Vessel Classification from Periscope Data

Quenten Thomas

School of Mechanical Engineering

CEED Partner: Raytheon Australia Pty Ltd

Abstract

In this paper we present a system for automatically recognising ships from periscope image data. The system is designed to be robust to noise as well as incomplete data in both the ship database and the image, while using features that may be extracted from an image reliably and relatively easily. The proposed system makes use of two heuristics involving vertical features (such as masts), and binary moment tables. A random ship generator has been developed to provide data on which to test the system, although tests of the system are not yet complete. We explain in detail our evaluation methodology and argue that it may provide a convincing proof-of-concept once tests are complete.

1.0 Introduction

There can be no argument that timely and reliable vessel classification skills are important on a naval submarine. For human operators, vessel classification from periscope images is a skill that develops with experience and some experienced submariners are adept at vessel classification. However a risk to the submarine lies in the loss of these personnel, or simply in mistaken or time-consuming identification. Raytheon Australia has commissioned this CEED project in order to develop a system that may assist Navy personnel with vessel classification.

A vessel classification system must involve a search for ship features in a database. (A 'feature' is any information about a ship that can help distinguish it from other ships.) The functional block diagram for a general vessel classification system is shown in Figure 1.

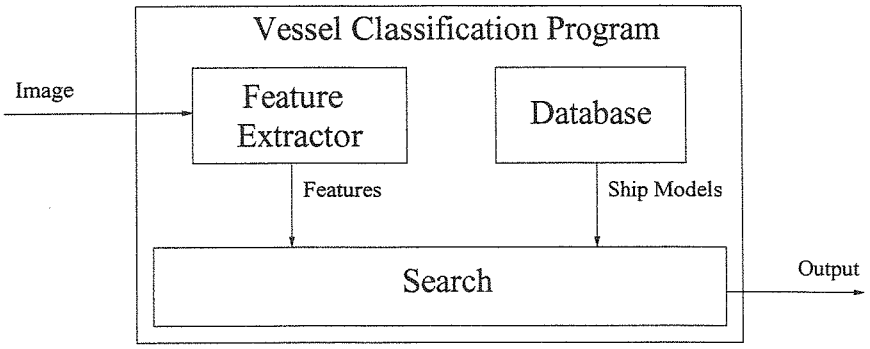

Figure 1. The components of a general vessel classification system.

As Figure 1 shows, the search algorithm, the database and the feature extraction components are related. We cannot design one without considering its interactions with the others. For example, there is no point designing the search to use features that cannot be reliably extracted.

In this project we have focussed on developing the search mechanism. While the implementation of the other components is beyond the scope of this project, we are still required to carefully consider of how the feature extractor and the database may be implemented in future – specifically, which features we can expect to be able to use. Our search must make use of those features that are both extractable and obtainable. That is, we must be able to extract the features from a ship image, and we must be able to obtain the data with which to build our database.

Therefore, our system is based on the assumptions that we can extract the features we wish to use, and that the information required to construct a database of these features is available (if not to us, then to the Navy who would ultimately implement the database.) This paper aims to show that if these two assumptions are correct, then our planned experiments will be able to verify the level of success of our approach.

In Section 2 we identify a number of key requirements that shaped the design of our vessel classification system. Throughout the rest of the paper, we will refer back to this list of requirements and explain how the proposed system has been designed to meet them.

In Section 3, we introduce the idea of rotating 3D models of the ships to determine whether there exists a view of the model that matches the ship image. Searching for a matching rotation is searching over the rotational variable space. There are many ways this search could be implemented. We consider the worst case, brute-force search. We go on to explain how this simplistic search will need to search over several dimensions, and is unlikely to succeed in returning a result quickly. In Section 4, we introduce a vertical feature heuristic, and explain why it is likely to significantly improve performance over the 'brute-force' approach. In Section 5 we examine a second heuristic that involves matching an image's binary moment vector to a model in the database.

At some point, the system will need to be tested, and so we require some data to test. That is, we need models of ships for the database, and test images to try and match to them. However, the data required to construct the models is not easy to aquire. Therefore, we have developed an automatic ship generator that can produce all the necessary data required to test our system. This is outlined in Section 6. In Sections 7 and 8 we explain how the project will proceed, and then offer some concluding remarks.

2.0 System Requirements

Table 1 shows a list of system requirements that were identified, which we refer to throughout the paper, to objectively assess the suitability of our approach.

	Requirement	Comment
1	Few False Positives (High Specificity)	It is better that the system select one or two ships as potential candidates, than a large group, so the system should aim not to match an incorrect ship.
2	Few False Negatives	If the ship in the periscope image exists in the database, the system should match the correct ship, from any viewpoint and under any conditions.
3	Robust to noise	System should be robust to noise and degrade gracefully with increasing noise in both the image and the ship model.
4	Robust to incomplete information	The system should be robust to incomplete information in both the image and in the model.
5	Timely Results	Submariners need to be able to identify a ship within seconds, so the system should run in real or close-to-real time.
6	Easily Obtained Ship Data and Model	The required ship data should be easily obtained, and the database entry (the ship model) easily constructed from this data.
7	Easy and Reliable Feature Extraction	The features that we match to the database model must be easily and reliably extracted from a periscope image.
9	System Testability	There must be some way to test the system without requiring detailed Naval data.
10	Extensibility	Ideally, the system will be extensible, allowing for inclusion of extra information or other feature extraction techniques.

Table 1. System Requirements

3.0 Matching via Simple 3D Ship Models

Let us consider the range of periscope images that a submariner may expect to encounter. Certainly, there will be many types of weather and sea conditions, various lighting conditions and of course the images will feature many different ships to be identified. However, another difficulty a submariner will face is that the ship will be facing in an arbitrary direction, so there is no single reference image that defines how a particular ship will appear. Consequently, there is no single view of a ship that can adequately represent the ship in a vessel classification system. It is possible that a number of images from viewpoints around the vessel will contain enough information for identification from any angle, and together could be used as a model of the ship (Lowe 2001). However, these models are sensitive to lighting conditions and so lack robustness, which we require.

The alternative that we have adopted is to represent ships using a simple three dimensional model, which may then be rotated to determine if there exists a view that matches the image.

This allows us to identify ships from any viewpoint, which should improve the false negative rate.

3.1 The 3D Model

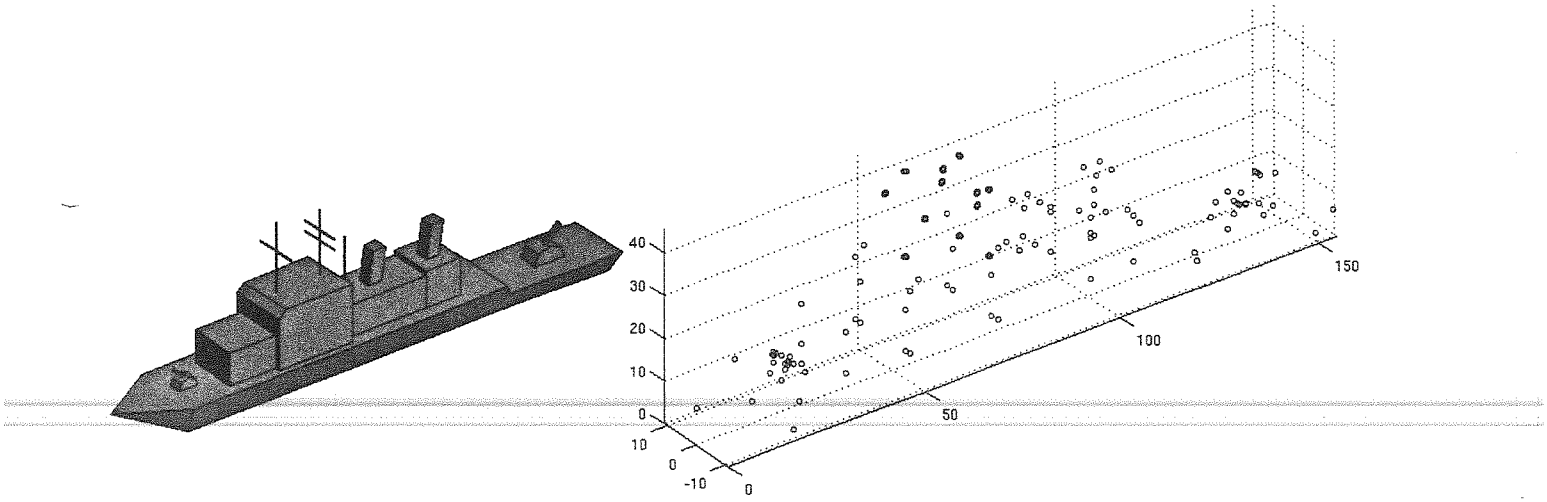

Figure 2. A model of a ship and it's associated corner points.

The 3D model simply consists of a set of 3D coordinates that represent features. In our system the basic features that are used are a ship's 'corner points' – those points at which a physical vertex occurs, as shown in Figure 2. The assumption behind using a ship's vertex points is that around these points in the image there will be high intensity gradients (colour changes). This is a reasonable assumption, as evidenced by Figure 2. There is a well-known Computer Vision technique for finding these points of high intensity gradient in images, called a corner detector (Harris and Stevens 1988). We hope to be able to match the corner points extracted from the image to the 3D corner points of the ship model.

There are good reasons for choosing corner points to construct our ship models. Firstly, the 3D corner points required to make a model of a ship are quite easy to obtain. Sources of the 3D corner points may include blueprints of ships, silhouettes, other 3D models, and even arbitrary photographs. Secondly, the 2D corner points are relatively easily and reliably extracted from images. (By 'reliable' we mean that if a particular physical corner point is extracted in one image, it is also likely to be found in another, different image, which also contains the point.) Finally, corner points are generally quite plentiful in images, and can be ranked according to prominence. Even those images in which the majority of the ship is occluded may contain enough corner points to facilitate a match. Therefore, the use of corner points satisfies Requirements 6 and 7.

The use of a 3D model also provides robustness to noise and incomplete information. Furthermore, it may well eliminate the false positive rate, and ensure high specificity. To show why this is the case, we consider what happens when a 3D model is rotated close to the matching viewpoint of a matching image.

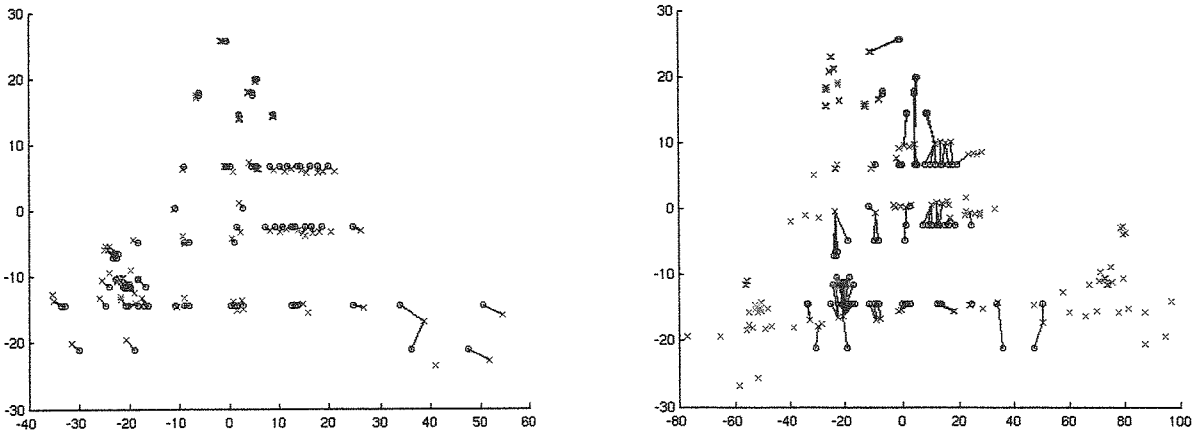

Figure 3. To calculate the distance measure, we match each image point (circles) to its closest model point (crosses), then sum up the distances between these matched points. Solid lines indicate matched points and the distance between them. On the left, the image matches closely to the model, unlike on the right.

Let us define a simple measure of the 'distance' of a particular model viewpoint to an image: it is the sum of the distances between each corner point in the image and its nearest point in the model for the viewpoint chosen, as illustraded in Figure 3. If the rotated model and the image line up exactly, the distance will be zero; the model points will correspond to matching image points. For each corner point in the image, the nearest point from the model is on top of it. For those viewpoints that come close to matching, the distance will still be small. However, as the viewpoint moves away to an arbitrary view, the summed nearest distances will no longer correspond to matching points, and will be more arbitrary. This sum of arbitrary values is extremely unlikely to reduce to levels as low as when the points actually correspond.

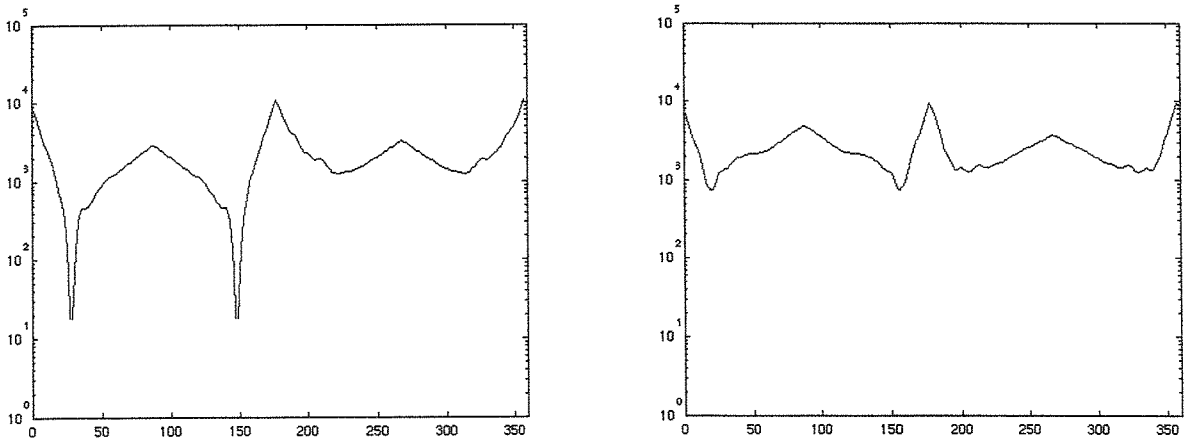

Figure 4. Distance measure plotted against vertical rotation. (a) The lefthand plot shows two zero points because the model used matched the image. (b) The plot on the right illustrates the behaviour of the distance metric in the case where the model does not match the image. In both these plots, the image was taken from a viewpoint 30 degrees about the vertical.

Figure 4 shows plots of how the distance metric for two models changes as they are rotated about their vertical axes. The two models were different, but tested against the same image. The model used to generate the lefthand plot was the correct model, while an arbitrary model was chosen to generate the righthand plot. In Figure 4a, where the model matches the image, we see that the correct viewpoint is either 30 or 150 degrees about the vertical (these are essentially the

same view from a distance), while in Figure 4b, the model is not a match at any viewpoint. In Figure 4a, we see that the distance drops sharply around the correct viewpoints to low values (note the logarithmic scale). In both plots, where the view is incorrect, the distance metric is the sum of more arbitrary numbers and hence, high distance measures are observed. From this we see that there is not much difference between a non-matching viewpoint of a matching ship model, and a non-matching viewpoint of a non-matching model. The only case in which the sum will be relatively low is where both the viewpoint *and* the ship model match. Therefore, we have a near binary function to test for matches; typically, the distance metric is high, but in the rare cases when the model matches and the view is close to correct, the points begin to correspond and the distance metric drops sharply indicating a match that we may be extremely confident in (as seen in Figure 4a). Therefore, the system can be expected to exhibit high specificity and very few false positives.

Should there exist noise in the position of 3D points, or image points, we may still expect corresponding points to be matched when the view is correct, or close to correct. When the corresponding points are matched, the distance metric is still expected to be low relative to viewpoints that do not match as well.

Robustness to incomplete information is due to the fact that the search does not crucially depend on any one piece of data. It simply works with the information available.

3.2 Searching Over 3D Models

Rotating the 3D model amounts to a search over a three dimensional space; with axes for yaw, pitch and roll of the ship.

So far, we have implicitly assumed that we are viewing the model at the appropriate magnification, and that we can easily move the superimposed image about on top of the other until the points match up. However, finding the correct magnification, and the horizontal and vertical positions for a match is actually a search over three variables. In general, these are three further dimensions to search over, increasing the total number of search dimensions to six.

A simple 'brute-force' approach to the search will require seven nested loops (six variables for each ship in the database), at the center of which, the distance metric must be computed, which is fairly expensive. (The distance metric has complexity of $O(nm)$ where n and m are the number of points in the image and the model.) The result is that while the brute-force search will almost certainly obtain the correct answer (we can make the step size arbitrarily small at the expense of extra computing time), preliminary tests indicate that it will take far too long – not the close-to-real time performance that we wished to achieve as per Requirement 5.

3.3 Improving Search Performance

Three approaches to improve search performance are: 1) somehow match corresponding points in the image to points in the model, 2) guess the viewpoint, and 3) quickly eliminiate some incorrect ship models before searching.

If we can determine six (or preferably more) likely 'model point – image point' correspondences, then it is possible to compute the transform that best projects the model onto the image points (Faugeras 1993). This transform is like a guess of the viewpoint. If the guess is a good one, then we would expect many corresponding points to have been matched, and we would observe a relatively low distance metric, and be confident of a match. So if we guess our

six (or more) likely corresponding point matches correctly, we can also match the other corresponding points and confirm that the viewpoint is correct, matching the ship. If we can guess the correct viewpoint, we do not have to search through the six dimensional space for it.

We may be able to use some other methods to estimate the view. We could begin our search from the estimate and expand the search around it. If the estimate is close, then we will not have to search for long to obtain the correct view.

If we are able to apply some quick test that will eliminate some ships from the search altogether, it is obvious that the search time will improve. Depending on the test, we may also be able to rank the models, so that the most promising are searched first.

In the next two sections, we present two heuristics that exploit these possibilities. We hope that their use will dramatically decrease the duration of the search and permit close-to-real time performance.

4.0 Vertical Feature Heuristic

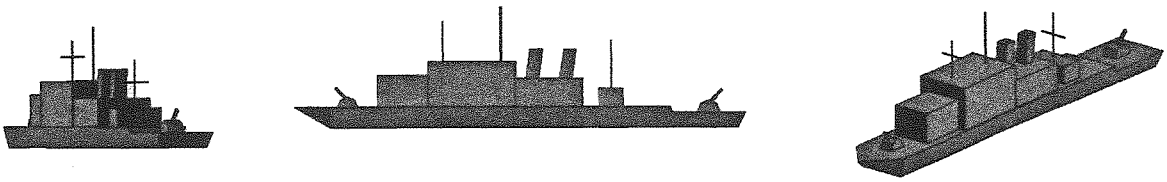

Figure 5. The vertical features of ships remain vertical in each image, even though the viewpoint has changed considerably.

The vertical feature heuristic is based on the fact that those vertical or close-to-vertical features of a ship, such as the masts, will likely remain vertical in an image, regardless of the direction in which the ship is facing, as illustrated in Figure 5. Using an edge detector and a Hough transform, we may apply standard Computer Vision techniques to identify vertical lines in the image (Duda & Hart 1972).

If the database contains information on where the vertical features lie in the model, it will be possible to establish correspondences between vertical features and the vertical image lines fairly easily. If enough correspondences can be found, we can calculate the associated transform and check if the model matches.

As well as finding point correspondences, we can use vertical features to quickly eliminate those models that cannot possibly match the image. For example, if we can determine which vertical features are masts, and we know how many masts are in the model, we can eliminate those models in which the number of masts precludes them from possibly matching the image.

The vertical feature heuristic only requires that 3D information on vertical features is included in the database. This data should be easy to obtain and also easy to include in the database, so this heuristic upholds Requirements 6 and 7.

The use of vertical features demonstrates the extensibility of the system (Requirement 10). All that is required is to include the new features in the database models, have some way to extract the features from an image, and a way of trying to match the two. In this case, we also use the heuristic in conjuction with the 3D search.

5.0 Moment Table Heuristic

Unfortuantely, the scope of this paper does not allow for a detailed explanation of the moment table heuristic, so this section serves only as an introduction to the concept.

A binary image is one in which the pixels are either white or black – a silhouette. If we segement the image (that is, if we identify those pixels that make up the ship and those that make up the background) we can form a silhouette $I(x,y)$ by setting all the ship pixels to one colour (eg white), and the background pixels to the other (eg black).

The *area* of the ship is simply the number of pixels in the silhouette (white pixels). We can work out the *centre of mass* of the silhouette by finding the mean of the sum of index-weighted pixels, in both the x and y direction:

$$area = \sum_{x,y} I(x,y), \quad \bar{x} = \frac{\sum_{x,y} xI(x,y)}{area}, \quad \bar{y} = \frac{\sum_{x,y} yI(x,y)}{area}, \quad \text{where } I(x,y) = \begin{cases} 1 \text{ if ship pixel} \\ 0 \text{ if background pixel} \end{cases}$$

These are the 0th and 1st order moments of the silhouette. In general, there are $n+1$, n^{th} order moments of a binary image given by

$$M_n = \sum_{x,y} x^p y^q I(x,y), \text{ where } n = p+q, \text{ and } p,q = 0,1,2...$$

We can form an image's moment vector, consisting of a sequence of increasing order moments. We can calculate an image's moment vector to arbitrary length. The moment vector encodes information about the shape of the silhouette, and if it is long enough, contains enough information to reconstruct the binary image.

The moment table heuristic is based on matching the moment vector of the segmented ship image against a list of precomputed moment vectors stored in what we have dubbed a 'moment table.' We need to match the moment vector of the image against a number of precomputed moment vectors because of the possible variances due to horizon occlusion and the direction in which the ship is facing.

Searching the list of moment vectors can be made efficient by employing an inverted index (Knuth 1997). Finding near-matching moment vectors will not solve the 3D search problem, but it will provide a ranked set of candidate ship models, with an initial guess of the appropriate view, as well as occlusion information, which will help to establish correspondences. Therefore, this heuristic may prove to be a valuable addition to a vessel classification system.

To include moment tables in the database, images of each ship from a number of known viewpoints are required. Images are easy to obtain, while the construction of the tables themselves should be quite simple, so Requirement 6 is satisfied.

6.0 Automatic Ship Generator

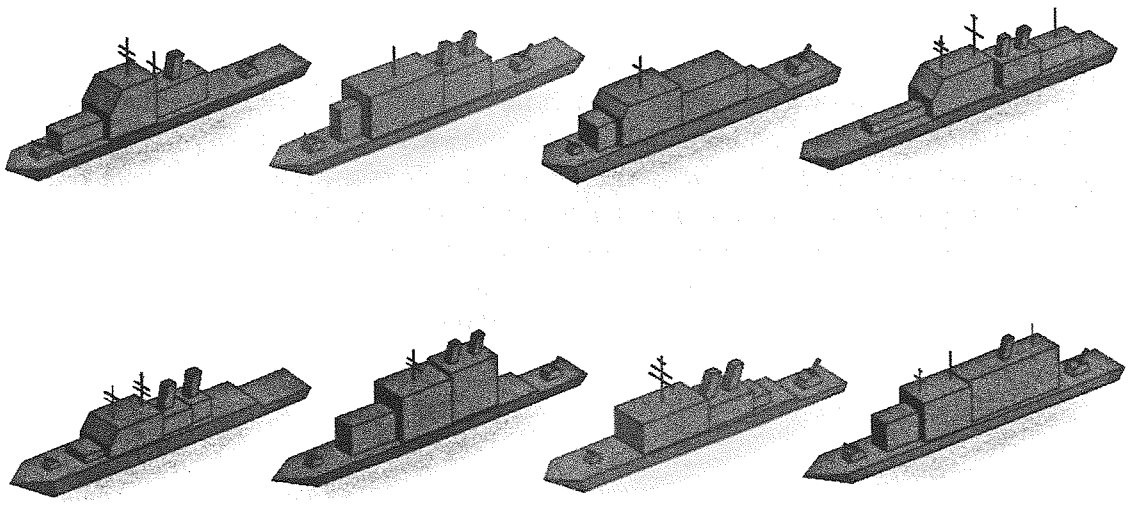

Figure 6. Eight ships from the automatic ship generator

We have designed the 3D corner point model search and the two heuristics with ease of data aquisition in mind. Should the system continue to be developed, it seems likely that the Navy will already have or be able to obtain all the necessary data. However, at this stage of development, we have almost no ship data to work with, and so we must generate our own test data.

In order to test the 3D corner point search, and the two heuristics, we designed a ship generator to generate corner point data, mast information, and images from arbitrary viewpoints to facilitate moment table generation and provide simulated 'periscope images' with which to test the system.

Figure 3 shows eight automatically generated ships. The ships are different enough to be distinctive, yet are probably more similar than most randomly chosen sets of ships would be. Therefore success in matching using the generated ships would suggest that the performance when applied to more distinct, real ships would be at least as good.

7.0 Further Work

We have explained that the 3D point search was designed to be robust to noise and incomplete data, but it would be desirable to measure exactly how robust the system is.

With the completion of the automatic ship generator, we are able to start measuring the performance of the system. We hope to implement and test both the 3D corner point model search and the vertical feature heuristic. Testing will involve measuring false positive and false negative rates, and the time taken to complete a search. This will require significant testing time. Unfortunately, there may not be time left for this project to also implement and test the moment table heuristic. This may prove to be a good starting point for future CEED students working on this project.

8.0 Conclusion

In this paper we have proposed a system for vessel classification that has been designed to be robust to noise and incomplete data, be easy to build and extend, and run in near to real time. A ship is represented in a database as a set of 3D corner points, vertical features and moment tables. We have explained how features to match this data can be extracted reliably from ship images. We noted that the data employed in the database may be reasonably easily obtained by the Navy should the project develop that far. However, to overcome the lack of data available at present, we developed an automatic ship generator that could generate database entries covering all the data we wish to search, as well as simulated periscope images with which to test the system. At the time of writing testing of the system is not complete, however where possible we show the reasons for design choices and provided explanations of how we expect the system to work.

9.0 References

Duda, R. O. and Hart, P. E. (1972) Use of the Hough Transformation to Detect Lines and Curves in Pictures, *Comm. ACM, Vol. 15*, pp. 11–15

Faugeras, O. (1993) Three Dimensional Computer Vision. MIT Press.

Harris, C. and Stevens, M. (1988) A Combined Corner and Edge Detector. *Proceedings of the Fourth Alvey Vision Conference,* Manchester, pp 147-151.

Knuth, D. (1997) The Art of Computer Programming, Volume 3: Sorting and Searching, Third Edition. Addison-Wesley, pp 560–563 of section 6.5: Retrieval on Secondary Keys.

Lowe D.G. (2001) Local Feature View Clustering for 3D Object Recognition. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition.*

Raytheon Australia Pty Ltd and The University of Western Australia have produced this paper collaboratively and copyright vests in both parties. No copy is to be made except for private study without the written consent of both parties.