithm marked location as `[0,0]` when the beacon was off, which are shown as the zero-distance errors. Any mistakes would have caused a large error.

The Dim video then caused problems, stemming from the problem identified in section 4.2.1; since the current number of erosion iterations is fixed at 10 the mask does not fit the beacon tightly enough, and the average intensity drops below the threshold. By increasing the erosion iterations to 20 and reducing the minumum area criteria of blobs from 300 to 100 we achieve good results in both Light and Dim



(a) Light (b) Dim



(c) Dark

Figure 6.7: The problems encountered with using a static number of erosions over different levels of light in the images, one number does not suit every light level and the beacon is at times missed.

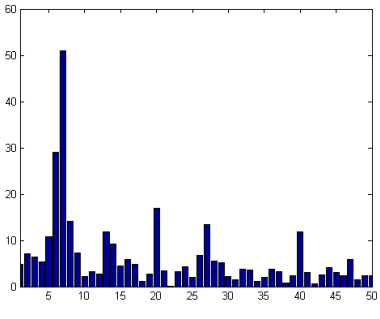
videos, with the blob mask focusing tightly around the centre of the beacon and yielding intensities in the area of 230-240, whereas the previous threshold necessary was 190.

Having expected to encounter the same problem again when moving to the darkest video, we found that the contrast enhancement performed automatically by the camera was filling the frame with artificial light, meaning that the aura was not as pronounced as in the dim video, and in fact that 20 erosions were excessive and causing the beacon to be missed. A middle ground was hoped to provide an answer, but Figure 6.7 shows that 16 iterations is once again not enough for the Dim video, and that one of the dynamic figures suggested in Chapter 4 will have to be used if illumination invariance is desired.

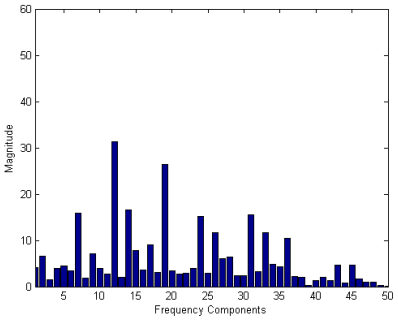
6.5.2 Frequency Identification

In order to assess the quality of our frequency identification, two extra videos were created from the video provided by DSTL. The frequency of the beacon was modified in each from ~1Hz to ~5Hz and ~0.6Hz. The DFT of a window from each beacon signal as measured by the AUV code is shown in Figure 6.8, and it is clear that the algorithm should be able to differentiate between them.

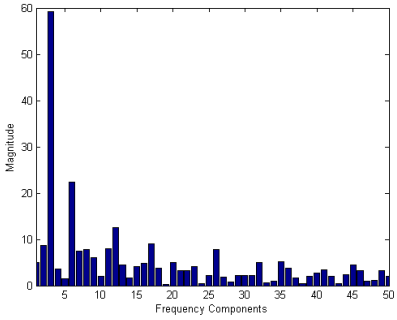
The results of beacon identification are shown in Table 6.15, where clearly the algorithm has not been affected by the two false beacons. We note however, that it has not performed particularly well on the correct beacon. As we have already ensured that the algorithm correctly identifies the beacon in individual frames, the problem must come from the signal identification. It was observed during the evaluation that



(a) Beacon Correct.avi: The competition beacon.



(b) Beacon Blip.avi: A short-on long-off beacon.



(c) Beacon LongOn.avi: A long-on short-off beacon

Figure 6.8: A snapshot of the three frequency spectrums of the beacon in testing video. The difference is pronounced enough to identify the correct video in this case, however the frequency components may change as the videos progress.

Table 6.15: The scores given to each run of the program. *The videos marked with an asterisk were scored relative to 0 while the correct video was scored relative to 1, since the target for the second two is to not make an identification.

Test Video	Correct Signal Classifications
Beacon Correct	62.6%
Beacon Blip*	100%
Beacon LongOn*	100%

despite the target frequency bin averaging 54 (a figure relative to the beacon's pixel intensity) across the whole video, it would spend short spells below our threshold of 46, therefore explaining the lack of identification. The threshold was empirically defined before the evaluations, and could be lowered, however it is already as low as comfort should allow. The higher frequency beacon generated a frequency spectrum with components around the 30 mark, and we would like to retain as much safety zone as possible with the threshold therefore alternative options are desirable. A Hann window could be used to rectify the spectral smearing taking place, which is the most likely cause of the error. Another option would be to alter the window size so that the measured frequency bins are shifted to lie more centrally to the discrete frequencies, minimising the smear.

6.6 Limitations

Although the utmost effort was made to ensure that the testing was carried out as rigourously as possible, it must be said that certain irregularities will have corrupted the results. Due mainly to the testing apparatus; fishing wire, foam boards and a hand-controlled AUV, many of the environment variables simply could not be controlled as well as a more detailed study might require, but hopefully this is understandable considering the limited time available for undergraduate dissertations.

Also, due to the original intention to gather an extensive quantity of test footage, and having limited disk space on the AUV, videos were captured in a compressed IYUV format. IYUV like any YUV scheme transforms the image from the original RGB-space representation into a new space. IYUV then subsamples the U and V dimensions of the image. This may have had an influence on the quality of videos available to the algorithms, particularly since much of the work was focused on colour itself. Unfortunately again, this is not something that could be controlled given the limited resources available to undergraduates.

Furthermore, our budget experiment apparatus unavoidably submitted every shape to projection in the testing videos. Which potentially explains the failure of the Shape Descriptors after the transition underwater.

With its limitations understood, this investigation should nevertheless provide a very good indication of the kind of results which could be expected if any of these methods were applied in the field, and certainly has shed some light on what will be necessary to perform well in the competition, and thus it has been fully worth its while.

6.7 Summary

Interestingly, although our Fourier Descriptors are more closely bound to the border representation, the Shape Descriptors suffer the most from border corruptions. The Fourier Descriptors' low fidelity representation of a high fidelity signal allows it to capture the fundamental shape, giving a tolerance to many imperfections where the signal of the border is attenuated, and ironically making it resistant to the highest frequency noisy information. It captures the essence of each shape, the six big peaks of a barnacle compared to the steadiness of a circle. Shape Descriptors on the

other hand, although based on region measures, supposedly more tolerant to border noise, put in an environment where the finer details of the border are 'rubbed away' actually struggle. This is understandabe, for example Russ' Aspect Ratio measure which measures only the longest caliper against the smallest, and remains ignorant of how many peaks there are. Once this longest caliper is rubbed away even slightly, the ratio can change dramatically, especially considering we use such similar shapes, and as we see the Shape Descriptors are easily confused.

Chapter 7

Conclusion

This project aimed to produce algorithms capable of aiding the University of Bath's upcoming SAUC-E entry, providing the eyes for its now named Autonomous Underwater Vehicle, Bathymysis. Aside from the specific tasks and their problems, it also aimed to evaluate techniques for general shape recognition in an underwater environment.

Some of the work was inspired by previous entrants to the US competition, in particular our rarity based thresholding bears resemblance to the rarity approach offered by the University of Florida (Dubel et al., 2005), whose apparent drawbacks allowed plenty of room for improvement. We have provided an enhancement to their approach, using a rarity measure for every frame while also accounting for the principal components of the scene's colour distribution, making a more general and versatile solution. Evaluation of the technique showed significant achievements, in particular we found it performing almost as well in our test cases as other algorithms which had the advantage of knowing the target colour in advance. While these pre-warned algorithms performed more accurately, it is suggested that the rarity measure developed is the true best candidate for autonomous application. Not stated in the evaluation is that the colour samples used contained one portion of underwater colour sample taken from the evaluation videos. Of course this then leads to a good recall of target pixels for all the evaluation videos, which were taken on the same occasion. As the scene changes however, perhaps the AUV moves into new, darker or less clear territory, such made-to-fit samples will break down for exactly the same reasons as we originally criticised Florida's technique. We conclude then that the ongoing rarity measure is the most autonomous algorithm in spirit, which only causes us a slight loss in precision, and that at the moment its primary weakness is its sub real-time performance.

Of the two techniques for shape representation chosen in the Literature Review, the Shape Descriptors showed more weakness in distinguishing between shapes than was expected, and showed more weakness under border degradation than was suggested by the literature, where in general region-based techniques are cited to be robust to border corruptions. The weakness in distinguishing shapes can be put down to the author's inexperience of the field, but as explained in Chapter 6, this does highlight the strengths and weaknesses of the technique in relation to the target environment. The Fourier Descriptors showed a good aptitude for the domain, however it was conluded during the Evaluation that the Radius Signal might not be as appropriate

as initially thought, due to its apparent confusion between non-similar borders, and something more closely bound to the changes in border direction might have resulted in less confusion. One of the main requirements of our techniques was that they were invariant to projection, and unfortunately problems during testing meant that this went unevaluated, however evaluation proved the requirements of scale and rotation invariance to be satisfied to a degree.

As for the more specific competition tasks, the Beacon finding and Gate localisation, both were implemented with a good level of success. The Evaluation showed that both techniques were capable of producing accurate localisation, however improvements were pointed out for each technique.

Hopefully one of the most useful outcomes of this project is the availability of a comprehensive resource for academics with very similar problems to solve, particularly those interested in segmentations of images affected by the weather or environment, or interested in the performance of Fourier Descriptors under border erosion. This dissertation provides a far more detailed and critical investigation than is found in any of the past competition papers, allowing the successes and failures described here to be built upon by others, or tailored for new domains.

7.1 Future Work

Although kept separate throughout this project for the purpose of comparison and investigation, there is no reason why the Fourier and Shape Descriptor techniques should not both be used for the competition. As found during evaluation the Shape Descriptors consistently performed well at least on the Moon, suggesting that if good measurements could be found for the circle and cross then a hybrid feature vector could provide a more robust solution than the Fourier Descriptors alone.

Furthermore, in order to better analyse the quality of the Shape Descriptors, more could be implemented, and the training sets created in the feature space could be subjected to principal component analysis in order to ascertain which of the descriptors are most powerful at discriminating between shapes likely to be found inside the competition environment, making the most effective use of our feature space.

Again, although the target-based and rarity-based models were kept seperate for the purposes of comparison and due to the shortage of time, there is potential for a hybrid object extraction algorithm, using both proximity to our target colour *and* distance from the perceived background colour (rarity) to distinguish targets from the background. Once again the two techniques should complement each other, leaving a doubly strong signal where rare target colours exist and weaker signals where regions are not rare *and* similar to the target colour.

7.1.1 Optimisations

Computing the Mahalanobis Distance of each pixel, from each target colour or rarity eigenmodel, is one of the most expensive calculations in the system. Calculating it once for every pixel makes it take the greatest portion of time in the processing of a single frame. This is especially significant in finding the beacon where a high

framerate is crucial to the success of the algorithm, for which at present the Mahalanobis Distance calculations are slowing to an unacceptable level, and which only succeeds since it is working on a video rather than sampling a scene in real time. Bruce et al. (2000) present a novel approach for fast segmentation of images based on predefined colour ranges. The technique uses an array per colour channel as a binary function of colour class membership, where the array/function ranges from 0 to 255. For instance, $R = [0, 0, 0, 0, .., 1, 1, 1, 1, ..., 1, 0, 0]$, $G = [0, 1, 1, 0, 0, 0, 0, ...]$ etc, creating a membership cube in the RGB space, and meaning that classification of a colour can be performed with three array lookups and two binary AND operations. This is not immediately applicable for our system since the best candidate for the segmentation of the scene was judged to be the frame-by-frame rarity based measure, and that eigenmodels form ellipsoids in 3d space not cubes. It is possible that these membership arrays could be created dynamically at run-time.

Firstly, for those applications that can take place with illumination invariance in HSV space, this is certainly possible. By calculating the frame's colour eigenmodel, the mean and principal components could then be used to draw a rectangle or more accurately an ellipse in H-S space. Colours external to this shape would then represent rare colours, and the binary membership arrays could be created based on colour space membership of this shape. Crucially, the principal components (and thus the ellipse) should be scaled relative to the rarity threshold (currently 99%), such that we simply need to use the membership arrays to classify each pixel.

The case for beacon recognition, requiring all three colour channels is not so straightforward. Having not had any time to investigate the possibilities, the author would like to speculate that an efficient technique could be found for the same purpose, such as creating an eigenmodel-bounding-cuboid in colour space from the eigenvectors and measuring all (or perhaps divided and conquered) pixels for intersection with this cuboid. Given this, or perhaps an intelligent 3d flood-fill algorithm, it is unknown whether or not this initial overhead would be so expensive as to be worth the subsequent gain, but with two months free time before the competition it is certainly something worth investigating.

Another possible technique might be to cache results from the Mahalanobis Distance calculation. As we have repeatedly stated, many of the pixels in the image will be of similar colour, and trading memory for speed is an oft used method of code optimisation. Conceivably, a 3D matrix representing colour space could be created along with the eigenmodel of the scene. Every time the Mahalanobis Distance is calculated for a colour this matrix could be updated with the result, potentially quantising the index in order to increase the recall rate. Subsequent pixels could poll this matrix to see if a result had already been calculated, skipping the floating point calculations. This would obviously only be more efficient if there were many similar pixels in the image. The median filter obviously would create an image with less variance between pixels, and a large enough kernel might do so enough to make this technique possible, but again further investigation must be undertaken to determine feasibility.

After all this, we may find that the simple solution is the obvious one, and to simply use intensity as a segmentation for the beacon will suffice.

During development many floating point calculations were used in order to keep accuracy, and to keep results predictable to aid in debugging. However floating point calculations are slow, particularly the use of square roots, and in many cases optimisations could take place, notably in the generation of our radius signal for Fourier analysis (see Equation 5.1.1), where there it is not even necessary to square root the squared summation - all that is required is consistency. In badly segmented images this could result in the saving of up to a thousand or so square root operations, just in this one function alone.

Many attempts at robustness were added to the code, such as macro definitions for accessing image pixels, rather than typing the full dereference each time, where bugs can easily creep in and ruin the program the day before the competition. However little time could be afforded to fully implementing this, and so a lot more time will be spent on improving the quality of the code after this dissertation is handed in. Both in making it more robust, sharing more common functions between the classifiers, as well as more error handling.

Currently the code uses binary training videos to know what each target looks like. The initial creation of these videos was time consuming, and the training at the beginning of each program execution is particularly slow and a hindrance to rapid development and debugging. The eventual representation of a trained object is simply an eigenmodel, which could be saved to a folder after the training video is used and then serve as a rapid shape recall.

In order to produce the evaluation results quickly, a 'learning mode' was created so that a user could inform the algorithm whether it was looking at the right blob or not, and to ouput statistics for it. This learning mode could easily be extended to produce the training eigenmodel directly and save it to a file. This will be invaluable in the competition where there will be little time to entertain with such worries.

The protocol implementation is still in the proof-of-concept stage, enough code was developed during the project so that its feasibility could be tested, firstly with a Dummy AI on the same host then across a network to the current AI implementation. However it has not yet been fully integrated with the rest of the algorithms, such that much of the code is currently in disjoint pieces. This will be one of the first things to receive attention when development starts again in June, followed by the implementation of a 'practice mode', in which the vision (and potentially the whole AUV) can be told to self-test and will run through a typical mission from the safety of the bench in the workshop. This will help iron out early software errors and reduce the spending on pool bookings, and mean that work on the software components can take place while the AUV is under mechanical development and is not pool-ready.

All our algorithms struggled fiercly with the video corruption created by the AUV's scratched and distorted dome, this will have to be changed for a clearer window before the robot is expected to function autonomously, as it is already hard enough to create the kind of all-purpose algorithms required for an autonomous vehicle without having to work around serious image degradation.

7.2 Critical Analysis

On a personal level two main errors were made during this project. Over the long christmas break code was implemented to walk the border of a Blob as fast as possible, however the time taken up working out its issues became too significant and the code had to be abandoned in favour of the OpenCV implementation. Other losses of time to low level coding problems, such as coding my own DFT, or difficulties caused by infamiliarity with the OpenCV library or the underlying maths meant that a large amount of time was spent programming and reduced the time available for investigating the more interesting high level aspects of this problem. Secondly, the author's background in Statistics is quite weak, as the basic modules taught at A-level were not a sufficient basis for understanding many of the more advanced techniques in the classification field. However throughout the project progress was made in this area, creating a stronger willingness and confidence to learn and providing a strong base of subject knowledge which can be used for further improving the algorithms implemented before the competition begins in August.

Overall though, the project experience has been a pleasurable one, and I am thoroughly looking forward to watching our AUV swimming through its first validation gate.

Bibliography

Robert C Altshuler, Joshua F Apgar, Jonathan S Edelson, David L Greenspan Debra E Horng, Alex Khripin, Ara N Knaian, Steven D Lovell, Seth O Newburg, Jordan J McRae, and Marvin B. Shieh. Orca-vii: An autonomous underwater vehicle. *AUVSI AUV Competition Archive*, 2004.

Joshua F Apgar. Orca-viii: An autonomous underwater vehicle. *AUVSI AUV Competition Archive*, 2005.

D H Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13:111–122, 1981.

R B Blackman and J W Turkey. Particular pairs of windows. *In The Measurement of Power Spectra from the Point of View of Communications Engineering*, pages 95–101, 1959.

Rives Borland, Eli Brown, James Buescher, Chris Caufield, Wei Min Chan, Shawn Chen, Chad Dize, Ben Evans, Marcelo Garza, Vikash Goel, David Hinkes, Kevin Kornegay, Jayant Kulkarni, Adrian Lai, Chris Lehman, Dennis Leung, Dominick Mancuso, Onimisi Ojeba, Sheng Wuey Ong, Jason Pagnotta, Ming Qiu, Neil Radia, Matt Robins, Karl Schulze, Alex Shih, Phillip Sieh, Bryan Silverthorn, Mike Stanish, Ryan Stenson, Ian Wang, and Sean Welch. Design and implementation of an autonomous underwater vehicle for the 2004 auvsi underwater competition. *AUVSI AUV Competition Archive*, 2004.

James Bruce, Tucker Balch, and Manuela Veloso. Fast and inexpensive color image segmentation for interactive robots. *International Conference on Intelligent Robots and Systems*, 3:2061–2066, 2000.

Daniel Bryant, Chris Crutchfield, Nick Gurtler, Chris LaFlash, Nick Rapp, and Karl Schulze. Intelligence by design: The development of an autonomous underwater vehicle. *AUVSI AUV Competition Archive*, 2001.

C K Chow and T Kaneko. Automated boundary detection of the left ventricle from cineangiograms. *Computers and Biomedical Research*, 5(4):338–410, 1972. ISSN 0010-4809.

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

Randal Droher, Carl Jantzen, Damian Ng, Dipayan Paul, Stewart Shearer, and Tim Soppet. Barracuda mark iv. *AUVSI AUV Competition Archive*, 2005.

William Dubel, James Greco, Aaron Chinault, Carlo Francis, Adam Barnett, Kevin Claycomb, Alan Melling, Eric M. Schwartz, and A. Antonio Arroyo. Subjugator 2005. *AUVSI AUV Competition Archive*, 2005.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience Publication, 2000.

M A Fischler and R C Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24:381–395, 1981.

Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, pages 179–188, 1936.

Ronald A Fisher. The statistical utilization of multiple measurements. *Annals of Eugenics*, pages 376–385, 1937.

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

David Forsyth, Joseph L. Mundy, Andrew Zisserman, Chris Coelho, Aaron Heller, and Charles Rothwell. Invariant descriptors for 3d object recognition and pose. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(10):971–991, 1991. ISSN 0162-8828. doi: http://dx.doi.org/10.1109/34.99233.

H. Freeman. On the encoding of arbitrary geometric configuration. *IRE Transactions on Electronic Computers, EC-10(2):260-268*, 1961.

M Heath, S Sarkar, T Sanocki, and K W Bowyer. A robust visual method for assessing the relative performance of edge-detection algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:1338–1359, 1997.

P V C Hough. *A Method and Means for Recognizing Complex Patterns*. U.S., Patent 3,069,654, 1962.

M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking, 1998. URL `citeseer.ist.psu.edu/isard98condensation.html`.

E Jacobsen and R Lyons. The sliding dft. *IEEE Signal Processing Magazine*, 20: 74–80, 2003. ISSN 1053-5888.

J. et al. Johnston. Duke university robotics teams autonomous underwater vehicle: Charybdis. *AUVSI AUV Competition Archive*, 2004.

Emil Kalman, Rudolph. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D): 35–45, 1960.

M Kass, A Witkin, and D Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1:321–331, 1988.

Hannu Kauppinen, Tapio Seppnen, and Matti Pietikinen. An experimental comparison of autoregressive and fourier-based descriptors in 2d shape classification. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(2):201–207, 1995. ISSN 0162-8828. doi: http://dx.doi.org/10.1109/34.368168.

Volodymyr Kindratenko. *Development and Application of Image Analysis Techniques for Identification and Classification of Microscopic Particles*. PhD thesis, University of Antwerp, 1997.

A Krzyzak, S Y Leung, and C Y Suen. Reconstruction of two-dimensional patterns from fourier descriptors. *Machine Vision and Applications*, 2:123–140, 1989.

X Li and Z Zhu. Group direction difference chain codes for the representation of the border. In *Digital and Optical Shape Representation and Pattern Recognition, SPIE Proceedings Vol. 938. Edited by Richard D. Juday. Bellingham, WA: Society for Photo-Optical Instrumentation Engineers, 1988., p.372*, pages 372–+, January 1988.

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

Tom Mathes and Justus H Piater. Robust non-rigid object tracking using point distribution models. In *Electronic Proceedings of The 16th British Machine Vision Conference*, 2005.

Mark Nixon and Alberto Aguado. *Feature Extraction and Image Processing*. Newnes, 2002.

Adriana Olmos and Emanuele Trucco. Detecting man-made objects in unconstrained subsea videos. In *Electronic Proceedings of The 13th British Machine Vision Conference*, 2002.

P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(7):629–639, 1990. ISSN 0162-8828. doi: http://dx.doi.org/10.1109/34.56205.

Anthony R Porteus and John P Collomosse. An investigation into the use of genetic algorithms for the automated solution of jigsaw puzzles, 2005.

Lawrence G. Roberts. *Machine perception of three-dimensional solids*. PhD thesis, Massachusetts Institute of Technology Lincoln Laboratory, 1963.

John C Russ. *The Image Processing Handbook (2nd ed.)*. CRC Press, Inc., Boca Raton, FL, USA, 1995. ISBN 0-8493-2516-1.

Irwin Edward Sobel. *Camera models and machine perception*. PhD thesis, Stanford University Artifical Intelligence Lab, 1970.

Ian Sommerville. *Software Engineering*. International Computer Sciences Series. Addison-Wesley, Harlow, UK, 6th edition, 2001.

Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing: Analysis and Machine Vision*. O'Reilly, 1999. SON m 99:1 1.Ex.

Robert Charles Warren. *Detection of Distant Airborne Targets in Cluttered Backgrounds in Infrared Image Sequences*. PhD thesis, University of South Australia, 2002.

P S Windyga. Fast impulsive noise removal. *IEEE Transactions on Image Processing*, 10:173–179, 2001.

Dengsheng Zhang and Guojun Lu. Content-based shape retrieval using different shape descriptors: A comparative study,. In *IEEE International Conference on Multimedia and Expo (ICME'01)*, page 289, 2001.

Appendix A

Experiment Plan

A.1 Gathering Training Data

- Training footage will be acquired out of the pool. As AUVs in general are autonomous, it is reasonable to expect that training might occur on footage taken out of water.
- Training footage will consist of each shape being held against a sufficiently contrasting background at a distance of approximately two metres, such that as little noise as possible is present in the training data.
- Each training footage video will contain a single shape and will be at least 100 frames long.
- Final training videos will be a sequence of binary images in which the shape is the only white component present, and will be produced directly from the training footage videos.

A.2 Gathering Testing Data

A.2.1 Distance, Colour, Shape, Size

- All shapes are placed side by side in the pool including the gate in front of a clear background, preferably open water. The shapes should be 2 metres from the AUV.
- The AUV is passed horizontally across the line of objects.
- The AUV will rest on each shape for approximately a minute before moving on.
- While focusing on one shape, the AUV will be held steady.
- This procedure is repeated with all shapes turned round to show the second test colour, and then repeated at 4 metres and 6 metres for a single colour.

A.2.2 Orientation

- All shapes are placed side by side in the pool, against the pool wall or in front of a clear background, including the gate.
- The AUV is passed horizontally across the line of objects.
- The AUV will rest on each shape for approximately a minute before moving on.
- While focusing on one shape, the AUV will be rotated clockwise and then counter-clockwise by approximately 45 degrees.

A.2.3 Projection

- All shapes are placed side by side in the pool, against the pool wall, including the gate.
- The AUV is passed horizontally across the line of objects.
- The AUV will rest on each shape for maybe a minute before moving on.
- While focusing on one shape, the AUV will undergo a clockwise movement whilst pointing at the shape, effectively making a cone with it.

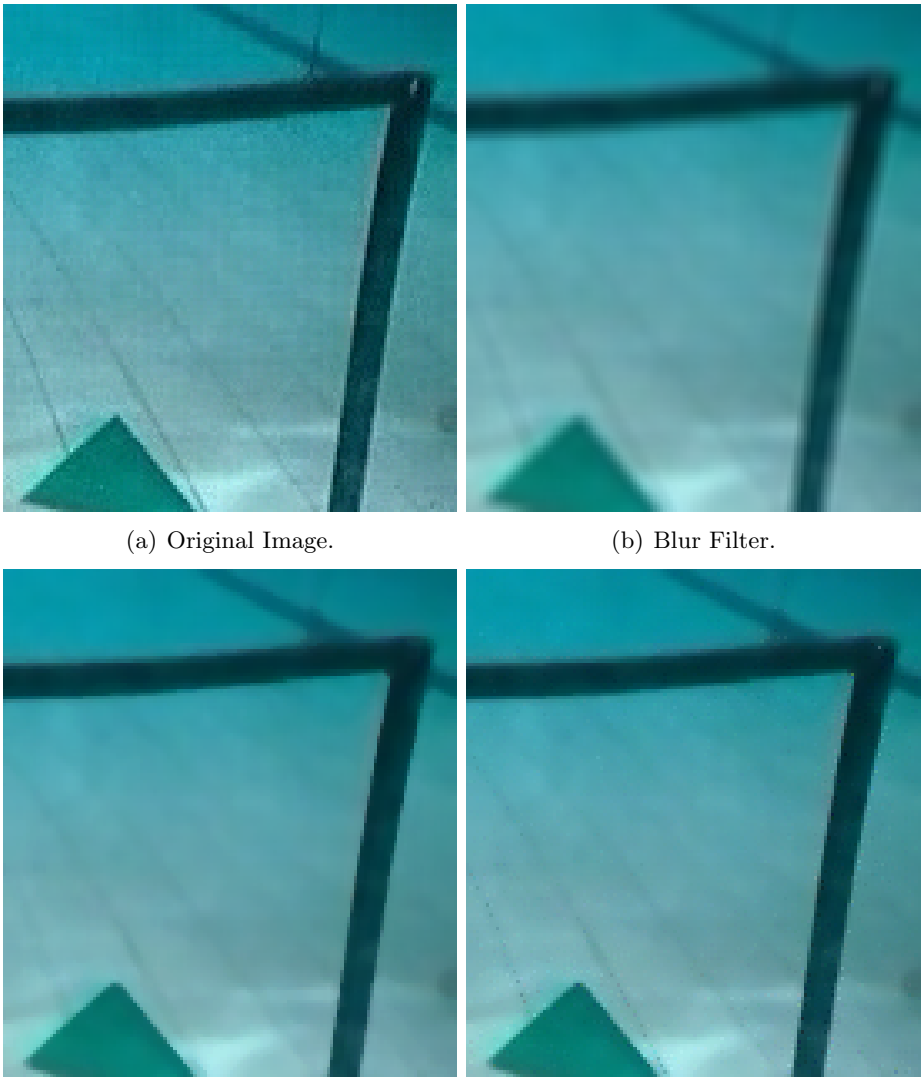
A.2.4 Image Resolution

- One of the more problematic shapes is selected to repeat the first scenario.
- Vidoes are taken at double resolution.

Appendix B

De-Noising Filters

Three noise reducers were used during the project, their effect on a magnified portion of a sample image is shown here for illustration purposes.



(a) Original Image. (b) Blur Filter.

(c) Median Filter. (d) Peak and Valley Filter.

Figure B.1: The results of the three noise reduction techniques chosen during the Literature Review.

Appendix C

Included Disc

The CD accompanying this dissertation includes this document, and many of the materials used in creating it. Full listings of statistics produced for evaluation are available, as well as the evaluation videos used. The current version of the code is also made available in full on the CD. Perhaps most interesting for the reader, there are videos of the algorithms functioning on the gate, beacon, and test shapes.