Feature based three-dimensional object recognition using disparity maps

Ahmad Shawky

University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation
Shawky, Ahmad, "Feature based three-dimensional object recognition using disparity maps" (2007). Electronic Theses and Dissertations. 2086.
https://scholar.uwindsor.ca/etd/2086

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000 ext. 3208.
Feature Based 3D Object Recognition Using Disparity Maps

By

Ahmad Shawky

A Thesis
Submitted to the Faculty of Graduate Studies through the Department of Electrical and Computer Engineering in Partial Fulfillment of the Requirements for the Degree of Master of Applied Science at the University of Windsor

Windsor, Ontario, Canada
2007

© 2007 Ahmad Shawky
NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.
ABSTRACT

The human vision system is able to recognize objects it has seen before even if the particular orientation of the object being viewed was not specifically seen before. This is due to the adaptability of the cognitive abilities of the human brain to categorize objects by different features. The features and experience used in the human recognition system are also applicable to a computer recognition system. The recognition of three-dimensional objects has been a popular area in computer vision research in recent years, as computer and machine vision is becoming more abundant in areas such as surveillance and product inspection. The purpose of this study is to explore and develop an adaptive computer vision based recognition system which can recognize 3D information of an object from a limited amount of training data in the form of disparity maps. Using this system, it should be possible to recognize an object in many different orientations, even if the specific orientation had not been seen before, as well as distinguish between different objects.
ACKNOWLEDGEMENTS

First and foremost, I would like to thank Allah for everything that has been given to me... Whatever good I do is in his name and by his will.

I would like to thank my mother for pushing me though the years, my father for always being there to guide me and brother for constantly challenging me.

I would like to acknowledge my deepest gratitude to Dr. Xiang Chen, who had provided me with the opportunity, facility, guidance and expertise to conduct my studies. I would also like to thank my research lab partners for their help with troubleshooting the various challenges I had faced in my post-graduate experience.
TABLE OF CONTENTS

ABSTRACT............................................................................................................ iii

ACKNOWLEDGEMENTS...................................................................................... iv

LIST OF TABLES.................................................................................................... ix

LIST OF FIGURES.................................................................................................. x

CHAPTER 1: INTRODUCTION............................................................................... 1

1.1 Introduction...................................................................................................... 1

1.2 Literature Survey........................................................................................... 3

1.3 Motivation and Objective.............................................................................. 5

1.4 Contribution................................................................................................... 6

1.4.1 Recovery of 3D Information...................................................................... 6

1.4.2 Invariant Value Based Object Recognition.............................................. 6

1.4.3 Neural-Fuzzy Inference Network............................................................... 7

1.5 Thesis Overview........................................................................................... 7

CHAPTER 2: PRELIMINARY THEORY.................................................................... 8

2.1 Intensity Matrix............................................................................................. 8

2.2 Prospective Projection.................................................................................. 9

2.3 Pixel Windows and Image Filtering.............................................................. 11

2.4 Edge Detection............................................................................................. 14

2.5 Stereo Vision................................................................................................ 18

2.6 Point Correlation........................................................................................ 20

2.7 Epipolar Geometry....................................................................................... 22

2.8 Fuzzy Logic.................................................................................................. 23
CHAPTER 3: OBJECT RECOGNITION

3.1 Image Capture

3.2 Pre-Processing

3.3 Disparity Map Generation

3.4 Invariant Values

3.4.1 Compactness

3.4.2 Histogram

3.4.3 Moments

3.4.4 Affine Moments

3.5 Image Normalization

3.6 Fuzzifier

3.7 Class Membership Functions

3.8 Learning Vector Quantization

3.9 Determining the LVQ Cluster Centers

3.10 Class Membership Functions with LVQ Scaling

3.11 Adapted Rival Penalizing Learning Vector Quantization

3.12 Training the Network

3.13 Recognition

CHAPTER 4: EXPERIMENTS AND RESULTS

4.1 Implementation

4.2 Vision Platform
LIST OF TABLES

Table 3.1: Trained Disparity Maps of each Class ................................. 45
Table 3.2: Recognition Indexes for each Inference Method .................. 46
Table 4.1: Final Network Training Set .............................................. 52
Table 4.2: Recognition Results ...................................................... 55
Table 4.3: Recognition under Different Training Sizes ...................... 56
Table A.1: Image Normalization Angle of Rotation ......................... 63
LIST OF FIGURES

Figure 2.1: Digital Image and Intensity Matrix_________________________________8
Figure 2.2: Pinhole and Modified Pinhole Camera Representations________________11
Figure 2.3: Pixel Window Example_________________________________________11
Figure 2.4: Median Filtered Image and Intensity Matrix_________________________14
Figure 2.5: Sobel Edge Detection___________________________________________17
Figure 2.6: Canny Edge Detection Result_____________________________________18
Figure 2.7: Stereo Vision Geometry_________________________________________19
Figure 2.8: Epipolar Geometry_____________________________________________22
Figure 2.9: Triangular Fuzzy Membership Function____________________________23
Figure 2.10: Fuzzy Union and Intersection___________________________________25
Figure 2.11: Two-Layer Neural Network_____________________________________26
Figure 2.12: Two-Layer Competitive Neural Network__________________________27
Figure 3.1: Class Individual and Class Membership Functions____________________37
Figure 3.2: Cluster Center of a Set of Points__________________________________39
Figure 3.3: Simple Union vs. LVQ Scaled Class Membership Functions___________40
Figure 3.4: Comparison of Methods to Form Class Membership Functions__________41
Figure 3.5: Training Routine for the Nth Invariant Value________________________43
Figure 4.1: Flow Chart of Training and Recognition Processes____________________48
Figure 4.2: Vision Platform_________________________________________________50
Figure 4.3: Robotic Arm End-Effector_______________________________________51
Figure B.1: Stereo Image Capture Program GUI________________________________65
Figure B.2: Initializing Servo Motors________________________________________66

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
1.1 Introduction

Using disparity maps in object recognition is very useful, as a disparity map displays information about the 3D shape of an object. However, since the information is only available from a certain perspective, much like a 2D image, disparity maps are not fully 3D, and they are commonly referred to as 2.5D. To approach full 3D information, disparity maps can be taken from different perspectives around an object and the information can be combined to describe the object. This is based on the theory that if you observe an object from a satisfactory amount of views, you should be able to identify that object from any orientation, even if you had never encountered that specific orientation before.

Recognition is conducted by comparing one object with another, and in computer vision these objects are found in images. Though it is possible to compare an entire image to another, the amount of data provided by an image can make this comparison cumbersome and inefficient, and therefore, it is desirable to reduce the amount of information used in the comparison. One such way to reduce the informational burden of a recognition scheme is to characterize images through quantifiable metrics, or values. These values should be able to describe the state of the objects in the images in an efficient manner that is useful in recognition schemes. However, since there are many factors that can affect the images from which the values are calculated, it is desirable that the values are not affected (are invariant) to said changes. The majority of computer vision based object recognition schemes adopt the use of these invariant values as the arguments for each object. There are many different types of invariant values that can be used in object recognition, including Fourier descriptors, moment invariants and Hough transformations. Each of the preceding types is invariant to different conditions found in images such as translation, rotation, scaling and skewing; therefore, the choice of the invariants to be used should be such that all of these conditions are accounted for.
There have been a number of studies on object recognition using invariant values on 2D images [5][18], and using range data [12][15][17] with each study using a different set of invariants. This study will examine a few known invariants and their effectiveness for use in 3D object recognition using disparity maps: In particular, compactness [28] (Fourier descriptor), Histograms (Fourier descriptor), Hu’s seven invariant moments [18] (moment invariants) and Affine moment invariants [24] (moment invariants).

Since the invariant values are calculated from disparity maps, which are known for being sensitive to the various conditions of the images used in their generation, it is beneficial to introduce a certain tolerance when using the invariant values in recognition. Fuzzy logic was developed for similar situations as it takes the “crisp” invariant values and “fuzzifies” them to produce a weighted range of acceptable values from a single value. Another reason for the use of fuzzy logic is that it is desired to identify the entire object from all angles while being provided only with a limited set of images. This fuzzy data can be input into a neural-network to create what is known as a neural-fuzzy network, where the inference methods operate on a range of fuzzy values, rather than a single crisp value.

It is commonly known that Neural-Networks are widely used in the field of artificial intelligence for application such as comparative object recognition. As the data to be recognized is input into these networks, the “winner” is chosen based on a comparison with the data of other objects. In choosing the “winner,” a modified Learning Vector Quantization (LVQ) clustering technique is applied to the fuzzified data sets in order to properly scale the data for use in a competitive recognition scheme. The LVQ scheme is adapted in order to factor in the competitive nature of the other classes, or “rivals,” in the recognition scheme. There are many ways to compare the data and chose the winner, with these methods known as inference methods. This study will examine three different inference methods for neural-fuzzy networks [20]: Fuzzy set intersection, thresholded or “alpha-cut” fuzzy sets, and the value of the fuzzy set at a single input value.
1.2 Literature Survey

The idea of robot vision has been a largely explored topic in the last three decades. This is because many applications could be fulfilled by the implementation of a smart vision system on many of the robotic tools used today. One particular area which would benefit by such a system is the area of automated robotics. By implementing a smart vision system, the robot could work autonomously and actively gain information about its surroundings. Such systems have already been explored in areas of agriculture, where autonomous harvesters have been successful in harvesting over 80 acres of alfalfa in the United States of America [1]. The use of a camera is fundamental in the area of robotic vision systems, where the camera would be used as a robotic eye to gain the visual knowledge of the surrounding area; however, the use of only one camera lacks depth of field information which could be critical in certain applications. However, using when multiple cameras, the disparity of object position information between cameras can be used to calculate the depth of the object. Depth information, relative to the reference camera could be used in areas of agriculture and construction, to name a few, where a robotic arm would need to manipulate objects in its surroundings autonomously. While it might be possible to complete certain tasks using 2D vision, the added information provided by 3D information allows the system to function faster and more robustly than its 2D counterpart.

There have been numerous studies on vision based object recognition in both 2D and 3D cases. Many of these cases can be grouped into two cases; the first case being that the object geometry is known or assumed, and the second being that the object geometry is unknown. In the first case, the geometry of the object can be used to form generalizations about the shape of the object and possibly recover a simplified representation of said object. Recognition is then conducted on this representation by either directly comparing the geometry of the recovered shape with that of other trained objects or by comparing shape based geometric invariant values. In the second case, the geometry of the object is not known or assumed and recognition is done by calculating Fourier descriptor, moment, or Hough Transform based invariant values. There are also many different configurations of equipment used in the works that were surveyed,
including single camera setups, two or more camera setups, and the use of cameras in conjunction to 3D modeling software. The following is a brief list of studies conducted in the area of vision based object recognition:

- A region descriptor (Compactness) and a moment invariant have been used to implement a Fuzzy classification and recognition database for 2D shapes [7].

- Using 2D moment invariants, 3D objects have been modeled using a hybrid multi-layer perception network with one hidden layer [24]. 3 cameras were used to take 72 pictures of an object revolving on a turntable.

- The 2D moments of the object in each image is computed and used to calculate 7 moment invariants [25]. Best results were found when using the lower order moments and not using all 7 moments.

- Using descriptors such as illumination pose, illumination colour and specularity of an object in various images with varying illumination conditions an object’s colour distribution has been found and used in recognition [20]. This method required a large set of images and has only shown limited success.

- Object recognition has been conducted on range images using histograms, surface normals and curvature as object features [19]. Histograms are invariant to translations and rotations. Surface normals and curvature are calculated through the first and second derivatives of the image, respectively and are both invariant to translation.

- Using multiple 2D images of a 3D object, an adaptive neural network has been constructed for object recognition [21]. The input images are first aligned using the first moment and zero-order moment invariants to find the centroid and the location of the image center. A log-polar transform is taken to classify the coordinates of each point in the image using radial magnitude \( \log r \) and angle \( \theta \).
The coordinates are now invariant to scale and rotation since figural sizes and rotations are now converted to figural shift parameters. Recognition is conducted on the figural shift parameters.

- Variations in object shape and reflectance have been used for 3D object recognition based on numerous synthetically generated 2D views images from a 3D polyhedral mesh [22]. A 3D model was used to generate various 2D viewpoints. Image size does not dramatically affect recognition through these features. The more views provide higher recognition rates. Change of some object aspects can cause misclassification during recognition. Using a 2D “footprint” views can be aligned to resolve pose issues in the generated views.

1.3 Motivation and Objective

The purpose of this thesis is to explore and expand upon some of the ideals of two-dimensional object recognition and adapt them for three-dimensional object recognition using two cameras and passive 3D vision techniques. The challenges in such an adaptation include the generation of 3D information given only 2D inputs from regular CCD cameras, selection of the features to be used in the recognition scheme and ensuring that the recognition scheme is able to differentiate between different objects regardless of the orientation of the object.

In 2D recognition schemes it is assumed that most of the important information can be represented in a single 2D image, however, if the object contains different information that can only be viewed at certain angles, these 2D schemes cannot effectively work without modification. Another challenge in 3D object recognition is to be able to identify an object from a number of different views. While it is possible to simply train a recognition scheme with images of an object that were taken from all possible orientations of the object, a question might arise as to how many images are necessary to fully gain the information of an object. It is easy to visualize that as the number of images used to classify an object increases, so to does the memory requirements to store
such information, therefore, another important question might be arise as to how many images (views) are enough.

The primary goal of this study is to develop a 3D recognition scheme using 2D images while keeping the informational burden to a minimum in order to allow for the recognition scheme to be scalable to real time. In order to accomplish this goal, different techniques were examined and compared leading to the formulation of an optimal scheme. This scheme was then tested using a computer vision testbed platform to ensure that the ideals are applicable to real world situations. Finally, the techniques employed in this study could be used to develop 3D object information database and a smart 3D vision sensor which is capable of recognizing many objects from any 3D orientation.

1.4 Contribution

In this section, the main points of this research will be outlined.

1.4.1 Recovery of 3D Information
In order to gain 3D information of an object the use of disparity maps was proposed. Using disparity maps, 2.5D images can be formed using two regular 2D images. To approach full 3D information about an object, disparity maps were taken from numerous orientations of the object and the information is grouped into a class which defines said object. Different techniques of disparity map generation were studied and examined for their relative accuracy and execution time. Also the number of disparity maps needed to approach full 3D information was studied given different object shapes and orientations.

1.4.2 Invariant Value Based Object Recognition
Given that different invariant values display different strengths under certain condition it is important to select the right combination of values in order to achieve an optimal scheme for recognition. Through many studies, different invariant values have been used for similar applications, most only stating the results of their specific study, comparing these results against other methods is rarely addressed. In this study a selected list of invariant values has been chosen based on specific criteria and examined under different
conditions and their effectiveness has been gauged. Based on the performance of each invariant value, a final set of invariants to be used in the recognition scheme of this study has been selected.

1.4.3 Neural-Fuzzy Inference Network

Supervised learning systems have been employed in many previous studies in order to achieve adaptability in comparative recognition schemes. Many studies have adopted the use of data scaling to achieve system adaptability, based on the data that is input. Among the scaling techniques used in previous studies is a technique called Learning Vector Quantization (LVQ) which provides a measure of the distance between a data point and what is referred to as the cluster center of all data in its specific class. This study uses this distance measure to scale the data of each class so that data closest to the cluster center will retain most of its original value, while data farther from the cluster center will be downscaled to a fraction of its original value. A distance measure can also be conducted with respect to the cluster centers of other classes of data and all data can be scaled based on its relation to data of other classes.

1.5 Thesis Overview

This thesis is organized as follows: Chapter 2 outlines the preliminary theory for the necessary image processing, fuzzy logic and neural network operations conducted in the following chapters. Chapter 3 details the design methodology for the object recognition scheme, used in the experiments of Chapter 4. Chapter 4 contains the experiments that were conducted with an explanation of the results. Finally, Chapter 5 contains the conclusions from the experiments and recommendations for future work.

A user's manual for the vision platform used in this study can be found in Appendix A. This manual outlines the simple operations that were coded into the GUI that was used while conducting the experiments of Chapter 4.
Chapter 2: Preliminary Theory

This chapter will cover the preliminary image processing, fuzzy logic and neural network theory that is necessary to understand the operations found in the following chapters.

2.1 Intensity Matrix

Digital images are represented by a 2D array, or matrix, of numbers. Each of the elements in the array represents the brightness, or intensity, of light at the specific location in the image. Figure 2.1 shows a grayscale digital image and its corresponding matrix of intensity values. In an 8 bit representation, these intensity values will range from 0 to 255, where the larger the number, the more light is present at the pixel (0 = black, 255 = white).

(a)

![Image](a)

(b)

![Image](b)

Figure 2.1

C(a): 20 x 20 image (b): 20 x 20 intensity matrix representation of (a)

University of Windsor
Chapter 2: Preliminary Theory

Colour images are represented in a more complicated, yet similar manner. There are many different ways to represent colour in digital images, however there are two main colour representations used in digital imaging: The RGB (Red, Green, Blue) scale and the HSL (Hue, Saturation, Luminance) scale. In the more commonly used RGB scale, colour is represented by three separate matrices, with each matrix containing the intensity of light in the red, green and blue light spectrums, respectively. Image processing software such as C++ and stores colour information in 32-bit, “truecolour” format where the least significant 8 bits contain the red scale data, the next 8 bits contain the green scale data and the following 8 bits contain the blue scale data (BBGGRR). The most significant 8 bits are not used in this representation, making the colour data 24-bits in actuality, however since most processors deal with 32-bit data, the information is padded with null data and represented in 32-bits (00BBGGRR).

The location of pixels in an image are referenced by their $x$ and $y$ pixel offset from the top left corner of the image, where $x$ increases from left to right and $y$ increases from top to bottom. This is somewhat in contrast to the way in which matrices are usually referenced since the location of an element of a matrix is referred to by (row, column) and pixels are essentially referred to by (column, row).

2.2 Prospective Projection

The most commonly used optical representation of a camera can be found in Figure 2.2 (a). In this optical representation, a point in the real world will be translated to the image plane by a straight line through the origin of the camera plane. After passing through the camera plane, all points end up on the inverted quadrant in the image plane, leading to the image actually being captured upside-down and backwards. Since all real world points pass through the same point, this projection technique is referred to as the pinhole camera representation. Using this projection technique, the mapping from the real world to the image plane is conducted as follows:
where $X$, $Y$ and $Z$ are the coordinates in the real world, with respect to the center of the camera plane, $x$ and $y$ are coordinates in the image plane and $f$ is distance between the camera plane and the image plane (focal length). To simplify the representation and the associated calculation, the image plane can be visualized in front of the camera plane as seen in Figure 2.2 (b), and the calculation are simplified to the following:

\[ x = f \frac{X}{Z} \]  
\[ y = f \frac{Y}{Z} \]

It is important to note that the coordinate $z$ on the image plane does not exist. This is because a camera captures a 2D representation of a 3D scene and depth information is lost during the translation from 2D to 3D.
2.3 Pixel Windows and Image Filtering

A pixel window is an area in a digital image of a predetermined size. The window is square in shape, encloses a certain number of pixels of the image and always contains a central pixel, making the format of the window size $M \times M$, where $M$ is an odd number.

An example of this is found in Figure 2.3 which displays the bounding of a $3 \times 3$ pixel window around the pixel with an intensity of 122.
Pixel windows are used in a number of image processing techniques such as mask filtering. In mask filtering, a window of pixels in the image is manipulated by a same sized array of pixels of a filter (referred to as a mask). The mask is then convolved with the pixel window from the image and the result replaces the central intensity value of the pixel in the image window. An example of the 2D convolution of the mask and the pixel window can be seen in the following example:

**Example 2.1:**

Pixel Window from Image: \( W = \begin{bmatrix} 89 & 143 & 127 \\ 77 & 122 & 115 \\ 85 & 114 & 114 \end{bmatrix} \)

Mask Filter: \( M = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \)

\[ W \ast M = \frac{1}{9} (89 \times 1 + 143 \times 1 + 127 \times 1 + 77 \times 1 + 122 \times 1 + 115 \times 1 + 85 \times 1 + 114 \times 1 + 114 \times 1) \]

\[ = 109.56 \]

Pixel Window from Image After Applying the Mask: \( W = \begin{bmatrix} 89 & 143 & 127 \\ 77 & 109.56 & 115 \\ 85 & 114 & 114 \end{bmatrix} \)

It is important to note that during the image filtering process, the new intensity values of the central pixels are placed into a new image, not the original image, as doing so would erroneously propagate the filtered value whenever this value is used in another pixel window.
Median filtering is conducted similar to mask filtering by first forming a window of pixels. However, after the window of pixels is formed, the values of the intensities found in the window are ordered from lowest to highest value. The value of the new center pixel is the value that is found in the middle of this list of intensities as seen in the following example:

**Example 2.2:**

$$\begin{bmatrix} 89 & 143 & 127 \\ 77 & 122 & 115 \\ 85 & 114 & 114 \end{bmatrix}$$

Sorted list of intensities: 77 85 89 114 114 115 122 127 143.

114 is the median value.

$$\begin{bmatrix} 89 & 143 & 127 \\ 77 & 114 & 115 \\ 85 & 114 & 114 \end{bmatrix}$$

Figure 2.4 shows the image from Figure 2.1 (a) after median filtering with a 5 x 5 window as well as its associated intensity matrix.
In order to manipulate the pixels around the border of the image, the pixel windows for these pixels must be padded with zeros in order to maintain the filtering window size as illustrated when trying to form a pixel window around the pixel at (0,0) in the matrix of Figure 2.1 (b):

\[
\begin{bmatrix}
82 & 101 & 220 \\
85 & 89 & 143 \\
75 & 77 & 122 \\
\end{bmatrix}
\]

Pixel Window from Image: \( W = \) [[82, 101, 220], [85, 89, 143], [75, 77, 122]]

Padded Pixel Window from Image: \( W = \) [[0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 82, 101, 220, 0], [0, 0, 85, 89, 143], [0, 0, 75, 77, 122]]

### 2.4 Edge Detection

An edge point in an image is a pixel around which the neighboring pixels exhibit a large change in intensity. Edge points are very useful in computer and machine vision.
processes as they help to accent the boundaries of the different shapes and colours present in an image. There are many ways to find (detect) these edges; from simple image filtering techniques, such as Sobel edge detectors, to more complex multi-process techniques, such as Canny edge detectors.

Sobel edge detection is conducted by first filtering the image to reduce the noise in the image, then filtering the image with two different masks. The results of the image filtering are the edges in the horizontal and vertical directions. After the two filtered images are generated, they can be combined by calculating the Euclidean distance of the intensity values of each image and forming a new image with the result.

Figure 2.5 shows the steps taken in the Sobel edge detection technique. Figure 2.5 (a) shows the original, unprocessed image and Figure 2.5 (b) shows the result after applying a median filter with a window size of 5 x 5 pixels. After the image is filtered to remove any undesired noise, the image gradients are generated by applying two filter masks to the filtered image as seen below:

*Horizontal Gradient Mask:*

\[
\begin{bmatrix}
-1 & -2 & -1 \\
0 & 0 & 0 \\
1 & 2 & 1
\end{bmatrix}
\]

(2.5)

*Vertical Gradient Mask:*

\[
\begin{bmatrix}
-1 & 0 & 1 \\
-2 & 0 & 2 \\
-1 & 0 & 1
\end{bmatrix}
\]

(2.6)
Applying these masks will generate the horizontal \((I_x)\) and vertical \((I_y)\) gradients of the image as seen in Figure 2.5 (c) and 2.5 (d), respectively. By using the Euclidean distance of the values of the two gradients at each pixel, the final edge image is formed, as seen in Figure 2.5 (e). This Euclidean distance is calculated one pixel at a time as follows:

\[
G(i, j) = \sqrt{I_x^2(i, j) + I_y^2(i, j)}
\]  

(2.7)
Figure 2.5:
(a) Original Image (b) 5 x 5 Median Filtered Image (c) Horizontal Image Gradient
(d) Vertical Image Gradient (e) Combined Edge Image
Chapter 2: Preliminary Theory

Canny edge detection is a more complex edge detector that not only enhances the edges, as done with Sobel edge detection, but also conducts non-maximum suppression, which makes all edges have a single pixel width, and hysteresis thresholding, which eliminates edge values that are under a certain value. The result of Canny edge detection on the image in Figure 2.5 (a) can be seen in Figure 2.6.

![Canny Edge Detection Result](image)

Figure 2.6: Canny Edge Detection Result

2.5 Stereo Vision

There are two main methods to gain 3D special knowledge from 2D images. The first way is referred to as active 3D vision and uses a dedicated light source, such as a laser line, to infer 3D shape based on the distortion of shape that the line exhibits when it contacts an object in view of a camera. Another technique is called passive 3D vision and does not rely on any other equipment besides cameras. Stereo vision is a passive 3D vision technique that uses two or more 2D images of the same scene to gain 3D information about that scene. Figure 2.7 shows the simple geometry of a 2 camera stereo vision setup and displays the way in which a common point, seen in both images would be placed in each respective image. In this figure the pinhole projection technique is adopted and the translation line from the real world point to the two image planes is displayed.
Notice that the location of an object, with respect to the center of the associated image planes, will be different in the two images due to the orientation of the cameras with respect to the object. This difference in position is called disparity and is the basis for most passive 3D computer vision techniques. This disparity is calculated as follows:

\[
Z = b \frac{f \lambda}{x_l' - x_r'}
\]  

(2.8)

where \(Z\) is the calculated depth of the point, \(b\) is the distance between the left and right cameras (baseline), \(f\) is the focal length of the cameras, \(\lambda\) is a distance to pixel ratio, and \(x_l' - x_r'\) is the disparity of the point between the left and right images.
2.6 Point Correlation

Using Equation 2.8, the depth of all points in the images can be calculated, however, in order to calculate the correct depth, the correct point pairs must be found. The general method of finding matching points is to compare a certain point to a set of other points and calculate a matching, or correlation value. By using a point in one image as the argument and finding its correlation with a set of points in the other image, the disparity can be calculated from the point pair with the best match, or best correlation value. There are many different methods for finding the correlation of points in stereo images; among these correlation methods are the popular sum of square difference (SSD) method and the normalized cross correlation (NCC) method.

The SSD method first takes a window of pixels around a central pixel from one image and compares it to a window of pixels around a central pixel in the other image. The correlation values are calculated as follows [7]:

$$SSD(dx, dy) = \sum_{W(x,y)} [I_L(x,y) - I_R(x+dx, y+dy)]^2$$  \hspace{1cm} (2.9)

where $I_L$ and $I_R$ are the left and right images, respectively, $(x, y)$ is the location of the argument pixel in the left image, $W(x, y)$ is a window of pixels around the pixel at $(x, y)$, and $(dx, dy)$ is the displacement of the pixel in the right image with respect to that of the argument pixel. Using the same argument pixel from the left image, the SSD of all pixels in the right image is calculated. The pixel with the smallest SSD value is the pixel from the right image that best matches the argument pixel, and the value of the disparity map at location $(x, y)$ is calculated from $dx$ and $dy$.

The NCC method is conducted in a similar manner to the SSD method with the major difference being that the average intensities are calculated for the windows of pixels in the left and right images. Another difference is that the correlation values range from -1 to +1, with +1 corresponding to a best match. The correlation values are calculated as follows [7]:
Chapter 2: Preliminary Theory

\[ NCC(\vec{d}) = \frac{\sum_{W(\vec{x})} [(I_L(\vec{x}) - \bar{I}_L)(I_R(\vec{x} + \vec{d}) - \bar{I}_R)]}{\sqrt{\sum_{W(\vec{x})} (I_L(\vec{x}) - \bar{I}_L)^2 \sum_{W(\vec{x} + \vec{d})} (I_R(\vec{x} + \vec{d}) - \bar{I}_R)^2}} \]  

(2.10)

where \( \vec{x} \) is the position vector \((x, y)\), \( \vec{d} \) is the displacement vector \((dx, dy)\), \( \bar{I}_L \) and \( \bar{I}_R \) are the average intensities of the window of pixels, \( W(\vec{x}) \) and \( W(\vec{x} + \vec{d}) \), in the left and right images, respectively, and are calculated as follows:

\[ \bar{I}_L = \frac{1}{N} \sum_{W(\vec{x})} I_L(\vec{x}) \]  

(2.11)

\[ \bar{I}_R = \frac{1}{N} \sum_{W(\vec{x} + \vec{d})} I_R(\vec{x} + \vec{d}) \]  

(2.12)

where \( N \) is the total number of pixels in the pixel windows. By incorporating the average intensity into the correlation calculation, NCC becomes less affected by changes in lighting between the left and right images. These changes can arise from a stereo setup with a wide baseline, or from a background with dynamic lighting. As with the SSD method, the argument pixel in the left image is compared to pixels in the right image.

Though possible, conducting an exhaustive search of one image for a specific point match is very time consuming. This type of search can also lead to false matches if there are many regions in the images with a similar appearance. One way to limit this search is to conduct a feature based match by only examining the correlation of feature points, such as edge points [8]. The tradeoff with this method is that correlation will only be found for feature points while other points remain unknown. It is also possible to reduce the search for point correspondence with a more extensive knowledge of a stereo vision system, as discussed in the following section.
2.7 Epipolar Geometry

Figure 2.8 shows a more detailed example of the geometry of a two camera stereo vision configuration [30]. In this setup, the path that connect the point $P$ in the real world to the center of the left camera projection pinhole, $C_l$, is seen as a single point $p_l$ in its own image. However, if the cameras view the point not only from different locations along the baseline, but at different angles as well, the same path that produced a single point $p_l$ in the left image can be viewed as the line $e_l p_r$ in the right image. This line is known as the epipolar line. Epipolar geometry states that if the same point exists in two images, it can be found on epipolar lines. The same holds for points in the right image leading to the generalization that a point in one image can be found on a line in the other image. Using epipolar geometry, it is possible to reduce the search for corresponding pixels in two images from a 2D search problem to a 1D search problem, thus saving time and eliminating a large percentage of false matching possibilities.

2.8 Fuzzy Logic

The main principle of fuzzy logic is to describe uncertainty or partial truth [20]. In fuzzy logic, uncertainty, or fuzziness, replaces rigid (crisp) truth to allow for more flexibility in
terms of perception. For example, expressing truth with binary logic (1 or 0) would lead to an absolute classification of a situation as "right" or "wrong," however with the added uncertainty of fuzzy logic, it is possible to express the same situation as a set of many possible classifications such as "mostly right" and "not entirely wrong," or any different number of various degrees of "right" and "wrong."

The translation of crisp values into fuzzy sets is done through what is referred to as a fuzzifier. The fuzzifier takes a single "crisp" value and transforms it into a "fuzzified" distribution of a desired shape. These shapes include triangular, trapezoidal and Gaussian distributions. For example, the fuzzification of a crisp value into a triangular fuzzy set [5] is done through the following equation:

\[
f(x) = \max \left( \min \left( \frac{x-(1-k)a}{2ka}, \frac{(1+k)a-x}{2ka} \right), 0 \right)
\]  

(2.13)

where \(a\) is the crisp input value, \(k\) is the support factor coefficient which determines the width of the triangular membership function and is usually set to a desired value by trail and error, and \(x\) spans the universe of discourse (x-axis). An example of this fuzzification can be seen in Figure 2.9.

\[\text{Figure 2.9: Triangular Fuzzy Membership Function (a=10, k=0.3)}\]
2.9 Fuzzy Logic Operations

Once crisp values have been fuzzified, a number of logical operations can be conducted on their fuzzy sets. Among these operations is the fuzzy union operation. This operation is essentially a Boolean OR operation which forms a new set based on the maximum values of the elements of the two argument sets as seen by the following equation:

\[
\mu_j(x) = \mu_1(x) \lor \mu_2(x) \lor \ldots \lor \mu_j(x) \quad x \in X
\]  

(2.14)

where \(x\) is the self-variable, \(X\) is the universe of discourse, \(\mu_i(x)\) is the value of the \(i\)th fuzzified data set at \(x\) and \(\lor\) is the maximum comparator operation. An example of a fuzzy union of two fuzzy sets can be seen in Figure 2.10 (c). The fuzzy union operation is useful when combining data from many different fuzzy sets into a single set.

Another fuzzy logic operation is the fuzzy intersection operation. This operation is essentially a Boolean AND operation which forms a new set based on the minimum values of the elements of the two argument sets as seen by the following equation:

\[
\mu_j(x) = \mu_1(x) \land \mu_2(x) \land \ldots \land \mu_j(x) \quad x \in X
\]

(2.15)

where \(\land\) is the minimum comparator operation. An example of a fuzzy intersection operation can be seen in Figure 2.10 (d). The fuzzy intersection operation is useful when trying to find the degree of similarity of fuzzy sets as it can form a new fuzzy set that is based on the overlapping of many different fuzzy sets.
Figure 2.10
(a): Fuzzy Set A \((a=10, k=0.3)\) (b): Fuzzy Set B \((a=15, k=0.4)\)
(c): Fuzzy Set \(A \cup B\) (d): Fuzzy Set \(A \cap B\) (shaded region)

2.10 Neural Networks

Artificial neural networks are used to model the simplified activity of neurons in the human brain during a decision making process. Figure 2.11 shows a simplified two-layer neural network with two inputs and one output. The first layer is the input layer, which is composed of a set of features that are used to distinguish between objects. Each input feature is then connected to every neuron in the second layer, called the hidden layer. The hidden layer contains a specific excitation factor which determines the output of the neuron. This factor can be seen as the neurons firing rate. This rate is used to scale the value of an input based on certain criteria of the neuron and therefore can be seen as a multiplicative weighting factor. More detail on this scaling factor will be given in Chapter 3.8. The output layer contains only one neuron and which can be seen as the final decision of the neural network after an examination of the inputs in the hidden layer.
Figure 2.12 displays a more complicated example of a neural network which includes a competitive decision process. This type of neural network is ideal for the object recognition algorithm of this study because the final decision is a competitive comparison of the degrees of membership between each input object and each pre-trained class. In the input layer $p_i$ through $p_m$ represent the different inputs that are used for the recognition scheme. The firing rate in the hidden layer of Figure 2.11 acts similar to a multiplicative weight factor, therefore in Figure 2.12 the hidden layer is replaced by scaling factors. The input values are first scaled by the hidden layer weight factors $w_{1i}, \ldots, w_{1n}$ through $w_{mi}, \ldots, w_{mn}$ then used to form the recognition index at the output neuron for each class. These outputs are calculated using what is referred to as inference techniques, a few of which will be examined in Chapter 3.11. In this example, the final recognition, or best matching criteria is decided by the class with the maximum recognition index.
Chapter 2: Preliminary Theory

Figure 2.12: Two-Layer Competitive Neural Network
CHAPTER 3: OBJECT RECOGNITION

This chapter outlines the methodology used in the design of the object recognition scheme of this study. This recognition system can be used in various applications such as automated tracking and object inspection. After an object has been recognized using this system, object specific processes can be selected and conducted autonomously. Such a system could be useful in recognizing a specific component on an assembly line, which could potentially be present in any orientation, then conducting a specific task such as automatically picking it up with a robotic arm or further inspecting it for certain defects.

3.1 Image Capture

The input stereo images were captured using 2 Prosilica 1350C single CCD colour cameras with a Bayer filter pattern of RGRG-GBGB. The cameras were used with Computar H6Z0812MP mechanical lenses with a possible focal length of 8-48mm. In order to cut down on the preprocessing procedures, the cameras were configured to capture full colour images with a resolution of 640x480. The cameras use IEEE 1394 Firewire communication and the lenses use serial communication though separate VL1C-12V RS-232 controllers for each lens.

3.2 Pre-Processing

In order to reduce computation time, grayscale images are used in the generation of the disparity maps, which are used during the training and recognition processes. Though there is an available grayscale capture mode on the cameras, the Bayer filter on the cameras produces a grid-like pattern on the images due to the way in which the filter interprets each pixel; therefore, grayscale images were generated from the colour images through the following transformation:

\[ I_{\text{Gray}} = 0.59I_G + 0.30I_R + 0.11I_B \]  

(3.1)
where $I_r$, $I_g$, $I_b$ and $I_{Gray}$ are the red, green, blue and resulting grayscale values of the pixels of the image, respectively.

Preprocessing also includes a comparative routine to extract the object in the images from its surrounding background. First, an image of the background is captured, next, the object is placed in that background and an image is captured. Once these images are captured the values of the images are compared on a pixel-to-pixel basis, and if the difference is under a certain threshold the pixel is set to black while the pixels that are highly different from the background retaining their value and the object extracted.

### 3.3 Disparity Map Generation

After the colour images are transformed into grayscale images the disparity of each pixel of the images can be generated and stored. The disparity map is a 2D array with each element containing a disparity value. The disparity of a point is inversely proportional to its depth to the camera, as seen in Equation 2.8. Due to this relationship, a disparity map can also be seen as a range, or depth, map. Also, since the disparity map is a 2D array, the information can be conveniently displayed in an image representation, with the value of pixel intensity equal to the value of the disparity at that pixel; the brighter the pixel, the greater the disparity and the smaller the distance of that point to the camera. A disparity map is generated from a stereo pair of images, which were taken with a stereo camera setup from a left view and a right view perspective. To lower the computational time, the SSD function (Equation 2.9) was used, as the trade off of accuracy to time was seen as favorable when compared to using the NCC function.

To generate a complete disparity map, the disparity of all points in the stereo images must be found. This task, though simple in theory, has distinct drawbacks when implemented on actual stereo images. It can be seen that the computational burden increases exponentially with image size, as an increasing number of pixels in one image must each be compared to an increasing number of pixels in the other image. This has a large affect on the disparity map generation as the time required to make the disparity map increases and the larger amount of similar regions of pixels can lead rise to false correspondence. To reduce these problems a range can be defined to limit the search for correspondence;
this range is defined such that the computational load is minimized and the possibility of false matches is limited. One way to reduce the range is to calibrate the stereo camera setup so that all epipolar lines of the stereo pair are parallel to each image's respective y-axis, forcing all corresponding point pairs to exist at the same y-axis coordinate, and thus eliminating the need for computationally intensive stereo image rectification. Another way to reduce the range is to set up a maximum disparity argument, which the search for correspondence will not exceed; if satisfactory correspondence is not met in the range below this maximum disparity, the argument point is considered occluded and its disparity value is set to zero. Both of these methods have been adopted in the image capture and disparity map generation in this study. Once the disparity map has been generated, all of the necessary preprocessing for the recognition phase has been completed.

3.4 Invariant Values

The use of a limited number of features in object recognition helps to reduce the information used in the recognition process while directing the focus on the specific aspects which will be used as the arguments during the matching process. Since there are a number of different conditions that the same object in a number of different images can be subject to, it is important that these features have invariance to these conditions. These conditions include translation, scaling, rotation, which are linear transformations and skewing, which is a non-linear transformation. There are many different classes of invariant values, with each class displaying invariance to a different combination of image transformations. Among these classes are Fourier descriptors which are used to describe regional distributions in an image and are invariant to translation and rotation. Other classes of invariant values include Moment invariants, which are invariant to scaling and rotation, and Hough transformations, which are invariant to translation and rotation. All of the preceding invariant values have been used on 2D images; however the use of these values can beneficially extended to range data. The following invariants were examined in this study for their effectiveness in use in object recognition from disparity maps.
3.4.1 Compactness

Compactness is a Fourier descriptor that describes a distribution of intensity values in an enclosed region. Introduced to describe colour distributions in 2D images, when applied to a disparity map, it can describe disparity (distance), and therefore shape distributions and is invariant to translation and rotation. As described in [28], the compactness of an image can be calculated as follows:

\[
C_{2D} = \frac{\left(\sum_{x=1}^{h} \sum_{y=1}^{w} f_{\text{boundary}}(x, y)\right)^2}{\sum_{x=1}^{h} \sum_{y=1}^{w} f(x, y)}
\]  

(3.2)

Where \( f(x, y) \) is the value of the intensity at pixel \((x, y)\) scaled to fall in range from 0 to 1, and \( \sum_{x=1}^{h} \sum_{y=1}^{w} f_{\text{boundary}}(x, y) \) is the perimeter of a region (object).

3.4.2 Histogram

Histograms are Fourier descriptors, and in 2D images they are used to examine the distribution of colours found in the image and can be calculated as follows:

\[
H(v) = \frac{\sum_{i=0}^{w} \sum_{j=0}^{h} \mathbf{1}_{I(i,j)=v}}{w \times h}
\]

(3.3)

where \( H(v) \) is the value of the histogram at index \( v \), \( w \) and \( h \) are the width and height of the image, respectively and \( I(i,j) \) is the intensity of image \( I \) at pixel \((i,j)\). When applied to disparity maps, a histogram will describe the distribution of heights found in the view of the stereo camera setup. Histograms are invariant to translation and rotation since they are a function of the intensities of the pixels in an image, not their location or orientation. If the histograms are computed for two different images, a comparison can be calculated using the following equation [12]:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
where $I_1$ and $I_2$ are the two images, $i_{1i}$ and $i_{2i}$ are the $i$th elements of the first and second histograms, respectively, and $M$ is the final element in the histograms. The result of this comparison is a single value which is small in scale if the two histograms are similar and large if the two histograms are different.

### 3.4.3 Moments

In his 1962 paper, Ming-Kuei Hu derived seven invariant moments which have been widely used in the field of image recognition [18]. The use of moment invariant values is very popular in object recognition and tracking as the moment values are invariant to the linear transformations of translation, rotation and scaling. The following is a summary of the derivation of the Hu’s seven invariant moments.

The $pq$th moment of a density distribution can be found as follows:

\[
m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q \rho(x, y) \, dx \, dy
\]

where $\rho(x, y)$ is the value of the distribution at location $(x, y)$. To make this moment invariant to translation, the values of the displacement from the distribution’s centroid are used in place of $x$ and $y$.

\[
\mu_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - \bar{x})^p (y - \bar{y})^q \rho(x, y) \, dx \, dy
\]

and the distribution centroid $(\bar{x}, \bar{y})$ can be found as follows:

\[
\bar{x} = \frac{m_{10}}{m_{00}}
\]
\[
\bar{y} = \frac{m_{01}}{m_{00}}
\]
In a digital image, the Riemann sum can be used to calculate the integrals and the equation for the central moments becomes:

$$\mu_{pq} = \sum_{x=0}^{w} \sum_{y=0}^{h} (x - \bar{x})^p (y - \bar{y})^q f(x,y)$$  \hspace{1cm} (3.9)

where $f(x,y)$ is the intensity of the image at location $(x, y)$ and $w$ and $h$ are the width and height of the image, respectively. Next, the central moments are normalized to make them invariant to scale:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}} \text{, where } \gamma = \frac{p + q + 2}{2}$$  \hspace{1cm} (3.10)

Finally, the normalized central moments are used to calculate Hu’s seven invariant moments as follows:

$$\Phi_1 = \eta_{20} + \eta_{02}$$  \hspace{1cm} (3.11)
$$\Phi_2 = (\eta_{20} - \eta_{02})^2 + 4 \eta_{11}^2$$  \hspace{1cm} (3.12)
$$\Phi_3 = (\eta_{30} - 3 \eta_{12})^2 + (3 \eta_{21} - \eta_{03})^2$$  \hspace{1cm} (3.13)
$$\Phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$  \hspace{1cm} (3.14)
$$\Phi_5 = (\eta_{30} - 3 \eta_{12}) (\eta_{30} + \eta_{12}) \left[ (\eta_{30} + \eta_{12})^2 - 3 (\eta_{21} + \eta_{03})^2 \right] + (3 \eta_{21} - \eta_{03}) (\eta_{21} + \eta_{03})$$  \hspace{1cm} (3.15)
$$\Phi_6 = (\eta_{30} - \eta_{02}) \left[ (\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2 \right] + 4 \eta_{11} (\eta_{30} - \eta_{12}) (\eta_{21} + \eta_{03})$$  \hspace{1cm} (3.16)
$$\Phi_7 = (3 \eta_{21} - \eta_{03}) (\eta_{30} + \eta_{12}) \left[ (\eta_{30} + \eta_{12})^2 - 3 (\eta_{21} + \eta_{03})^2 \right] - (\eta_{30} - 3 \eta_{12}) (\eta_{21} + \eta_{03})$$  \hspace{1cm} (3.17)

Moments $\Phi_1$ through $\Phi_6$ are absolute orthogonal invariants, and are invariant to translation, rotation and scaling while the moment $\Phi_7$ is useful in recognition of mirror images.
3.4.4 Affine Moments

Among image transformations such as translation, rotation, scaling and skewing is what’s known as the 2D affine transformation. The general 2D affine transformation is a combination of the four basic transformations and can be expressed as follows:

\[
\begin{align*}
\hat{X} &= AX + B \\
\begin{bmatrix}
\hat{x} \\
\hat{y}
\end{bmatrix} &=
\begin{bmatrix}
a_{11} & a_{12} \\
a_{21} & a_{22}
\end{bmatrix}
\begin{bmatrix}
x \\
y
\end{bmatrix}
+ 
\begin{bmatrix}
b_1 \\
b_2
\end{bmatrix}
\end{align*}
\]

(3.18)

where \(a_{11}\) through \(a_{22}\) are the affine coefficients which determine the rotation, scaling and skewing of the image and \(b_1\) and \(b_2\) are the affine coefficients which represent the translation of the image. Hu’s moments are not invariant under this affine transformation and affine invariants have been introduced to address this [24]. The first two affine invariant moments can be expressed using the central moments as follows:

\[
I_1 = \frac{1}{\mu_{00}^4} (\mu_{20}\mu_{02} - \mu_{11}^2)
\]

(3.19)

\[
I_2 = \frac{1}{\mu_{00}^6} (\mu_{30}^2\mu_{03}^2 - 6\mu_{30}\mu_{21}\mu_{12}\mu_{03} \\
+ 4\mu_{30}\mu_{11}^3 + 4\mu_{20}\mu_{21}^3 - 3\mu_{21}^2\mu_{12}^2)
\]

(3.20)

It is important to note that disparity maps are inherently noisy images, and that the noise in the images is affected by various conditions of the input stereo images. Among these conditions is the presence of uniform or non-textured regions. Since disparity is found by finding the correlation of windowed regions of pixels in both left and right images, if there is a region where all pixels are relatively similar it is hard, if not impossible, to accurately find the proper correspondence of points. The result is a region on the disparity map with untrue values, which can be viewed as noise. This noise will have an impact on the calculated invariant values since the invariant values are derived from the
intensity of the pixels in the disparity map. This effect is easy to observe when calculating Hu's seven invariant moments; as the higher-order moments use increasing powers of the central moments, the results under different conditions of noise vary more dramatically than that of the lower-order moments. This leads to the conclusion that higher-order Hu moments are not suited for this specific recognition application, as can be said about the affine invariant moments.

3.5 Image Normalization

Image normalization is a process which takes any image subject to a particular transformation and produces an image which is invariant to all of the transformations. For example, if an image is subject to skewing, it would produce the same image as the un-skewed image after both images were normalized. The image normalization scheme used in this study is a multi-step process, which uses the strengths of Hu's moments to first compact the image, rendering it invariant to translation, scaling and skewing. After the image is compacted Hu's second order moments are used to render the image invariant to rotation. The following equation is applied to the coordinates of each pixel in the image.

\[
\begin{bmatrix}
    \bar{x} \\
    \bar{y}
\end{bmatrix} =
\begin{bmatrix}
    \cos \phi & \sin \phi \\
    -\sin \phi & \cos \phi
\end{bmatrix}
\begin{bmatrix}
    \frac{\sqrt{\lambda_1 \lambda_2}}{\lambda_1} & \frac{\sqrt{\lambda_1 \lambda_2}}{\lambda_2} & 0 \\
    \sqrt{\lambda_1} & \sqrt{\lambda_2} & 0 \\
    0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
    e_{1x} & e_{1y} & x - \bar{x} \\
    e_{2x} & e_{2y} & y - \bar{y}
\end{bmatrix}
\]

(3.21)

For a more detailed explanation of image normalization, please refer to Appendix A.

3.6 Fuzzifier

After the invariant values are calculated they are used as the inputs of a two layer neural-fuzzy network. The invariant values are used to form what is referred to as a class membership function which will define the acceptable values of the invariants for different training views of the object of each class. One of the reasons for including fuzzy
logic is to limit the possible effects of noise in the disparity maps on the calculated invariant values. Another reason is so that the invariant values do not rigidly define the images they were derived from, but rather these values are used to define a range of acceptable values for the object, allowing for recognition from a larger variety of views of the object based entirely on information from a minimum number of views. Before the invariant values can be used in the network, they must be brought into the fuzzy domain by a fuzzifier transformation. The fuzzifier chosen for this study is a Gaussian due to its attribute of displaying a near uniform weight around the mean value. The invariant values are fuzzified as follows:

\[
F(x, a, \sigma) = e^{-\frac{(x-a)^2}{2\sigma^2}}
\]  

(3.21)

where, \(x\) is the self-variable, which spans the universe of discourse of the domain of the fuzzy distribution and \(a\) is the mean value (the invariant value in this case). The value \(\sigma\) is the standard deviation around the mean value which is used to set the tolerance during the recognition phase and is determined though examination of the input invariant value, \(a\); for example, if input \(a\) has a magnitude of around 100, it may be desirable to set \(\sigma\) equal to 10 to produce a fuzzified distribution of values in the domain of 90 to 110 with the values in the range having the highest weight at 100.

### 3.7 Class Membership Functions

Each invariant value is fuzzified separately with an appropriate standard deviation and added into a membership function that will define the object of each specific class. The simplest method of combining the fuzzified data is to use the fuzzy union operation on the separately generated fuzzy distributions of each invariant value. Figure 3.1 shows individual fuzzy membership functions and the resulting class membership function after the union operation.
Chapter 3: Object Recognition

Figure 3.1
(a): Individual Fuzzy Membership Functions
(b): Combined Class Membership Function
3.8 Learning Vector Quantization

The problem with simply combining the fuzzy data with the union operation is that outliers to the class will be given the same weight of importance as the more appropriate data. One way to soften the impact of outliers on the set is to use the Learning Vector Quantization (LVQ) clustering technique.

3.9 Determining the LVQ Cluster Centers

When using the LVQ methods, the cluster centers of the crisp input data must be found. To find the cluster centers first a random number is chosen as an estimate of the cluster center. Next, using a specified learning rate, the estimate is iteratively changed over a number of training epochs until it converges upon a certain number, which is the actual location of the cluster center. In practice the average of the input crisp invariant values can be used in place of the estimate to ensure proper convergence. The learning rate for similar applications, as found in [5] is expressed as follows:

\[ \alpha = e^{(-0.13n-0.69)} \]  

(3.22)

where \( \alpha \) is the training rate and \( n \) is the number of data sets in the class. After the training rate is found, the estimated cluster center is changed as follows [29]:

\[ w_i(N+1) = w_i(N) + \alpha[\rho - w_i(N)] \]  

(3.23)

where \( w_i(N) \) is the value of the cluster center after the \( N \)th iteration and \( \rho \) is the value of the crisp invariant value associated with the cluster center. This calculation is repeated for a specified number of training epochs on every crisp value that is associated with the cluster center. The number of epochs is usually chosen as a relatively large number such as 1000 to ensure convergence of the cluster center. Figure 3.2 shows an example of a set of points and their associated cluster center.
3.10 Class Membership Functions with LVQ scaling

The LVQ scheme first finds the location of the cluster center of the crisp data that is to be input into the class membership function, and then scales each fuzzy input by a measure of the Euclidean distance from the crisp input data to the associated cluster center as shown below:

\[ A_y = A_y e^{-\frac{|w_i - p_y|}{|w_i + p_y|}} \]  

(3.24)

where \( w_i \) is the location of the cluster center in the \( i \)th class, \( A_y \) is the \( j \)th fuzzy input data of the \( i \)th class, and \( p_y \) is the \( j \)th crisp input data in the \( i \)th class. As the distance between the cluster center \( w_i \) and input \( p_y \) increases, \( A_y \) approaches zero, thus reducing the contribution of data that is far from the cluster center of the class. After the data is scaled, the union operation is performed on the individual fuzzy membership functions. The result of forming the class membership function after applying LVQ to the set of
individual fuzzy membership functions can be seen in Figure 3.3 and it can be seen that the outlying data is downscaled more than the data around the average of the class.

![Figure 3.3: Simple Union (dashed) vs. LVQ Scaled Class Membership Functions (Solid)](image)

3.11 Adapted Rival Penalizing Learning Vector Quantization

A situation might also arise where the invariant values to be input into a class more appropriately define another class. This situation could prove problematic in distinguishing the proper recognition given the method described above as ambiguity would arise as to which class is the better match. However, this situation can also be addressed by using the cluster centers to scale data from other classes as shown below:

\[
A_y = A_y e^{- \frac{w_r - p_i}{2\sigma^2}} (1 - e^{\frac{w_r - p_i}{2\sigma^2}})
\]  

(3.25)

where \(w_r\) is the location of cluster center of a rival class. This method adapts the common LVQ clustering scheme to penalize the data of a set given the presence of rival
sets. Figure 3.4 shows two class membership functions formed using the simple union (a), LVQ scaled (b) and adapted rival penalizing LVQ scaled (c) methods.
Figure 3.4

(a): Simple Union Class Membership Functions
(b): LVQ Scaled Class Membership Functions
(c): Adapted Rival Penalizing LVQ Scaled Class Membership Functions

3.12 Training the Network

Figure 3.5 shows an example of the structure of the training routine employed in this study for the Nth invariant value.
The calculated invariants are used to find the cluster centers for each class and the Euclidean distance is calculated with respects to the cluster center of its own class and that of the rival classes. Using this measure, the invariant values are assigned an individual scaling factor, calculated by Equation 3.25 and fuzzified. The individually fuzzified invariant values are then combined into the Fuzzy Class Membership Functions and training is completed for the specific invariant value.

3.13 Recognition

After the class membership functions have been formed and the training process is completed, each invariant value for each class will have its own class membership function. During the recognition phase, the invariant values of the input disparity map are calculated and the degree of membership between each object class is calculated using a fuzzy inference method. Various inference methods were explored in this study to find the most appropriate method for recognition. Among these inference methods is membership based on fuzzy set intersection (Method 1), area of alpha-cut fuzzy sets.
(Method 2), and membership based on a single crisp input value (Method 3). The fuzzy set terminology expressions for these methods can be found below:

Membership based on fuzzy set intersection (Method 1):

\[ \mu_a = \bar{\nu} \left[ \mu_j(x) \land \mu_i(x) \right] = b \quad (3.26) \]

Membership based on alpha-cut (thresholded) fuzzy sets (Method 2):

\[ \mu_a = \bar{\nu} \left[ \mu_j(x) \lor \mu_j(x) \right] = b \quad (3.27) \]

Membership based on single, crisp input data (Method 3):

\[ \mu_a = \bar{\nu} \left[ \mu_j(x) \land I(x) \right] = b \quad (3.28) \]

The output of each fuzzy inference method, \( \mu_a \), is a single value, dictating the degree of membership between the input value and the class it was compared against. This inference is used to compare all invariant values to the appropriate class membership function and the final recognition index is calculated as follows:

\[ \mu_{\text{Final}} = \frac{1}{N} \sum_{i=1}^{N} \mu_i \quad (2.29) \]

where \( \mu_i \) is degree of membership between the \( i \)th invariant value and class membership function and \( N \) is the number of invariant values being used in the network. With this equation all invariant values are given an equal weight of importance, however it might be worth further study to examine the effects of uneven weighting of the discriminating features. These three methods were used on the same set of input data from Table 3.1 and the results can be found in Table 3.2. It can be seen that Method 1 misclassifies the object more frequently than Methods 2 and 3, and is therefore not a recommended choice.
and will eliminated from further discussion. Methods 2 and 3 are both similar in their accuracy of correct classification, even under different image conditions; however it should be noted that Method 3 quantified the degree of similarity between the object in question and the various classes in the inference network better than Method 2, as the value of recognition with Method 3 is represented by the class with the largest percentage, whereas that of Method 2 is represented by the class with the largest value, with this value belonging to a range with no specifically set limit.

<table>
<thead>
<tr>
<th>Set 1</th>
<th>Set 2</th>
<th>Set 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image 1" /></td>
<td><img src="image2.png" alt="Image 2" /></td>
<td><img src="image3.png" alt="Image 3" /></td>
</tr>
<tr>
<td><img src="image4.png" alt="Image 4" /></td>
<td><img src="image5.png" alt="Image 5" /></td>
<td><img src="image6.png" alt="Image 6" /></td>
</tr>
<tr>
<td><img src="image7.png" alt="Image 7" /></td>
<td><img src="image8.png" alt="Image 8" /></td>
<td><img src="image9.png" alt="Image 9" /></td>
</tr>
</tbody>
</table>

*Table 3.1: Trained Disparity Maps of each Class*
### Chapter 3: Object Recognition

#### Table 3.2: Recognition Indexes for each Inference Method

<table>
<thead>
<tr>
<th>Method 1</th>
<th>Method 2</th>
<th>Method 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fuzzy Set Intersection</td>
<td>Alpha-Cut Fuzzy Set</td>
<td>Value of Fuzzy Set at Crisp Input Point</td>
</tr>
<tr>
<td><img src="image1.png" alt="Image 1" /></td>
<td><img src="image2.png" alt="Image 2" /></td>
<td><img src="image3.png" alt="Image 3" /></td>
</tr>
<tr>
<td><img src="image4.png" alt="Image 4" /></td>
<td><img src="image5.png" alt="Image 5" /></td>
<td><img src="image6.png" alt="Image 6" /></td>
</tr>
<tr>
<td><img src="image7.png" alt="Image 7" /></td>
<td><img src="image8.png" alt="Image 8" /></td>
<td><img src="image9.png" alt="Image 9" /></td>
</tr>
</tbody>
</table>

| ![Input Data](input_data.png) | ![Input Data](input_data.png) | ![Input Data](input_data.png) |

*University of Windsor*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
CHAPTER 4: EXPERIMENTS AND RESULTS

This chapter contains an explanation of the experiments that were conducted in this study. This chapter also details the computing processes and steps taken while conducting the experiments.

4.1 Implementation

The steps taken in this experiment are displayed in the flow chart in Figure 4.1. The first step is the capture of the images from each camera. After the images are captured they are transformed into grayscale images in the preprocessing step. Next, the left and right preprocessed images are used to generate the disparity map, from which the invariant values are derived. If the data is to be trained, in the supervised training phase, the class of the calculated invariants must be entered. The invariants are stored as crisp values until the all of the desired images of the object are input and the training phase is completed. If there are still incoming images into the algorithm, the process repeats from the image capture step, otherwise the crisp invariant values of each class are used to calculate the associated cluster centers of the LVQ scheme. Using the LVQ cluster centers, each individual invariant value is fuzzified and scaled using the self and rival penalizing LVQ scheme. Finally, the individually scaled fuzzy invariant values are added to its class membership function.
Figure 4.1: Flow Chart of Training and Recognition Processes
4.2 Vision Platform

An image of the vision platform used in the experiments of this study can found in Figure 4.2. The components of the platform used in this study consists of two Prosilica 1350C single CCD colour cameras, used with Computar H6Z0812MP lenses which have mechanically controllable zoom, focus and iris. As seen in Figure 4.3, the cameras were positioned on the end-effector of a Mitsubishi RV-1A 6-axis robotic arm with a maximum reach radius of 418cm. The cameras were also placed on a pivot, controlled by Hi-Tec HS-645MG servo motors to allow the cameras to change their relative rotation, independent of the robotic arm. During the experiments, an object was placed inside the platform at a distance to robotic arm and the robotic arm was programmed to move at around the object at a fixed radius from the object while the cameras captured the left and right images which were used to generate the disparity map for that specific view.
Figure 4.2: Vision Platform
4.3 Experiment

The recognition scheme developed in Chapter 3 has been tested using the data found in Appendix D. Appendix D is comprised of 3 objects, of which multiple stereo images were taken from the cameras on the robotic arm found in Figure 4.3. These stereo images were used to generate the disparity maps found in Appendix E. A selected number of these disparity maps were used to generate the invariant values that were used while training the data of each class. After all of the data had been entered, the LVQ cluster centers were found and used to scale the fuzzified invariant values of each respective class (Equation 3.24). Finally the class membership functions were made for each invariant value of each class, completing the training phase. The system was tested using a number of different combinations of the disparity maps in Appendix E.

The disparity maps found in Table 4.1 were used during the training phase. To test the recognition phase of the system, all of the disparity maps of Appendix E were used as inputs to test the accuracy of the system. Table 4.2 shows the results of the recognition operation on the system with LVQ self-scaling under the different inputs.
## Table 4.1: Final Network Training Set

<table>
<thead>
<tr>
<th>Object #1</th>
<th>Object #2</th>
<th>Object #3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td><strong>Result</strong></td>
<td><strong>Input</strong></td>
</tr>
<tr>
<td>1.1</td>
<td>Correct</td>
<td>2.1</td>
</tr>
<tr>
<td>1.2</td>
<td>Correct</td>
<td>2.2</td>
</tr>
<tr>
<td>1.3</td>
<td>Correct</td>
<td>2.3</td>
</tr>
<tr>
<td>1.4</td>
<td>Correct</td>
<td>2.4</td>
</tr>
<tr>
<td>1.5</td>
<td><strong>Incorrect</strong></td>
<td>2.5</td>
</tr>
<tr>
<td>1.6</td>
<td>Correct</td>
<td>2.6</td>
</tr>
<tr>
<td>1.7</td>
<td>Correct</td>
<td>2.7</td>
</tr>
<tr>
<td>1.8</td>
<td>Correct</td>
<td>2.8</td>
</tr>
<tr>
<td>1.9</td>
<td>Correct</td>
<td>2.9</td>
</tr>
<tr>
<td>1.10</td>
<td>Correct</td>
<td>2.10</td>
</tr>
<tr>
<td>1.11</td>
<td>Correct</td>
<td>2.11</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
<td>------</td>
</tr>
<tr>
<td>1.12</td>
<td>Correct</td>
<td>2.12</td>
</tr>
<tr>
<td>1.13</td>
<td>Correct</td>
<td>2.13</td>
</tr>
<tr>
<td>1.14</td>
<td>Correct</td>
<td>2.14</td>
</tr>
<tr>
<td>1.15</td>
<td>Correct</td>
<td>2.15</td>
</tr>
<tr>
<td>1.16</td>
<td>Correct</td>
<td>2.16</td>
</tr>
<tr>
<td>1.17</td>
<td>Correct</td>
<td>2.17</td>
</tr>
<tr>
<td>1.18</td>
<td>Correct</td>
<td>2.18</td>
</tr>
<tr>
<td>1.19</td>
<td>Correct</td>
<td>2.19</td>
</tr>
<tr>
<td>1.20</td>
<td>Correct</td>
<td>2.20</td>
</tr>
<tr>
<td>1.21</td>
<td>Correct</td>
<td>2.21</td>
</tr>
<tr>
<td>1.22</td>
<td>Correct</td>
<td>2.22</td>
</tr>
<tr>
<td>1.23</td>
<td>Correct</td>
<td>2.23</td>
</tr>
<tr>
<td>1.24</td>
<td>Correct</td>
<td>2.24</td>
</tr>
<tr>
<td>1.25</td>
<td>Correct</td>
<td>2.25</td>
</tr>
<tr>
<td>1.26</td>
<td>Correct</td>
<td>2.26</td>
</tr>
<tr>
<td>1.27</td>
<td>Correct</td>
<td>2.27</td>
</tr>
<tr>
<td>1.28</td>
<td>Correct</td>
<td>2.28</td>
</tr>
<tr>
<td>1.29</td>
<td>Correct</td>
<td>2.29</td>
</tr>
<tr>
<td>1.30</td>
<td>Correct</td>
<td>2.30</td>
</tr>
<tr>
<td>1.31</td>
<td>Correct</td>
<td>2.31</td>
</tr>
<tr>
<td>1.32</td>
<td>Correct</td>
<td>2.32</td>
</tr>
<tr>
<td>1.33</td>
<td>Correct</td>
<td>2.33</td>
</tr>
<tr>
<td>1.34</td>
<td>Correct</td>
<td>2.34</td>
</tr>
<tr>
<td>1.35</td>
<td>Correct</td>
<td>2.35</td>
</tr>
<tr>
<td>1.36</td>
<td>Correct</td>
<td>2.36</td>
</tr>
<tr>
<td>1.37</td>
<td>Correct</td>
<td>2.37</td>
</tr>
<tr>
<td>1.38</td>
<td>Correct</td>
<td>2.38</td>
</tr>
<tr>
<td>1.39</td>
<td>Correct</td>
<td>2.39</td>
</tr>
<tr>
<td>1.40</td>
<td>Correct</td>
<td>2.40</td>
</tr>
<tr>
<td>1.41</td>
<td>Correct</td>
<td>2.41</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
<td>------</td>
</tr>
<tr>
<td>1.42</td>
<td>Correct</td>
<td>2.42</td>
</tr>
<tr>
<td>1.43</td>
<td>Correct</td>
<td>2.43</td>
</tr>
<tr>
<td>1.44</td>
<td>Correct</td>
<td>2.44</td>
</tr>
<tr>
<td>1.45</td>
<td>Correct</td>
<td>2.45</td>
</tr>
<tr>
<td>1.46</td>
<td>Correct</td>
<td>2.46</td>
</tr>
<tr>
<td>1.47</td>
<td>Correct</td>
<td>2.47</td>
</tr>
<tr>
<td>1.48</td>
<td>Correct</td>
<td>2.48</td>
</tr>
<tr>
<td>1.49</td>
<td>Correct</td>
<td>2.49</td>
</tr>
<tr>
<td>1.50</td>
<td>Correct</td>
<td>2.50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.51</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.52</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.53</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.54</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.55</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.56</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.57</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.58</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.59</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.60</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.61</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.62</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.63</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.65</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.66</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.67</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.68</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.69</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.70</td>
</tr>
</tbody>
</table>

University of Windsor

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Table 4.2: Recognition Results

(Numbers Correspond to Disparity Maps in Appendix E)
4.4 Explanation

The results in Chapter 4.3 show that even with a small training set, the system is adaptive enough to recognize a large variety of images. It is important to note that the selection of the training set is very important in that it must define the foundation of the different possibilities of which the images in each class are derivative. For the Object#1 (camera), this foundation can be defined by disparity maps taken from the front, left side and back of the object (images 1.1, 1.21, 1.32 from Appendix E). For Object #2 (toy car), the foundation can be defined by defined by disparity maps from the front, left and right sides (images 2.11, 2.33, 2.81 from Appendix E). For Object #3 (toy truck), the foundation can be defined by front, back and 40° from the left side disparity maps (images 3.1, 3.12, 3.49 from Appendix E). The reason for selecting these images was to test the computer aided drafting convention that three views are usually enough to understand the shape of an object.

<table>
<thead>
<tr>
<th>Test #1</th>
<th>Training Set</th>
<th>Class #1</th>
<th>Class #2</th>
<th>Class #3</th>
</tr>
</thead>
<tbody>
<tr>
<td>(No data scaling)</td>
<td>(1.1, 1.21, 1.32)</td>
<td>94%</td>
<td>93.81%</td>
<td>86.67%</td>
</tr>
<tr>
<td></td>
<td>(2.11, 2.32, 2.81)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(3.1, 3.12, 3.49)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test #2</td>
<td>(LVQ self-scaling only)</td>
<td>98%</td>
<td>98.97%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>(1.1, 1.21, 1.32)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(2.11, 2.32, 2.81)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(3.1, 3.12, 3.49)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test #3</td>
<td>(LVQ self-scaling with Image Normalization)</td>
<td>100%</td>
<td>68%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>(1.1, 1.21, 1.32)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(2.11, 2.32, 2.81)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(3.1, 3.12, 3.49)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test #4</td>
<td>(LVQ Self and Rival Scaling)</td>
<td>100%</td>
<td>72%</td>
<td>90%</td>
</tr>
<tr>
<td></td>
<td>(1.1, 1.21, 1.32)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(2.11, 2.32, 2.81)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(3.1, 3.12, 3.49)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test #5</td>
<td>(LVQ self-scaling only)</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>(1.1, 1.21, 1.32, 1.44)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(2.11, 2.32, 2.50, 2.81)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(3.1, 3.12, 3.39, 3.49)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.3: Recognition under Different Sized Training Sets
(Numbers Correspond to Disparity Maps in Appendix E)
4.4.1 Test #1: No Data Scaling

The first test that was conducted used the training set of Table 4.1 in a system which did not conduct any type of data scaling. This test shows that the disparity maps for Class #1 were properly identified with an accuracy of 94%, while those of Class #2 were properly identified with an accuracy of 93.81% and those of Class #3 were identified with an accuracy of 86.67%. The misclassifications of Class #1 and Class #3 were mostly erroneously recognized as belonging to Class #2, while those of Class #2 were erroneously recognized as mostly belonging to Class #3.

4.4.2 Test #2: LVQ Self-Scaling Only (3 Views)

Detailed results of the second test can be found in Table 4.2. This test shows that the training set of Table 4.1 is good enough to recognize objects in Class #1 with an accuracy of 98%, objects in Class #2 with an accuracy of 98.97%, and objects in Class #3 with an accuracy of 100%. Different sized training sets and different training styles were also tested on in this recognition system with results found in Table 4.3.

4.4.3 Test #3: LVQ Self-Scaling with Image Normalization

The third test was conducted to see if image normalization offered any improvements in the recognition of the maps. From Table 4.3 it is easy to recognize that applying the image normalization scheme detailed in Appendix A leads to a large number of misclassifications on the disparity maps in Class #2. The reason for image normalization not increasing the accuracy of the recognition system could be that the different image distortions which image normalizations can handle are already collectively handled by other invariant values. The reason why image normalization leads to less accurate recognition could be due to the false assumption of this scheme that the images of the toy car are really skewed images of the toy truck.

4.4.4 Test #4: LVQ Self and Rival Scaling

The fourth test was conducted by applying LVQ self and rival scaling. This test showed lower accuracy than that of tests 1 and 2 for Class #2, the toy car and Class #3. This lower accuracy is due to the fact that the variances in the shape of the toy car at different
angles lead to a wide range in the values of the compactness invariant value for the class. This range confuses the network during the determination of the cluster centers and the value for the invariant is not only scaled down during self-scaling, but it is also scaled down again during the rival-scaling. A conclusion about this type of data scaling is that it is only appropriate for sets of data which are relatively consistent.

4.4.5 Test #5: LVQ Self-Scaling Only (4 Views)
The last test that was conducted was to use front, left, right and back disparity maps for the objects, again using only LVQ self-scaling. This test showed an improvement over the second test and saw that every disparity map was properly classified as belonging to the right object.
CHAPTER 5: CONCLUSIONS

This chapter outlines the conclusions of this study based on the experiments conducted in Chapter 4. Potential future work is also outlined based on the assumptions and simplifications taken during the experiments as well as possible extensions of this study.

5.1 Conclusions

This paper examines various different invariant values and inference methods for use in recognition using disparity maps and proposes a combined approach of these methods. The invariant values which involve higher order exponents are ill-suited for this type of recognition, as the values are very sensitive to noise and disparity maps are inherently noisy. The most suited invariant values that were examined in this study are histogram difference (Equation 3.4), compactness (Equation 3.2) and low-order moments (Equation 3.11).

LVQ and a modified LVQ scheme were both examined for their effectiveness on the adaptive scaling of data distributions and it was found that the self-scaling LVQ scheme works best on competitive recognition systems due to the ability to scale data based on a wide range of training data of a specific class. The most suitable inference method using this LVQ self-scaling fuzzy neural network is to find the value of the class membership function at the value of a crisp input as it better recognizes and quantifies the degree of recognition than the other methods explored in this paper. The final proposed method has shown the ability to reliably recognize 3D data of objects from disparity maps in a variety of views based on a limited number of trained views.

Through the experiments in Chapter 4, it was found that it is possible to correctly identify the shape of an object based on the knowledge of a limited set of viewpoints. With knowledge of a front, back, left and right view, it was shown that all viewpoints in between can be properly identified with a high accuracy rate, even when compared to that of other objects. It was also shown that applying the image normalization scheme to the
Chapter 5: Conclusions

disparity maps in Appendix E leads to increased false recognition as the skewing compensation of this scheme can confuse the network.

5.2 Future Work

In this study the objects extracted from an image using simple background subtraction. The drawback of this method is that pixels with intensities that are similar to the background are eliminated from the extracted object, resulting in areas on the object with large holes. A possible extension of this study could be to include an object segmentation scheme that separates an object from a noisy background using a more accurate method that is not solely dependent on an intensity comparison of two images. Another simplification that was used in this study was that the stereo camera setup was constructed to simplify the epipolar geometry of the stereo system by forcing all epipolar lines to be parallel to the image y-axis. This study can be made more robust by including a stereo image rectification scheme and allow for stereo setups that are not as physically constrained. The invariant value based inference system in this study was conducted by granting each invariant value an equal importance on the recognition result. As stated earlier, it might be worth while to examine the effects of a similar inference system under selectively different weighting for the importance of each invariant value.
APPENDIX A

Image Normalization

The following steps in this image normalization process were provided by [19].

First calculate the image centroid as in Equations 3.7 and 3.8. Next, calculate the second order \((p + q = 2)\) central moments as in Equation 3.9. Using the second order moments, form the covariance matrix, \(M\), is formed as seen below:

\[
M = \begin{bmatrix}
\mu_{20} & \mu_{11} \\
\mu_{11} & \mu_{02}
\end{bmatrix}
\] (A.1)

Covariance matrix \(M\) is formed such that the covariance matrix of a transformed image, \(M'\), has the following relation to the covariance matrix of an untransformed image:

\[
M' = A M A^T
\] (A.2)

Next the eigenvalues of the covariance matrix are calculated. After finding the eigenvalues, the eigenvectors of \(M\) are calculated and the coordinates of the pixels are rotated parallel to the eigenvectors. The eigenvalues of \(M\) are calculated as follows:

\[
\begin{vmatrix}
\mu_{20} - \lambda_1 & \mu_{11} \\
\mu_{11} & \mu_{02} - \lambda_2
\end{vmatrix} = 0
\] (A.3)

Solving Equation A.3 for the eigenvalues \(\lambda_1\) and \(\lambda_2\):

\[
\lambda_{1,2} = \frac{\mu_{20} + \mu_{02} \pm \sqrt{(\mu_{20} - \mu_{02})^2 + 4 \mu_{11}^2}}{2}
\] (A.4)
After finding the eigenvalues, the eigenvectors of $M$ are found such that $e_1 = [e_{1x} \ e_{1y}]^T$ and $e_2 = [e_{2x} \ e_{2y}]^T$ corresponding to eigenvalues $\lambda_1$ and $\lambda_2$, respectively. These eigenvectors are found from the following relationship:

$$
\begin{bmatrix}
\mu_{20} - \lambda_1 & \mu_{11} \\
\mu_{11} & \mu_{02} - \lambda_1
\end{bmatrix}
\begin{bmatrix}
e_{ax} \\
e_{ay}
\end{bmatrix} = 0
$$

(A.5)

Solving Equation A.5 for $e_{ax}$ and $e_{ay}$:

$$
\begin{bmatrix}
e_{ax} \\
e_{ay}
\end{bmatrix} =
\begin{bmatrix}
\frac{\mu_{11}}{\sqrt{(\lambda_1 - \mu_{20})^2 + \mu_{11}^2}} \\
\frac{\lambda_1 - \mu_{20}}{\sqrt{(\lambda_1 - \mu_{20})^2 + \mu_{11}^2}}
\end{bmatrix}
$$

(A.6)

The rotational matrix which is to be applied to the pixel coordinates is formed as follows:

$$
E =
\begin{bmatrix}
e_{1x} & e_{1y} \\
- e_{2y} & e_{2x}
\end{bmatrix}
$$

(A.7)

Since $M$ is symmetric, the eigenvectors are orthonormal to each other and the cross-product of the elements of $E$ are equal to zero, leading to two possible solutions where either $e_{2x} = e_{1y}$ and $e_{1x} = -e_{2y}$ or $e_{2x} = -e_{1y}$ and $e_{1x} = e_{2y}$. Therefore the rotational matrix $E$ can be simplified into the following:

$$
E =
\begin{bmatrix}
e_{1x} & e_{1y} \\
- e_{1y} & e_{1x}
\end{bmatrix}
$$

(A.8)

Using the image centroid and the rotation matrix $E$ the translation and skewing of the image are compensated. The next step is to compensate for the scaling of the image. This is done through a scaling matrix, $W$, which is formed using the eigenvalues of $M$ found in Equation A.4. $W$ is formed as follows:
The last step is to compensate for the rotation of the image. As conducted in [19], the rotation of the image is found through a conditional check of the second-order central moments as found in Table A.1.

<table>
<thead>
<tr>
<th>$\mu_{20} - \mu_{02}$</th>
<th>$\mu_{11}$</th>
<th>$\phi$ ( \left( \xi = \frac{2\mu_{11}}{\mu_{20} - \mu_{02}} \right) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td>Zero</td>
<td>0</td>
</tr>
<tr>
<td>Zero</td>
<td>Positive</td>
<td>+45°</td>
</tr>
<tr>
<td>Zero</td>
<td>Negative</td>
<td>-45°</td>
</tr>
<tr>
<td>Positive</td>
<td>Zero</td>
<td>0</td>
</tr>
<tr>
<td>Negative</td>
<td>Zero</td>
<td>-90°</td>
</tr>
<tr>
<td>Positive</td>
<td>Positive</td>
<td>( \left( \frac{1}{2} \tan^{-1} \xi \right) )</td>
</tr>
<tr>
<td>Positive</td>
<td>Negative</td>
<td>( \left( \frac{1}{2} \tan^{-1} \xi \right) )</td>
</tr>
<tr>
<td>Negative</td>
<td>Positive</td>
<td>( \left( \frac{1}{2} \tan^{-1} \xi + 90° \right) )</td>
</tr>
<tr>
<td>Negative</td>
<td>Negative</td>
<td>( \left( \frac{1}{2} \tan^{-1} \xi - 90° \right) )</td>
</tr>
</tbody>
</table>

*Table A.1: Image Normalization Angle of Rotation*

After the angle $\phi$ has been found it is used to form a simple 2D rotational matrix as follows:

$$
R = \begin{bmatrix}
\cos \phi & \sin \phi \\
-\sin \phi & \cos \phi
\end{bmatrix}
$$

(A.10)

The final image normalization scheme is summarized by the following equation:

$$
W = \begin{bmatrix}
\frac{\sqrt{\lambda_2}}{\lambda_1} & 0 \\
0 & \frac{\sqrt{\lambda_1}}{\lambda_2}
\end{bmatrix}
$$

(A.9)
\[ \tilde{X} = R W E (X - \tilde{X}) \]  

(A.11)

\[
\begin{bmatrix}
\tilde{x} \\
\tilde{y}
\end{bmatrix} =
\begin{bmatrix}
\cos \phi & \sin \phi \\
-\sin \phi & \cos \phi
\end{bmatrix}
\begin{bmatrix}
\frac{\lambda_2 - \lambda_1}{\sqrt{\lambda_1}} & 0 \\
0 & \frac{\lambda_2 - \lambda_1}{\sqrt{\lambda_2}}
\end{bmatrix}
\begin{bmatrix}
e_{1x} & e_{1y} \\
e_{1y} & e_{1x}
\end{bmatrix}
\begin{bmatrix}
x - \tilde{x} \\
y - \tilde{y}
\end{bmatrix}
\]  

(A.12)

where \((\tilde{x}, \tilde{y})\) is the new coordinates of the pixel at location \((x, y)\) of the original image.
APPENDIX B


Following is an explanation of the program used to control the robotic arm, mechanical lenses, camera mounted servo motors and CCD cameras of the vision platform during the image capture in the experiments of Chapter 4. This GUI controls all of the equipment of the vision platform of Figure 4.2 during the experiments of this study, but can be modified to use the equipment in a different manner.

Figure B.1 shows the GUI for the Stereo Image Capture program StereoCapture.exe. The steps taken in operating this program are listed below in order of operation.

1. Operating Camera Mounted Servo Motors

To turn on the camera mounted servo motors first turn on the switch for the servo controller. This will activate the green LED light on the servo controller. Next, press the “Servos” toggle switch on the GUI and the message “Initializing servo positions…” will appear in the Status box of the GUI as seen in Figure B.2. Once this message appears turn on the servo motor power supply and the servo motors on the platform will initialize to their starting positions.
Once the servo motors have reached their starting positions, the Servos toggle switch will move to the “On” position and the Status box of the GUI will be updated to indicate that the initialization of the servo motors is done, as seen in Figure B.3.

2. Operating RV-1A Robotic Arm

After the servo motors have been turned on the next step is to activate the robotic arm. Before toggling the “Robot” On/Off switch in the GUI, turn on the CR1-571 Robotic Arm Controller. If an error is encountered by the controller, record the error number and refer to the MELFA BFP-A5993-K Troubleshooting manual. To clear the error status, press the RESET button on the CR1-571 controller or the ERROR RESET button on the R28TB Teaching Box seen in Figure B.4.
To manually move the robotic arm, first wait until the Teaching Box displays the message "ANY KEY DOWN," then insert the key into the provided slot on the Teaching Box and switch the operations mode to ENABLE. Next, while holding the "Deadman" trigger on the back of the Teaching Box, begin to hold the STEP MOVE button on the face of the Teaching Box until you hear a clicking on the robot accompanied by a high frequency sound. This will turn on the servo motors in the joints of the robotic arm as long as the Deadman Trigger is held. To move the arm by changing the angle of each joint, hold the Deadman Trigger and the STEP MOVE button and press the JOINT button, or to move the arm using Cartesian coordinates press the XYZ button. The mode currently in use will be displayed in the screen of the Teaching Box. Next, while still holding the Deadman Trigger and the STEP MOVE button, press and hold the appropriate button for the desired movement (i.e. To move the J1 clockwise, hold the "-X (J1)" button). To increase or decrease the speed of the servo motors press the +FORWD or the -BACKWD buttons and the speed will be increased or decreased by an interval of 10, respectively, with 100 being the maximum speed*. When finished moving the robotic arm, use the key to select DISABLE mode and retrieve the key from the Teaching Box.

*CAUTION: Using the equipment as seen in Figure 4.3, it is NOT recommended to operate with a servo speed above 30.
Starting a Robotic Arm Program

Insert the key into the provided slot on the Mitsubishi MELFA CR1-571 controller (seen in Figure B.5) and switch the operation mode to AUTO (Op.). In this operations mode the robotic arm will be able to run using programs stored on the controller without the use of the Teaching Box. Press the CHNG DISP button on the controller until to view the current servo speed (o. XXX), program (P.XXXX) and line of the current program (XXXXX) on the STATUS NUMBER display of the controller. To change the current servo speed or load a different program, press the UP or DOWN buttons while viewing the current settings. Once the desired speed and program are loaded, press the SVO ON button to turn on the servo motors of the robotic arm. Once the servo motors are on, press the START button to begin the loaded program. The default setup for execution of a program is for the robot to run in a continuous loop. In order to ensure that the program only executes once, press the END button shortly after the robot begins its program and the LED light for the END button will flash to indicate that the robot will stop after executing the current program.

![Figure B.5: Mitsubishi MELFA CR1-571 Controller](image)

NOTE: To stop the robot during the execution of a program, press the STOP button on the controller and the robot will immediately halt all movement. While stopped it is
possible to turn off the servos of the robotic arm by pressing the SVO OFF button or reset the current program by pressing the RESET button. To start the robot again, ensure that the servos are turned on and press the START button and the robot will continue executing the loaded program from where it stopped, unless the program was reset, in which case it will maneuver itself into position and start the program from the first command.

After the robotic arm has started its program and reached the first waiting position, press the “Robot” toggle switch in the GUI to activate the communication of the software program and the robot controller as seen in Figure B.6.

![Figure B.6: Robot Communication Activated](image)

3. Activating the Mechanical Lenses
After the servo motors and the robot communication are activated, the mechanical lenses can be activated. To activate the mechanical lenses, first ensure that the V1LC lens controllers, as seen in Figure B.7, are connected to power by checking if the green LED on each controller is turned on.
Once the lens controllers are powered, press the “Lenses” toggle switch in the GUI to start the initialization of the lenses. When initializing, the motors in the lens will be set to the positions defined in the file \texttt{vp\_capture.c}. Once the lenses have finished initializing, the Lenses toggle switch will indicate that the lenses are On and the GUI Status box will display the message “Initializing lens settings...done.” as seen in Figure B.8.

4. **Calibrating the Image Capture**

After the robot, servos and lenses are all activated, the Calibrate and Start Capture buttons will become available. The Calibrate process of this setup is the capture of images of the background that will surround the target object during the actual stereo image capture so that the object can be easily extracted from the images; therefore, first
set up the necessary supports for the object, then press the Calibrate button of the GUI as seen in Figure B.8. This will start the robot and image capture programs. During the calibration process the captured images will be saved to the folder BGImages of the same directory in which the GUI program is present.

5. Capturing the Stereo Images
After the background images have been captured in the Calibration process the stereo image capture process can be started. Place the target object on its supports in the platform and press the Start Capture button in the GUI, as seen in Figure B.8, to start the robot and image capture programs. During the image capture two image windows will pop up to display the left and right camera images after background subtraction. The background subtraction scheme works by comparing the background image, captured in the Calibration process, to the image captured in the Start Capture process. The intensities of the two images are compared pixel by pixel, and if the difference is under a certain threshold value, the pixel is set to black. The threshold of the background subtraction scheme can be set from the file vp_capture.c. The resulting images are stored in the folder ObImages of the same directory in which the GUI program is present.
APPENDIX C

Object Recognition Program: Users Manual

The object recognition program used in this study is compiled in the executable file 3DRecognition.exe.

To start generate the disparity map, first open the left image using the Open Image command from the file tab in the main menu of the GUI, as in Figure C.1.

Using this command will open an image in the left image canvas of the main window. Next, open the right image using the Open Image 2 command from the File menu. This will open an image in the right image canvas of the main form. Once the left and right images are loaded into their associated canvases, select the Generate Exhaustive Disparity Map command from the 3D Reconstruction tab in the main menu off the GUI, as in Figure C.2.
After the disparity map is generated, it will be displayed in the left image canvas as seen in Figure C.3.

To save this image, select the Save Image option from the File tab in the main menu. Saving the disparity maps may make the training process easier and faster for later sessions.

To start the training process, first choose the Initialize command from the 3D Training tab in the main menu as seen in Figure C.4.
Once the network is initialized disparity maps can be entered into a desired class of the network. Ensure that a disparity map is generated or loaded into the left image canvas and choose the Input Image option from the 3D Training tab. Once this option is selected, the Object Class window will pop up and allow the appropriate class to be entered, as seen in Figure C.5.

After the class is chosen, press the OK button and the Object Trainees window will pop up, displaying the current trainees of the class that the image was assigned to, as seen in Figure C.6.
There are a maximum of three classes in this program, with a limit of eight images able to be trained into each class. After the images have been assigned to each class, select the either the Train LVQ Self Scaling Network or Train LVQ Self and Rival Scaling option from the 3D Training tab. Once the method has been selected the LVQ Epochs window will appear and allow the epochs (number of iterations) of the training cycle to be entered, as in Figure C.7.

Once the desired number is entered, press the OK button to complete the training process.

Now that the training process is completed it is possible to use the conduct the object recognition process. Before starting the object recognition ensure that the network is properly trained. Also, ensure that a disparity map is present in the left image canvas of the main window. Finally, select the Recognition command from the 3D Training tab to conduct the recognition process on the disparity map in the left image canvas. Once
the process is completed the Recognition window will pop up displaying the first disparity map of the recognized class, as seen in Figure C.8.

Figure C.8
APPENDIX D

The images in this appendix are the preprocessed images that were captured using the stereo vision platform in Figure 4.1. The images were captured at a fixed radius from the object in roughly 20° intervals using the robot program The2.prg. The images been processed with background subtraction and have been converted from RGB colour to grayscale using Equation 3.1.

Object #1 – Camera
Object #2 – Toy Car
Object # 3 - Toy Truck
APPENDIX E

This appendix contains the disparity maps that were generated using the stereo images of Appendix E and used in the experiments of Chapter 4.

Object #1 – Camera
Object #2 – Toy Car
Object #3 – Toy Truck

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
BIBLIOGRAPHY


VITA AUCTORIS

Ahmad Shawky was born on May 19, 1982 in Windsor, Ontario. He graduated from Vincent Massey Secondary School in 2001. After high school Ahmad enrolled in the engineering program at the University of Windsor and graduated with a Bachelor of Science in the Honors Electrical Engineering program with distinction in 2005. He is currently a candidate in the Masters of Applied Science program at the University of Windsor and hopes to graduate in 2007.