1997

Three-dimensional object recognition

Kehang Chen
Iowa State University

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Three-dimensional object recognition

by

Kehang Chen

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Mechanical Engineering

Major Professor: Jerry Lee Hall

Iowa State University

Ames, Iowa

1997

Graduate College
Iowa State University

This is to certify that the Doctoral dissertation of

Kehang Chen

has met the dissertation requirements of Iowa State University

Signature was redacted for privacy.

Major Professor

Signature was redacted for privacy.

For the Major Program

Signature was redacted for privacy.

For the Graduate College

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

Many of the real life applications of object pattern recognition aim to use robots to perform human-like tasks. In many tasks, robots are used to replace the human-being to do tedious, heavily repeated, and hazardous jobs. Robotic part assembly, automatic feature measurement in quality control, mail sorting, fingerprint matching, deep sea searching and target detection/identification are some examples of such applications.

In the early 1970's, a robot equipped with a digital TV camera and a microcomputer [1] was successfully developed at Stanford University to solve the "Instant Insanity" puzzle. This motivated the potential uses of object pattern recognition systems in robotic applications.

Today, object pattern recognition in the process of robotic assembly is one of the important applications of the manufacturing industry. Since an object pattern recognition system can help to recognize and locate objects in random sequences and positions, a robot coupled with this kind of system provides flexibility for part assembly processes and reduces the required fixturing designs in an assembly line. In contrast, a traditional assembly line must be equipped with some special fixtures to position and hold parts. The parts being assembled must be marked or identified with some features so that a special sensor can distinquish them. The advantage of using robotic assembly lines is limited due to rapid changes in product design imposed by highly competitive global markets. Many manufacturers require systems that can quickly and easily adapt to new product designs and market changes. Also, increasing demands being placed on robots require that they be capable of performing multiple functions such as assembly and inspection with the same robotic arm. Thus, the technology associated with a traditional assembly line must expand in its scope. One solution to these problems is to apply an object pattern recognition system to a robotic assembly line.

In order to make judgments regarding part feature identification in assembly lines, it is necessary to provide intelligence to the robots. Robotic systems aided with sensing tools, such as vision systems, provide such desired intelligence into these assembly lines. Apart from flexibility, an assembly system needs to satisfy another important criteria of most manufacturing industries - speed.

Many robotic systems have been developed to achieve the goals of intelligence and speed. Some have object recognition systems devoted to handle two-dimensional objects. These systems are usually designed for the applications of circuit board automatic manufacturing [2, 3, 4, 5, 6]. Other systems are designed to identify three-dimensional (3D) objects [7, 8, 9, 10, 11, 12]. But the applications of assembly lines integrated with three-dimensional object recognition systems are limited in intelligence or speed due to the lack of efficient representation of 3D objects. This is because the current state of 3D object recognition systems requires large databases and exhaustive searches for matching their patterns. The execution time increases rapidly if an object is very complicated or the number of different objects to be recognized or identified is large.

In the late 1980's, the integration of Computer-Aided Design and Computer-Aided Manufacturing (CAD/CAM) became the hot issue in the manufacturing industry. By means of integration, CAD models are directly used by CAM systems to produce parameters of correlated operations, such as material selection, planning, production, inspection, management, and marketing of the products. The advantage of using CAD/CAM integration can reduce the manufacturing cycle and increase product quality. Thus, it is the current trend to integrate CAD systems with automated assembly lines.

1.1 Focus of This Study

1.1.1 Research goals

Few attempts have been made to provide three-dimensional object recognition systems for automated assembly lines. To date little effort has been made to integrate with any CAD systems to 3D object recognition or to have used the CAD models to extract example features

of parts. Furthermore, all the attempts require large database structures to store sophisticated example features and time-consuming search mechanisms to find matches between example features and derived features. Very little effort has been devoted to find simple and representative features that are suitable for neural network classifiers to recognize three-dimensional objects. Therefore, the overall objective of this research is to build a three-dimensional object recognition system with use of a neural network that is applicable to automated assembly lines based on the needs of speed, flexibility, and CAD/CAM integration. The major specific goals of this study are:

1. To develop methodologies to derive compact and representative features for a neural network classifier to recognize three-dimensional objects.

2. To investigate the inter-relationships between derived features of objects (features vector robustness)by studying the correlations of the part with the output values from a neural network classifier.

3. To implement a prototype system based on the proposed methodologies and to evaluate the system performance by the criteria of classification rates and processing speed.

1.1.2 Organization of this dissertation

This dissertation consists of eight chapters. The second chapter starts with the outline of the necessary components of an object pattern recognition system and then presents a literature review. The first section focuses on three areas: (1) image processing methods employed in recognition systems to obtain clean images or to enhance recognition features; (2) feature extraction techniques to derive compact representative features; and (3) classifiers for identifying derived features. The last section briefly describes different existing systems with emphasis on the recognition systems coupling with neural network classifiers.

In Chapter 3, several neural network algorithms used in object pattern recognition are discussed. The comparisons between these networks are summarized in Section 3.2 based on their properties. Chapter 4 presents the motivation of choosing the Radon transform technique

to extract features for recognition as well as its mathematical properties and its suitability of coupling with artificial neural networks.

In the first three sections of Chapter 5, three algorithms are proposed based on use of the properties of the Radon transform to extract recognition or training features. The forth section provides a mathematical comparison of these algorithms by using the developed probability function for getting a specific vector from different images. Detailed information about the system implementation along with the evaluation criteria of the system performance are given in Chapter 6.

Chapter 7 demonstrates the experimental results of the recognition systems using the proposed algorithms. The comparisons based on the experimental data are also outlined in this chapter. Chapter 8 presents the conclusions and the recommendation for future investigation.

2 OBJECT PATTERN RECOGNITION

2.1 Recognition System

In general, the complete process of an object pattern recognition system may be formulated as follows: obtain an image from a sensor (typically a television-type camera), process the image and extract the features, then compare these features with some pre-defined standard to determine if these features satisfy some given conditions or specifications. No matter how simple or complex the system is, an object pattern recognition system essentially consists of image acquisition, image processing, feature extraction, and classification, as described by Figure 2.1.

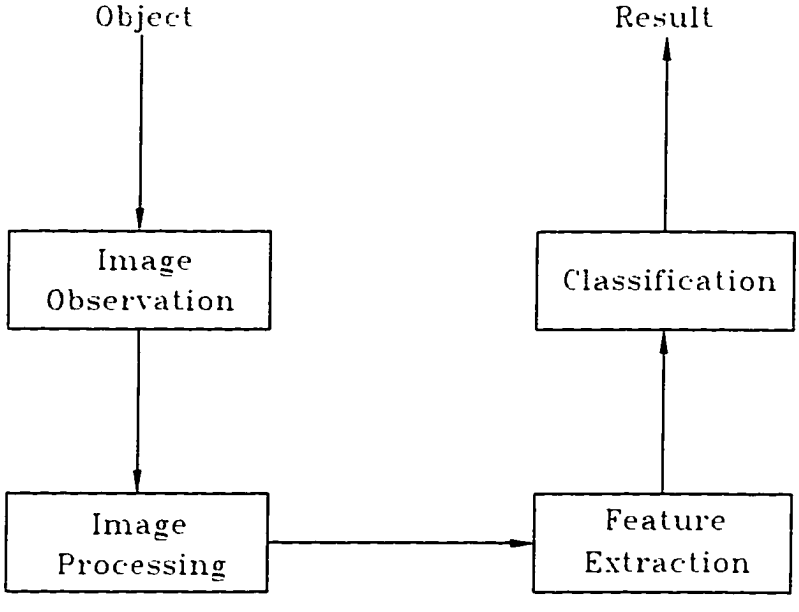


Figure 2.1 The complete process of an object pattern recognition system

The process of image acquisition can be described as a camera acquiring light energy from a scene and converting the energy to a set of data that a computer can understand. The light energy representing an image can be obtained in two ways: sequentially scanning a scene with tube type cameras (i.e, rostering) or reading the output of multiple detectors of an array sensor such as those used in solid state cameras. The detailed technique of how a camera obtains the image is beyond the scope of this research, and the reader can refer to [17] and [31] for more information about this issue.

Image processing in an object recognition system essentially improves the quality of images from the cameras so that the desired features can be easily detected in the feature extraction stage. Image processing involves image enhancement and segmentation as well as other necessary operations to obtain an "undistorted" or "clear" image of the object to be recognized. The clear image will then be further processed to extract the application-dependent features. Based on these features, a specific classifier is used to compare images with a number of reference objects and decide whether this object exhibits a sufficient similarity to one of the reference objects to allow identification. The following three subsections discuss the general algorithms and methods used in image processing, feature extraction, and classification.

2.1.1 Image processing

Image processing serves many purposes in real world applications, and the objective is dependent on the application. In general, image processing can be described as the process of producing, based on some criteria, desirable images from the original images. From a manufacturing point of view, the objective may be to increase the part identification rate for robotic assembly or the feature measuring accuracy for quality control.

Image processing encompasses several basic tasks: *image enhancement*, *image restoration*, *image reconstruction*, *image compression*, and *image analysis*. *Image enhancement* modifies an image to improve its quality so that the result is more suitable than the original image for a specific application. *Image restoration* corrects the degradation in the image, such as blurring and/or distorting effects introduced in the image forming process. *Image reconstruction* deals

with reconstructing the image from a set of one-dimensional projections of the original image. The Radon transform is one of the techniques used for the purpose of 3D object or image reconstruction [13, 14, 15, 16]. It is used in this research for extracting features from images. The mathematical representation of the Radon transform and the feasibility of using the features obtained from it for object pattern recognition will be discussed in Chapters 4 and 5, respectively. *Image compression* reduces the image data storage or transmission requirements based on information redundancy, while maintaining acceptable image quality. *Image analysis* focuses on the region of interest in images to find information in a form compatible with later feature extraction processes.

One of the objectives of this research is to reduce the cost of the required equipment for a recognition system. Thus, only one inexpensive camera with low resolution is proposed in the implementation. Of significant concern is the quality of the image obtained from the camera. Because of this low quality image, the techniques of image enhancement are involved in the implementation of object recognition in this study. The camera is at a stationary position while it is acquiring an image. It is assumed that it is located far enough from the scene to effectively capture the image of the subject.

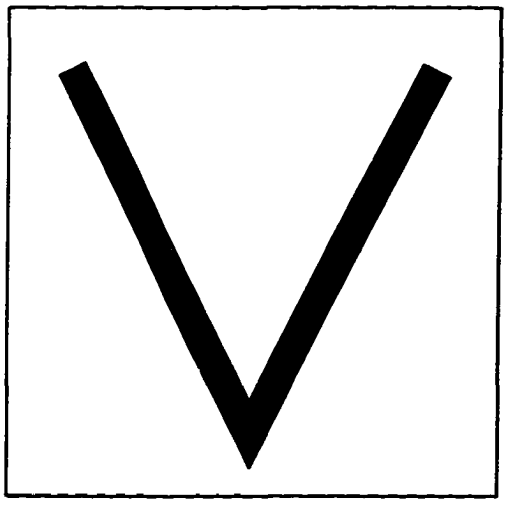
The blurring and/or distorting effects introduced in the image forming process are beyond the scope of this study and therefore not included herein. Thus, it was not necessary to perform image restoration in this recognition system. Image compression is considered by communication engineers to reduce the extremely long time of transmission and requirements for extensive storage. Image compression is usually applied for satellite and television image transmission. The system developed in this research requires only one image at each process and does not need long time or long distance transmission of the image signal. Thus, image compression was not utilized in the implementation of this work. In addition, the more information about an object inherit in an image, the more accurate the recognition rate the system will have. The following sub-sections summarize some useful techniques in image enhancement and analysis.

2.1.1.1 Image enhancement techniques

Intuitively, an image is a picture, photograph, or other form giving a visual representation of an object or scene. However, an image in digital image processing is represented by a two dimensional array of numerical values. These numerical values define the appearances of the corresponding pixels in an image (see Figure 2.2). Pixel is the short name for picture element. An image can be mathematically represented by an image function

$$f(x, y), \tag{2.1}$$

where f is the intensity level of a pixel in the (x, y) location, and x and y are spatial coordinates related to the physical locations of pixels in an image. In practice, the function f is related to the light energy distribution of the scene being imaged.



(a)

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	
0	1	1	0	0	0	0	0	0	0	0	0	1	1	0	
0	0	1	1	0	0	0	0	0	0	0	1	1	0	0	
0	0	1	1	0	0	0	0	0	0	0	1	1	0	0	
0	0	0	1	1	0	0	0	0	0	1	1	0	0	0	
0	0	0	1	1	0	0	0	0	0	1	1	0	0	0	
0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0
0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0
0	0	0	0	0	1	1	0	0	1	1	0	0	0	0	0
0	0	0	0	0	1	1	0	0	1	1	0	0	0	0	0
0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(b)

Figure 2.2 (a) An image of character "V", (b) The simple digital image of (a).

Although an exact formula for $f(x, y)$ of an image is almost never known, many techniques in image processing, such as those described in this sub-section and the next sub-section, have been developed based on the concept of the $f(x, y)$ representation of an image.

Two well known methods in image enhancement are low-pass filtering and high-pass filtering [17, 18, 19, 20]. The aim of low-pass filtering is to remove only the high intensity or high frequency components of a signal. Conversely, high-pass filtering retains only the high intensity or high frequency components of a signal. Since image degradation is often introduced by wide-band random noise, the use of low-pass filtering can reduce a large amount of noise. But the filtering also removes a small amount of signal that might be critical to the recognition process, such the edges or details of an object. Thus, when using low-pass filtering, the tradeoff between information reduction and noise reduction must be considered.

The principle operation of low-pass filtering can be described as assigning an intensity value to a pixel at the (x, y) coordinates based on the original intensity levels of its own and its neighboring pixels. A neighborhood about (x, y) is defined as a square or rectangular sub-image area centered at (x, y), as shown in Figure 2.3(a). The mathematical representation of low-pass filtering of an image at pixel (x, y) is given by

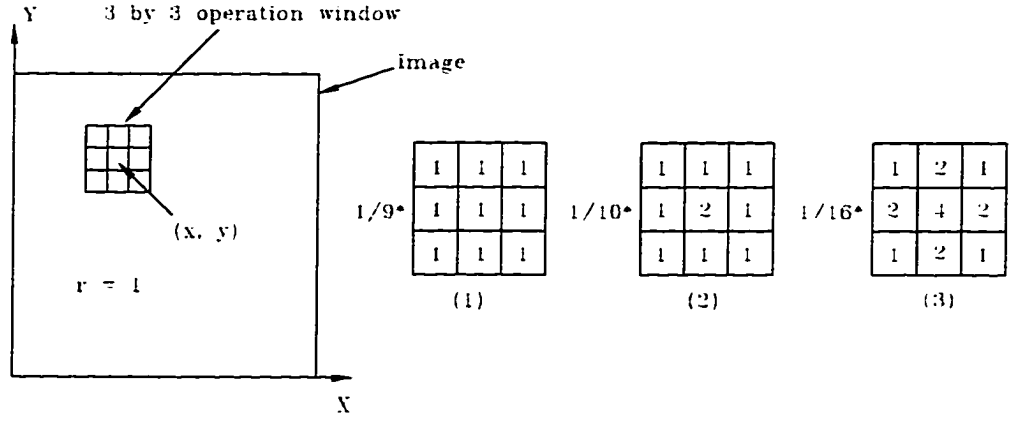
$$f^{new}(x, y) = \sum_{i=x-r}^{x+r} \sum_{j=y-r}^{y+r} t(i, j) f^{old}(i, j), \tag{2.2}$$

where $f^{new}(x, y)$ and $f^{old}(i, j)$ are the intensity values of the pixel at (x, y) coordinates after the low-pass filtering process and the pixels inside the square sub-image area before the filtering process, respectively. The function $t(i, j)$ is the coefficient of a transformation operator (also termed window, or mask) associated with the pixels in the sub-image area, and $(2r + 1)$ is the size of a square sub-image. For example, a low-pass filtering operator shown in Figure 2.3(b) is used to filter an image. Then, the pixel value at (x, y) coordinates after the filtering process is

$$\begin{aligned} f^{new}(x, y) &= \sum_{i=x-1}^{x+1} \sum_{j=y-1}^{y+1} t(i, j) f^{old}(i, j) \\ &= \frac{1}{10}[1 \times f^{old}(x-1, y-1) + 1 \times f^{old}(x-1, y) \\ &+ 1 \times f^{old}(x-1, y+1) + 1 \times f^{old}(x, y-1) \end{aligned}$$

$$+ \quad 2 \times f^{old}(x,y) + 1 \times f^{old}(x,y+1)$$
$$+ \quad 1 \times f^{old}(x+1,y-1) + 1 \times f^{old}(x+1,y)$$
$$+ \quad 1 \times f^{old}(x+1,y+1)]. \tag{2.3}$$

Thus, $f^{new}(x,y)$ is the weighted average of the original pixel values inside a 3×3 operation window. Other examples of low-pass filtering operators are shown in Figure 2.3.



(a) 3 by 3 neighborhood of pixel (x, y).

(b) Examples of low-pass filtering operators.

Figure 2.3 Low-pass filtering process.

The high intensity or high frequency components of a signal are often associated with edges or details of objects in an image. High-pass filtering will help retain this information for image analysis. Since background noise typically has high intensity or high frequency components, this filtering process will increase the relative influence of background noise. This limits the use of high-pass filtering in image enhancement, especially if the image is processed for the purpose of object recognition, since the large background noise will significantly weaken the reliability of the extracted features, and consequently reduce the recognition rate. The mathematical representation of high-pass filtering is the same as that for low-pass filtering as previously shown in Equation 2.2. Figure 2.4 illustrates three typical transformation operators in 3×3 operation window. If the coordinates of the center location of an operation window is at (x,y),

the intensity value of the (x, y) pixel after using Figure 2.4(a) filtering operator is, for example,

$$\begin{aligned} f^{new}(x,y) &= 0 \times f^{old}(x-1,y-1) + (-1) \times f^{old}(x-1,y) \\ &+ 0 \times f^{old}(x-1,y+1) + (-1) \times f^{old}(x,y-1) \\ &+ 5 \times f^{old}(x,y) + (-1) \times f^{old}(x,y+1) \\ &+ 0 \times f^{old}(x+1,y-1) + (-1) \times f^{old}(x+1,y) \\ &+ 0 \times f^{old}(x+1,y+1). \end{aligned} \tag{2.4}$$

0	-1	0
-1	5	-1
0	-1	0

(a)

1	-2	1
-2	5	-2
1	-2	1

(b)

-1	-2	-1
-2	19	-2
-1	-2	-1

(c)

Figure 2.4 High-pass filtering operators.

Median filtering [18, 19, 22] is a nonlinear operation that combines the advantages of both low-pass and high-pass filterings. Not only can it be used to reduce high-frequency noise like low-pass filtering, but it can also preserve the edges of objects in an image like high-pass filtering without losing image details. The filtering procedure for this method is similar to previous filters and can be outlined as follows: The center of the operation window (see Figure 2.3) is moved around the image. At each pixel position in the image, all pixel intensities in the sub-image area (or operation window) are ranked. The median of these pixel intensities is found and replaces the center pixel intensity of the current sub-image area.

When using median filtering, the size of the operation window is an important factor affecting the result of the filtered image. Although the size of the window depends on the application, a large size will generally distort the signal and a small size cannot remove the isolated noise. Thus, it is necessary to experiment with different window sizes and choose the one with the best resulting image before applying median filtering to an application. Figure 2.5 shows the results of the three techniques discussed above.

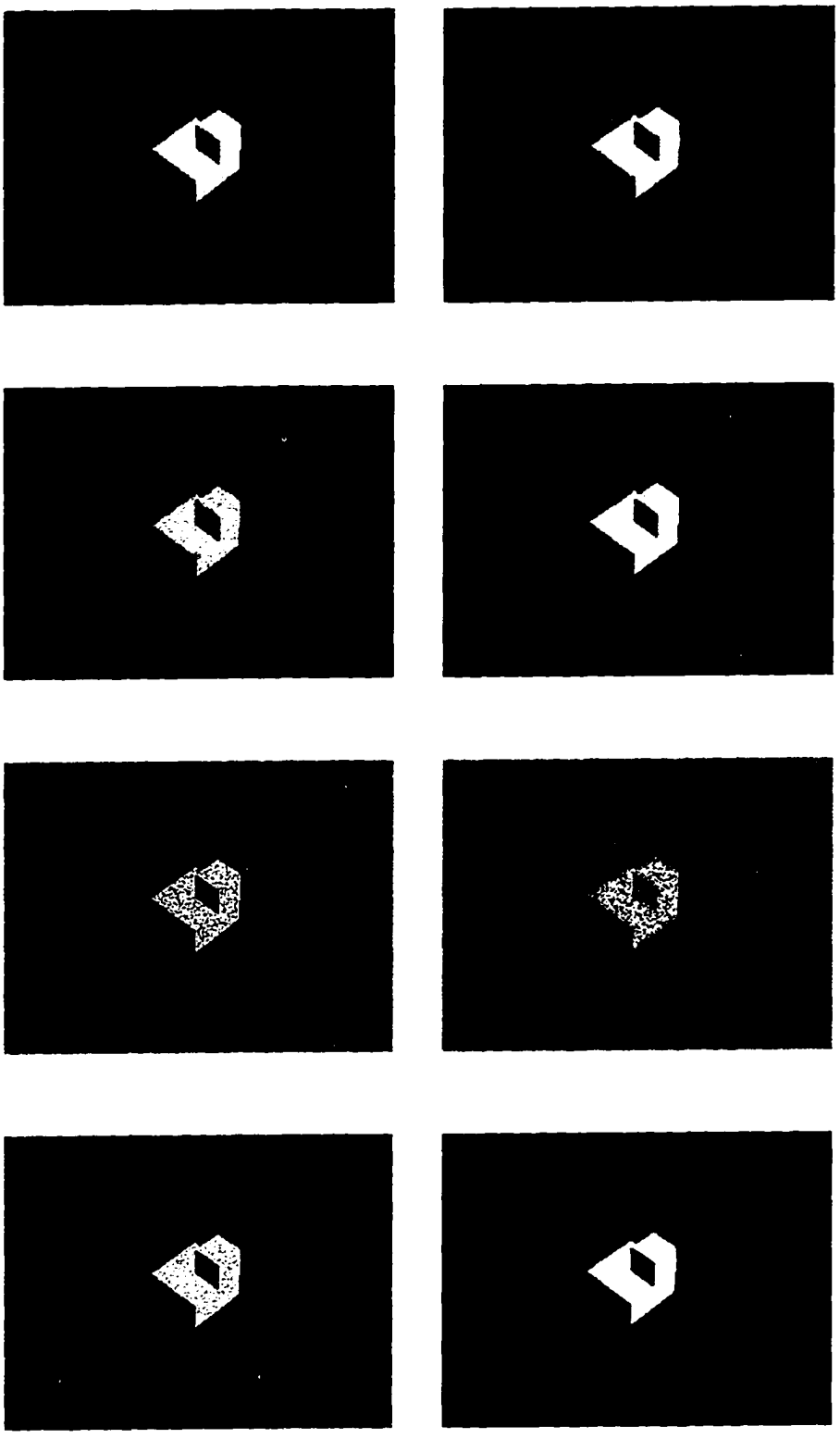


Figure 2.5 The results of different image enhancement techniques.

There are many other sophisticated methods for image enhancement, such as adaptive filtering and Wiener filtering, that require either more computational effort and/or strong prior knowledge about a degraded image. To process an image, adaptive filtering [23, 24] adjusts the operation window size and the operator coefficients based on the details of image characteristics. One example of such image characteristics is to consider the differences between image regions that may have various degradation. Weiner filtering [20] is frequently used in extracting a signal from background noise. Considering the problem of an original signal combined with background noise, the methods presented before will only smooth the original signal itself. They will not separate the real signal from the background noise. Weiner filtering aims to overcome this problem by deriving a transformation operator that approximately transforms the original signal to the real signal. The transformation operator is a function of the variance of the noise distribution. Thus, previous knowledge of the noise distribution is required to use this filtering technique.

Histogram modification of gray-level [17] is another major technique used in image enhancement. This technique is based on the histogram of image pixels to transform the image. A good result requires intensive study of the characteristics of the histogram of the input image pixels. Experienced personnel must be involved to find a suitable histogram for every individual image [17]. Because of the difficulty in using this technique, it was not considered for use herein.

2.1.1.2 Image analysis techniques

The goal of image analysis is to describe physical objects in a scene from an image of the scene. One of the most important elements in image analysis is image segmentation, in which objects or other entities of interest are extracted from an image for subsequent processing. The fundamental idea of image segmentation is based on the properties of pixel intensities (or gray-level values): discontinuity or similarity. Algorithms using the first property partition an image into regions based on the differences between neighborhood pixel intensities. The meaningful features derived from these algorithms are generally points, lines, edges of objects,

edges of object shadows. Algorithms using the similarity property are based on thresholding and relaxation (region growing). These algorithm will have the better segmentational results in situations where the regions in question are insufficiently homogeneous so that the transition between two adjacent regions cannot be detected from intensity-level discontinuities alone.

Although points and lines certainly represent some information of an image content, edges and curves are by far more meaningful in describing objects in an image. Most object description or recognition systems only extract edges and curves as the useful features without using points or lines in an image. This is because the points and lines are often isolated features or noise that does not belong to or represent the objects of interest. Thus, edge detection is the most important approach for detecting meaningful discontinuities in pixel intensity-level.

The idea underlying most edge-detection techniques follows the mathematical definition of a *gradient*. The method calculates the differences in intensity levels of two orthogonal directions and combines the differences to give the edge intensity, the arc-tangent of their quotient is then computed to yield an estimate of the edge direction. The three general methods of combining two differences of intensity levels are: (1) the square root of the sum of two square differences; (2) the sum of two absolute differences; (3) the maximum of two absolute differences.

In the edge detection process, the operator usually uses a 2×2 or 3×3 operation window with two orthogonal masks. An operation window is swept across every pixel in an image in order to find every possible discontinuity that may represent an edge of an object in the scene. An example of an operator using 2×2 operation window is the *Roberts* operator [25] (see Figure 2.6(a)). It is derived by fitting a planar surface to the intensity-levels in the 2×2 neighborhood of the given pixel and by taking the gradient of this surface as an estimate of the differences of intensity-levels in two orthogonal directions. Each orthogonal direction will have its own mask. Based on a similar approach, the *Prewitt* operator [26] used a 3×3 operation window can be obtained by fitting a quadratic surface to the intensity-levels in a 3×3 neighborhood of the given pixel (see Figure 2.6(b)). Another operator [26] proposed by Prewitt to overcome the shortcoming of the 3×3 Prewitt operator (that does not use the pixel intensity at the center of the operation window) is the 4×4 operator. It is again derived by

fitting a quadratic surface by least squares to the intensity-levels in a 4×4 neighborhood of the given pixel. The complete process of 3×3 Prewitt operator is first to calculate the edge magnitudes at the (x, y) location in horizontal and vertical directions based on two masks and neighboring pixel intensity values

$$\begin{aligned} e^h(x,y) &= \sum_{i=x-1}^{x+1} \sum_{j=y-1}^{y+1} t^h(i,j) f(i,j) \\ &= (-1) \times f(x-1,y-1) + 0 \times f(x-1,y) + 1 \times f(x-1,y+1) \\ &+ (-1) \times f(x,y-1) + 0 \times f(x,y) + 1 \times f(x,y+1) \\ &+ (-1) \times f(x+1,y-1) + 0 \times f(x+1,y) + 1 \times f(x+1,y+1), \end{aligned}$$

and

$$\begin{aligned} e^v(x,y) &= \sum_{i=x-1}^{x+1} \sum_{j=y-1}^{y+1} t^v(i,j) f(i,j) \\ &= 1 \times f(x-1,y-1) + 1 \times f(x-1,y) + 1 \times f(x-1,y+1) \\ &+ 0 \times f(x,y-1) + 0 \times f(x,y) + 0 \times f(x,y+1) \\ &+ (-1) \times f(x+1,y-1) + (-1) \times f(x+1,y) + (-1) \times f(x+1,y+1). \end{aligned}$$

Then, the sum of $|e^h|$ and $|e^v|$ is calculated as the estimate of an edge intensity at (x, y) position, if the sum is larger than a pre-defined cutoff value. The $atan\left(\frac{e^v}{e^h}\right)$ is computed as the estimate of the edge direction,

The *Sobel* operator [27] is closely related to the Prewitt operator, but is less sensitive to noise. Its application is equivalent to the combined execution of first averaging the intensity-levels with a 2×2 operation window that has all 1's, and then computing the horizontal and vertical directional derivatives of the averaged intensity-levels using a 2×2 operation window. The Sobel operator is illustrated in Figure 2.6(c). All the above operators have been used in this research to find the best one for this application. Results of these experiments are shown in Figure 2.7.

Template matching is another approach to edge or curve detection that is also based on the concept of gradient analysis. Instead of computing the first derivatives in only two orthogonal directions, an operator of this kind has more than two masks to provide estimates of the

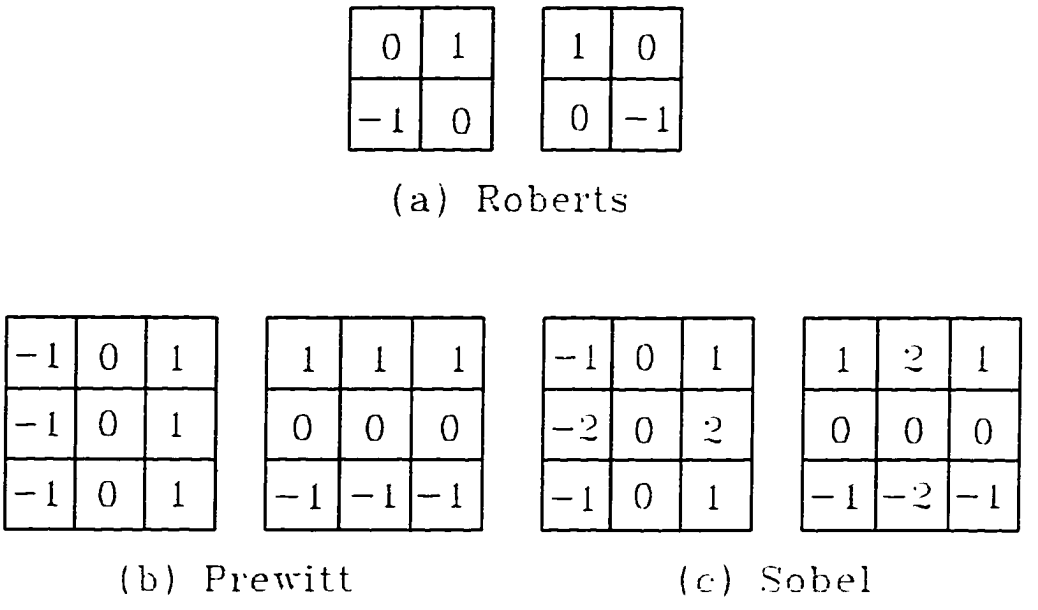


Figure 2.6 Different operators used for edge detection.

first derivatives (or differences of intensity-levels) in more than two directions for a specific operation window. The edge intensity at the center of an operation window is then estimated as the maximum value of the computed differences and the corresponding mask orientation provides the estimate of the edge direction. Many operators of template matching have been proposed since Prewitt [26] first suggested this method. Figure 2.8 shows several operators: (a) Prewitt operator; (b) Robinson operator [28]; and (c) Kirsch operator [29]. The results of these operators are displayed in Figure 2.9.

A template matching technique developed by Frei and Chen [30] has the advantage of detecting combinations of points, lines, and edges. Its operator contains nine masks as shown in Figure 2.10. The first four masks are used to detect edges; the second four masks are suitable for line detection; and the last mask is proportional to the average of the intensity-levels in the 3 x 3 neighborhood of the given pixel. The decision that the pixel in the center of the operation window belongs to a certain type of feature is based on the value of the following equations

$$\theta_e = cos^{-1}\left\{\frac{1}{||\mathbf{x}||}\left[\sum_{i=1}^{4}(\mathbf{w}_i'\mathbf{x})^2\right]^{1/2}\right\}, \tag{2.5}$$

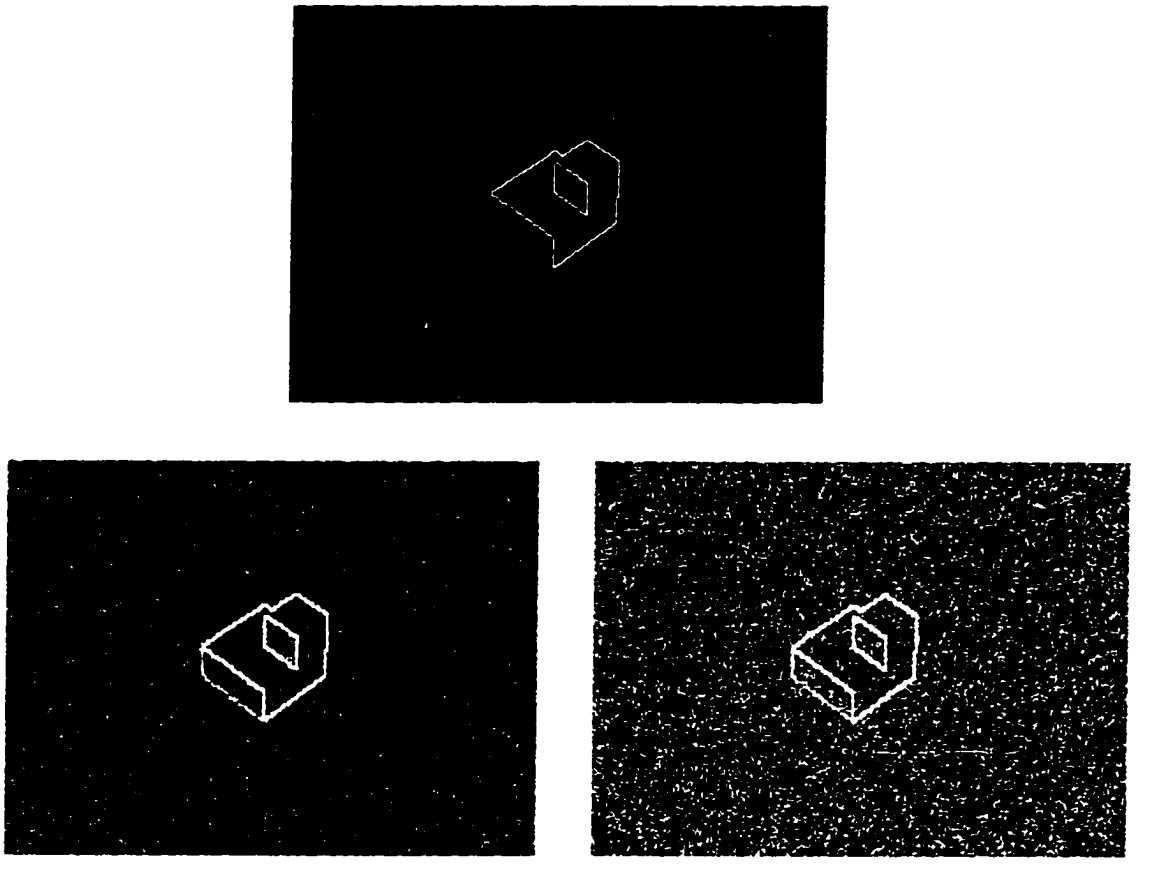


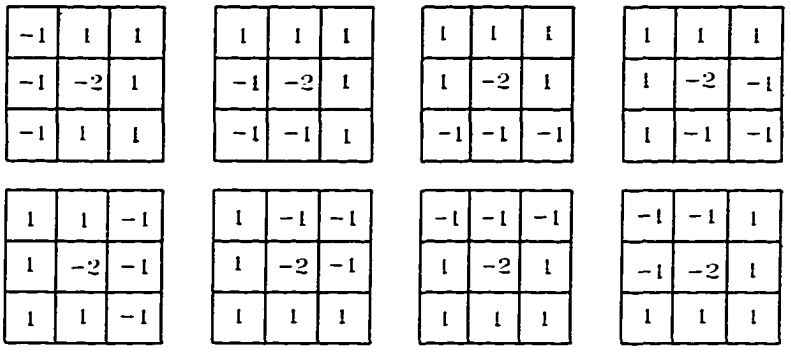
Figure 2.7 The results of different operators for edge detection.

$$\theta_l = cos^{-1}\left\{\frac{1}{||\mathbf{x}||}\left[\sum_{i=5}^{8}(\mathbf{w}_i'\mathbf{x})^2\right]^{1/2}\right\}, \tag{2.6}$$

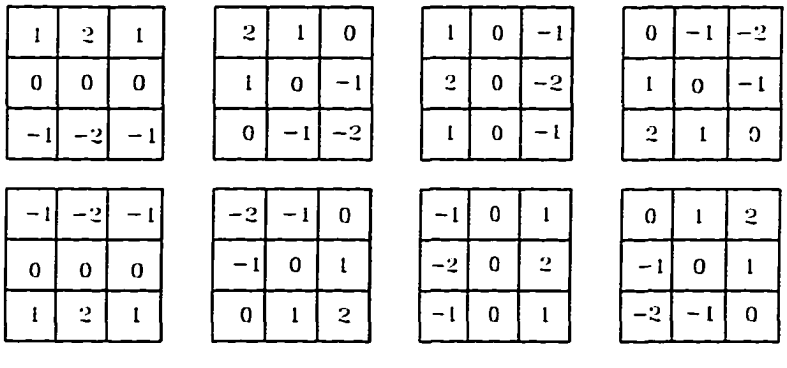
and

$$\theta_a = cos^{-1}\left\{\frac{1}{||\mathbf{x}||}|\mathbf{w}_9'\mathbf{x}|\right\}, \tag{2.7}$$

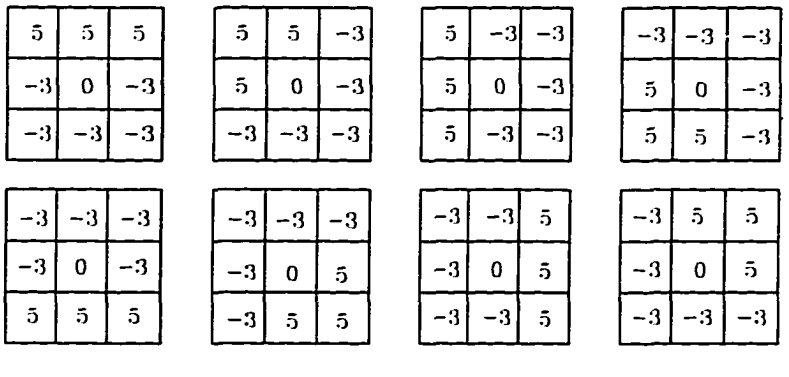
where $\mathbf{x}$ is a vector of intensities of a 3×3 array of pixels, and $\mathbf{w}_i$ is a vector of elements for i-*th* mask. If $\theta_e = max\{\theta_e, \theta_l, \theta_a\}$, the pixel in the center of the operation window is classified as an edge of an object; if $\theta_e = max\{\theta_e, \theta_l, \theta_a\}$, it will be treated as a line feature; otherwise it is a point. Once the pixel is concluded as an edge, the maximum response of the first four masks provides an estimate of the edge magnitude, and the orientation of the corresponding mask provides an estimate of the edge direction. In the same manner, the second four masks estimate the magnitude and direction of a line.



(a) Prewitt masks in eight directions



(b) Robinson masks in eight directions



(c) Kirsch masks in eight directions

Figure 2.8 Different operators for template matching.

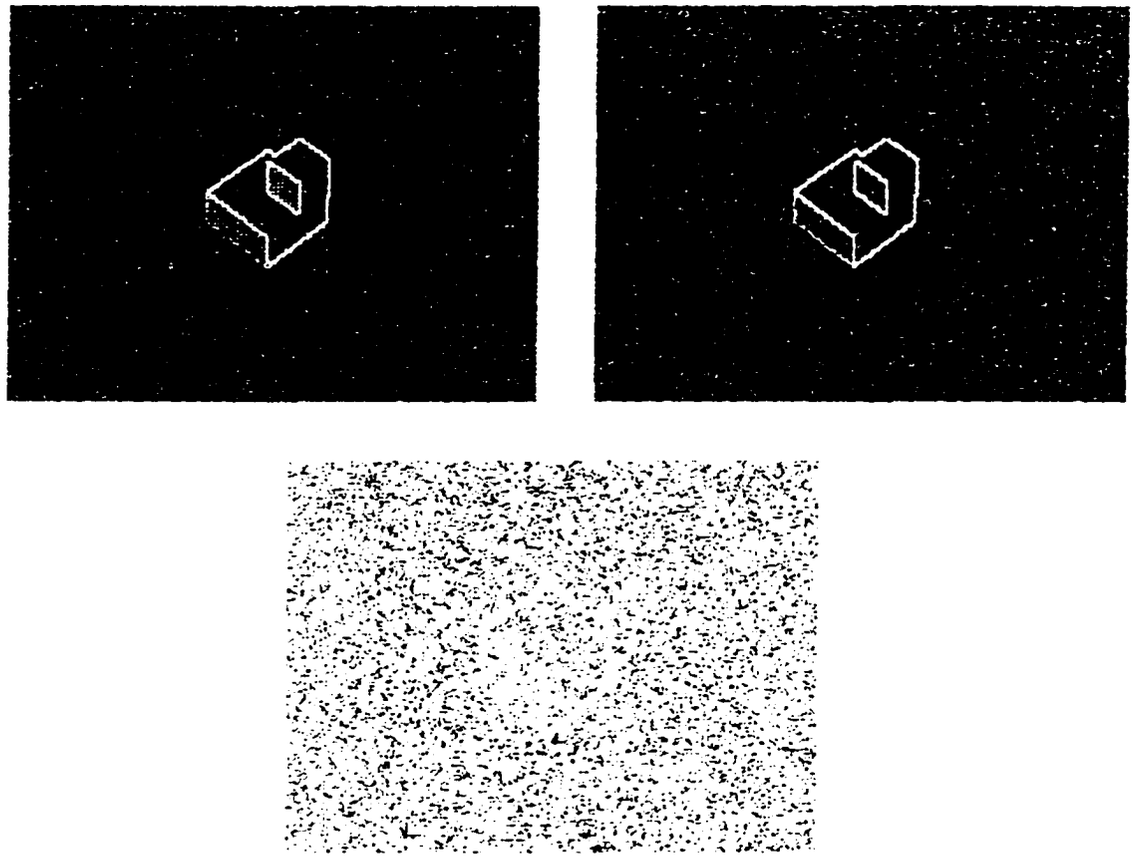


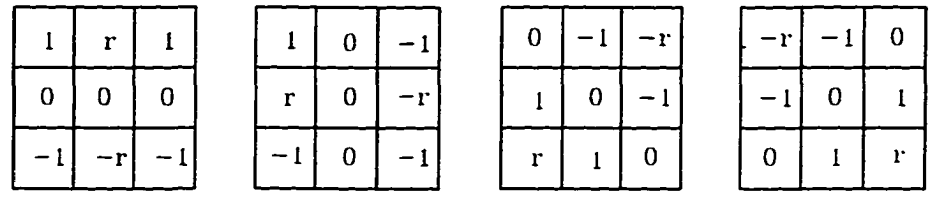
Figure 2.9 The results of different operators for template matching.

Unlike the previous operators that use the first order differentiation, the operators outlined below use second derivative estimates and have only one mask for each operator. These type of operators are often called *Laplacian* operators. The mathematical formulation of a Laplacian is defined as

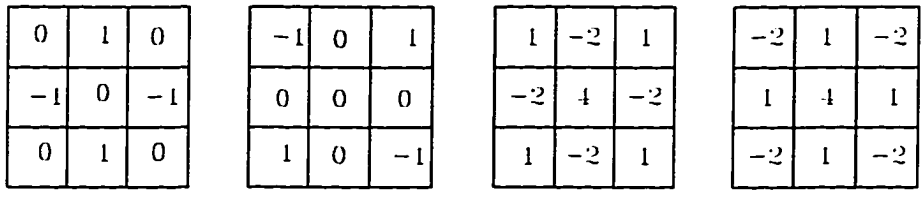
$$\Delta^2 f(x,y) = \frac{\alpha^2 f(x,y)}{\alpha x^2} + \frac{\alpha^2 f(x,y)}{\alpha y^2}, \tag{2.8}$$

where $f(x,y)$ is defined as (2.1). Based on a 3×3 operation window, a digital version of the Laplacian can be derived as

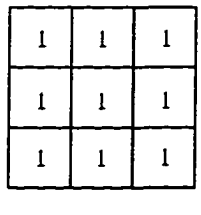
$$\begin{aligned} \Delta^2 f(x,y) &= \Delta_x^2 f(x,y) + \Delta_y^2 f(x,y) && (2.9) \\ &= [f(x+1,y) + f(x-1,y) \\ &+ f(x,y+1) + f(x,y-1)] - 4f(x,y), && (2.10) \end{aligned}$$



Basis of edge subspace $r = 2^{-1/2}$



Basis of line subspace



"Average" subspace

Figure 2.10 Template matching operator for detecting combined features, such as edges, lines, and points.

if the derivatives are estimated across horizontal and vertical directions such as shown in Figure 2.3(a). The equivalent operator is apparent in Figure 2.11(a). Figure 2.11 also shows other Laplacian operators obtained by taking different directions. Figure 2.12 shows the results of edge detection using these operators.

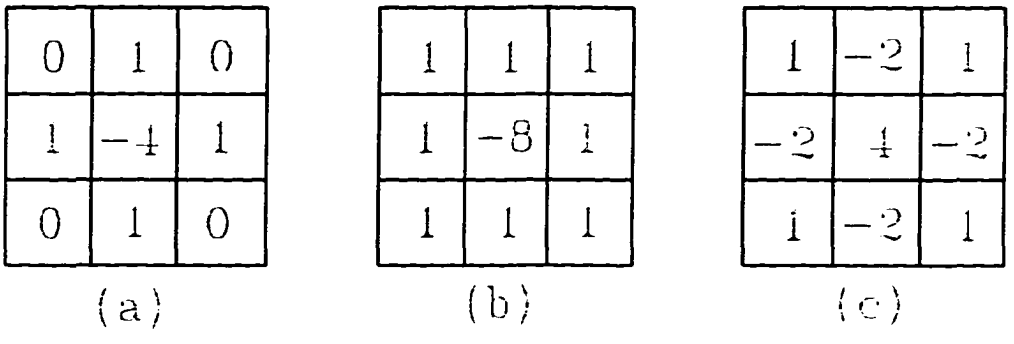


Figure 2.11 Laplacian operators.

The techniques outlined above are suitable for most applications. Generally, the results obtained from sophisticated approaches are not significantly better than those from simple operators. If an application requires precise locations and direction of edges, it usually needs a bigger operation window. But the tradeoff is the requirement of larger computational effort.

In reality, breaks in edges and between edges always exist due to nonuniform illumination and other effects that introduce spurious intensity discontinuities, thus the description of object boundaries is always incomplete. In order to complete the segmentation process after applying the edge-detection techniques, an edge linking process must be used to get a meaningful set of object boundaries. One of the simplest approaches to accomplish edge linking is *local analysis*, which uses the similarities of pixels' intensity levels and directions after the edge-detection process to link pixels. Other more complex techniques may consider not only the similarities of pixels but also some other properties such as distances between pixels sharing some other common properties.

The fundamental concepts of segmentations by *thresholding* and *relaxation* will be briefly outlined without detailing any specific algorithm. As mentioned earlier, these types of tech-

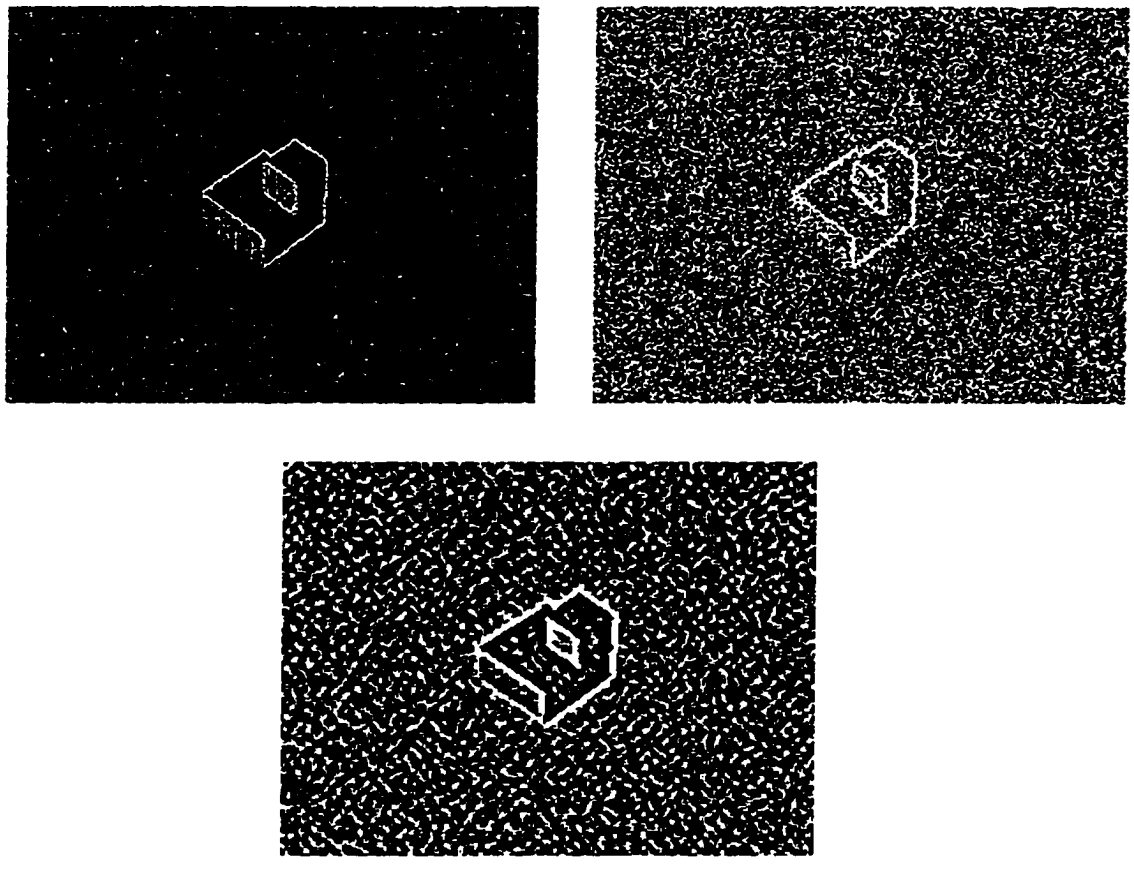


Figure 2.12 The results of using Laplacian operators in edge detection.

niques are based on the similarity of the intensity level of pixels. Segmentation by thresholding attempts to cluster pixels which have similar intensity levels in an image by using one or more thresholds. In the case of using a single threshold, for example, pixels with intensity-levels lower than the threshold will be clustered into one region and otherwise into another. The values of thresholds in some algorithms are treated as constant, but in other approaches are variated globally or locally. In practice, the values of thresholds are defined based on the distribution of pixel intensity-levels. More information about segmentation by thresholding can be found in [31] as well as other books about image processing.

Segmentation by relaxation is also called segmentation by *region growing*. It is a process that simply groups pixels or subregions into larger regions that satisfy some criteria. There are many ways to define these criteria. Some are just based on the absolute difference in

intensity-levels, and more sophisticated ones use statistical models. References [31, 32, 33, 34] contain more complete and formal characterizations.

2.1.2 Feature extraction

After image segmentation, the meaningful features are enhanced, such as the regions characterizing objects and the edges bounding the useful regions in an image. The next step in a recognition process is to extract features. Feature extraction is the standard task that compacts the segmented image into representations that are considerably more useful or sufficient in the computation of classification, although the segmented image data are sometimes used directly to classify objects. The type of features that are used is vital for a system to have good recognition results. It is evident that the number of features needed to successfully perform a given recognition task depends on the discriminatory qualities of the selected features. However, the problem of feature selection is usually complicated by the fact that the most important features are not necessarily easily measurable, or, in many cases, their measurement is inhibited by economic considerations.

In general, features representing objects usually exhibit two characteristics, *wide domain* and *unique*. A set of features is said to have *wide domain* if it can represent a large class of objects adequately. A set of features is said to be *unique* if every distinct member of its domain has a single distinct representation. Another important property of features is invariance with respect to the small variations in the scene. Also, it is desirable to obtain features with computational simplicity, data reduction, or invariance to translation, rotation, and scaling. For the most part, the features outlined in this section satisfy one or more of these properties.

Although many different kinds of features have been used in variety of object pattern recognition applications, these features can be summarized into three fundamental categories: *intensities*, *edges*, and *regions*. It is not uncommon to find a combination of these features being selected as the input data to classifiers in recognition systems. Since features are strongly problem-oriented in the sense that their uses involve the development of specific algorithms which fit the situation at hand, it is not intended in this discussion to describe features for any

specific application. Instead of that only discuss the above three fundamental features without losing the generalization.

Identifying features strictly based on intensity-levels of images is very intuitive. Each pixel has its own intensity-level and can provide some information about the image. If a feature uses every pixel's information, it can represent an image well. This type of feature can be easily derived without any computation effort. But at the same time, it means the original image data is input into the classifier without any data reduction. Therefore, large computational effort in the classifier is necessary for any system using this type of feature, especially for large images. In industrial applications, the performances of recognition systems are not only evaluated by the recognition rate, but also the classifying speed. Thus, it is not desirable or applicable to directly use every individual pixel intensity to construct features. In fact, by applying a simple additive effort, features with a great reduction in their dimensions can be obtained by using a Radon transform (projection) of the images. Note that it is unavoidable to lose some information about objects in the images, and consequently the recognition rate is degraded, but the significant increase in classifying speed dominates this tradeoff. The mathematical representation of the Radon transform and its properties are outlined in Chapter 4, along with its feasibility to represent three-dimensional objects in images.

Features using intensity-level histograms also depend on pixel intensity-levels of images. The underlying concept of this type is to model the features to represent the intensity-level probability density functions. It is obvious, for example, that images with different areas of white regions will provide different intensity-level probability density functions.

From the above discussion, it is easy to notice that features based on intensity-levels are not suitable for recognition systems that deal with multiple objects. This is because these features consider only one global property of images with no information about local feature relation.

Although features based on intensity-levels can be extracted without even pre-processing the original images and are easy to implement, the lack of information on the local feature relation makes them unfavorable for most recognition systems. Alternatively, features based

on edges or curves will provide some local information. Edge features can take various forms to represent objects in images. Some of them directly utilize the parameters associated with edges such as their lengths, curvatures, and moments, to mention a few. Others may be established at higher levels of complexity. The length of a set of edges that bounds regions of interest is one of the simplest features used to describe objects. It can be approximated by counting the number of pixels along the region after the enhanced region has been thinned along its boundary edges. Further studies for computing the length can be found in references [35, 36, 37].

Curvature, the rate of change of slope, is another simple feature. One useful method to derive it is to use the difference between the slopes of adjacent boundary edges. Features using length or curvature are useful for systems that deal with multiple objects in the same scene, and can be obtained without difficulty. But they can not uniquely represent objects if a system only involves length or curvature feature individually. Features using moments are by far the most popular technique in describing the shape of edge or curve segments. Its advantage includes straightforward implementation and good physical interpretation of the boundary shape.

If an edge or curve segment is represented as a one-dimensional function $g(r)$ of an arbitrary variable r, then the n-th moment of r about its mean is

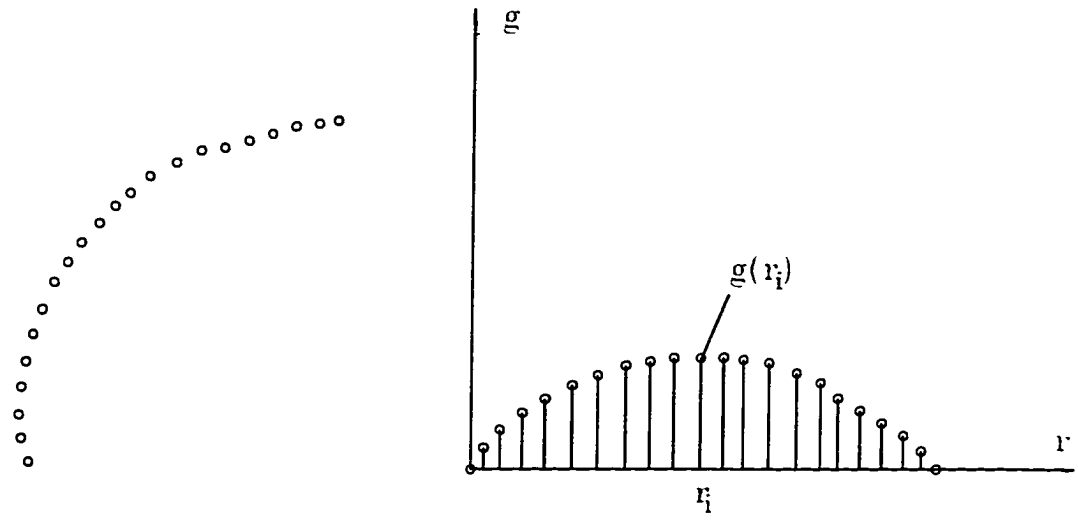
$$\mu_n(r) = \sum_{i=1}^{N} (r_i - m)^n g(r_i), \tag{2.11}$$

where

$$m = \sum_{i=1}^{N} r_i g(r_i),$$

and N is the number of points on the curve segment, see Figure 2.13. From the equation above, it is not difficult to see that the second moment $u_2(r)$ is the measure of the spread of the curve about the mean value of r and the third moment $u_3(r)$ is used to measure its symmetry with respect to the mean. In practice, it is normal to use both moments simultaneously to describe a given boundary segment.

Classifying techniques using edges or curves as features with higher levels of complexity are actually based on the features discussed above. For example, the boundary matching



(a) Digitized boundary segment

(b) Representation as a one-dimensional function

Figure 2.13 The representation of moment.

technique essentially matches the lengths of two region boundaries in some specific way to check if two regions are similar. The syntactic technique uses a set of pre-defined primitive elements to exploit any structural relationships between boundaries or regions. Often, these primitive elements are edges of boundaries. All features extracted from the edge or curve thus far require some sort of thinning process for the enhanced image. This implies that the time spent in the image processing stage is longer. However, the requested time in the final process, classification, might be significantly reduced.

The last set of features is based on the enhanced regions. The area and perimeter of a region might be used as features. In most cases, they are applied to the situations in which the size of the objects of interest is invariant. A more frequently used feature that measures the *compactness* of a region can be established as $(perimeter)^2/area$. Since $(perimeter)^2/area$ is a unitless quantity, this feature is insensitive to rotation and scale changes.

Features based on moments of regions are also popular in recognition systems. A two-dimensional function $g(x, y)$ represents a region of interest. Then, the central moment of order

$(p+q)$ for a digital image can be expressed as

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q g(x, y), \tag{2.12}$$

where

$$\bar{x} = \frac{m_{10}}{m_{00}}, \qquad \bar{y} = \frac{m_{01}}{m_{00}},$$

and

$$m_{pq} = \sum_x \sum_y x^p y^q g(x, y).$$

Based on the second and third central moments that can be obtained from the above equations, a set of seven *invariant moments* [31] can be derived. This set of moments is very useful in constructing features and has been adapted by many researches because of its invariant properties of translation, rotation, and scale changes [38]. Other features based on region are detailed in reference [17]. Features of this type can be applied in recognition systems that deal with multi-object environments. As with features based on edges, features based on regions require their associated images to go through the properly enhancement and thinning processes.

This subsection has discussed different types of features that might be extracted from images and outlined their general advantages and disadvantages. As indicated earlier in this subsection, the choice of some features over other is dictated by the problem at hand. But the fundamental concept should be borne in mind when selecting features. That is, the chosen features can be measured for each individual object and they should be able to capture essential differences between objects, or classes of objects, while maintaining as much independence as possible to changes such as location, size, and orientation.

2.1.3 Classification

The last process involved in an object pattern recognition system, of course, is the task of identifying or classifying patterns. Once features are chosen, a classifier or recognizer needs to be appropriately designed or selected. There are many classifiers that have been developed or implemented in object pattern recognition systems. The basic design concepts for these

classifiers have been implemented by three principal categories of methodology: *heuristic*, *mathematical*, and *syntactic*. It is not uncommon to find classifiers that use combinations of these methods.

Classifiers that use heuristic methods are an important branch of classifier family. They usually classify different objects by matching a set of stored prototype features or objects with an unknown object in a scene. Once a match is found, the unknown object is classified as the member of the corresponding prototype class. This type of classifying process is also known as the *template matching* process.

The use of this type of classifier is a simple solution to object recognition systems in certain applications. For instance, if a set of objects with different shapes are stored in a object recognition system as prototypes, an object in an image may be recognized by comparing it with the prototypes as long as the image is not distorted by noise due to the resolution of the sensing device, lighting, or blurring effect. Clearly, this is a simple-minded classifying method. However, the number and type of prototypes are crucial to the performance of this type of recognition system. Too many or complex stored prototypes would require extremely long time computation. On the other hand, too few or poorly chosen prototypes would not classify objects in images sufficiently to permit categorization into their respective object classes. In fact, the performance of a heuristic classifier mainly depends on the cleverness and experience of system designers.

Many classifiers of earlier versions of recognition systems have been designed by using heuristic methods. But it is difficult to conclude a generalized principle for developing these classifiers since their developments and implementations heavily depend on human intuition and experience, and also the application in hand. Thus, we will not further discuss this type of approach.

Classifiers based on the mathematical approach utilize the mathematically formulated common properties underlying a set of features to recognize different objects. In order to have an acceptable rate of classification using this type of classifier, objects belonging to the same class need to possess certain common properties which reflect similarities among them. These

common properties can be derived by formulating the extracted features from a finite set of sample objects. Several methods of extracting features have been discussed in the previous subsection.

The principal methodology of the mathematical approach is to find a set of decision boundaries that can separate different classes of objects based on the common properties of the extracted features. In a mathematical framework, a set of decision boundaries is simply constructed by a set of decision functions. In fact, decision functions have played a central role in the development of object pattern recognition theory. They not only provide a feasible and meaningful tool, but also they can often be formulated by using representative features from each object class. There are many ways to find the decision functions for a set of sample (mathematically formulated) features (patterns). Some of them use the statistical properties of the pattern classes under consideration and others do not. Therefore, the classifiers based on the mathematical methods can be subdivided into two categories: *statistical* and *deterministic*. As the name implies, the statistical method attempts to classify objects by employing explicitly a statistical model underlying a set of extracted features. One popular statistical classifier is known as the Bayes classifier. With some small variations, other statistical classifiers can be developed.

The idea of the Bayes classifier is to minimize the total expected misclassification with respect to all decision boundaries. Let $\mathbf{x}$ be a vector mathematically representing a feature; let ω_i denote the i-th pattern class with probability $p(\omega_i)$. Then, the Bayes classifier will classify feature $\mathbf{x}$ to class ω_i, if $r_i(\mathbf{x}) < r_j(\mathbf{x})$ for $j = 1, 2, ..., M, j \neq i$, where M is the number of classes, and

$$r_i(\mathbf{x}) = \sum_{i=1}^{M} L_{ij} p(\mathbf{x}/\omega_i) p(\omega_i), \tag{2.13}$$

where L_{ij} is the *loss function.*

It is evident from the equation above that a priori knowledge or estimation of the probability density function $p(\mathbf{x}/\omega_i)$ is the most important problem in the implementation of a Bayes classifier. Several methods [39] have been developed to estimate the density function for some given problem. In general, a normal distribution is suitable for most practical applications.

The performance of a statistical classifier is, in general, strongly dependent on the quality of the estimation of the probability density function as well as the complexity of its decision functions. Unlike their statistical counterparts, however, the deterministic classifiers do not require knowledge about the statistical properties of the object classes under consideration. Simple deterministic classifiers can be approached by applying distance functions, which is the most obvious way of establishing a measure of similarity between pattern vectors in Euclidean space. These pattern vectors may be, for example, the vectors whose components are the invariant moments of regions of interest or the intensities of pixels inside regions of interest. One example of simple deterministic classifiers is the minimum-distance classifier. It uses prototypes of every pattern class to compute the decision functions based on the minimum-distance concept. It is worthwhile to note that the ability to determine characteristic prototypes in a given set of patterns plays an important role in the design of this kind of classifiers. Due to their incapabilities of dealing with overlapping of different pattern classes, simple deterministic classifiers suffer from a high rate of misclassification.

More sophisticated classifiers of deterministic method have been developed based on the fast-growing technique of neural networks. Not only can this type of approach always produce a set of decision functions which will correctly classify all patterns if their classes do not share identical ones, but also they can achieve the same performance as the statistical methods without concerning the statistical properties of the patterns being considered. It is also worth noting that neural network classifiers are in general better than statistical classifiers in their convergence rate [39]. Although many advantages have been found for neural network approaches, some practical considerations of these approaches must be emphasized on the economic and computational efforts. In many cases a perfect classification is not achievable within the time or complexity constraints placed on the classifier. Therefore, a certain percentage of tolerable mis-classifications needs to be determined to meet the design requirement of a classifier. The detail information about neural network models that can be implemented as classifiers is outlined in Chapter 3.

The classifiers in the preceding paragraphs of this subsection are generally applied to in-

dividual boundaries and regions of interest in an image. They deal with feature patterns on a strictly quantitative basis without concerning any structural relationship that may exist between individual features. In contrast to these classifiers, syntactic classifiers use the information of hierarchical structures to provide promising results in the research of object pattern recognition.

A syntactic classifier, also called a linguistic classifier, grammatical classifier, or structural classifier in the literature, can be implemented by using the concept of formal language theory. Like natural languages such as English, a set of grammars in syntactic classification plays the central role for interpreting objects. Therefore, the construction of pattern grammars is an essential step in the implementation of a syntactic classifier. It is important to note that the performance of a syntactic classifier is affected by the selection of pattern primitives, the assembling of the primitives, and their relationships into pattern grammars.

The syntactic classifier is a very powerful tool for classifying a large set of complex patterns which cannot be conveniently described by mathematical or heuristic approaches or when local properties cannot be identified and global features must be used. It is suitable to recognize occluded or overlapping objects. To gain more complete information about syntactic techniques, the reader should refer to references [39, 40, 41].

In this section, all processes required in an object pattern recognition system have been generally outlined. The next section will discuss some specific recognition systems that have been implemented. Since this research uses a neural network as the classifier in the recognition system, an emphasis will be made on those recognition systems that use neural networks as the recognizing tools.

2.2 Existing Systems

Object recognition is particularly important in industrial applications such as part inspection and robotics assembly due to the large variety of production parts. There are many object recognition systems currently available in industry or under development. Among these systems, many of them make use of the so called model-based recognition paradigm. In the

model-based approach, a set of geometrical models are created and stored in a database and then retrieved to match against features extracted from a 2D image. The features associated with these systems are usually the vertices, edges, angles of edges, etc. In most cases, however, these systems require the design of efficient retrieval mechanisms to search models in a database. In general, they use simple forms of mathematical classifiers, such as distance measure of similarity. Another major issue for this type of system is the missing edge or line segments during the process of image segmentation. Some systems employ heuristic methods to hypothesize the most likely missing line or edge segments. Therefore, knowledge about objects must be acquired to design a system and these systems are in general very problem oriented. Some of these systems are listed in the references [42, 43, 44, 45].

Other systems utilize the topological relation based on graph theory. They require the construction of a relational graph and the design of an interpretation tree search mechanism. In many applications, the relational graph is represented by a set of nodes connected by arcs. The nodes correspond to the junctions of the line or edge segments and the arcs correspond to the edges linking the junctions. The recognition process is simply to find a subgraph in the model database to match with the image graph. Therefore, the design of the interpretation tree search mechanism is very important to the performance of a system. It is obvious from the above discussion that this recognition scheme is a branch of syntactic object pattern recognition that we have briefly outlined before. Most of these schemes are capable of recognizing overlapping and multiple objects in an image. Some excellent references about this technique can be found in [46, 47, 48, 49, 50].

Due to the recent development of artificial neural networks, neural network classifiers have been replacing the conventional classifiers in many newly developed recognition systems because of their relatively simple mechanisms and less required knowledge about the input data. Mitzias and Mertzios [51] proposed an algorithm that is based on the polygon approximation technique coupling with neural network classifiers. It gives a fast recognition time for 2D binary shapes. It first uses a polygon with a specific number of vertices to approximate a contour region and then calculates the internal angles of the consecutive sides of the polygon and the

normalized errors. The normalized errors is the ratio of the real length over the polygon's edge length for two consecutive vertices of the polygon. These internal angles and normalized errors are treated as the feature vectors to input into two separate three-layer perceptron neural networks for training and recognition. The dimensions of input layers of the networks are defined by the number of vertices, while the number of their outputs equals the number of training classes. The third network is used to combine the outputs of the previous two networks, according to the specified strength of each feature vector. Since a fixed number of vertices are chosen as the information carrying points to characterize a given shape, this number could be small if the complexity of the given region is simple. Thus, the training time and the recognition time for the networks are fast. But, as the author intended, this algorithm works only for 2D shapes.

Al-Shaykh [52] developed a method that utilizes the singular values of a projection matrix as the features to input to a three-layer perceptron neural network for recognizing 2D objects. Each row vector of the projection matrix is derived by using the Radon transform of an image rotated a certain angle around the object's mass center. Since these features achieve invariance to translation, rotation, and scaling, this algorithm can be used to recognize 2D objects inside an image in any orientation and location. The dimensions of the input layer of the neural network is the same as the number of singular values of the projection matrix. The time for training the neural network is small because there is only one training pattern for each class. Due to the time spent on the singular value decomposition, the recognition speed is relatively slow compared to other methods using the same neural network.

A system called VIEWNET conducted by Bradski and Grossberg [53] can recognize 3D objects under noisy environments. The original image containing an object is first pre-processed to achieve the object-background separation by using neural networks. Then, the CORT-X 2 filter [54, 55] is used to filter noise and segment the object's boundary. The resulting image is further processed through a sequence of operations in order to have features with rotation, translation, and scale invariance. The Gaussian averaging method is applied to the image for reducing the feature dimensions. The pixel intensities in the finalized image are fed into

a neural network, called Fuzzy ARTMAP, for training and recognition. Also, the system employed a mechanism that can accumulate evidence from several images of a 3D object being viewed to derive the final decision about the classification. Thus, more views about a 3D object that the system can have in the recognizing stage will give a more accurate classification rate. The required training data set for this system is derived from a sequence of images that are obtained by rotating the object one full revolution around the vertical axis at each angle above horizontal. Therefore, the number of training patterns is large. Another disadvantage is the low recognition speed due to the heavy image pre-processing for deriving the invariant features. Other research closely related to this approach can be found in [56, 57].

Although there are many existing object recognition systems, very few of them, especially in the manufacturing area, employ simple features as the input data. In fact, most commercially available systems are model-based systems that require large database to store models and have a slow recognition speed. Many research papers have proposed the use of neural networks as the classifiers and simple features as the input data, but many of these are still in the developing or experimental stages. Typically, it is difficult to find an object recognition system that is suitable for manufacturing assembly or inspection applications based on a neural network architecture.

3 ARTIFICIAL NEURAL NETWORKS

3.1 Artificial Neural Networks and Their Algorithms

In the early 1900's, the motivation for understanding and emulating the human brain and its strength brought out the concept of the artificial neural network. Since then, many studies and research have been presented in this field. But until the late 1970's, the limited available computational power was the major deficit for the application and development of artificial neural networks. The successful implementation of more sophisticated network architectures that were able to handle the shortcomings of the earlier networks required tremendous computational effort. Thus, the interests in this field were overshadowed by the insufficiency of manual calculation. Due to the rapid movement in digital computers, artificial neural networks are not only being studied to understand the human brain and its strength, but are also becoming important tools in many research areas. Such areas include signal processing, control theory, medicine, speech production and recognition, physics, business, and object pattern recognition.

Artificial neural networks are based on the human biological neural system where a set of neurons are interconnected to each other to process incoming information [58]. They are essentially the generalizations of mathematical models to emulate the human brain in processing and translating information. In general, an artificial neural network consists of a large number of simple processing elements, *neurons* (also called *units*, *cells*, or *nodes*) [59]. These neurons are connected to themselves or others by means of directed communication links, each with an associated weight. Each neuron processes the incoming weighted information from other neurons or the external data based on its defined *activation function* and sends its activation or processed information as a signal to several other neurons.

There are many ways to classify artificial neural networks. They can be classified by one or more of the following: (a) the connecting architecture of neurons, (b) the processing function of a neuron, or (c) the method of determining the weights on the connections. The connecting architecture of neurons is simply the arrangement of neurons in a neural network, and it defines the connection patterns within and between layers. Using this idea, neural networks are often categorized as single layer, multi-layer, or competitive layer.

Based on the processing function of a neuron, neural networks can be classified as linear or nonlinear classifiers [60]. The most popular processing functions (activation function) used by neural networks are (1) identity function, (2) binary or bipolar step function with threshold, (3) binary sigmoid, and (4) bipolar sigmoid. The choice of the activation function is important to a neural network. It has direct influences on the performance of a multi-layer neural network. In most cases, multi-layer networks require nonlinear activation functions. Unlike multi-layer nets, linear and nonlinear activation functions make no difference for single layer nets.

The method of determining the weights on the connections is another way to classify neural networks. It can divide neural networks into the three categories of supervised training, unsupervised training, and fixed-weight. Supervised training is the most typical method of obtaining the weights for a neural network. It uses a set of training vectors with their associated target vectors to update the weights according to its learning algorithm. In unsupervised training neural networks, the training is accomplished by grouping similar input vectors together without knowing each associated target vector. A learning algorithm is responsible to the way of modifying the weights so that the clusters will be formed and exemplars will be produced. As the name states, fixed-weight nets will not update the weights when each input vector is presented. The primary use of this net is to solve the optimization problems that are difficult to solve by traditional methods. Table 3.1 categorizes some typical nets by the above three classifying methods.

One of the primary concerns of this research is the neural network models that can be used for object classification. Since many papers and books have been presented regarding a wide variety of neural networks with their architecture and training algorithms, only several

Table 3.1 Classification of several typical neural networks

	Single-layer	Two-layers	Multi-layer	Competitive layer
Unsupervised Learning	Additive Grossberg Discrete Hopfield Continuous Hopfield OLAM	BAM ABAM TAM DR	SDM CPN	ART1 ART2 Kohenon Maps
Supervised Learning	BSB	Perceptron Adaline LVQ ARP	Perceptron Madaline BM AHC	

networks that are suitable for object pattern classification will be outlined.

3.1.1 The Hopfield net

The Hopfield net developed by Hopfield [61, 62] is an iterative associative neural network with fully interconnected nodes. Figure 3.1 illustrates this single-layer architecture neural net. The figure does not display the full connection. There are actually lateral connections from node a_i to every other others. It uses the supervised training method (Hebb learning rule) to adjust the weight vector as each input vector is presented. It could be categorized as a single-layer iterative autoassociative neural network with a supervised training algorithm. The algorithm of a Hopfield net is outlined as follow:

1. **Initialize weights to store patterns**

$$w_{ij} = \begin{cases} \sum_{p=1}^{M} s_i(p)s_j(p), & i \neq j \\ 0, & i = j, 1 \leq i, j \leq M \end{cases}$$

where, w_{ij} is the weight between nodes i and j; $\mathbf{s}(p), p = 1, ..., M$, is a input pattern with bipolar valued elements.

2. **Initialize initial activation of net equal to the external input vector x**

$$y_i = x_i, (i = 1, ..., N).$$

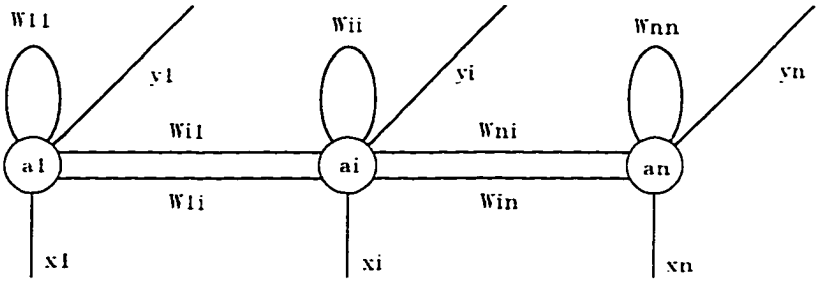


Figure 3.1 The architecture of the Hopfield net.

3. **Iterate until convergence**

$$y_j(t+1) = f_h\left[\sum_{j=1}^{N} w_{ij} y_i(t)\right], 1 \leq j \leq M$$

where f_h is the nonlinear function.

4. **Test for convergence**

If the convergent condition is not satisfied, go back to step 2; otherwise, the training is completed.

The trained net will have a symmetric weight matrix with zero values for all diagonal elements, i.e., no self-connections for every node. The characteristics of asynchronous updating of the units in a Hopfield net allows a Lyapunov function to be found for the net. The convergence of a stable set of activations in a Hopfield net can be then proven by using the Lyapunov function.

After training, the Hopfield net will recall a stored exemplar when it is given an input vector that is sufficiently similar to a vector that it has learned. If there is no stored exemplar that can be recalled due to the unsimilarity of the input vector with the learned exemplars, the net will converge the input vector to a *spurious stable state* that is a new pattern that does not belong to any of the existing exemplars.

As Lippmann stated [65], the use of a Hopfield net is "most appropriate when exact binary representations are possible as with black and white images where input elements are pixel

values, or ASCII text where input values could represent a bit in the 8-bit ASCII representation of each character." Also, it is "less appropriate when input values are actually continuous, because a fundamental representation problem must be addressed to convert the analog quantities to binary values." So the use of a Hopfield net is restricted to binary input vectors. In order to store a relatively large number of patterns, the Hopfield net requires a large number of nodes and a huge number of connection weights to maintain the net with reasonable accuracy. Hopfield [61] shows that the relationship between the number of nodes, n, and the number of patterns, P, that can be stored, is

$$P \approx 0.15n.$$

Another limitation of the Hopfield net is the convergence of an exemplar to other exemplars. This is produced by the result of two or more than two exemplars that share many bits in common.

The Hopfield net above is only suitable for binary input vectors. It is also called the discrete Hopfield net. With the modification of the discrete Hopfield net, Cohen & Grossberg [63] and Hopfield [64] developed the continuous Hopfield net. It has the same topology as the discrete one and its weight matrix is also symmetric. As its name states, the continuous Hopfield net can handle the continuous-valued input vectors. Details for the continuous Hopfield net can be found in references [63, 64].

3.1.2 The Hamming net

The Hamming neural network is one of the fixed-weight competitive nets. It uses Winner-Take-All competition as part of its operation. By setting the weights and the bias of the lower subnet to one-half the exemplar vector and one-half the number of the input nodes, respectively, the lower subnet essentially obtains the values of the number of the input nodes minus each *Hamming distance*. Then, the output of the lower subnet is fed into Maxnet (upper subnet) to find the largest net input. This largest net input to the Maxnet is the stored exemplar with the most similarity to the input vector. The architecture of a Hamming net is shown in Figure 3.2.

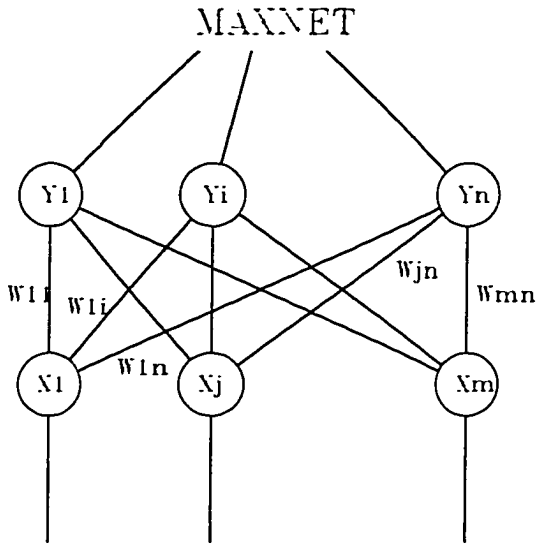


Figure 3.2 The architecture of the Hamming net

The mechanism of the Hamming net is based on the idea of choosing an exemplar vector with the smallest *Hamming distance* to an input vector. The *Hamming distance* is defined as the number of components in which the two vectors differ. The Hamming net is easy to understand and can apply to the situation that no more than one of the exemplars should respond to the same input vector. It has two advantages over the Hopfield net. First, the performance of the Hamming net is at least as good as the Hopfield net if the errors in an image are random and independent. Second, it does not require as many connections as a Hopfield net does. This is because the number of connections in a Hamming net grows linearly as the number of inputs while the number of connections in a Hopfield net grows quadratically as the number of inputs. A comparison of the Hamming and the Hopfield nets can be found in Lippmann, Gold, and Malpass's technical report [66].

Handling continuous-valued vectors in a Hamming net displays a serious problem because of the use of the Hamming distance. For example, a Hamming net with n input nodes has two exemplars, $\mathbf{e}_1$ and $\mathbf{e}_2$, and is going to classify an input vector, $\mathbf{v}$. Suppose the Hamming distance between $\mathbf{e}_1$ and $\mathbf{v}$ is h_1, and between $\mathbf{e}_2$ and $\mathbf{v}$ is h_2. If h_1 is larger than h_2, the Hamming net will find that the input vector is more similar to exemplar $\mathbf{e}_2$ than exemplar $\mathbf{e}_1$.

Now, considering the situation where the largest contribution to h_1 is the *j-th* element of **v**. If the *j-th* element is excluded from the exemplars and the input vector, the hamming distance h_1 could be much smaller than h_2. The Hamming net would find that the exemplar $\mathbf{e_1}$ is closer to the input vector **v**. Since the *j-th* element in the input vector could be noise infected, this net is not appropriate for handling continuous-valued vectors with additive noise. The above situation will not give a controversial result by excluding only one element in a vector for the binary-valued case.

3.1.3 Kohonen self-organizing maps

Kohonen self-organizing maps utilize the square of the Euclidean distance with or without the topological structure information to group a set of continuous-valued vectors into number of clusters [67]. The use of the topological structure information in the algorithm is based on the observed principle of operation in the brain. This information has not been applied to other neural networks.

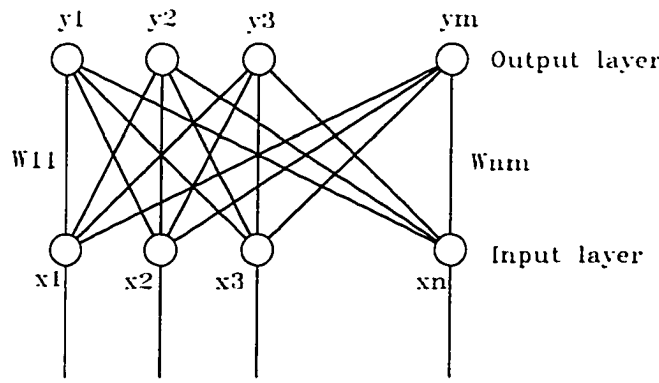


Figure 3.3 The architecture of the Kohonen self-organizing map

A Kohonen self-organizing map usually contains n input units and a one- or two-dimensional array of cluster units (m output nodes) (see Figure 3.3). There are many different topological structures that can be considered among input units to strengthen the correlations between units and shorten the training time. Figure 3.4 shows three typical topological structures for

the winning node and its neighborhoods: (a) linear structure; (b) rectangular structure; and (c) diamond structure. After defining the topological structure, the training vector with n-tuples is matched with the weight vector of each cluster unit to find out their corresponding square of the Euclidean distance. The cluster unit with the smallest square of the Euclidean distance is chosen as the winning unit. The winning unit and its neighboring units that are defined by the topological structure update their corresponding connection weights. The clustering algorithm of the Kohonen self-organizing maps is as follows:

1. **Initialize weights w_{ij}, set topological structure, and learning rate**

   Initialize weights from every input to all output nodes to small random values. Set the initial radius of the neighborhood. Set the learning rate α between [0, 1].

2. **Compute distance to all nodes**

   For each input vector x, compute its square of the Euclidean distance:

$$D(j) = \sum_i (w_{ij} - x_i)^2, \qquad j = 1, ..., m.$$

3. **Find index J such that $D(j)$ is a minimum.**

4. **Update weights to node J and its neighbors**

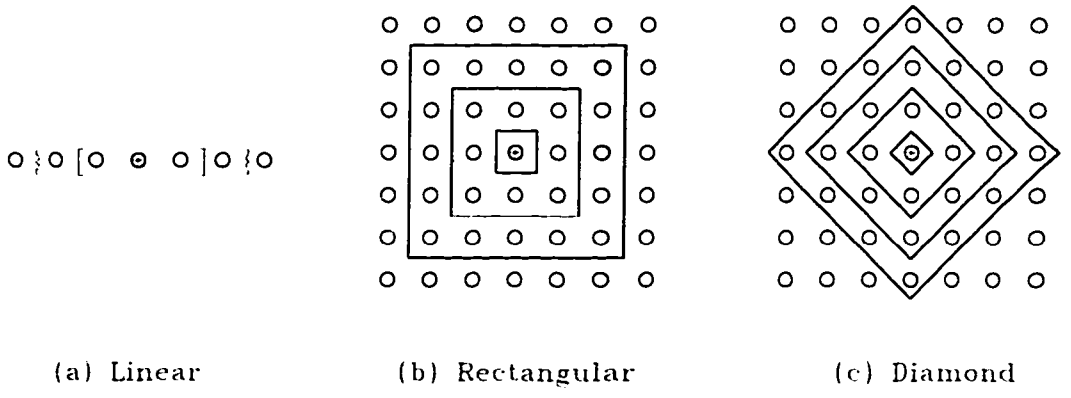
   For all units j within a specified neighborhood of J, and for all i:

$$w_{ij}(new) = w_{ij}(old) + \alpha[x_i - w_{ij}(old)].$$

5. **Update learning rate**

   Linearly or geometrically decrease the learning rate, α, and the size of the topological structure. If the stopping condition is false, go back to step 2.

The Kohonen self-organizing maps are categorized as unsupervised training neural networks, since the way of updating the weights on the connections are self-adjusting based on a Euclidean distance measure of pattern similarity. By the nature of the unsupervised training nets, it is difficult to control the similarity of the input data except through a great deal of intuition and experimentation or using some expensive reset mechanism. In practice, this net



(a) Linear (b) Rectangular (c) Diamond

Figure 3.4 Three typical topological structures for a winning node and its neighborhoods.

requires extensive experimentation with different starting points in order to gain useful insight into the geometrical distribution of the training data. The clustering results depend on the first training data chosen (it defines the initial cluster center), the order in which the patterns are considered, and the geometrical properties of the data. This net requires the training data to exhibit characteristic "regions" which are reasonably well separated in order to yield desirable clusters.

3.1.4 Adaptive resonance theory neural nets

Adaptive resonance theory (ART) neural networks were introduced by Carpenter and Grossberg [68, 69] based on the work of Grossberg in 1976 [70]. They are two-layer unsupervised training neural networks. ART1 is designed for clustering binary data, and ART2 was developed for handling continuous-valued data. The topologies are shown in Figures 3.5 and 3.6. The basic architecture of an ART neural network contains three groups of nodes: the input processing units (the F_G layer), the cluster units (the F_H layer), and the reset mechanism. The F_G layer consists of the input portion (denoted as $F_G(p)$) and the interface portion (as $F_G(f)$). Since one objective of developing ART was to allow the user to control the degree of similarity of patterns placed on the same group, the interface portion is designed to achieve

this objective. It combines signals from the input portion and the F_H layer for use in comparing the similarity of the input signal to the weight vector for the cluster unit that has been selected as a candidate for learning. Therefore, the units in the $F_G(f)$ layer are bi-directively connected to the cluster units in the F_H layer. In Figures 3.5 and 3.6, w and v designate the bottom-up weights and top-down weights, respectively.

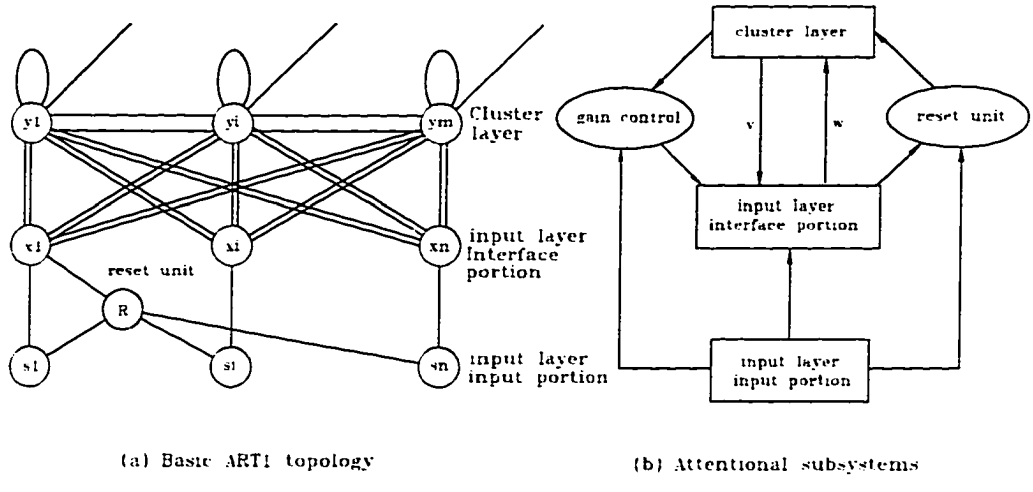
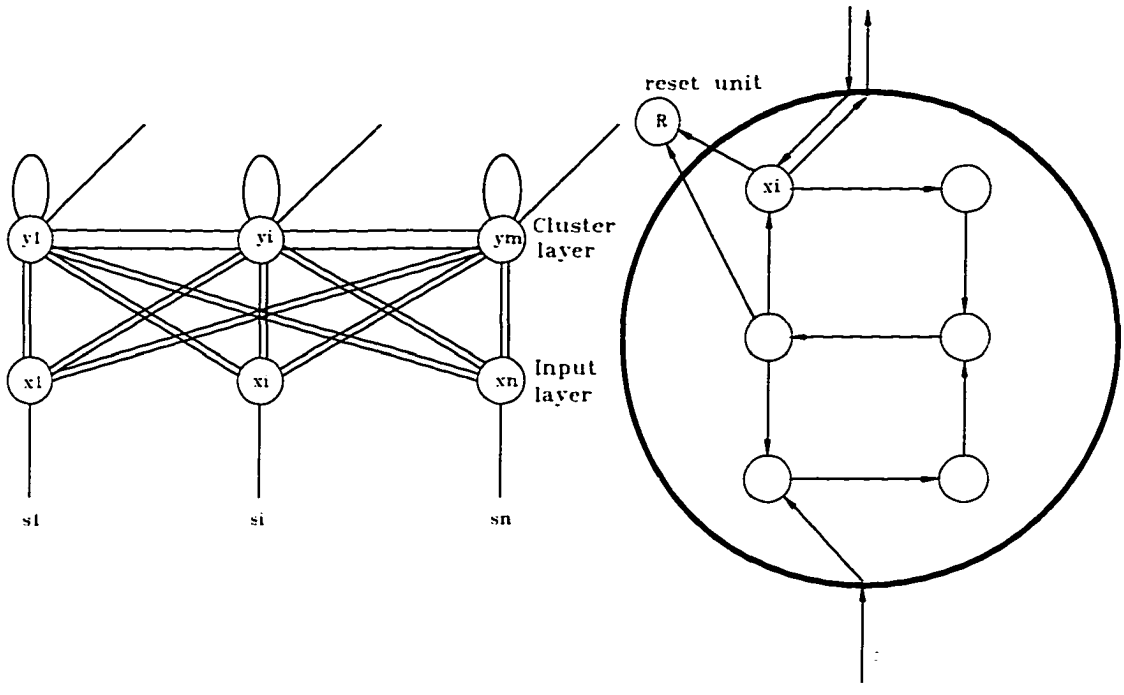


Figure 3.5 The topological structure of ART1

The clustering algorithms of ART1 and ART2 are essentially the same except the difference in the reset condition. The first input data is selected as the exemplar for the first clustering unit. The second input data is then matched with the first clustering exemplar to determine whether or not the distance between them is less than a threshold. If it is less, the second input data is clustered with the first clustering unit. Otherwise, a new clustering unit is assigned to it as a new exemplar. This process is repeated for all input data. The clustering algorithm implemented in ART neural networks generally is as follows:

1. **Initialize weights and parameters**

   Set w_{ij} and v_{ji}, for $1 \leq i \leq n$ and $1 \leq j \leq m$. Also set the *vigilance* threshold ρ, $0 \leq \rho \leq 1$. ρ is the parameter that determines how close an input data must be to a stored exemplar to match.



(a) Basic topology of ART2

(b) An expanded view of Xi unit in the input layer

Figure 3.6 The topological structure of ART2

2. **Apply new input**

   For each input vector, do steps 3-6.

3. **Compute the Matching Scores**

4. **Select the best matching exemplar**

   Find the unit in F_H with largest input to learn the current input pattern.

5. **Test reset condition (Vigilance test)**

   If reset is true, then the current candidate unit is rejected; go back to step 3. Otherwise, the current candidate unit is accepted for learning; proceed to step 6.

6. **Update the weights**

   Change weights according to differential equations.

7. Test stopping condition

If the stopping condition is false, return to step 2. Otherwise. stop the training.

As outlined in the algorithm, the *vigilance* threshold which ranges between 0.0 and 1.0 must be set before any process proceeds. It is the parameter involved in the interface portion $F_G(f)$ of the input processing layer F_G to control the similarity of an input pattern and a stored exemplar. An appropriate value of the *vigilance* threshold is critical for obtaining the useful clustering results. In general, clustering a set of similar patterns requires a value close to one and clustering a set of unsimilar patterns requires only a small fraction of one. In practice, a disturbing noise in patterns is unavoidable. With even a small amount of noise, the clustering algorithms in ART neural networks will create problems. The net may run out of the clustering units to store the exemplars in the noisy condition even though these exemplars can be held in the same clustering unit in perfect condition (no noise), using the same value of the *vigilance* threshold. Extensive experimentation is needed to resolve the feasible value of the threshold.

3.1.5 Back-propagation neural networks

Back-propagation neural networks are a powerful network that overcomes many shortcomings existing in many other nets. A back-propagation neural net is a multi-layer perceptron using a back-propagating error training algorithm. The algorithm that propagates information about errors at the output unit back to the hidden units was discovered in the 1970's [71] and successfully implemented in the 1980's [72, 73, 74]. This training method brought new life to the neural network industry for training multi-layer perceptions.

Perceptron refers to a two-layer pattern classification neural network with linear threshold units. Its original concept was introduced and developed by Frank Rosenblatt [75, 76, 77] and other researchers [78, 79]. It can be used with both continuous valued and binary inputs. The perceptron had the most significant impact to the neural network development in the early research.

A perceptron can be categorized as a two-layer supervised neural net coupling with a

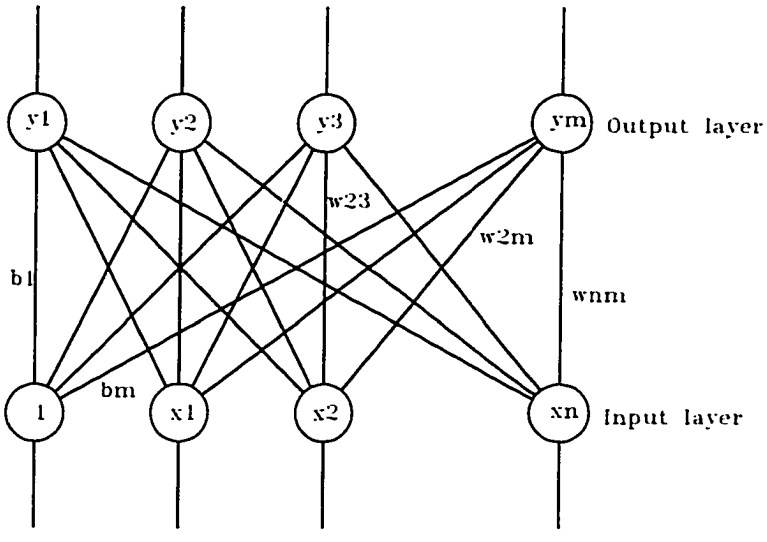


Figure 3.7 The topology of an elementary Perceptron.

nonlinear processing function. A typical perceptron consists of an input layer connected by paths with weights to the output layer (see Figure 3.7). These connection weights are adjusted according to errors occuring in the output nodes using an iterative training method. As a supervised training net, a sequence of training patterns, each with an associated target output vector, are presented to a perceptron neural network. Then, the neural network decides whether an input pattern belongs to one of several classes by comparing the associated target vector with the computed output vector from the output layer. If the output vector does not agree with the target vector, an error occurs for this pattern, and the weights are updated. The original training algorithm developed by Rosenblatt [77] is described with small modifications as follows:

1. **Initialize weights and bias**

   Set weights w_{ij}, for $1 \leq i \leq n$ and $1 \leq j \leq m$, and bias b to small random values. Set the learning rate α within the range of (0, 1].

2. **Present every training pair**

   For each input training pair: input vector and target vector, do steps 3-5.

3. **Calculate actual output**

$$y_j = f\left(\sum_{i=1}^{n} w_{ij}x_i + b_j\right), \qquad \text{for } 1 \leq j \leq m.$$

where, f() could be the bipolar step function,

$$f(y\_in) = \begin{cases} 1, & \text{if } y\_in > \theta \\ 0, & \text{if } -\theta \leq y\_in \leq \theta \\ -1, & \text{if } y\_in < -\theta \end{cases}$$

In the original formula, the threshold $\theta = 0$.

4. **Update weights and bias**

If output vector does not match with the target vector **t**,

$$\begin{aligned} w_{ij}(new) &= w_{ij}(old) + \alpha t_j x_{ij}, \\ b_j(new) &= b_j(old) + \alpha t_j, \qquad \text{for } y_j \neq t_j. \end{aligned}$$

Otherwise,

$$\begin{aligned} w_{ij}(new) &= w_{ij}(old), \\ b_j(new) &= b_j(old). \end{aligned}$$

5. **Test stopping condition**

If no weight has changed in step 2, stop; else, continue.

Note that the threshold, θ, on the bipolar step function (the processing function) is a fixed, non-negative value. From the mathematical point of view, this function produces an "undecided" region that is determined by the chosen value of θ, and that separates one "decided" region from others. Figure 3.8 shows an example for two-dimensional input vectors.

Note also that the perceptron training algorithm only updates the training patterns that do not produce the correct output vectors. Thus, the training time depends on the correct response of the number of training patterns. The learning rate, α, is a factor that also affects the training time. It could be adjusted during the training process to overcome the confliction of fast convergence for new training patterns with changes and averaging of past inputs to provide stable weight estimates.

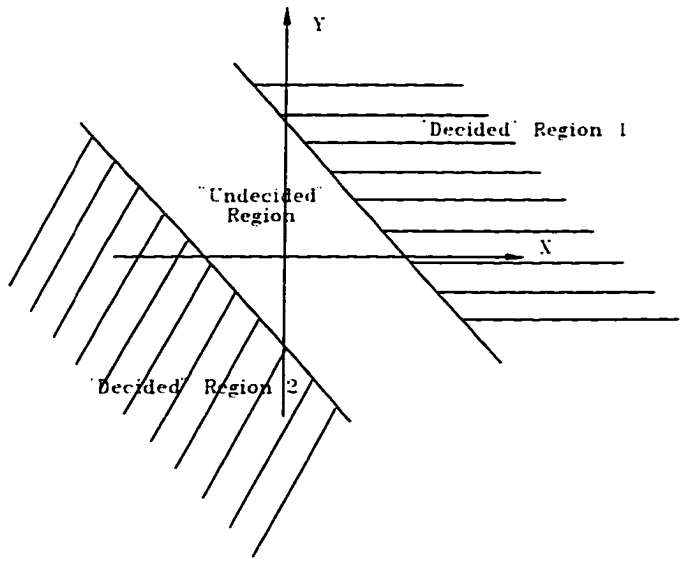


Figure 3.8 The geometric representation of a bipolar step function with threshold θ.

The perceptron training rule has been proven to have more advantages than the Hebb training rule that directly uses the target vectors to update weights without concerning the computed output vectors and the learning rate. Fausett uses two simple examples in his book [59] to demonstrate this advantage for separating the AND function with binary input vectors and bipolar targets. Several researchers [80, 79, 81] have also proven that the perceptron training rule can converge to the correct weights in a finite number of steps to give the correct response for all training patterns. These proofs require an assumption that such weights exist. In other words, there exist some hyperplanes such that each can separate one corresponding class from others (see Figure 3.9). The detail procedure of the proof can be seen in chapter 2 of Fausett's book [59]. The proof is derived by carrying out the inequality,

$$k \leq \frac{M\|\mathbf{w}^*\|^2}{m^2},$$

where k is the number of steps updating the weights; $M = max\{\|\mathbf{x}\|^2$ for all $\mathbf{x}$ in the training set$\}$; w^* is the weight vector that produces the correct output vector for all training input patterns; $m = min\{x \cdot \mathbf{w}^*\}$ and $\mathbf{x}$ is an input training vector. From the preceding inequality, the number of weight updates, k, is bounded by the value of $\frac{M\|\mathbf{w}^*\|^2}{m^2}$. However, this value

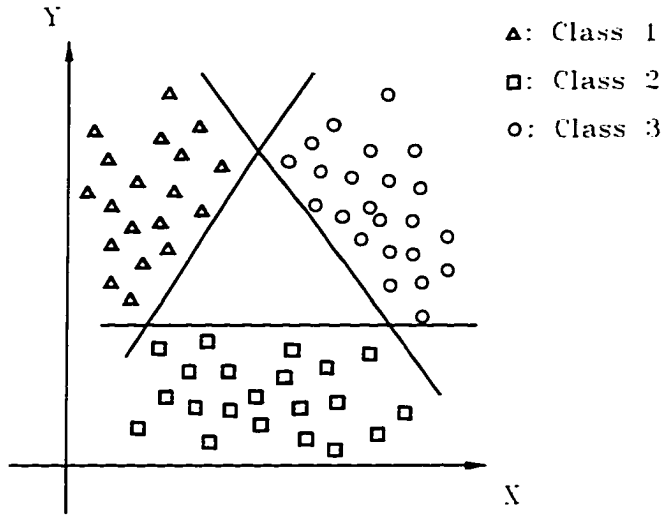


Figure 3.9 Example of hyperplanes that separate one class from others.

becomes bigger, i.e., the training time becomes longer, if some training vectors are very small in norm, i.e., m has a very small value.

The perceptron neural networks obviously have an unavoidable weakness that is common to all single or two-layer neural networks when they are used to classify classes that are well correlated. Figure 3.10 shows two well correlated classes. It is clear that no straight line or hyperplane can separate these two classes. The motivation of overcoming the limitation in single and two-layer networks prompts the development of multi-layer networks.

Although multi-layer perceptrons had been proposed many years ago, they were not widely used until the development of an effective training algorithm in the mid 1980's. A multi-layer perceptron consists of one or more layers between the input and output layers. Conventionally, these layers are called hidden layers. The activation for each node requires a nonlinear processing function. Otherwise, multi-layer perceptrons would not have any advantage over two-layer perceptrons [65], and the extra calculation would be a waste.

A one-hidden layer (three-layer) perceptron (Figure 3.11(a)) is capable of classifying features that can be separated by convex polygons, such as the situation in Figure 3.10, using a step processing function. During the training stage, the first layer processes the incoming

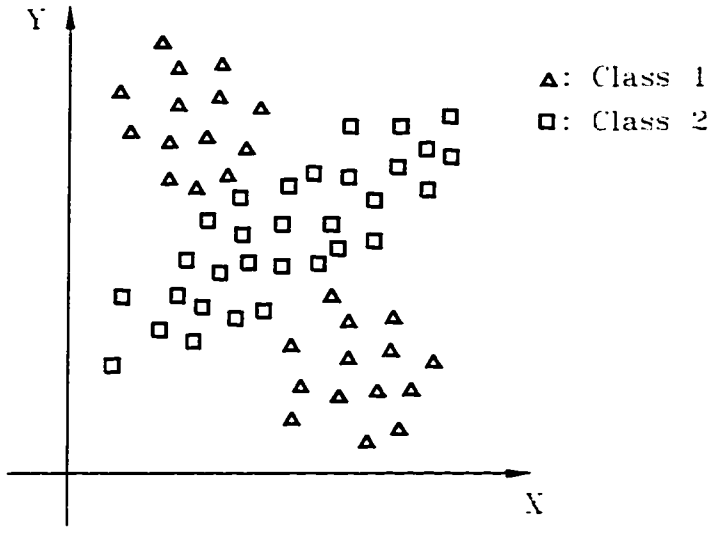
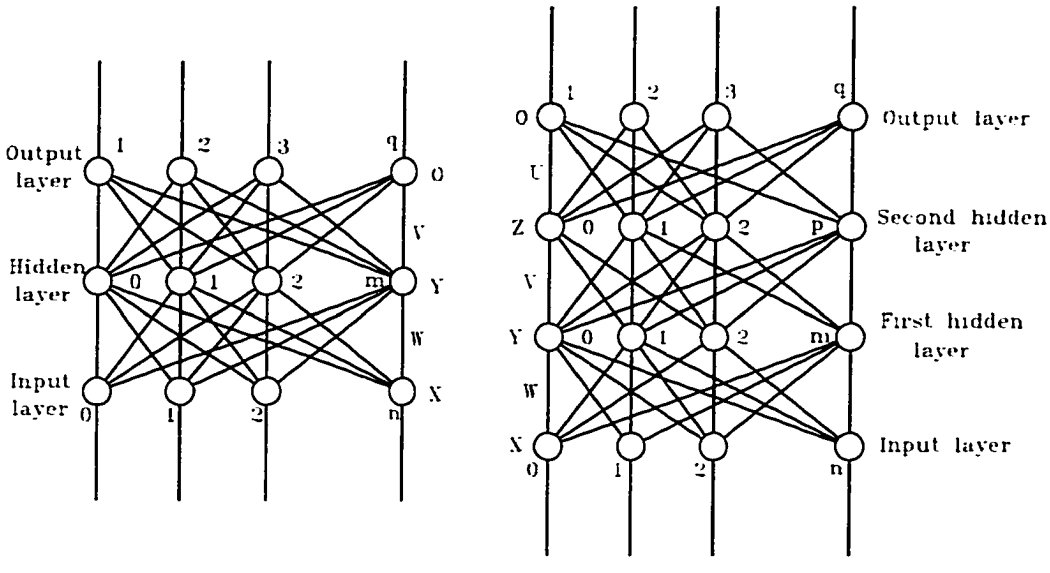


Figure 3.10 Two well correlated classes that can not be separated by a straight line or a hyperplane.

patterns as the same behavior as a single-layer perceptron. Each node in the hidden layer produces a half-plane to divide one class from *some of others.* Using the phrase "some of others" in the proceeding sentence is due to the incapability of separating one class from all the others by only one half-plane. The output layer of the one-hidden layer perceptron performs the logical AND operation to join these half-planes together. This results in a convex polygon that separates one class from others. It is easy to see that the number of sides for the convex polygon resulting from a one-hidden layer perceptron cannot exceed the number of nodes in the hidden layer.

One-hidden layer perceptrons retrieve the limitations inherit with two-layer perceptrons. But they still have problems when they are utilized to recognize classes that cannot be separated by convex polygons, such as those shown in Figure 3.12. To solve the problems existing in one-hidden layer as well as in two-layer perceptrons, two-hidden layer (four-layer) perceptrons (see Figure 3.11(b)) have been studied [82, 83, 84] and implemented in several applications [85]. The progress of the training is much more complicated in a two-hidden layer perceptron. The first hidden layer partitions the training patterns into small hypercubes. If there are n inputs in each pattern, it requires $2n$ nodes in the first hidden layer to define a hypercube, one



(a). One-hidden perceptron. (b). Two-hidden perceptron

Figure 3.11 The architectures of multi-layer perceptron nets.

for each side of the hypercube. The second hidden layer performs the logical AND operation to form a convex hyperhedra by joining the hypercubes created in the first hidden layer. Then, the output layer acts as a logical OR operation to gather the disconnected convex hyperhedra to represent output classes. So, the number of nodes in the second hidden layer cannot be less than the number of disconnected convex hyperhedras. The final decision region from a two-hidden layer perceptron can be any arbitrary shape. This is because the OR operation in the output layer essentially unions each individual convex region together to produce a concave, discontinued, and/or looped region. From the classification capability of the two-hidden layer perceptron, it is easy to see that there has been no research conducted with more than a four-layer perceptron.

The back-propagation training algorithm is one of the most successful multi-layer training algorithms that is suitable in multi-layer perceptron neural networks. The *back-propagation training rule*, also called the *generalized delta rule*, is simply an iterative gradient descent

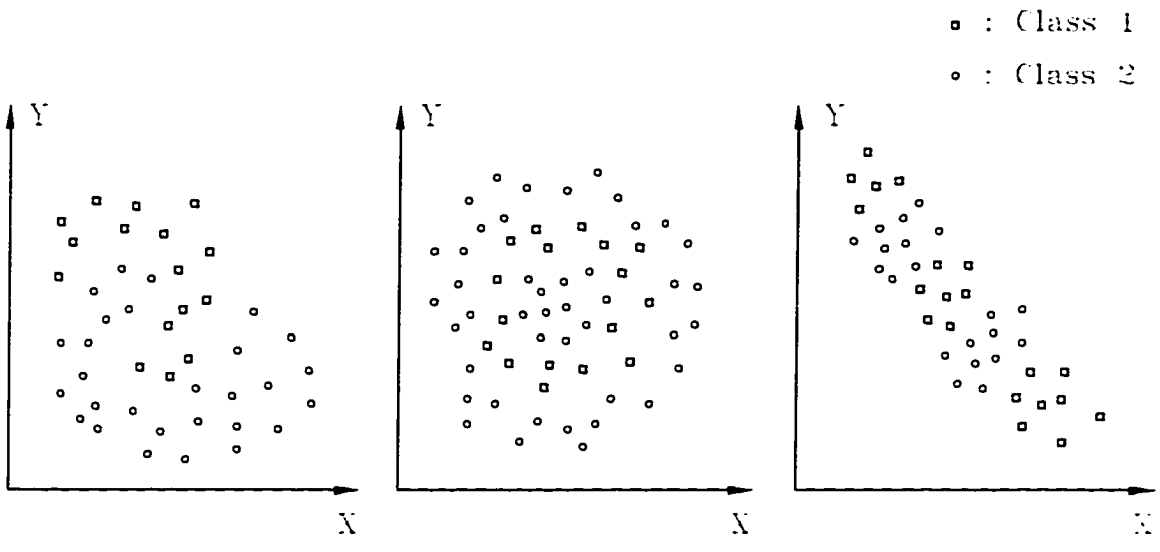


Figure 3.12 Classes that cannot be separated by convex polygons

method to minimize the total mean square error between the actual output computed by the net and the desired output. It uses a binary or bipolar sigmoid function as the processing (activation) function to achieve the requirement of continuous differentiability. The training process involves three stages: (1) feed-forward of inputs, (2) back-propagation of errors, and (3) adaptation of weights.

The first stage is similar to the perceptron training method to compute the output vector. Each layer obtains the input signals from the lower layer or the input patterns and computes each activation, then sends the computed results to the next layer or the output. The second stage is the essential component of the algorithm. The computed output vector is compared with the target vector to determine each associated error in an output unit for that input pattern. Then, the errors are formulated in a specific way and are distributed back to the previous layer. By doing this iteratively, the errors associated with the units in every layer are defined. The last stage updates the connection weights in every layer based on the errors associated with the units. The back-propagation training algorithm is outlined as follow:

1. **Initialize weights and bias**

   Set all weights and bias to small random values.

2. **Present every training pair**

   For each input vector **X** with n elements and target vector **T**, do steps 3-9.

3. **Compute the output signals for the first hidden layer**

   Each hidden unit $(Y_j, j = 1, ..., m)$ sums its weighted input signals and applies its processing function to compute its output signal,

$$y_j = f(w_{0j} + \sum_{i=1}^{n} x_i w_{ij}),$$

   and sends this signal to all units in the layer above.

4. **Compute the output signals for the second hidden layer**

   If it is not a two-hidden layer perceptron, go to next step; otherwise, do the following: Each hidden unit $(Z_k, k = 1, ..., p)$ sums its weighted input signals and applies its processing function to compute its output signal,

$$z_k = f(v_{0k} + \sum_{j=1}^{m} y_j v_{jk}),$$

   and sends this signal to all units in the layer above.

5. **Compute the output signals for the output layer**

   Each output unit $(O_l, l = 1, ..., q)$ sums its weighted input signals and applies its processing function to compute its output signal,

$$o_l = f(u_{0l} + \sum_{k=1}^{p} z_k u_{kl}).$$

   The processing functions in the above three steps are the sigmoid function,

$$f(x) = \frac{1}{1 + exp(-x)},$$

   or

$$f(x) = \frac{2}{1 + exp(-x)} - 1.$$

6. **Compute the errors in the output layer**

Each output unit $(O_l, l = 1, ..., q)$ computes its error term,

$$\delta_l = (t_l - o_l) f'(u_{0l} + \sum_{k=1}^{p} z_k u_{kl}).$$

7. **Compute the errors in the second hidden layer**

If it is not a two-hidden layer perceptron, go to next step; otherwise, do the following:

Each hidden unit $(Z_k, k = 1, ..., p)$ calculates its error term,

$$\delta_k^1 = f'(v_{0k} + \sum_{j=1}^{m} y_j v_{jk}) \sum_{l=1}^{q} \delta_l u_{lk}.$$

8. **Compute the errors in the first hidden layer**

Each hidden unit $(Y_j, j = 1, ..., m)$ calculates its error term,

$$\delta_j^2 = f'(w_{0j} + \sum_{i=1}^{n} x_i w_{ij}) \sum_{k=1}^{p} \delta_k^1 v_{kj}.$$

9. **Update weights and biases**

Update the bias and weights connecting the second hidden layer and the output layer:

$$u_{kl}(new) = u_{kl}(old) + \alpha \delta_l z_k,$$

$$u_{0l}(new) = u_{0l}(old) + \alpha \delta_l.$$

Update the bias and weights connecting the first hidden layer and the second hidden layer:

$$v_{jk}(new) = v_{jk}(old) + \alpha \delta_k^1 y_j,$$

$$v_{0k}(new) = v_{0k}(old) + \alpha \delta_k^1.$$

Update the bias and weights connecting the input layer and the first hidden layer:

$$w_{ij}(new) = w_{ij}(old) + \alpha \delta_j^2 x_i,$$

$$w_{0j}(new) = w_{oj}(old) + \alpha \delta_j^2.$$

10. **Test stopping condition**

    Repeat step 2 until δ_l, for each $l = 1, ..., q$, and every input training pattern is either sufficiently small or zero.

Back-propagation neural networks are basically the multi-layer perceptron neural networks coupling with the back-propagation training algorithm. They have the same properties as the multi-layer perceptron described before. However, the use of the Sigmoid function instead of the step function as the processing function results in a curved boundary for the final decision regions. The advantage of regions with curved boundaries is obvious. The number of nodes in the hidden layers and of course the training time can be reduced. In the other hand, the primary goal of the back-propagation training algorithm is to minimize the total mean square error in the output layer, so it does not guarantee to find the global minimum error. Thus the decision regions after training for these nets might not be the optimum regions, and therefore these nets might not classify classes perfectly. Also, Back-propagation neural networks usually need extensive computation and require a long period of training time. They are incapable of online training.

3.2 Comparison

The neural networks presented above are, in some forms, suitable for pattern recognition. Many pattern recognition systems have been developed or implemented using these networks either as the pre-processor or classifier. In most cases, the advantages or the disadvantages of using these networks in pattern recognition systems are application dependent. In this section, the neural networks discussed in the previous section are compared with respect to the characteristic properties: *learning abilities*, *training stabilities*, *handling capabilities* of input data, *computational complexities*, *noise handling abilities*, *generalization capabilities*, *storage capacities*, and *mapping abilities* without considering any specific application.

Before we proceed to the main point of this section, we need to define the characteristic properties for a neural network. The *learning ability* of a neural network is defined as its ability to do *online* training. If a neural network has the *online* training ability (also called *plasticity*),

it will learn (or cluster) unseen patterns properly without completely re-training other patterns that have been learned by the network before. Otherwise, a network is limited to only *offline* training. A recognition system, of course, desires a classifier or recognizer with an online training ability. *Training stability* refers to the weight changes reaching equilibrium during training for the neural networks with only feed-forward connections, such as a back-propagation net, or the activations reaching equilibrium for a recurrent net, such as a Hopfield net that has feedback connections. It is the measure of the convergence of a neural network. Some networks may convergent to find the global error minimum or the global optimum solution; some may only reach the local error minimum or the local optimum solution; others, at the worst, cannot even find a local error minimum. The occurrence of the last case is unusual. since most neural networks have been designed, in some way, to achieve at least a local error minimum or to find a local optimum solution. But the improper use of training patterns in a particular neural network leads to severe oscillations in the weight changes and results in a long period or even an infinity of training time.

The ability of a neural network to handle data defines its *data handling ability.* Some networks are developed to handle only binary or bipolar patterns. However, most of them are designed to deal with analog data. Obviously, analog data catches more detailed information within patterns than binary or bipolar data does. The *computational complexity* is simply the time required in each training or recognizing *epoch.* A training *epoch* is one cycle through the complete set of training patterns and a recognizing *epoch* is defined as one presentation of each input pattern. Higher computational complexity means longer training and recognizing time.

Images obtained from any kind of sensor can not avoid carrying some sort of noise. Although the steps of image processing involved in most systems reduce noise, images still contain noise. The ability of handling a noisy image is necessary in the classification stage. Thus, as a classifier or recognizer in a recognition system, a neural network with good *noise handling ability* will have reasonable responses to noise affected input patterns and is always desirable. This ability in a network is strongly related to its *generalization ability.* The *generalization ability* is the ability to produce reasonable responses to input patterns that are similar to some training

patterns. In general, a neural network having a good generalization ability will possess a good noise handling ability.

Another property that measures the performance of a neural network is the *storage capacity*. It is defined as the number of patterns that a network can store relative to its number of processing units. The last characteristic property of a neural network discussed in this section, the *mapping ability*, is an important factor to be concerned about in most applications. Usually, the applications in pattern recognition involve classifying complex data regions from simple convex to disconnected concave polygons. The degree of complexity of regions a neural network can handle defines its mapping ability.

Table 3.2 shows a comparison of the neural networks outlined in the previous section, along with their training algorithms.

3.3 Application to This Research

In this research, an object pattern recognition system is implemented using a neural network as the classifier. The more information about the original images that is contained in the training patterns, the higher the rate of identification that the classifier will have. Thus, the neural network considered here needs to have an analog (or continuous) data handling ability. From Table 3.2, the discrete Hopfield net, Hamming net, and ART1 net are not suitable for this application. Although the continuous Hopfield net has the ability to handle analog patterns, it has a low storage capacity, especially when the number of input nodes (the dimensions of input vector) is large. It is not chosen as the classifier since the dimensions of the input vector are large and the number of stored patterns needs to be a reasonable high value in this proposed method.

As discussed above, noise is unavoidable in pattern images. So, a neural network classifier that has a good noise handling ability is certainly expected. Even though the ART2 net has the unique property of online learning ability, it is not selected as the classifier in the implementation, due to its poor performance in a noisy environment.

Table 3.2 A comparison of neural networks

Property	Discrete Hopfield	Continuous Hopfield	Hamming	Self Organizing Map
Learning Algorithm	supervised	supervised	competitive	competitive
Learning Ability	offline	offline	online	offline
Training Stability	stable	stable	stable	oscillation to stable
Data Handling Ability	binary	analog	binary	analog
Computational Complexity	extensive	extensive	simple	median
Noise Handling Ability	good	good	good	fair
Generalization Capability	good	good	good	fair
Storage Capability	low	low	median	median
Mapping Ability	poor	fair	poor	good

Table 3.2 (Continued)

Property	Learning Vector Quantizer	ART1	ART2	Back Propagation
Learning Algorithm	supervised	unsupervised	unsupervised	supervised
Learning Ability	offline	online	online	offline
Training Stability	oscillation to stable	stable	stable	oscillation to stable
Data Handling Ability	analog	binary	analog	analog
Computational Complexity	extensive	extensive	extensive	extensive
Noise Handling Ability	fair	poor	poor	excellent
Generalization Capability	fair	poor	poor	excellent
Storage Capability	median	high	high	high
Mapping Ability	fair	fair	fair	fair to excellent

The Kohonen self-organizing map uses a competitive training algorithm. It can cluster arbitrary analog pattern regions and has a similar performance to the back-propagation neural network considering the properties listed in Table 3.2. As a neural network based on competition, however, it may use one or more output units to represent one class. This requires an extra mechanism or another network to gather these output units in order to have the unit respond to the same class of input patterns. The Learning Vector Quantizer (LVQ), an extension of the Kohonen self-organizing map, also has a similar performance to the back-propagation neural network. But its initialization of connection weights needs to have some knowledge about the distribution of the input data.

The back-propagation neural network has been chosen as the classifier in the object pattern recognition system in this study because of its superior mapping ability, noise handling ability, and generalization capability. The primary concerns when using it are the training stability (or convergence) and the computational complexity. Several efforts will be considered in the implementation of the neural network classifier to increase the training stability and quicken the training time. Also, a compact data representation is necessary to decrease the number of input nodes, and consequently the computational complexity. Two successful algorithms will be presented in Chapter 5 for obtaining a set of compact training data. Several algorithms were attempted and some were not successful for this study. Before presenting the algorithms, the Radon transform of an image that uses the algorithms will be outlined.

4 THE RADON TRANSFORM

In this chapter, the motivation of using the Radon transform of an image as the recognition feature is outlined in the first section. Then, the mathematical overview of the Radon transform is presented in the second section, followed by the third section which describes its suitability of coupling with artificial neural networks.

4.1 The Radon Transform in Object Pattern Recognition

Due to the large amount of image data contained in the usual pixel coordinate format (a two dimensional array of numerical values), image data manipulations are always a problem for many applications involving a large size of image. The utilization of alternative representations for image data is a definite solution for reducing the computational complexity of various applications in image processing as well as object pattern recognition. The Radon transform provides this alternative.

The theory of the Radon transform [86] presented by Radon in 1917 is the key to many application problems. A well-known medical application is the computerized tomographic scanner developed by Hounsfield who shared the Nobel prize in medicine with Cormack in 1978. The Radon transform has been applied to many fields other than medicine, such as chemistry, economics, image processing, and pattern recognition. In the area of image processing, it has been used for image analysis [87, 88], image segmentation [89, 90], image filtering & restoration [91], and image reconstruction [92, 93, 94].

Another area for applying the Radon transform is object pattern recognition, which is the primary concern in this research. Many researchers have used it to derive specific features [91, 95, 96], and others have developed recognition systems using features extracted from the Radon

representation of images for recognizing characters [97, 98, 99] or objects [52, 100, 101, 102]. In general, most of these algorithms follow the underlying concept of image reconstruction from its projections to construct features for recognition. Features that can be used to reconstruct an image suggest that they are adequately representing the image. In fact, with an acceptable reconstruction error, features collected from a finite set of image projections can approximately reconstruct an image by using the inverse Radon theory. These features can be essentially used as the input patterns or data to classifiers for training or classifications.

The Radon theory provides a suitable representation of a 2D image from a set of one-dimensional features. Although it is possible to extend this theory to reconstruct a 3D object from a set of 2D images, it requires a very large amount of computer storage and processing time. The direct use of 2D image data for the application of 3D object pattern recognition is impractical for the same reason. But the ideas motivate our search for new features representing 3D objects.

The main core of this research is to develop a 3D object pattern recognition system. The derived features are not intended to reconstruct 3D objects, but they will exhibit the desirable feature characteristics, a wide domain and approximately unique. Although some constraints are specified in order to obtain these features, they are based on the conditions of a manufacturing environment and can be easy to achieve, especially in the field of robotics assembly and automatic quality inspection. The feature observation and constraint specification will be described in detail in the next chapter, but first the mathematical representation of the Radon transform and its properties are outlined in the following section.

4.2 The Radon Transform and Its Properties

4.2.1 The Radon transform

The theory of the Radon transform, or projection, of a two-dimensional function is well-established mathematically. It can be stated as follows. Let $f(x, y)$ denote a two-dimensional function. A line, $xcos\theta + ysin\theta = t$, passing through $f(x, y)$ is called a *ray* (Figure 4.1). A *ray integral* is derived by integrating $f(x, y)$ along a ray, and a set of ray integrals forms a

projection, sometimes called a *ray sum.* Thus, the definition of the Radon transform of a two-dimensional function $P_\theta(t)$ can be stated as its ray integral along a ray inclined at an angle θ from the y axis at a distance t from the origin. That is,

$$\begin{aligned} P_\theta(t) &= \int_{ray_t} f(x,y)ds \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\delta(xcos\theta + ysin\theta - t)dxdy, \end{aligned} \tag{4.1}$$

$$-\infty < t < \infty, 0 \leq \theta < \pi.$$

From equation (4.1), $P_\theta(t)$ is a function of t at a specific value of angle θ. A *parallel projection* can be derived by taking a projection along a set of parallel rays (Figure 4.2). And the so called *fan-beam projection* can be generated by integrating a function along a set of rays emanating from a point source (Figure 4.3).

The Radon theory can be extended to deal with projections of n-dimensional functions $f(\mathbf{x})$, where $\mathbf{x}$ is a point in $\Re^n$ Euclidean space. Let $a_1x_1 + a_2x_2 + \cdots + a_nx_n = \mathbf{a}\cdot\mathbf{x} = t$ be a hyperplane in $\Re^n$ Euclidean space, then the Radon transform of a function $f(\mathbf{x})$ along this hyperplane is:

$$P_{\mathbf{a}}(t) = \int f(\mathbf{x})\delta(\mathbf{a}\cdot\mathbf{x} - t)d\mathbf{x}. \tag{4.2}$$

$P_{\mathbf{a}}(t)$ is defined on $\Re \times T^{(n-1)}$, where $T^{(n-1)}$ is the unit hypersphere in $\Re^n$ space.

Since an image contains two-dimensional digitized data and can be represented as equation (2.1), a digital version of the Radon transform can be applied to generate projection data from an image. A digital version of the Radon transform can be obtained by approximating the integral in function (4.1). It can be formulated as follows:

$$P_\theta(t) \cong \sum_{(x_k,y_k)\in ray_t^d} f(x_k,y_k)\Delta_k, \tag{4.3}$$

where $f(x_k,y_k)$ is the pixel intensity at (x_k,y_k) location, Δ_k is the Euclidean distance between pixels at (x_k,y_k) and (x_{k-1},y_{k-1}) locations, and ray_t^d is a digital approximation to ray_t.

Although function (4.3) is not the only way to approximate the integral in function (4.1), it can be implemented easily and does not require extra computational efforts such as the use of interpolation or a weighting mechanism [16]. If all digital rays in every projection are

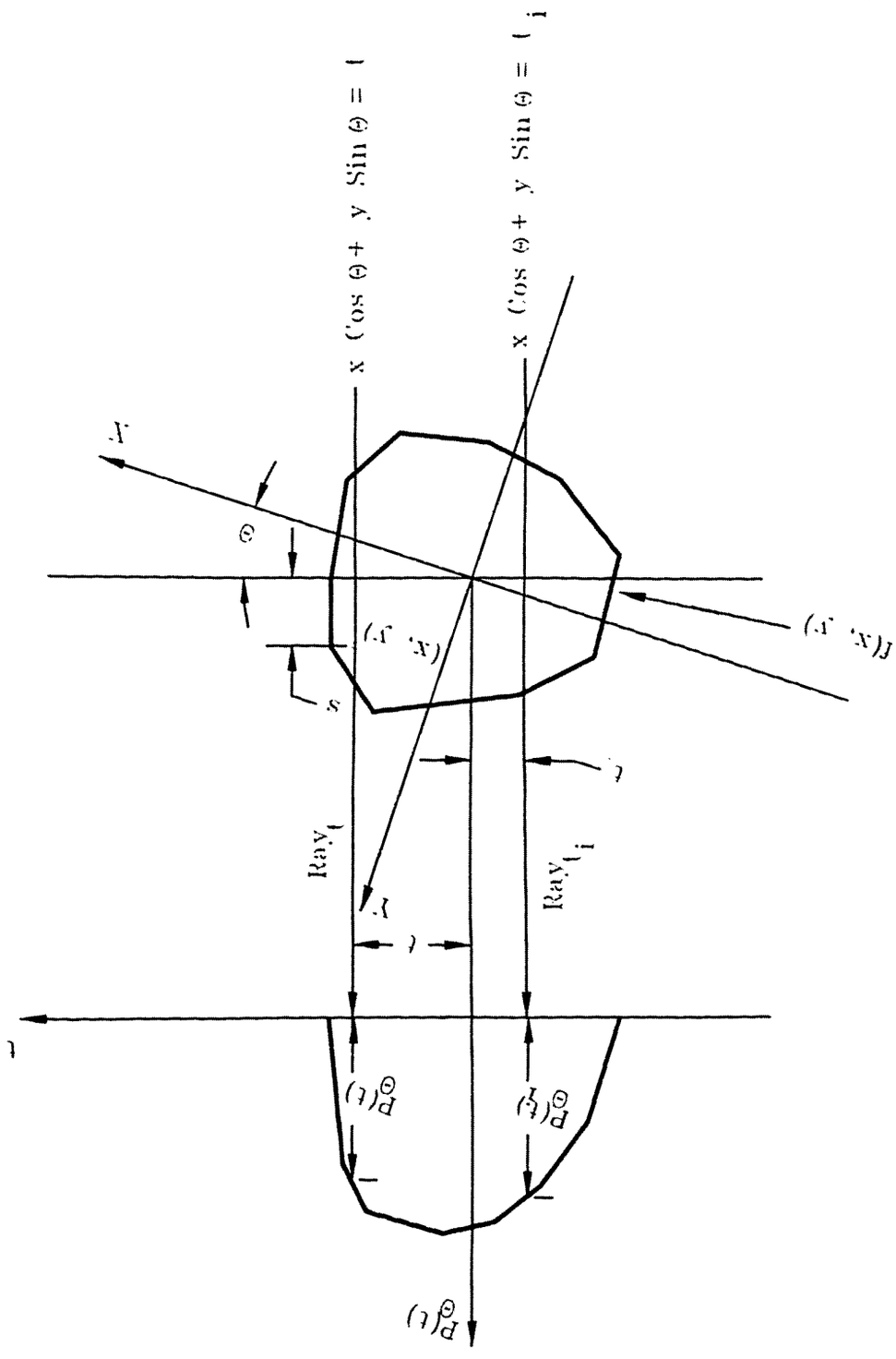


Figure 4.1 The projection function $P_\theta(t)$ of $f(x, y)$ at angle θ.

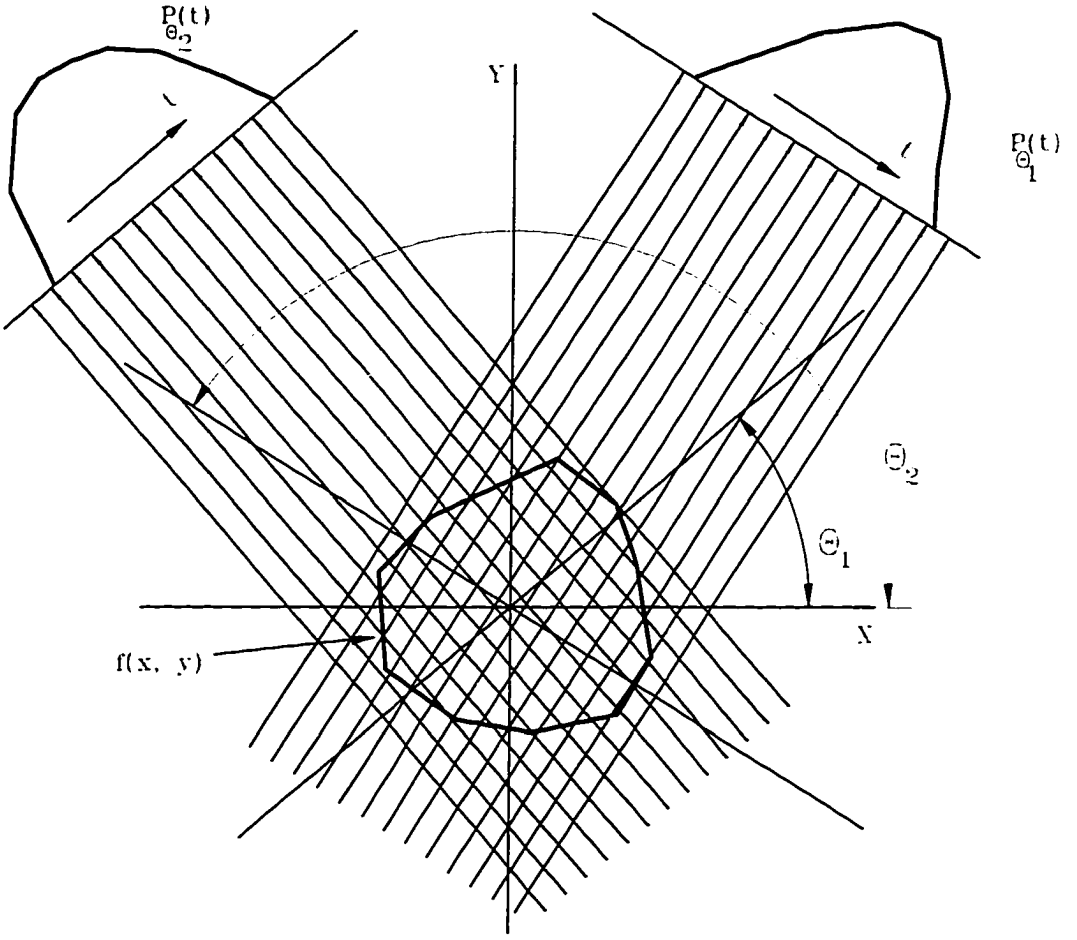


Figure 4.2 Parallel projection.

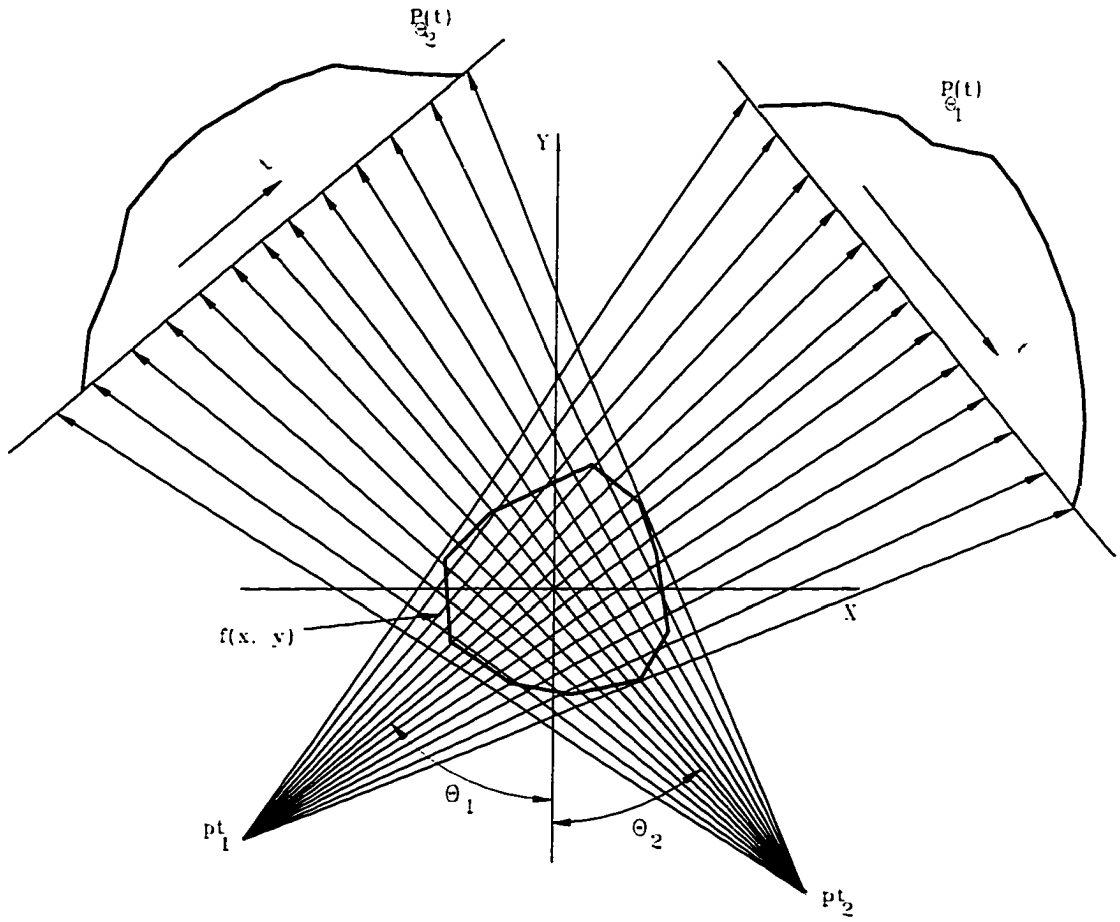


Figure 4.3 Fan-beam projection.

designed to have the same length and in such a way that each pixel belongs to one, and only one, ray, the computational time for observing the digital projection data is much faster and the function (4.3) can be simplified as

$$P_\theta(t) \cong \Delta \sum_{(x_k,y_k)\in ray_t^d} f(x_k, y_k). \tag{4.4}$$

It is not intended for this research to concern with image reconstruction from its projections, but image reconstruction from its projects suggests that projection data uniquely represent a specific image. Therefore, these data or features are suitable for pattern recognition. Essentially, image reconstruction from its projections is the inverse operation of the Radon transform of an image at various angles. It is based on the back-projection theorem that is formulated as

$$f(x,y) = \int_0^\pi Q_\theta(xcos\theta + ysin\theta)d\theta, \tag{4.5}$$

where

$$Q_\theta(t) = \frac{1}{2\pi^2}\int_{-\infty}^{\infty}\left[\frac{dg_\theta(s)}{ds}\right]\frac{1}{t-s}ds. \tag{4.6}$$

Basic to the back-projection theorems is the Fourier slice theorem which relates the one-dimensional Fourier transform of a projection of a function $f(x,y)$ to its two-dimensional Fourier transform. Let $F(u,v)$ be the Fourier transform of the image $f(x,y)$, and $G_\theta(w)$ be the Fourier transform of the projection $P_\theta(t)$ from the Fourier slice theorem, then

$$F(w,\theta) = G_\theta(w), \tag{4.7}$$

if $F(w,\theta)$ denotes the values of $F(u,v)$ along a line at an angle θ with the u axis.

The theorems of the Radon transform and inverse Radon transform were briefly outlined in this subsection. They have been applied in the field of three-dimensional object reconstruction. It is rare to find algorithms that directly use the Radon transform of images from a regular camera as the primitives on the basis of pixel intensities to reconstruct three-dimensional objects. This is due to the difficulty of observing depth information from pixel intensities, which is important to reconstruct three-dimensional objects. In fact, most of these algorithms are based on data that comes from electron micrographs, ultrasonic/x-ray tomography, or a

radiograph. These algorithms are often developed for medical applications. Therefore, we should not be surprised by the lack of recognition algorithms based on the Radon theorem in the field of manufacturing applications. To complete this section, the properties of the Radon transform are described below.

4.2.2 The properties

The Radon transform inherits several useful properties that can be utilized to obtain invariant features from images for object pattern recognition. These properties can also be used for other applications, such as finding an object's location and orientation, reducing the number of projections in the process of image reconstruction, and increasing the computational speed in obtaining projection data.

Let $P_\theta(t)$ or $\mathbf{R}\{f(x,y)\}$ be the Radon transform of an image with its two dimensional function $f(x,y)$ in Cartesian coordinates. Then, the property of *linearity* can be written as

$$\mathbf{R}\{c_1 f_1(x,y) + c_2 f_2(x,y)\} = c_1 g_\theta^1(t) + c_2 g_\theta^2(t). \tag{4.8}$$

The properties of *symmetry* and *periodicity* are,

$$P_\theta(t) = P_{\theta \pm \pi}(-t), \tag{4.9}$$

and

$$P_\theta(t) = P_{\theta + 2n\pi}(t), \tag{4.10}$$

respectively. The *scaling*, *translation*, and *rotation* properties can be formulated as

$$\mathbf{R}\{f(\rho x, \rho y)\} = \frac{1}{|\rho|} P_\theta(\rho t), \tag{4.11}$$

$$\mathbf{R}\{f(x - x_0, y - y_0)\} = P_\theta(t - x_0 cos\theta - y_0 sin\theta), \tag{4.12}$$

and

$$\mathbf{R}\{f_p(r, \phi + \phi_0)\} = P_{\theta + \phi_0}(t), \tag{4.13}$$

if $f_p(r,\phi)$ denotes the image function in the polar coordinates.

Another property of the Radon transfrom is the *space limitedness*. It states that, if

$$f(x,y) = 0, \qquad for \qquad |x|, |y| > \frac{A}{2}$$

then

$$P_\theta(t) = 0, \qquad for \qquad |t| > \frac{A}{2}.$$

The *mass conservation* is

$$M = \int\int f(x,y)dxdy = \int P_\theta(t)dt. \tag{4.14}$$

The last property of the Radon transform is the *convolution* property. The Radon transform of a convolution is the convolution of the Radon transform, and can be mathematically represented as

$$\mathbf{R}\{f_1 * f_2\} = P^1 * P^2. \tag{4.15}$$

The next section provides a brief overview of the feasibility for using projection data as input to artificial neural networks. The detailed mathematical representation and constraint specifications are shown in the next chapter.

4.3 The Feasibility of The Radon Transform in Artificial Neural Networks

Obviously, every image has a set of projection data. And as stated earlier, a set of projection data taken at various angles can approximately uniquely represent an image. If this data set is manipulated appropriately, one can achieve the invariance to object translation, rotation, and scaling within an image. Therefore, this data set can be utilized as the input data to a classifier for recognizing a two-dimensional object.

If the Radon transform of an image is obtained on the basis of image intensity-levels, the projection data is well suited for the use of quantitative basis classifiers such as neural networks introduced in the previous chapter. The mapping or classifying capability of neural networks makes this algorithmic scenario possible. Especially, a four-layer perceptron using the backpropagation of error training method can separate a data set of any complexity. This

is extremely useful for classifying sets of projection data from different images that present looped or concave characteristics.

The advantages of using neural networks as classifiers are manyfuld. They do not require a library to store example objects and thus, there is no need to develop searching mechanisms. The only storage being occupied using neural networks is the well-trained weights. The recognition speed is very acceptable if a neural network has a moderate size of processing elements. Also, many neural network applications do not consider any structure relation between edges, regions, or objects in images when characteristic features are extracted. They do not require experienced personnel or extensive study of the object classes to design a recognition system. But at the same time, the disadvantages are easy to see. They cannot classify multiple objects in the same image. If the number of processing elements is large, the recognition speed is slow thus causing difficulty in matching the requirements for manufacturing applications.

The discussion up to now has considered only a two-dimensional object recognition scheme. The development of three-dimensional object recognition is much more complicated and difficult. If the Radon transform of a three-dimensional scene to a two-dimensional image is directly used as the input data to a classifier, a large number of two-dimensional images covering every possible view of a three-dimensional object needs to be acquired to have a unique set of data representing the object. It takes a tremendous amount of computer storage for hundreds of two-dimensional images. The recognition speed then becomes a severe problem, especially for neural network classifiers. Therefore, without invoking some constraints, it is difficult to develop a three-dimensional object recognition system that is suitable in today's manufacturing environment.

In the following chapter, three algorithms for obtaining effective object representation are presented along with several neccessary constraints that will help to achieve the goal of fast training and recognition of a three-layer perceptron neural network. Since the objective of this research is to accomplish the goal of automatic manufacturing, all the constraints specified here are well suited for the applications of robotic assembly or automatic quality inspection.

5 OBJECT PATTERN RECOGNITION BASED ON EFFECTIVE OBJECT REPRESENTATION

Data accurately representing an object is difficult to obtain in the application of 3-D object recognition development. Although there are several studies of this kind, some of them are model-based techniques as mentioned in Chapter 3. This technique has many advantages over the others. One advantage is the simplicity of its implementation. It requires only to create a set of prototype objects and find a set of representative features that can be utilized as the measure of similarity. It is capable of recognizing a single object in a scene ranging from a simple to complex configuration. Since the representative features are vertices, edges, and angles of edges, some extra processes must be involved besides image enhancement and segmentation in order to obtain non-distorted features. Also, this technique requires an efficient retrieval mechanism to search a model's database, and its classifying speed is relatively slow.

Another technique that utilizes the topological relation has the goal of recognizing multiple objects in the same scene. Its primary features are also the vertices, edges, regions, or/and angles of edges. It employs similar image processing procedures as those required in the model-based technique. Without considering the extra quantitative measure of features, this technique will not be capable of classifying similar or scaled objects since it only makes use of the topological relation between chosen features.

Using neural networks as a tool in the research of object recognition is developing rapidly today. Neural networks have been replacing many traditional techniques in object pattern recognition systems in the areas of image pre-processing [53, 55, 103], image processing [55, 54, 104, 105], feature extraction [106, 107, 108, 109], and classification [51, 52, 109, 110]. As classifiers, most of the successful research in the field of object recognition emphasizes 2-D

objects. The insufficiency of collectible data representing the shape of an object is the major problem blocking the development of 3-D object recognition using neural networks.

One effort of this research is to focus on finding a set of reliable data to represent an object without losing computational power or decreasing neural network recognition performance. In this research, the classification based on the feed forward PNN with the back-propagating error training algorithm will be optimized using reliable data in order to reduce the training and recognition time, as well as to increase the recognition flexibility and accuracy of the object recognition.

The proposed algorithms adopt the intensities of image pixels as the measurement primitive to construct features. However, it does not directly use each individual pixel intensity. This is because the number of pixels in an image may be very large. For instance, a 256 pixel-wide and 256 pixel-height image contains $256^2 = 65536$ pixels. Using the features constructed from the intensities of all these pixels will seriously increase the number of required input nodes of a neural network and result in an extremely long period of training and recognition time. This makes such an algorithm impractical. In this research, the features representing objects are derived based on the property of the Radon transform of an image. As outlined in previous chapters, the goal of feature extraction is to reduce the complexity or dimensionality of input vectors. The extracted features should have the characteristics of a wide domain and uniqueness. The Radon transform of an image provides a way to achieve the goal of feature extraction. Through some proper manipulation, these modified features will have the sufficiently wide domain and the approximate unique characteristics. These features will be used to train a MPNN for classifying different object classes. Several algorithms for finding features that can represent 3-D objects from 2D images are investigated.

Although the dimensionality of a feature vector from a single image could be reduced by using the Radon transform of an image, an object that appears in an image would still have thousands of different views. That is, there are thousands of images of an object. For example, suppose a simple object, such as an one inch cube (Figure 5.1) is arbitrarily put on a platform. If a camera can move horizontally and vertically within some distance to observe

the object's image, an object could appear in many forms in an image (Figure 5.1). Due to the characteristics of the neural network, it tries to classify objects that have been seen or are at least similar to those having been seen before. Therefore, it is necessary to show the neural network all the possible views of an object including different orientations, and locations so that the neural network can correctly classify the object. But the number of all possible views of an object is so large that it is very difficult, if not impossible, to obtain a complete set of object views. Even if a complete set of object views could be obtained, a neural network would take an extremely long time to learn these views. These problems block the development of 3-D object recognition using a neural network. In order to overcome this difficulty and to make a neural network feasible for 3-D object recognition, the number of possible appearances of an object in an image must be reduced. This requires that several conditions be specified.

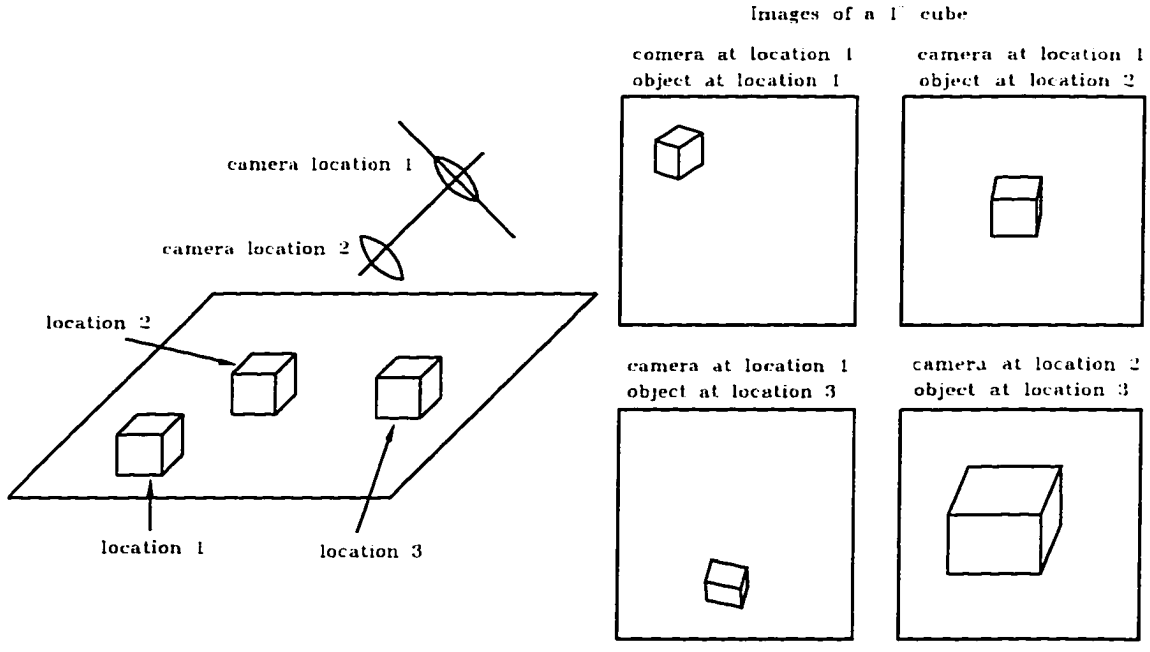


Figure 5.1 Some of the appearances of an one inch cube.

CONDITION 1: The viewing angle of a camera ϕ and the distance between the lens center of a camera and the platform h are fixed.

This condition can be satisfied easily for the application of a robotic assembly process. A camera can be mounted anywhere on a robot as long as it does not interfere with the assembly process and can be moved around a work-cell to search objects. Figure 5.2 visually displays this required condition.

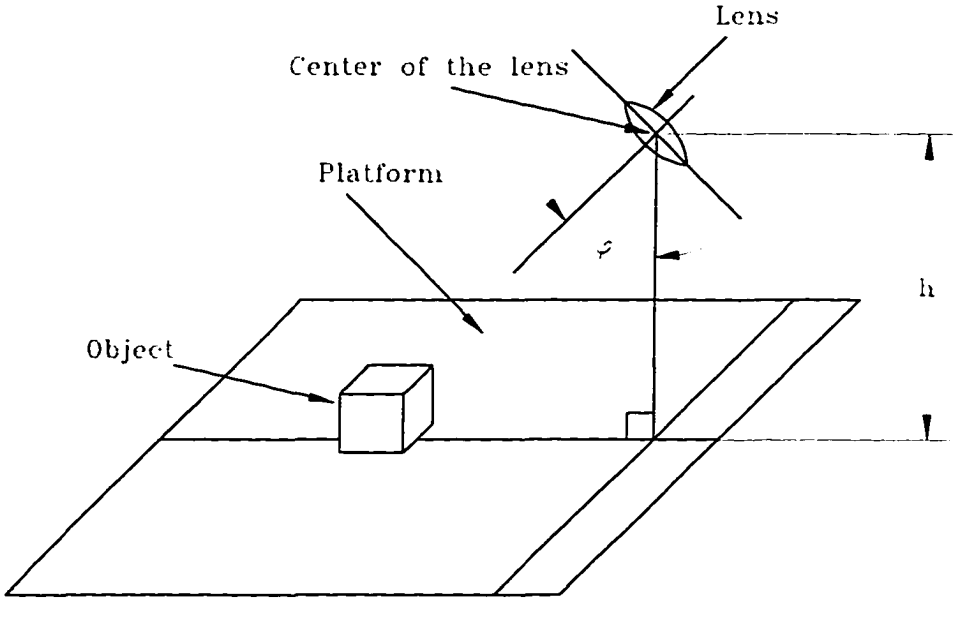


Figure 5.2 Condition 1.

CONDITION 2: An object appearing in an image is pre-defined at a specific location.

Figure 5.3 illustrates two specified locations where an object will appear in an image. One positions the geometric center of a projected 3-D object to the center of a 2-D image. Another matches the minimums of a projected 3-D object in the x and y directions with the pre-specified horizontal and vertical lines in an image. This condition can also be satisfied without difficulty. Since condition 1 has been satisfied, a mathematical formulation can be established to calculate the difference between the geometric center of a projected 3-D object and the center of a 2-D image or the differences between the minimums and the pre-defineed horizontal and vertical lines (See Figure 5.3).

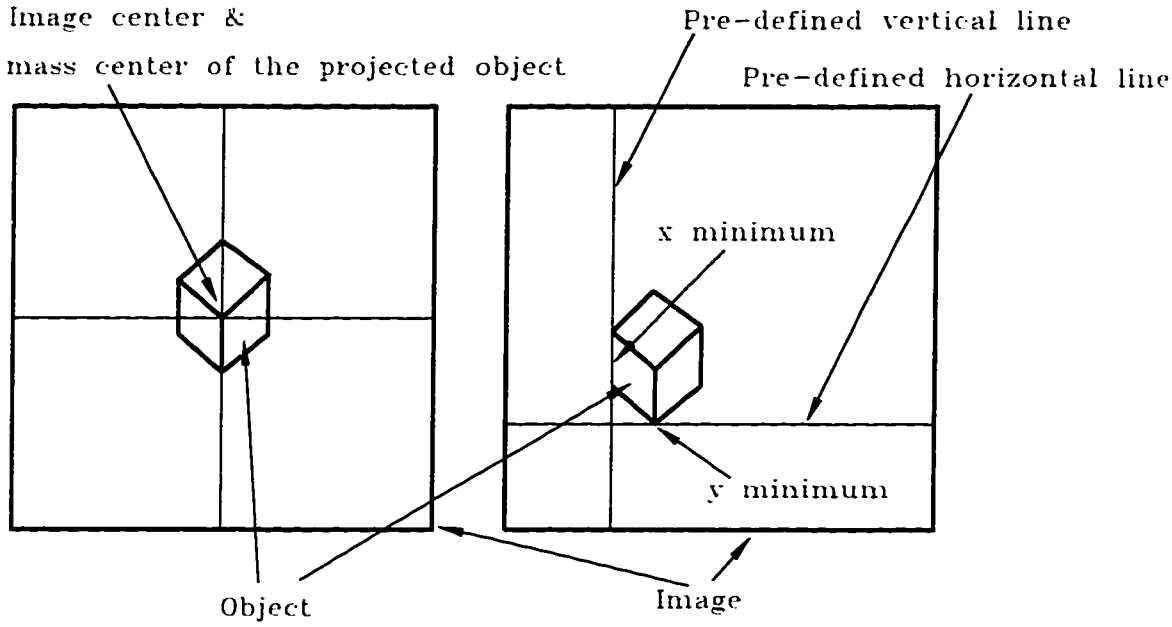


Figure 5.3 Condition 2.

By specifying these two conditions, the number of possible appearances of an object in an image obtained by a pre-located camera is reduced by $W \times H \times D$ times, where W and H are the width and height of a 2-D image, and D is the vertical movable distance of the camera. Considering the previous example, the appearances of the cube in an image under these conditions is shown in Figure 5.3.

Since there are now a limited number of views for an object, using these constrained views to train a neural network for recognizing objects becomes feasible. In this research a multi-layer perceptron neural network (MPNN) with back-propagation of error training method followed by a maximum-output-selector is used as the classifier to classify different feature vectors. Due to the nature of neural networks, it is necessary to show all the possible appearances of an object in order to obtain well-trained networks. (This is the same as for human-beings when they are recognizing objects, although humans are much more talented.) If some characteristic views of an object have not been seen before, humans can also make mistakes in recognizing

the object. Under the conditions specified above, all possible views of an object on a platform can be obtained by rotating the object 360^o on the platform at a fixed location. The following sections describe several algorithms selected in this work to construct feature vector for training as well as recognition of a MPNN classifier.

5.1 Data from Singular Valued Decomposition in Each Image (SVDI)

This SVDI algorithm is strictly based on the theory of the inverse Radon transform. It follows the underlying concept of image reconstruction from its projections to construct feature vectors. Once the images of an object at each orientation are observed at a specific location, projections of each image yield a set of projection vectors. Due to the symmetrical property of the Radon transform, projections taken over 180^o are sufficient to reconstruct the original image. They are then utilized to extract the feature vector for training and recognition.

The recognition rate based on these feature vectors depends on the *angular resolution* of the object rotation and the *angular resolution* of the image projection period. Experiments are conducted in this research to study the variation of the recognition rate with the changes of these angular resolutions. Results are presented in Chapter 7.

Suppose there are n projections taken over 180^o around each image. Let $\vec{p}_i$ denote the i-th projection vector. If all projection vectors are designed in such a way that they all have the same number of elements m, i.e., the intervals of every projection are the same, then a matrix is produced by stacking n vectors together. This $(n \times m)$ matrix is called the *projection matrix* of an image and can be represented as,

$$A = \begin{pmatrix} \vec{p}_1 \\ \cdot \\ \vec{p}_i \\ \cdot \\ \vec{p}_n \end{pmatrix} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,j} & \cdots & a_{1,m} \\ \cdot & \cdot & \cdots & \cdot & \cdots & \cdot \\ a_{i,1} & a_{i,2} & \cdots & a_{i,j} & \cdots & a_{i,m} \\ \cdot & \cdot & \cdots & \cdot & \cdots & \cdot \\ a_{n,1} & a_{n,2} & \cdots & a_{n,j} & \cdots & a_{n,m} \end{pmatrix}. \tag{5.1}$$

Figure (5.4) graphically represents how the projection matrix is created. By applying a process of singular value decomposition to A, a set of non-negative and real numbers can be derived.

They are called the singular values of matrix A and denoted as $\vec{s}$. This set of values are used as the feature vector to input to a neural network based classifier. The number of input nodes of a classifier depends on the number of the singular values, which is the minimun of m and n ($\min(m, n)$). Since m and n are fixed when deriving a projection matrix from each image, the ranks of all projection matrices are the same. Thus, every image can extract a feature vector that has pre-defined dimensions. Therefore, the use of singular values of a projection matrix as a feature vector provides the wide domain characteristics that a feature should have.

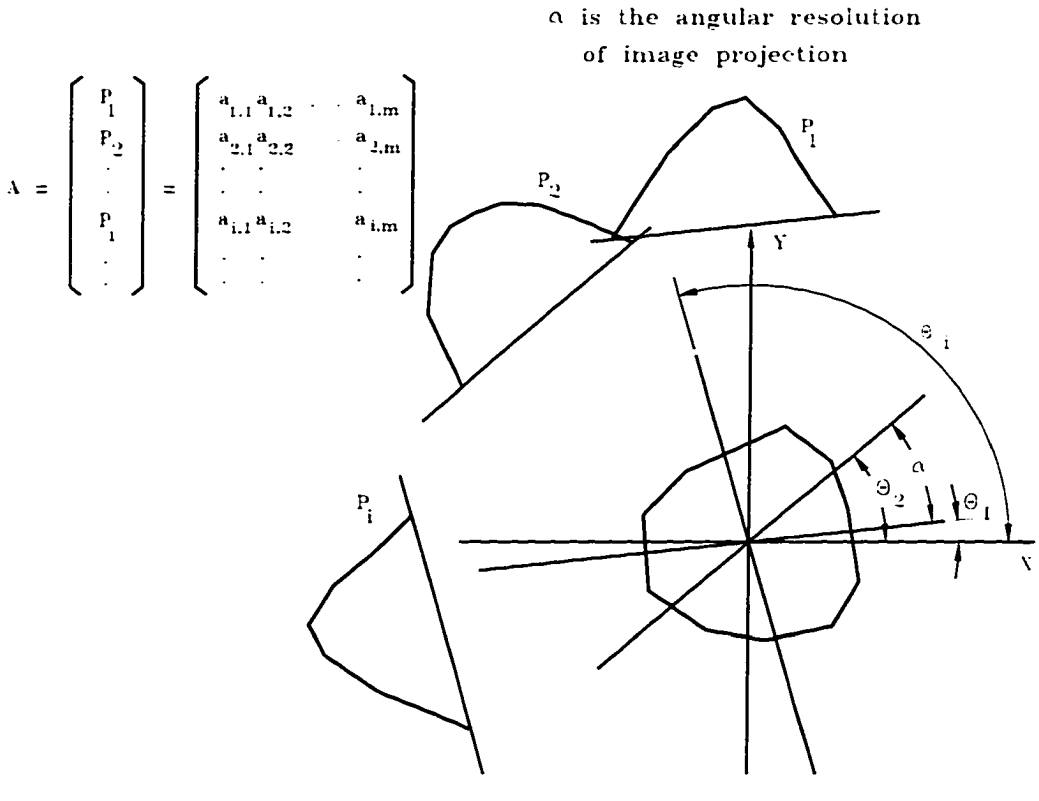


Figure 5.4 Construction of a projection matrix.

Theoretically, a set of projections can exactly reconstruct their corresponding image by using the theorem of inverse Radon transform if the angular resolution is infinitely small. These projection vectors can uniquely represent the image. However, the number of projections is so large that it is not efficient to use them all to reconstruct the image. Hence, data in the original image may be much smaller than that in this set of projections. Thus, it is impractical

to pursue an exactly reconstructed image. Instead, it is comman to use a much smaller set of projections to approximately reconstruct an image. Problem arises when a feature vector is extracted from a small set of projections of an image for an object recognition system. This feature vector does not uniquely represent a specific image because the same set of projections can be theoretically derived from different images. But if the possibility of getting the same set of projections from different images is small and acceptable to an application, the feature extracted from these projections is considered to have an approximately unique characteristic.

Since a feature vector used to recognize an object is the singular values of a projection matrix, the time to derive a feature partially depends on the dimensions of the projection matrix. The dimensions of a projection matrix depend on the number of projections and the size of an image. Also, the recognition rate relates directly to the unique characteristic of a derived feature. Therefore, the number of projections of an image affects the classification performance of a system. That is, too many projections will increase the recognition time and too few projections will decrease the recognition rate.

A feature using singular values of a projection matrix is a quantitative base feature that is suitable to neural network classifiers. Let K be the number of objects that need to be recognized, and I_k be the number of images for every possible appearances of k-th object. There are total of $\sum_{k=1}^{K} I_k$ images in a system. Consequently, $\sum_{k=1}^{K} I_k$ feature vectors can be extracted. They are used to train a MPNN classifier with back-propagation of error training method. Due to the characteristic of a MPNN, if any feature vector that belongs to one object class is not identical to any feature vector that belongs to other object classes, a MPNN is capable of classifying these objects without difficulty. The number of layers used in a MPNN depends on the complexity of $\sum_{k=1}^{K} I_k$ feature vectors. For features with concave or looped characteristic such as shown at Figure 3.12, a four-layer PNN needs to be implemented to recognize their associated objects. Otherwise, a three-layer perceptron neural network is adequate to perform the classification. Unfortunately, it is impractical to directly perform an analysis for these feature vectors since the characteristics of feature vectors from a set of objects is significantly different from that of a set of different objects. Experimental results are given in Chapter 7

showing the performances of two MPNN classifiers with one or two hidden layers in recognizing the same set of objects.

5.2 Data from Singular Valued Decomposition in Each Object (SVDO)

In the SVDI algorithm, n projections are taken from each individual image to construct a projection matrix, which is subsequently used to obtain a feature vector for training or recognition. In the training process, $\sum_{k=1}^{K} I_k$ feature vectors are used to train a MPNN classifier. Therefore, extensive training time is required to obtain a well trained MPNN classifier, especially when K and I_k are large. To overcome this problem, the following SVDO algorithm is proposed.

This method uses only one projection of each image to construct a *characteristic matrix* of an object. If there are I_k images for k-th object, the characteristic matrix of the k-th object is a $(I_k \times m)$ matrix, where m is the same as before. Figure 5.5 shows the construction of a characteristic matrix. If $I_k = I, (k = 1, \cdots, K)$, the characteristic matrix of every object will have the same dimension. Thus, the number of singular values from each characteristic matrix will be $\min(I, m)$. It is easy to see that different objects will have different characteristic matrices and consequently different sets of singular values.

In this algorithm, only one feature vector for each object is extracted rather than I_k feature vectors. K object classes provide K training vectors. The training time for a MPNN classifier is shortened significantly due to a much smaller set of training data. As mentioned earlier, the number of singular values of a characteristic matrix which defines the dimensions of a feature vector and the number of input nodes of a MPNN classifier depends on I or m whichever is smaller. In general, I is much smaller than m, and it is associated with the angular resolution of an object rotation. If the angular resolution of an object rotation is β, an object will be rotated $\frac{360^o}{\beta}$ steps to complete a 360^o full cycle. Thus, the characteristic matrix of an object contains $\frac{360^o}{\beta}$ rows. Therefore, the number of singular values obtained from the matrix is $\frac{360^o}{\beta}$. A chosen β value plays a crucial factor in this algorithm. A large value of β reduces the correlation between two consequent vectors while a small value of β increases the number of

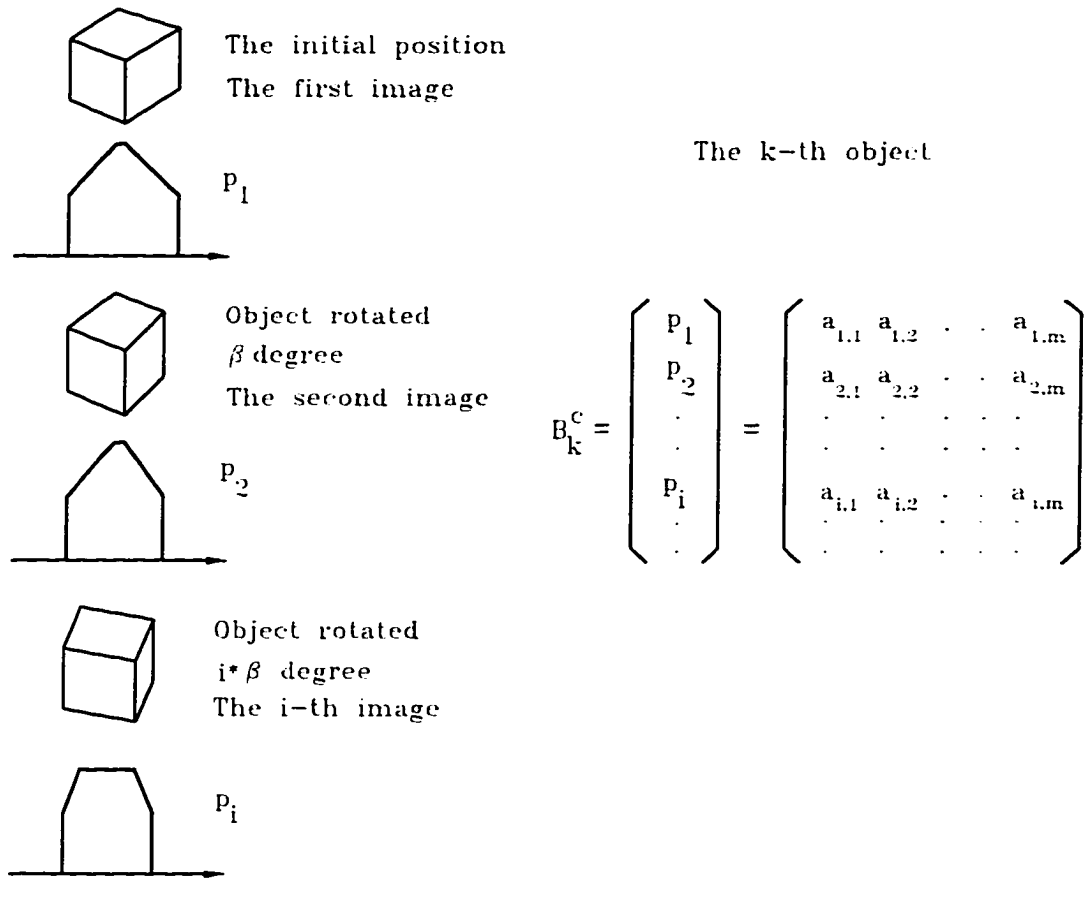


Figure 5.5 Construction of a characteristic matrix.

singular values. Consequently, a large value of β reduces the capability of generalization of a MPNN classifier, and a small value of β increases the training and recognizing time. In this research, β is considered as one of the important factors in evaluating this algorithm as well as others being proposed.

A projection of an image is a vector in $\Re^m$ Euclidean space. Since there are $\frac{360^o}{\beta}$ images for each object, $\frac{360}{\beta}$ vectors are obtained during a training stage. For convenience, these vectors are named *training vectors*. In a recognition process, the method of the minimum distance between vectors is proposed to derive a feature vector for recognition. After an object image is received by a camera, a projection of the image is conducted to get an m-dimensional vector. This vector is called the *recognition vector*. The distance between each training vector and the

recognition vector is calculated, and the one with the minimum distance to the recognition vector is replaced by the recognition vector. By stacking these vectors together, a new matrix is constructed, which is defined as the *recognition matrix* and has the same numbers of columns and rows as the characteristic matrix. The singular values calculated from a recognition matrix are then used to input to a well trained MPNN classifier to recognize the corresponding object.

The mathematical representation of this algorithm is as follows. Let $\vec{p}_l$ be a projection of the l-th image of an object, and B_k^c be the $(I \times m)$ characteristic matrix of the k-th object. The B_k^c is constructed as,

$$B_k^c = \begin{pmatrix} \vec{p}_1 \\ \cdot \\ \vec{p}_l \\ \cdot \\ \vec{p}_I \end{pmatrix} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,j} & \cdots & a_{1,m} \\ \cdot & \cdot & \cdots & \cdot & \cdots & \cdot \\ a_{l,1} & a_{l,2} & \cdots & a_{l,j} & \cdots & a_{l,m} \\ \cdot & \cdot & \cdots & \cdot & \cdots & \cdot \\ a_{I,1} & a_{I,2} & \cdots & a_{I,j} & \cdots & a_{I,m} \end{pmatrix}. \tag{5.2}$$

The singular values calculated from B_k^c are denoted as $\vec{T}_k^t = [s_1^t, \cdots, s_j^t, \cdots, s_I^t]$. Let $\vec{p}^r$ be the recognition vector that is a projection of an image obtained from a camera. Its elements are denoted as $[b_1, \cdots, b_m]$. Assume that

$$\min_{l=1}^{I} |\vec{p}_l - \vec{p}^r| = |\vec{p}_L - \vec{p}^r|. \tag{5.3}$$

Replacing vector $\vec{p}_L$ with vector $\vec{p}^r$ in a matrix B_k^c results a corresponding recognition matrix for the k-th object, denoted as B_k^r.

$$B_k^r = \begin{pmatrix} \vec{p}_1 \\ \cdot \\ \vec{p}_{L-1} \\ \vec{p}^r \\ \vec{p}_{L+1} \\ \cdot \\ \vec{p}_I \end{pmatrix} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,j} & \cdots & a_{1,m} \\ \cdot & \cdot & \cdots & \cdot & \cdots & \cdot \\ a_{L-1,1} & a_{L-1,2} & \cdots & a_{L-1,j} & \cdots & a_{L-1,m} \\ b_1 & b_2 & \cdots & b_j & \cdots & b_m \\ a_{L+1,1} & a_{L+1,2} & \cdots & a_{L-1,j} & \cdots & a_{L+1,m} \\ \cdot & \cdot & \cdots & \cdot & \cdots & \cdot \\ a_{I,1} & a_{I,2} & \cdots & a_{I,j} & \cdots & a_{I,m} \end{pmatrix}. \tag{5.4}$$

The singular values calculated from B_k^r are $\vec{R}_k = [s_1^r, \cdots, s_I^r]$.

There are three different cases according to Equation 5.3. Case 1: $\min_{l=1}^{I}|\vec{p}_l-\vec{p}^{\,r}| = |\vec{p}_L-\vec{p}^{\,r}| = 0$, this is $\vec{p}_L = \vec{p}^{\,r}$. The L-th training vector equals the recognition vector. B_k^c and B_k^r are then identical. Thus, the singular values from B_k^r will be identical to those from B_k^c, $\vec{R}_k = \vec{T}_k$. Since the neural network is trained by using $\vec{T}_k$ for k-th object, it will give $\vec{R}_k$ that have the same output category as $\vec{T}_k$. This means that the object captured from the camera has been recognized as the k-th object class being trained in a MPNN classifier. Case 2: $\min_{l=1}^{I}|\vec{p}_l - \vec{p}^{\,r}| = |\vec{p}_L - \vec{p}^{\,r}| = \delta$, where the smallest distance between the recognition vector and training vectors is δ. Depending on the value of δ, if it is smaller than some bounding value, the singular values from B_k^r, $\vec{R}_k$, would not have significant difference from $\vec{T}_k$. By the generalization property of a MPNN, the object would still be recognized as the same as k-th object class. Case 3: Instead of having a small δ as in Case 2, δ is larger than the bounding value. That is, the recognition vector is far away from any training vector and the trained MPNN classifier will not tolerate this difference. It will classify the object from the camera as not being the same as the k-th object.

There are only K training vectors if K objects need to be classified. So the number of hidden nodes of a MPNN can be very small, and a three-layer PNN is capable of separating these feature vectors because there is only one prototype for each object class. Thus, the training time using this method is very fast compared with other proposed algorithms of this study. In a recognition stage, since there are K characteristic matrices for K different objects, a recognition vector has to compare with $K \times I$ training vectors and K recognition matrices must be derived. Each recognition matrix is calculated to obtain a set of singular values. Then, K sets of singular values are input to a trained MPNN classifier sequentially. An output node with largest value for K sets of input vectors will conclude the object to be of the class corresponds to that maximum output node. The recognition speed is a major problem for this algorithm. It requires determining K processes of singular value decomposition for each $(I \times m)$ matrix. Therefore, the recognition speed is very slow.

5.3 Data from Simple Operation of Projections of An Image (SOPI)

Besides the problem of the recognition speed, the SVDO algorithm has another problem, which contributes to a relatively low percentage of classifying rate. This is because the derived features do not have a good uniqueness characteristic. Only one projection for each image has been used to construct a feature vector. Thus, the possibility of different images having an identical projection is relatively large.

To overcome these two problems which are of high concern in an object recognition system, the third algorithm is proposed that has higher percentage of classifying rate and much faster recognition speed. This SOPI algorithm uses a simple operation of more than one projection of each image to create a feature vector to train a MPNN classifier and recognize objects.

In order to have a better uniqueness characteristic for a feature vector, it is necessary to use more than one projection from each image. But it is necessary to find a way to combine these projections without additional lose of the feature's uniqueness characteristic. Initially, an averaging operation was applied to a set of projections to produce a feature vector. For consistency, a feature vector for k-th object in a training process is still denoted as $\vec{T}_k' = [s_1^t, \cdots, s_m^t]$, where m as before is the length of a projection vector. Therefore, an averaging operation can be formulated as,

$$\vec{T}_k = \begin{pmatrix} s_1^t \\ \cdot \\ s_j^t \\ \cdot \\ s_m^t \end{pmatrix} = \frac{1}{n}\left(\sum_{i=1}^{n} \vec{p}_i\right)' = \frac{1}{n}\begin{pmatrix} \sum_{i=1}^{n} a_{i,1} \\ \cdot \\ \sum_{i=1}^{n} a_{i,j} \\ \cdot \\ \sum_{i=1}^{n} a_{i,m} \end{pmatrix}, \tag{5.5}$$

where the prime "$'$" denotes a transpose of a vector, and n is the number of projections derived from each image. Since this operation is a linear combination of a set of projections, it does bring the additional possibility of getting the same feature vector from different images. In fact, replacing Σ's (summations) with Π's (multiplications) in Equation 5.5 will give similar results.

The additional possibility of getting an identical feature vector from different images will introduce extra mis-classifications of this system. Another simple operation is proposed to derive a feature vector from a set of projections without bringing such additional misclassification possibility or requiring extra computational effort. This operation, named "inserting operation", combines n projections of an image as follows:

$$\vec{T_k} = \begin{pmatrix} s_1 \\ \cdot \\ s_i \\ \cdot \\ s_{n\times m} \end{pmatrix} = \begin{pmatrix} a_{1,1} \\ \cdot \\ a_{i,1} \\ \cdot \\ a_{n,1} \\ \cdot \\ a_{1,j} \\ \cdot \\ a_{i,j} \\ \cdot \\ a_{n,j} \\ \cdot \\ a_{n,m} \end{pmatrix}. \tag{5.6}$$

From Equation 5.6, the i-th projection of an image is $\vec{p_i} = [a_{i,1}, \cdots, a_{i,j}, \cdots, a_{i,m}]$, and $i = 1, \cdots, n$.

Although no additional misclassification possibility is introduced by the inserting operation, dimensions of a feature vector become very large even with a small set of projections. This is a serious problem when applying it to a MPNN classifier since there are $n \times m$ input nodes if n is the number of projections of an image. The computational effort is significantly increased. A way to reduce the dimensionality of this feature vector is to reduce the dimensions of each individual projection vector, m. Theoretically, the effort in reducing the dimensionality of a feature vector not only increases the possibility of getting a feature vector from different images, but also introduces the aliasing problem which will also degrade the recognition performance of

a system. The aliasing problem is a typical problem in data acquisition processess. It strongly relates to the data sampling interval. If the data sampling interval is relatively small, aliasing will not give any influential effect to the unique characteristic of the sampled data. Thus, it is not desirable to reduce m, especially when a system is designed to recognize objects with complicated shapes.

Feature vectors extracted by application of this SOPI algorithm are desired in the recognition stage since simple operations are used to manipulate a set of projections. A recognition system implemented with this algorithm is suitable to the applications that require fast classification speed, such as robotic assembly processes. The drawback of this method is the additional probability of getting a feature vector from different images (misclassification). But if the classification rate meets an application requirement, the overall performance of a system using this algorithm is better than those using the previous two algorithms. The result of the experiments using these algorithms is presented in Chapter 7.

6 IMPLEMENTATION

6.1 Introduction

The proposed three-dimensional object recognition system has been implemented in a Silicon Graphic workstation (INDY) which has a UNIX based operating system. The computer has one hundred fifty mega hertz of clock speed and sixty four mega bytes of random access memory. The recognition system uses the relatively inexpensive camera attached with the workstation as the sensing device to acquire images from a scene. The experiment has been conducted in a controlled environment where the lighting is pre-set in order to increase the variations of light reflections from different surfaces of a 3D object. In addition the platform upon which the object is placed is properly colored (controlled) to signify the object-background separation.

In the implementation, a set of computer object models is first constructed to obtain a set of corresponding training vectors using the proposed algorithms presented in Chapter 5. These derived training vectors are then used to train a MPNN classifier. After the MPNN classifier has been well trained, the recognition system uses a camera to locate an object on a platform based on the mathematical formulation of perspective transformation. Once an object is properly located close to a pre-defined image position, the camera acquires an image from the scene. This image is referred to as the original image and will be further processed by using the techniques of image processing, such as noise filtering and edge detection. The recognition vector is extracted from the processed image and input to the well trained MPNN classifier for classification. Figure 6.1 shows the overall scheme of the system implementation.

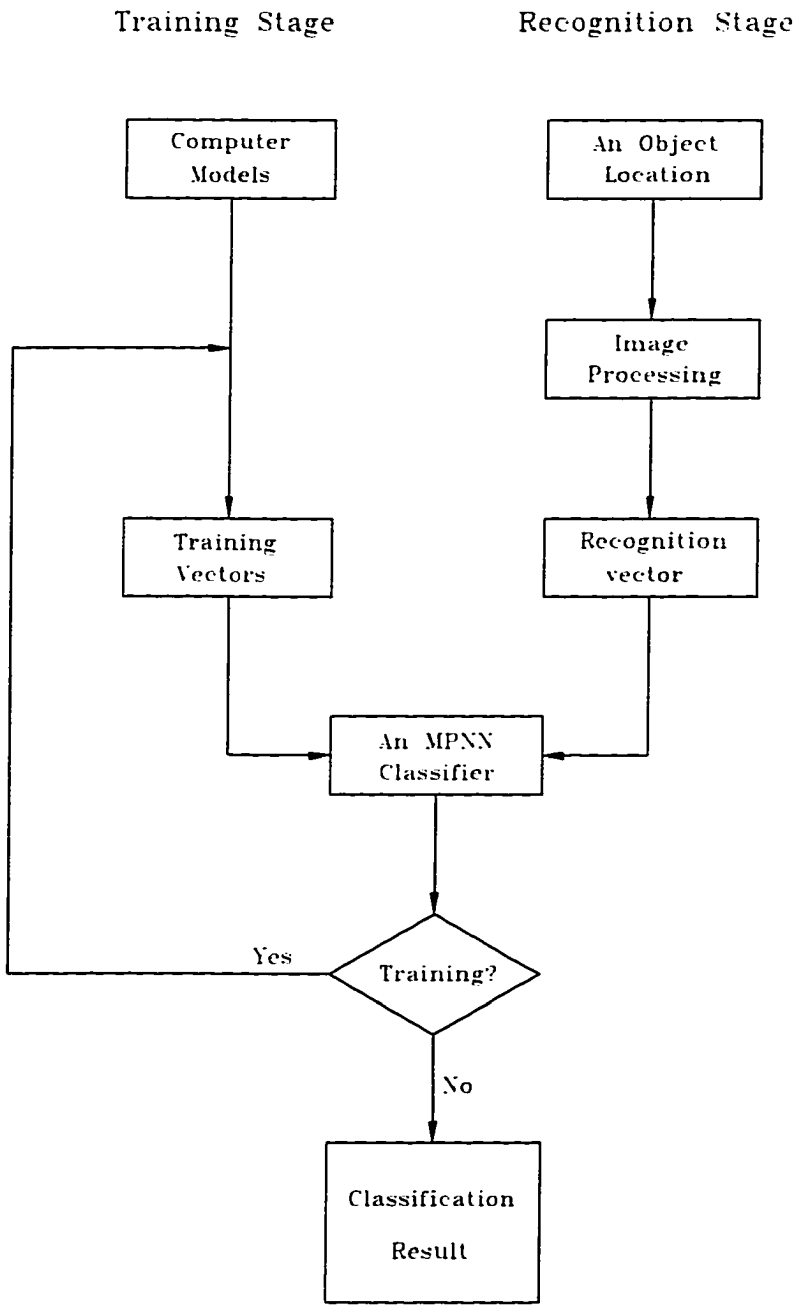


Figure 6.1 The overall schematic diagram of the system implementation

6.2 Training Process

Due the rapidly increasing use of 3D solid modeling in engineering design and manufacturing, the time span to production has been much shortened. This is because the detailed information about objects can be visualized without actually producing the parts. Taking advantage of 3D solid modeling in object recognition research is a relatively new subject to many recognition developers. Although there are some algorithms [43, 45, 56] that utilizes computer models, they are either in the testing stage or require large storage and complicated retrieval mechanisms. This proposed research utilizes 3D models generated from a computer to create the training pattern vectors. Once the training vectors are created from a 3D model, the 3D model is removed from the system. The training vectors can be also deleted from the system after the MPNN has been well trained. Therefore, the proposed system does not require extensive storage and consequently there is no need to design a complicated retrieval mechanism.

The sample objects are created by using the C programming language and OpenGL routines [112]. Once a sample object is created, the factors affecting the appearance of an object, such as its location, orientation, material property, and lighting model, are adjusted to approximate the appearance of a real-world object. The location and orientation of an object in an image are set according to the conditions specified in the previous chapter. Since the intensity values are the measuring primitives for constructing feature vectors, the values of RGB colors are all the same for the material of a sample object, and the values of RGB colors for light sources.

By the Condition 2, a 3D object produced by a computer (computer object) is modeled to give a similar visual effect as a real object appears in an image. This can be done by applying the rotation, translation, and perspective transformations of OpenGL routines to a 3D object. In order to obtain all possible views of a computer object, the computer object is rotated 360° around an axis. This axis is a vector that parallels to the YZ plane and has the ϕ angle from the Z axis if the X and Y coordinates coincide with the image or window coordinates. For each view, the different proposed algorithms are then applied to it to extract training vectors. Then, a new sample object will be created and old one will be removed. Another cycle of

deriving training vectors will be processed until the complete set of training vectors for every object has been observed.

The derived training vectors are used to train a multi-layer perceptron neural network (PNN) classifier. Several multi-layer PNNs are implemented by also using the C programming language. The number of input nodes is associated with the dimension or number of elements of a training or recognition vector. Consequently, the number of input nodes is related to different proposed algorithms. The number of output nodes is an adjustable variable which can be changed according to the number of objects required to be classified. The number of nodes in a hidden layer is usually chosen according to the characteristic of a data set. For one-hidden layer PNN, it has been determined that the number of hidden nodes defines the number of sides of a convex polygon which has resulted from a training process to separate one object class from others. It is desired to gain some knowledge about the data set. But it is extremely difficult to obtain the inside view of a data set that each data has more than three dimensions. Also, there are many different shapes of objects that may require recognition. It is not desirable to analyze every data set derived from any combination of objects. The number of hidden nodes is designed by using the trail-by-error method. Evaluation is based on the convergent speed and the classification rate using the same set of objects. In the experiment, this number is chosen from $[K, m]$, where K and m are, as before, the number of objects and the number of elements of a vector.

Two-, three-, or four-layer PNN is applied with different algorithms depending on their requirements. In the SDVI algorithm, there are I_k training features for each object and a total of $\sum_{k=1}^{K} I_k$ training features. These features together may inhibit looped or/and concave characteristics. Thus, the experiment has been conducted by using three- and four-layer PNNs for the algorithm. Since this algorithm requires long period of time for the singular valued decomposition of each image, four-layer PNN heavily increase training or recognition time of the system. Therefore, the experimental result herein are given only for the system coupling with three-layer PNN classifiers.

K object classes will provide K training features when applying the SVDO algorithm in

the system. This set of feature vectors does not exist with either looped or concave properties, since there is only one training feature for each object class. From the discussion in Chapter 3, a PNN with at most one-hidden layer is capable of separating these feature vectors. Thus, two- and three-layer PNN's have been implemented to test this algorithm.

Using the SOPI algorithm to extract feature vectors gives a training vector set with similar properties as those obtained by using the SVDI algorithm. It is necessary to use more than a two-layer PNN classifier to classify these vectors. But unlike the SVDI algorithm, this algorithm does not require time to do the singular valued decomposition process. Therefore, three- and four-layer PNN's are implemented in the system to evaluate this algorithm.

Several factors are also considered in the PNN implementation. The *training gain factor* denoted as *eta* is used to control the contribution of a node to the output error. Its value can be taken from the range of [0, 1]. Another important factor called *momentum factor* and denoted as *alpha* reduces the large or rapid change of a weight between two subsequent iterations. This is because a rapid change will often decrease the training stability of a PNN, and produce serious oscillation between two training cycles. Serious oscillation in a training process will result a very low convergent speed or even no convergence. The *flat-spot correction factor* is used to handle the situation when a PNN tries to find a solution in a flat surface where one does not exist. This correction will make a PNN move to other surface more quickly and the training time will thus be shortened.

In an experiment, the initial values of *eta*, *alpha*, and the *flat-spot correction factor* are set to be 0.5, 0.9, and 0.1, respectively. They will be steadily reduced during the training process in order to obtain an approximately optimum solution. The strategy of adjusting these values is based on the increase or decrease of the classification rate. After every feature has been presented to the PNN at each time, the classification rate is calculated by counting the number of correctly classified features. Once the classification rate shows a decreasing trend, after a sequence of increases in classification rate in previous training, these three values will be reduced. Also, the initial weights of a PNN are random values generated by the random function in the C programming language.

6.3 Locating an Object

After the PNN classifier is well trained, it is ready for the classification. The SGI computer then sends a signal to an IBM compatible computer (PC) which commands a robot to find an object using a camera. The communication between the SGI computer and a PC is implemented by using the TCP/IP routine [113] and SOCKET library [114] in the C programming language, respectively.

The first step in the process of locating an object is to find an object. To do that, an image is obtained by a camera mounted on a robot and then is projected to a line which is a horizontal line in the implementation. The projection vector is checked to see whether an object is contained in the image. If the image does not contain an object, the elements of the projection vector should have approximately the same values; else, the variations among these elements will be large. Once an object is found, the distance between the geometric center of the object image and the center of the image is calculated. If this distance is smaller than a desired value, the object has been properly located and the image will be processed to extract a recognition feature; else, the position of the object on a platform is computed and its coordinates are sent to a robot. A new searching process is then started.

The position of an object on a platform is computed by applying the perspective transformation [115] to an image that contains the object. Although a 2-D image does not provide any depth information about a 3-D object, by posting the Condition 1 described in the previous chapter, the geometric center of a 3-D object can be calculated approximately. Based on the coordinates of this geometric center, a robot will move a camera to another position to acquire a new image.

Let the image center and plane be the origin and XY plane of a coordinate system, respectively. A point and its image are denoted as $\mathbf{v_o} = (x_o, y_o, z_o)$, and $\mathbf{v_i} = (x_i, y_i, 0)$, respectively; see Figure 6.2. By the properties of similar triangles, the relationships between the coordinates of a point $\mathbf{v_o}$ and its image $\mathbf{v_i}$ are

$$x_i = \frac{\lambda x_o}{z_o + \lambda},$$

$$y_i = \frac{\lambda y_o}{z_o + \lambda}, \tag{6.1}$$
$$z_i = 0.$$

The inverse relationships are derived as

$$x_o = (z_o + \lambda)\frac{x_i}{\lambda}, \tag{6.2}$$
$$y_o = (z_o + \lambda)\frac{y_i}{\lambda}. \tag{6.3}$$

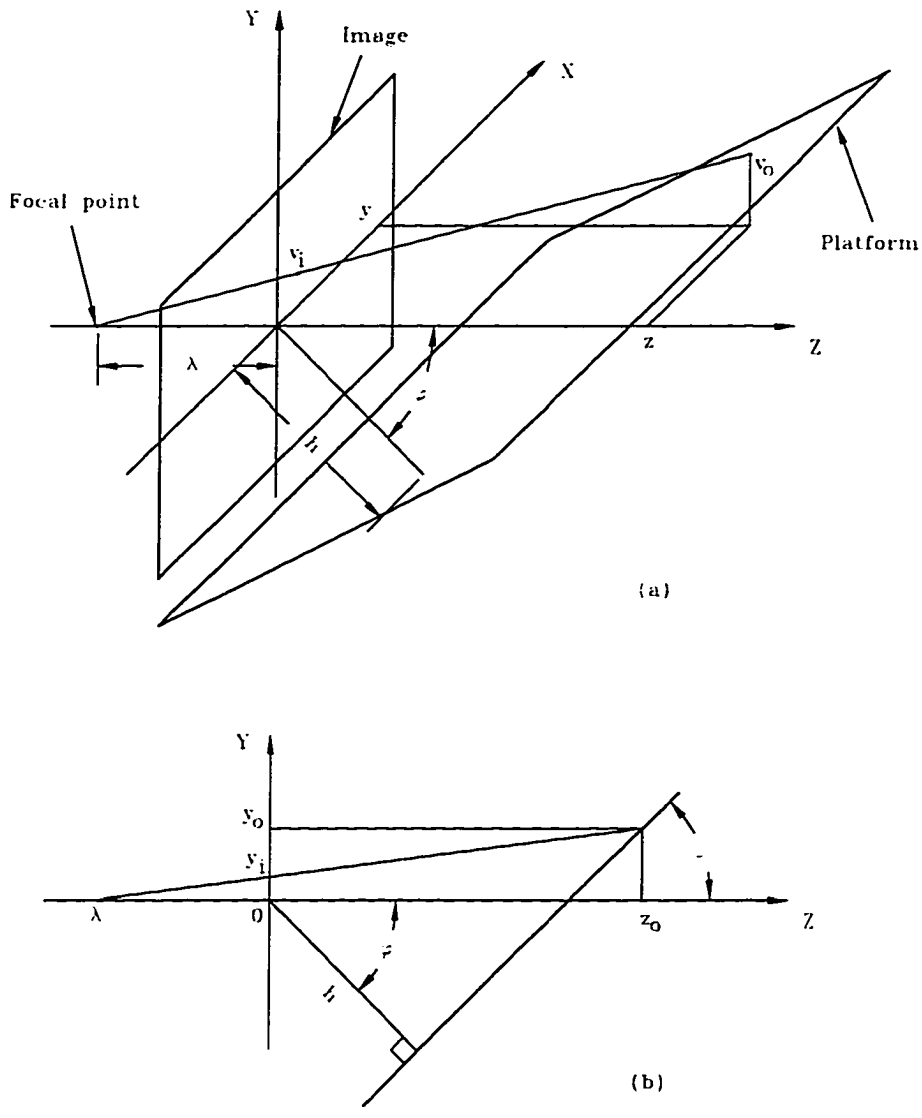


Figure 6.2 The representation of the perspective transformation.

From the above equations, the z value of the point $\mathbf{v_o}$ and its image $\mathbf{v_i}$ must be known to calculate its x and y coordinates. Figure 6.2(b) shows the key geometric ingredients for deriving z_o. If v_o is on the surface of a platform, z_o can be formulated as

$$z_o = y_o + \frac{h}{\cos\phi}, \tag{6.4}$$

where h is the distance between the center of a lens and a platform, and ϕ is the angle between the viewing direction and the normal of a platform, as specified in Condition 1. Then, substituting z_o into Equation (6.2), we obtain

$$y_o = \frac{y_i}{\lambda}(\lambda - y_o - \frac{h}{\cos\phi}). \tag{6.5}$$

Resolving Equation (6.5) yields

$$y_o = \frac{\lambda - \frac{h}{\cos\phi}}{\lambda + y_i}. \tag{6.6}$$

Consequently, z_o and x_o can also be derived. These image coordinates are transformed to the robot coordinates that are used by the robot to move camera to re-locate an object.

In the implementation, the geometric center of an object image needs to be approximately aligned with the image center so that the recognition vector extracted from an image will be consistent with the training vector. As outlined before, if the distance between these two centers is not smaller than a specified value, the coordinates of the geometric center v_i must be transformed to v_o for re-locating an object. But in the calculation, the point v_o is not on the surface of the platform. Therefore, it is necessary to estimate the distance between the platform and the point v_o. An estimate of this distance can be obtained by using the value of half of the average of the heights of K objects. After the distance has been estimated, the point v_o can be computed from the geometric center of an object image v_i by using Equations from (6.2) to (6.6). Although the above calculation only provides approximate coordinates of v_o, a few re-locating iterations will give very close values, especially when the distance between a camera and an object is large relative to the dimensions of the object. Also, a PNN classifier is capable of classifying features with small deviation from their prototypes. Thus, small distance between these two centers will not significantly affect the classification performance of the system.

The coordinates of the center point v_i of an object image is also derived by using projections of the associated image. Once an image is obtained from a camera, it is projected to the horizontal and vertical lines as shown in Figure 6.3. These projection vectors are first used to check if the image contains an object image in the way described at the beginning of this section. If the image contains an object, the portions of the two projection vectors that consist of an object are extracted. The mid-points of the portions are found. Since the mid-point of the portion derived from the horizontal projection vector provides the x_i coordinate of the geometric center of an object image. In the same way the mid-point of the portion derived from the vertical projection vector provides the y_i coordinate. Thus, the coordinates of the center point v_i are derived. Figure 6.3 demonstrates the above process.

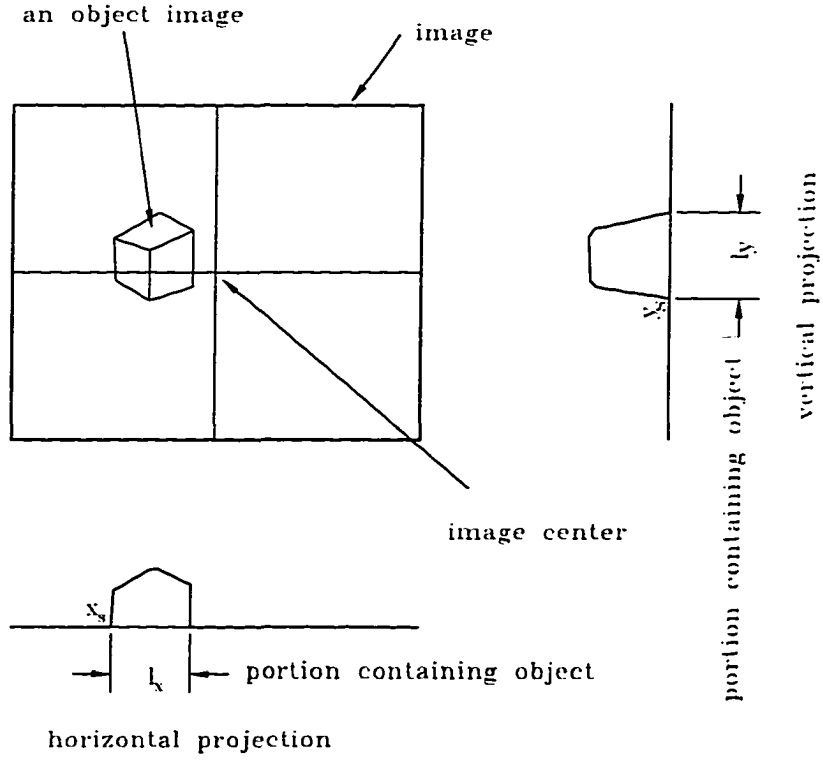


Figure 6.3 Deriving the geometric center of an object image.

If an image contains only a partial image of an object, the above process is still suitable for obtaining the coordinates of v_o. But if an image does not contain any object, the above process will not be applicable and no v_o can be determined. Thus no instruction can be send to a robot to command the next move. In this situation, a robot will follow a pre-defined path to search for objects. The path is defined in the manner that images obtained by a camera do not overlay each other but can cover the complete platform.

6.4 Image Processing in the Recognition Stage

After the geometric center of an object image is approximately aligned with the center of an image, the image will be processed and the recognition features will be extracted from it by applying one of the methods presented in Chapter 5. Several image processing techniques are used in the implementation to derive a "clean" image. In this study, noise filtering, boundary extraction, and object background seperation were used as described below to process the image before applying a algorithm to obtain object features.

Due to existing noise in an image, a noise filtering technique is first applied to an image. Although the process of a median filter is a nonlinear operation, its processing time is much less comparing to other more sophisticated methods. Its application is just like a low-pass or high-pass filter that does not require extensive study of image characteristics. Furthermore, a median filter has the capability of preserving edges of an object in an image and has the combining advantages of both low-pass and high-pass filters. Thus, a median filter was selected as the noise filtering method in the system even though the low-pass and high-pass filters have also been examined in the experiment.

The resulting images in Figure 2.5 can further explain why a median filter is used rather than a low-pass or a high-pass filter. The image smoothing by a low-pass filter results in an ambiguous boundary of an object. Also the image after the high-pass filtering process gives increasing contrast to an object boundary along with some noise. Both results are not desirable in this application. This is because the object-background separating process will be implemented in a later stage. Thus, the boundary of an object image needs to be as clear as

possible. Also, if the noise being signified is inside an object image, the recognition feature extracted from the image will be additionally distorted and may cause incorrect classification. No such distortion effects occur when an image is filtered by a median filter using a 5×5 operation window, as shown in Figure 2.5(d).

As outlined in Chapter 2, different sizes of operational windows may apply to a median filter. A median filter with a large size of operation window is usually used to remove a large area of isolated noise. It requires a relatively long period of processing time and may erase some useful information such as corners of an object image. A median filter with a small size operational window is desired when the processing speed is critical to the performance of a system. In contrast to a large size operational window, a small size operational window will not filter out a large area of noise, but it will not round off corners of an object image. Four images in Figure 6.4 display the original (top left) and the three smoothed images by using a median filter with 3×3 (top right), 5×5 (bottom left), or 7×7 (bottom right) operational window. From the filtered images, it is clear that the corners of an object being rounded off become more significant with an increase in the size of the operational window used in a median filter. Since it is important to retain all possible information that an object image contains and filter out noise as much as possible, a 5×5 operational window was chosen for the median filter of this study to remove noise in an image.

The next process in the implementation was to extract the boundary of an object image that will be used in the process of object-background separation. The process is conducted by utilizing an edge detection technique. It is desired to retain only edges in an image after an edge detection process is applied to the image. Also, the edges constructing a boundary of an object image should not have any gaps. Generally, algorithms that can obtain thick edges will leave strong noise in the image; and algorithms that can remove most noise will derive thin edges with gaps. It was noticed that the performance of an edge detection process also depends on the clarity of edges and amount of noise in the original image. Ambiguous edges and/or large amount of noise in an image will not result in a good quality of edge image for any edge detection techniques.

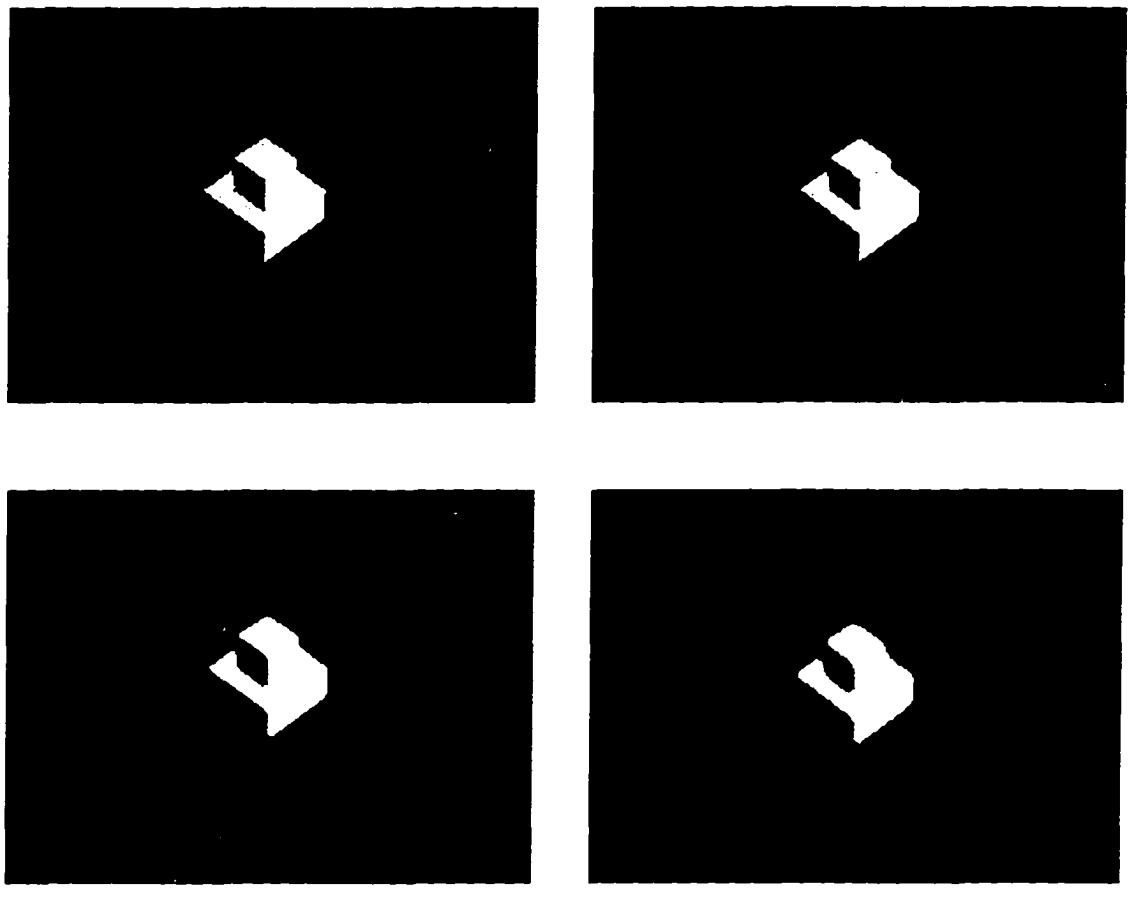


Figure 6.4 Images resulted from different sizes of operation widows for median filters.

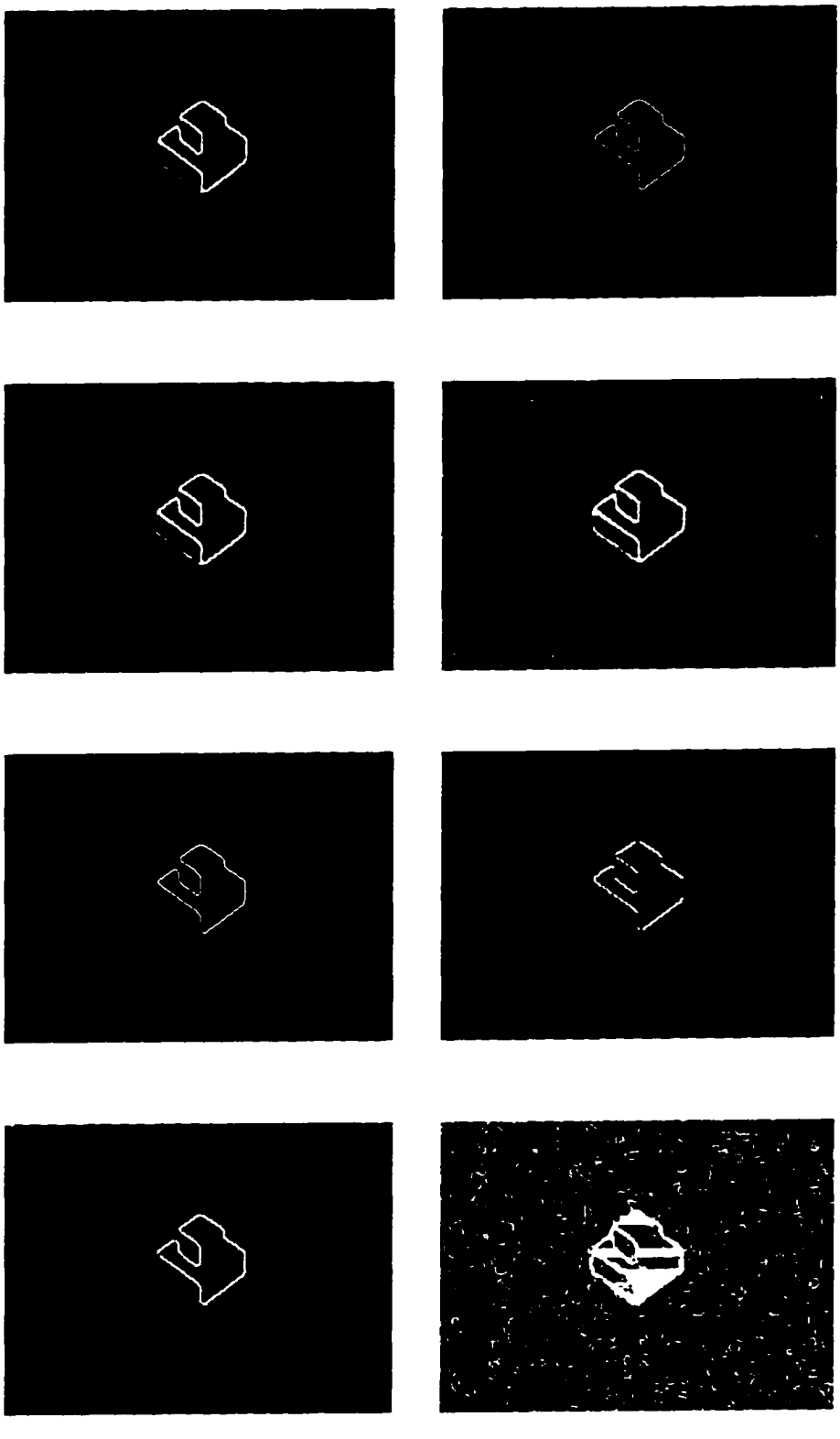


Figure 6.5 Images resulted from different edge detection techniques.

Ten edge detection techniques discussed in Chapter 2 have been implemented in the research in order to find a suitable one. Eight of them have relatively good results in this application and their resulting images are shown in Figure 6.5, they are from top left to bottom right, Prewitt, Robert, Robinson, Sobel, Laplacs 1, Laplacs 2, Laplacs 3, and Kersch techniques. The original image is obtained by the camera attached in a SGI INDY workstation and is pre-processed by a median filter with 5×5 operation window before applying an edge detection method. The results in these edge images demonstrate the discussion in the previous paragraph. The edge images derived from Prewitt and Robinson edge detection processes provide the desired outcome for this application. The Prewitt and Robinson operators used in the comparison have two masks and eight masks, respectively. There is no concern about the directions of edges. Therefore, the Prewitt edge detection technique was used throughout these experiments because its processing time is four time less than the Robinson operator. Figure 6.6 shows an image resulted from the Prewitt operator without having any noise filtering process. Comparing this image with the previous image which is also processed by the same Prewitt operator, it is clear that a noise filtering process needs to be applied to an image before edges are extracted from it.

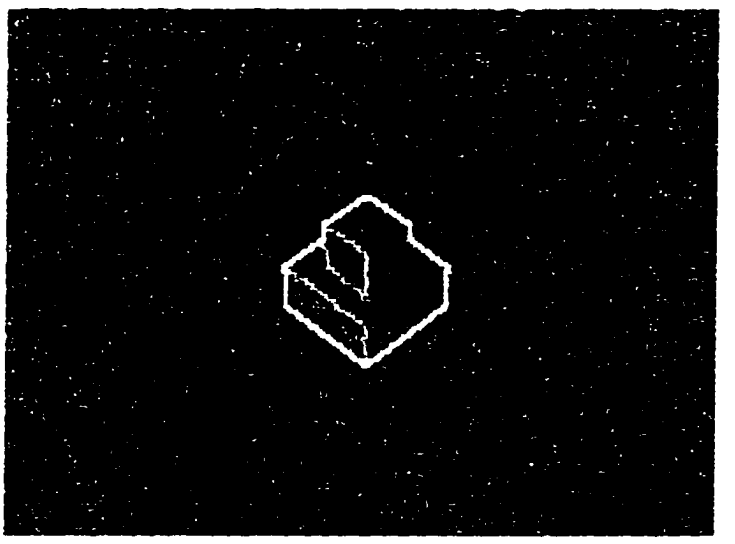


Figure 6.6 Images resulted from Prewitt technique without filtering noise.

Notice that the chosen edge detection operator may not be the best one for all different environments. The operator is dependent on the environment such as lighting, object's and platform's colors, etc. Thus, it is necessary to test different operators experimentally in order to find one with the desired result of an edge image in a specific environment.

The process of object-background separation was implemented to remove all pixels that do not belong to an object image. It is done by setting a pixel intensity-level to zero if the pixel location is outside the boundary that is extracted by an edge detection technique. In order to do that, a search mechanism must be developed to compare the location of a pixel with the location of a boundary pixel. Thus, there are $W \times H$ comparisons in the process if an image is W-pixel wide and H-pixel high. Extra computational time will be needed in the operation even though the effort in each individual comparison is small. Since the system performance is also evaluated by recognition speed, it is necessary to minimize this extra computational time.

As shown in Figure 6.3, the portions containing an object in two projection vectors has been extracted during the object locating process. They are used here as the bounding box or bounding frame. The search process is then operated only inside this bounding frame. The number of comparing operations is therefore reduced to $l_x \times l_y$ (see Figure 6.7). Inside the bounding frame, the left most pixel intensity value of the first row in the original image is set to zero if the corresponding pixel intensity value in the edge image is zero. Then, the second most left one of the same row is set using the same procedure. The process is repeated for every consecutive pixel in the same row until the corresponding pixel intensity value in the edge image is non-zero. From this point, the process is switched to the right most pixel and the same procedure is applied to every consecutive pixel starting from the right most one to the one that its corresponding pixel intensity value in the edge image is non-zero. This complete process is repeatedly applied in the same way to every row inside the bounding frame. Once each row inside the bounding frame has been processed, the object-background separation for an image is completed.

Since the geometric center of an object image might deviate from the image center even though a locating process has been used in the system to re-position the object image, it is

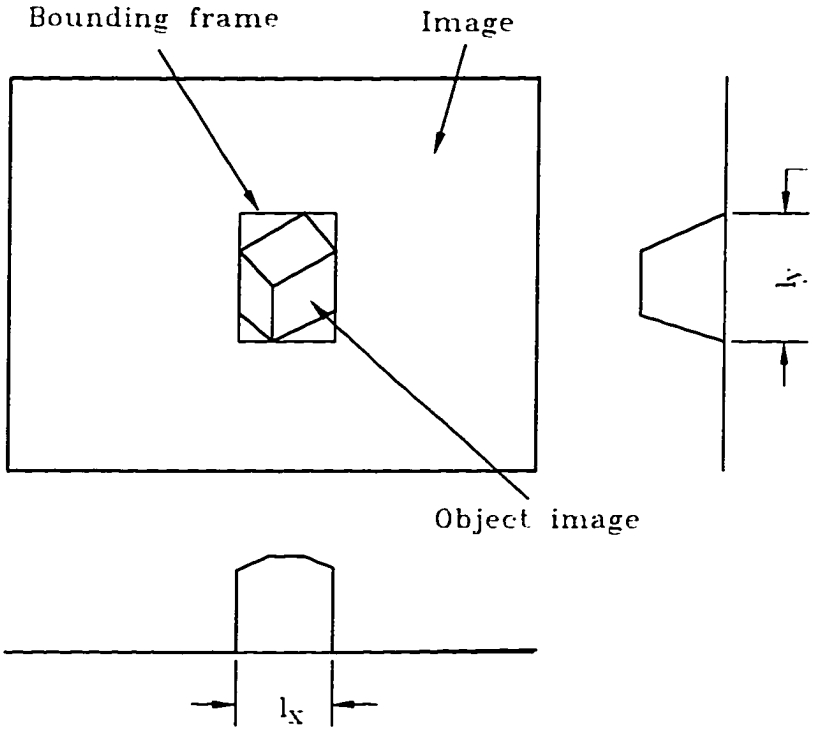


Figure 6.7 The bounding frame defines the processing area for the object-background separation.

desired to translate the object image to a position so that its geometric center aligns with the image center. This is because images that are used to extract the training vectors have the same position for the geometric centers of object images as the centers of their associated images. It is rated that, the projection vectors in two images shown in Figure 6.8 are not the same even if the object images are "identical". Notice that no object images are identical if locations of object images are not the same due to the characteristics of the perspective transformation used. Although a perceptron neural network classifier can tolerate small variation, large deviation between two recognition vectors will produce different recognition results. Thus, a simple mechanism was implemented in the system to overcome this problem. To each projection vector, the mechanism simply moves the portion of a projection vector which contains the object image d_x or d_y distance. This process will yield, for example, two identical projection vectors for images in Figure 6.8.

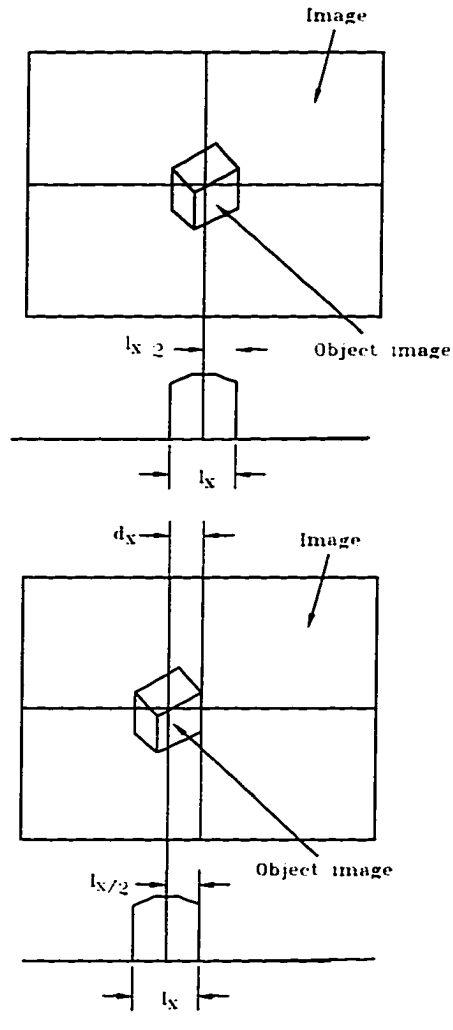


Figure 6.8 Different projection vectors resulted from two identical object images.

After the process of object-background separation, different algorithms can be applied to the image to extract a recognition vector. The recognition vector is then input to the well trained PNN classifier. The resulting classification is obtained from the maximum-output-selector which is coupled with the output layer of the PNN classifier. The classification performance for each algorithm considered was evaluated by using the criterion described in the following section.

6.5 Evaluation of the System Performance

One major effort in this research was to study correlations between features extracted from images of different objects. To accomplish this goal, the coefficient of correlation is used to measure the degree of association between the actual output of a node and the output of the node corresponding to the correct class. That is

$$\tilde{r}_{ij} = \frac{\sum(O_i - \bar{O}_i)(O_j - \bar{O}_j)}{n s_i s_j}, \tag{6.7}$$

$$s_i = \sqrt{\frac{\sum(O_i - \bar{O}_i)^2}{n-1}}, \tag{6.8}$$

$$s_j = \sqrt{\frac{\sum(O_j - \bar{O}_j)^2}{n-1}}, \tag{6.9}$$

$$\bar{O}_i = \sum O_i/n,$$

$$\bar{O}_j = \sum O_j/n,$$

where O_i is the output of i-th node which corrresponds to the features extracted from the i-th object, and n is the number of training vectors extracted from each object class. The coefficient of correlation $\tilde{r}_{ij}$ produces values between -1 and 1. When two output variables are perfectly associated, $\tilde{r}_{ij}$ is ± 1. When two output variables have no correlation, then $\tilde{r}_{ij}$ equals 0.

To study the classification performance of a system, it was necessary to establish a function to measure the similarity of features extracted from images of the same object class and the differences in features extracted from images of different object classes. This measuring function was designed as,

$$\eta_k = \frac{\sum_{i=1}^{K} |O_i^c + O_i^a|}{2K}, \tag{6.10}$$

where K is the total number of output nodes, O_i^c and O_i^a are the correct value and the actual value, respectively, of i-th output node when the feature vector being recognized is from the k-th object class. Notice that $O_i \in [-1, 1]$.

The values resulted from this measuring function are between 0 and 1. If η_k equals 1.0, the PNN is said to have learned this feature perfectly. This is the result when the output of

the node corresponding to the k-th object class has the value of 1.0 and the output of the rest of the nodes are -1.0. If η_k equals 0.0, the PNN is said to have not learned this feature at all. This is the result when the output of the node corresponding to the k-th object class has the value of 1.0 and all other node outputs have values of 1.0. In fact, this function is analogous to the membership function described in fuzzy set theory [116]. Thus, it is called the *membership function.*

Another criterion in evaluating the system classification performance is the classification reliability which is defined as the fraction of the correct classifications,

$$R_c = \frac{N_c}{N_t} = \frac{N_t - N_w}{N_t}, \tag{6.11}$$

where N_c and N_w are numbers of correct classifications and mis-classifications, respectively; and N_t is the total number of classifications, $N_t = N_c + N_w$.

Since computer models are used in this research to extract training vectors, they are also used here to obtain recognition vectors to evaluate the classification performance. When the computer models are used to evaluate the classification performance, the orientations of the object models will not be the same as those that have been used to extract training vectors. The classification rate is denoted as R_c^{model} if the recognition vectors are derived from the computer models. Recognition vectors extracted from images that are acquired by a camera are used to compute the classification rate, R_c^{real}. The membership function along with these two classification rates are used in this research to assess three different selected algorithms.

While the membership function and the classification rates (reliabilities) are the primary considerations to evaluate the classification performance of the system, the training time is also a definite factor in evaluating the overall score of the system. The training time refers to the number of learning cycles required by a PNN classifier to derive a set of optimum weights. The mis-classifications when using the PNN classifier with the set of optimum weights to classify the training vectors should be smaller than a pre-specified number. One learning cycle is defined as the complete process of presenting all training vectors to a PNN classifier. Thus, a learning cycle consists of $K \times I$ updates of a PNN if K and I are number of objects and number of training vectors in each object, respectively.

7 EXPERIMENTAL RESULTS

In this chapter, the selected algorithms are examined and their experimental results are displayed. All of these experiments are conducted by using a same set of simple computer models which are shown in Figure 7.1. These computer models are created from the CAD/CAM system, Pro/Engineering, to accomplish one step forward of CAD/CAM integration.

As mentioned before, one objective of this research was to investigate the inter-relationships between features derived from both similar and significantly different objects. Therefore, from the objects being chosen more focus is placed on their geometric similarities or distinctions than their own geometric complexities. The simple objects in Figure 7.1 were chosen to concentrate on this objective. It is noted that object 1 and object 4 are similar and the only distinction is their heights, with object 1 at 1.0 inch and object 4 at 0.9 inch. Object 2 is similar to object 3, except object 2 is topped with a cube and object 3 is topped with a cylinder. The last object (object 5) is not similar to any of the other four objects.

All the experimental results given in this chapter have been obtained by using the same perceptron neural network with the same training mechanism which is the back-propagation of errors learning algorithm. It consists of three layers, where the number of nodes in the input layer equals the length of the feature vector, the number of hidden nodes is forty-nine, and the number of output nodes is ten. Although two and four layer perceptron neural networks have also been attempted in this research, their results were overwhelmingly out-performed by the three layer network. Their results will not be listed in this dissertation, but they will be discussed in the final section of this chapter.

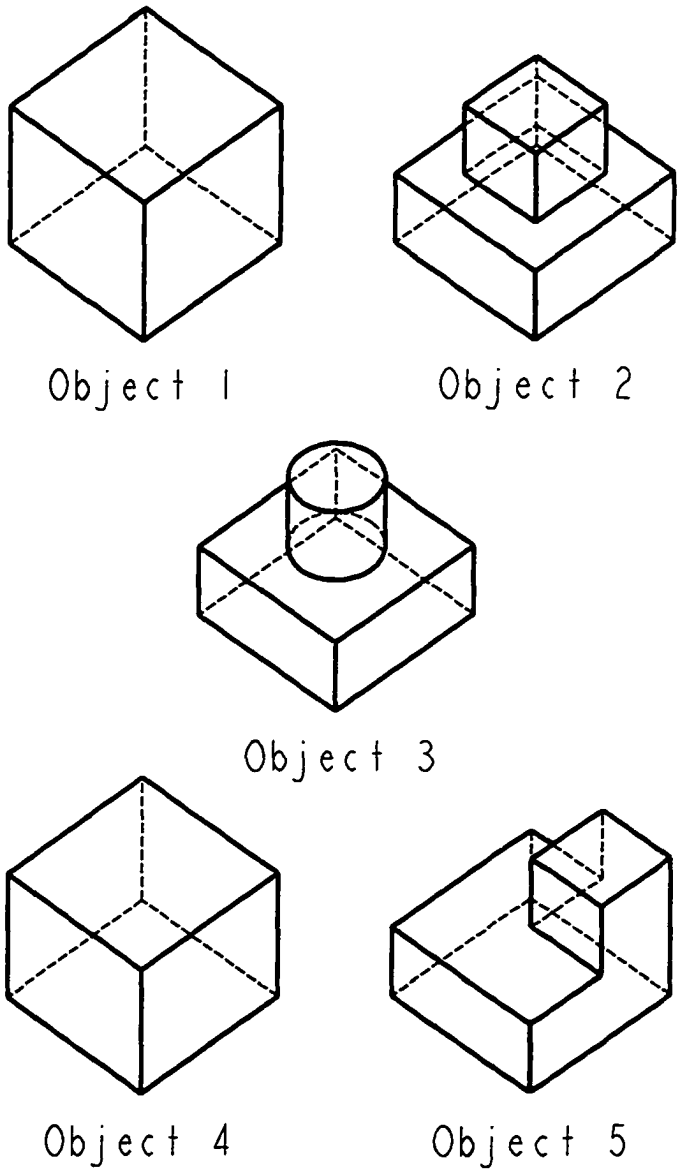


Figure 7.1 The five simple objects used in the experiments.

The feature vectors for training and recognition were derived from images which have pixel intensity values in single-precision floating points. Each intensity value was converted from the screen displaying an object by using the OpenGL [112] routine, *glReadPixels(origin_x, origin_y, width, height, GL_LUMINANCE, GL_FLOAT, &intensity)*.

This chapter is organized as follows: The first section lists different variables used in the experiments and the second section summarizes all experimental data for the algorithms used in this research. The third section provides the analytical result based on the experimental data. Finally different algorithms will be compared and the algorithms that have not been susseccfully implemented will also be discussed.

7.1 Experimental Variables

The experiment was conducted in a way such that one can determine how effective the different features are in recognizing 3D objects as well as in training a neural network. Changing features result from changing the number of projection vectors of each image used in the training and recognition. Changing features also results from changing the method of integrating two (or more) projection vectors. From the discussion in Chapter 5, features derived from different algorithms will have different effects on the probability of getting an identical feature vector from different images. This will affect the recognition rate as well as the training time. Table 7.1 lists these different features.

Table 7.1 Different features.

Property	Feature 1	Feature 2	Feature 3
Number of Projection Vectors	1	2	2
Method of Combining Vectors	x	$(x+y)/2$	(x, y)
Number of Neural Networks Used	1	1	2

The feature vector robustness is studied by varying the number of training features extracted from each object. Changing the number of training features results from adjusting the angular interval in which the projection vectors are taken. Changing the angular interval will mostly affect the recognition results of objects rotated with angles that are not integer multiples of the angular interval. Table 7.2 shows the number of training and recognition features with different angular intervals.

Notice that, the recognition features that are used to test recognition performacnes are the intermediant features that are derived from the mid-point of an interval, $\frac{(2n+1)interval}{2}$, where $n \in [0, \frac{360}{interval}]$. These features provide the weakest relation to the training features and the recognition results of these features demonstrate their robustness. In the experiment, the training features have two important uses. One is to train a neural network and another is to evaluate the membership function. The membership function represents the trainability of these features to the neural network.

The variables of a neural network are important to the training process. The changes of these variables affect the convergent time of a neural network. The initial values of these variables of a neural network are set to be the same at every beginning of the training processes. They will be gradually adjusted according to different feature extracting algorithms in order to achieve the minimal number of recognition errors. Their settings are displayed in Table 7.3.

7.2 Experimental Data

In the previous chapters, two characteristics of features have been outlined. It is obvious that the features extracted from the proposed algorithms have the wide domain property, because every image can have a set of projection vectors. However, the characteristic of uniqueness has not been contested even if it has been mathematically demonstrated for its feasibility in Chapter 5. Experimentally, there are several ways to illustrate this characteristic of our proposed features. Data tables A.1 through A.9 included in the Appendix are results from the training processes. The training time or the number of training epoches can be used

Table 7.2 Feature number variation in feature vector robustness experiments (5 objects).

Angular Invertal		Feature 1	Feature 2	Feature 3
10°	Number of Projection Vectors	360/10	2 × 360/10	2 × 360/10
	Number of Training Features	360/10	360/10	360/10
	Number of Recognition Features	4 × 9 + 36	4 × 9 + 36	4 × 9 + 36
6°	Number of Projection Vectors	360/6	2 × 360/6	2 × 360/6
	Number of Training Features	360/6	360/6	360/6
	Number of Recognition Features	4 × 15 + 60	4 × 15 + 60	4 × 15 + 60
4°	Number of Projection Vectors	360/4	2 × 360/4	2 × 360/4
	Number of Training Features	360/4	360/4	360/4
	Number of Recognition Features	4 × 23 + 90	4 × 23 + 90	4 × 23 + 90

Table 7.3 Experiment parameters of a neural network.

Parameters		Neural Network Setting
Gain Factor	Initial	0.5
	Adjusted	Variable
Momuntum Factor	Initial	0.9
	Adjusted	Variable
Flat Spot Handling Factor	Initial	0.1
	Adjusted	Variable
# of Input Nodes		240
# of Hidden Nodes		49
# of Output Nodes		10

to measure the uniqueness. From these tables (Table A.1 to Table A.9) we can see evidence that some types of features exhibit unique charateristics and some others might not. Notice that the criterion for stopping the neural network training in all the experiments is based on the stability of the output values of the neural network. Within some number of training epoches, if the changes between two consecutive output values for all output nodes are less than a specified value (.000005 was used in the experiment), the training process is terminated and the network is treated either as a well-trained classifier or as a non-well-trained neural network.

While the above data displayed some preliminary information about the trainability of the features to a perceptron neural network, Table A.10 to A.17 provide data for the detailed analytical results concerning uniqueness. The data is recorded from the output nodes of a trained neural network and the training features were used as the testing features. In the tables, the **Object Nodes** correspond to the features extracted from the corresponding objects while the **Null Nodes** correspond to no objects. The **Object Number** column represents which object was used to extract the corresponding training features.

The following experiments are conducted to evaluate the feature robustness or the correlations between the intermediate features and their neigboring training features. They are also used to evaluate the recognition performances of different proposed algorithms based on the classification ratios and the membership values. In Table A.18 to A.25, the **Object Number**

column represents object that was used to extract the corresponding recognition features.

7.3 Analytical Result

From Table A.1 to A.9, the training time for each method with different angular intervals can be seen from the column of **Number of Training epoches** in each table. The method using features extracted from only one projection vector has the longest training period compared to other methods. This is because the other methods use more than one projection vector to construct the training features. As detailed before, the more vectors used to construct features, the less chance there is to get misclassification of features. Thus, the neural network converge faster to the acceptable stage for other two methods using multiple projection vectors.

When training features are extracted from only the projection vector in the X direction, and with 4^o angular interval (See Table A.3), a three layer perceptron neural network fails to learn these training features. As the number of training features becomes larger, the chance of getting identical features from different images of different objects becomes bigger. Since the training features with 4^o angular interval has the largest number among three selected angular intervals, these features have the biggest possibility to have many common or similar images from different objects. Therefore, in this case, the neural network cannot reach the requested optimal stage. It misclassifies 250 of a total 450 training features after 2575 training epoches. It is noticed that 250 is not the minimum number of misclassifications. Table A.3 shows that 98 is the smallest number in the experiment. However, the network is very unstable after 720 training epoches. An unstable network has the characteristic of unconvergent weights and can be diagnosed from the values of the output nodes which, when unstable, jump largely in each training cycle.

The above example demonstrates the limitation of a perceptron neural network in handling a large number of training features. However, it can separate every feature vector in the experiment shown in Table A.1 with 1115 training epoches. As the number of training features increases, such as the experiments in Table A.2 and Table A.3, it cannot completely distinguish every training feature even with longer periods of training time.

Table 7.4 Summary of the experimental data listed in Table A.1 to A.9.

Feature Types	Angular Interval	Total Number of Training Epoches	Number of Errors	Misclassification Ratio
X	10^o	1115	0	0
	6^o	1386	2	$2/120 = .017$
	4^o	2575	250	$98/180 = .544$
$(X+Y)/2$	10^o	255	0	0
	6^0	360	0	0
	4^o	660	1	$1/180 = .006$
(X,Y)	10^o	120	0	0
	6^o	1770	1	$1/120 = .008$
	4^o	670	4	$4/180 = .022$

Generally, the more the number of features used in the training, the longer the time spent on the neural network training. However, there is an exception which can be seen in Table A.8 and Table A.9. The training time with more training features is shorter than that with less training features. This may be caused by the inefficient training process, and the choices of the adjustments for the neural network parameters.

Comparing the three types of features, the second type (which is derived from the average of two orthogonal projection vectors) has the best overall performace in term of trainability to the perceptron neural network. It only misclassified one training feature with 660 training epoches for features extracted at 4^o angular intervals. The third feature, that use two orthogonal projection vectors independently, also provides good results for the trainability. It only takes 120 training epoches to allow a perceptron neural network to learn features without misclassifying any of them when they are extracted at 10^0 angular interval. This illustrates the uniqueness and potential for this type of feature. Table 7.4 summarizes the experimental data lised in Table A.1 to Table A.9 with respect to the total number of training epoches and the number of misclassification for the neural networks in the well-trained state.

Table 7.5 shows the evidence of the unique characteristic of the proposed features since every membership value of different objects for the different type of features is very close to one on the average. This indicates that a perceptron neural network is capable of separating these features with good correlation.

Table 7.5 Average membership values of training features.

Feature Types	Angular Interval	Object				
		# 1	# 2	# 3	# 4	# 5
X	10^o	.973942	.966219	.969950	.964914	.969370
	6^o	.961162	.975111	.972115	.966638	.980638
	4^o	N/A	N/A	N/A	N/A	N/A
$(X+Y)/2$	10^o	.978722	.983512	.971475	.978326	.991039
	6^o	.971750	.970643	.970949	.974436	.976849
	4^o	.977311	.961535	.978571	.976859	.978785
(X,Y)	10^o	.939218	.977447	.975113	.941804	.946360
	6^o	.935623	.977459	.973525	.941285	.964730
	4^o	.899274	.967329	.979902	.908520	.944005

However, the main concerns for this research are not only the trainability, i.e., the wide domain and uniqueness, of the features, but also the robustness, i.e., the relationships between neighboring and intermediate features. The experimental data displayed in Table A.10 is collected from the output values of a well-trained neural network in classifying the training features which were constructed from projection vectors of object images. A set of object images is derived by rotating objects in 10^o steps around a circle. Although, there is no misclassification in the training features and high membership values for each object, the ratio of misclassifications are high for the intermediate features.

Table A.17 shows experimental data with the misclassified features in bold face. Notice that all features in the robustness experiments are taken from the mid-points of two consecutive training features which correspond to the examination results in Table A.10. The total number and ratio of misclassifications are 37 and 0.514, respectively as shown in Table 7.6. This is contradiction to the zero misclassification in the training features. From this point of view, it could be concluded that the neural network has not yet been taught to establish the relationship between neighbouring features based on this set of training features. Therefore, the classifier has not been trained to achieve the required robustness.

In fact, the other two types of features have the same problem of building the neighbouring feature relations when training features are extracted at 10^o angular intervals. It can be seen from Table 7.6 that misclassification ratios for the second and third types of features were

Table 7.6 Misclassified intermedian features for experiments in Table A.18 to A.25.

Feature Types	Angular Interval	Misclassified Features in Object					Misclassification	
		# 1	# 2	# 3	# 4	# 5	Number	Ratio
X	10^o	7	4	3	7	16	37	.514
	6^o	10	8	3	8	12	41	.342
	4^o	N/A	N/A	N/A	N/A	N/A	N/A	N/A
$(X+Y)/2$	10^o	1	5	4	4	2	16	.222
	6^o	6	4	6	5	12	33	.275
	4^o	0	0	0	0	8	8	.044
(X,Y)	10^o	7	1	2	3	6	19	.264
	6^o	5	1	2	5	9	22	.183
	4^o	0	0	0	0	8	8	.044

reduced to about half comparing with the first feature type. This provides direct evidence that the feature types, $(X+Y)/2$ and (X,Y), are better than feature type X.

When the robustness experiments are conducted in 6^o angular intervals for the three types of features, the second and third types have the same advantage over the first type in term of misclassification ratios. With the exception of the second type of feature, the smaller the angular interval, the less the misclassification ratio will be. The exception may be due to insufficient training time of the experiment. Even though the network can find the optimal solution for this set of features with few training epoches, the inter-relations between nodes have not been established in the same time frame. Comparing the number of training epoches of this experiment for feature two with the others, its number is much smaller. This points out a disadvantage of utilizing the perceptron neural network as the object classifier, and the difficulty of deciding on the feasible or workable trained network.

The experiments shown in Table A.20 and A.25 give very good results in feature robustness. This is, when the training features are extracted at 4^o angular intervals, their relationships with their neighboring features are strong enough to take effect on the intermediate features. The misclassification ratios are significantly reduced to 4.4 percent for both $(X+Y)/2$ and (X,Y) features.

It has been realized that the object images used to extract the intermediate features in the

above two experiments are only 2^o away from the images used to extract the training features. Therefore, it is necessary to investigate the misclassification ratios for the features deviated 2^o from the training features using the networks that are trained by features extracted from 6^o angular interval. Table A.26 and A.27 display the experimental results and Table 7.7 lists misclassification numbers as well as the ratios.

Table 7.7 Misclassifications for features 2^o deviated from the training features (see Table A.26 and A.27.

Feature Types	Misclassification Number in Object					Misclassification	
	# 1	# 2	# 3	# 4	# 5	Number	Ratio
$(X+Y)/2$	8	10	8	9	20	55	.229
(X,Y)	8	4	5	11	16	44	.183

Even if the neural networks are used to classify features that are only 2^o away from the training features, their misclassification ratios are still larger than the ratio that is obtained by the networks trained at 4^o angular interval. This is another way to demonstrate the characteristic of robustness for the proposed features.

Another emphasis in this section is to mathematically or statistically model the output node values of a trained neural network. The effort of this modeling is try to predict which intermediate feature will be misclassified from the information provided by the output values of a neural network examining the training features. A model based on the average membership values provides a relatively good prediction for classifying intermediate features and will be outlined here, although several models have been analyzed.

Considering a membership value of examining a training feature, if it is large relative to other values in the same object class, it is said its corresponding feature is definitely distinct from other features. Thus, the feature can be easily learned. On the other hand, if this feature is significantly distinct from its neighbouring features, a neural network will not be able to establish the relationship between it and other features. Conversely, if a membership value is too small, it is said that the neural network has not been able to learn its corresponding feature. The worst case happens when these two situations occur consequently. The intermedian feature will not be correctly classified because its neighboring training features either is not well learned

or does not train to build correlation to it.

Therefore, the statistical model is based on the above senario to predict the results of recognizing intermedian features. Let η_i^k and η^a be the individual and average membership values of object class k, repectively, and let σ^k be the standard deviation of η_i^k. If a membership value $\eta_i^k \in [\eta^a - \tau_l\sigma^k, \eta^a + \tau_u\sigma^k]$, it will be considered as an extreme value and at least one of the intermedian features adjacent to its corresponding training feature will be misclassified as predicted.

However, the choices of τ_l and τ_u that will be considered as the extremes are the problem issue here since there is no prelimenary data to be studied. The only way to resolve this issue is to study the experiments conducted in this research. The experiments shown in Table A.17 to A.25 as well as many others that are not shown in this dissertation have been analyzed. It has been found that the values of τ_l and τ_u depend on the type of features and the angular interval used in the experiment. For experiments in Table A.18, A.20, and A.23, when τ_l and τ_u are 1.25 and .5, respecively, 80 percent of the misclassifications can be predicted. If $\tau_l = 1.25$ and $\tau_u = 0.75$, the model will give 80 percent correct prediction of misclassifications for experiments in Table A.19, A.21, and A.24. If $\tau_l = 1.25$ and $\tau_u = 1.0$, 70 percent of misclassifications can be predicted for experiments shown in Table A.22 and A.25.

7.4 Comparison and Discussion

The comparison among the three types of features favors the type 2 and type 3 features ($\frac{X+Y}{2}$ and (X, Y), respectively). They provide better trainability and exhibit relatively high feature correlations. Although the first feature type requires less computation effort, a feature of this type does not present the necessary good characteristic of uniqueness. The classifiers trained by this feature have the highest misclassification ratios among eight robustness experiments. When a classifier is trained by features extracted from 4^o angular intervals, its weights can not converge to a stable stage. It misclassifies at least more than 25 percents of the training features.

It is not easy to categorzie the overall advantage or disadvantage between the type 2 and

3 features. Computationally, type 2 requires less operations. It needs only one more additive compared to the type 1 feature, but the classifier trained by this type of feature extracted from 4^o angular interval only missed one training feature and provides 95.6 percent correct identification of intermediate features. In addition, if only a simple maximun selector is used in the classifier without the use of threshold, there are only 4 misclassifications in the experiment shown in TableA.22. Notice that, the threshold in a classifier is used to constrain the maximum value among output nodes. If the maximum value among output nodes is smaller than the threshold, the feature will be categorized as unknown. All classifiers except those trained by feature type 3 use 0.25 as the threshold value.

Experimentally, feature type 3 has not shown any advantage over feature type 2. It requires two seperate perceptron neural networks with identical topology. The computational effort for training or recognizing the type 2 features is twice as much as the feature type 1. Thearetically, this feature has more uniqueness characteristic than the others since its two feature vectors are inputed into a classifier independently and combined in the output nodes. If more projection vectors are used in training such as in the case of 2^o or 3^o angular interval, a classifier trained by this type of feature will be much easier and faster to converge to a stable stage.

As mentioned before, the study of feature robustness was performed by varying the angular interval in which the projection vectors are taken. Changing the angular interval gives the resulting classification ratios as seen in Table A.18 to A.25. Besides this type of robustness experiment, there was no other type of experiment performed in this reseach to test the robustness associated with random noise, image distortion, scale and translational variation, etc. Since the noise filtering mechanism has been implemented in the research and the training and recognition features are extracted in the way to overcome a small amount of translational variation, the robustness of the type 2 and type 3 features would not be the main concern for the proposed features.

Image distortion and scale variation are the main considerations in affecting the classification ratio. This is because the features proposed in the research are established to distinguish very small variation between different objects, such as object 1 and object 4 in Figure 7.1.

Therefore, a classifier trained by these features will misclassify objects with small amount of image distortion and scale change.

In the previous section, a statistical model has been considered to predict the future misclassification of intermediate features. The advantage of establishing this type of model is obvious. Due to the nature of three-dimensional object appearances in the real world, it is very difficult if not impossible to derive a classifier with perfect identification of every object. If a model can be established successfully, misclassifications can be avoided by carefully rearranging the object orientation.

In order to obtain a statistical model, it is necessary to find the relations between input features and output values. However, the relations between input features are very difficult to analyze since each feature contains a large number of elements. The values from output nodes seem feasible for such analysis since they only have 10 elements. However, by trial and error, only the model presented in previous section gives a relatively good prediction result. Even with this model, the choice ofτ_l and τ_u are ambiguous. They are strictly based on the number of misclassifications in the robustness experiments. If a better model can be found, intensive study must be performed to analyze every data set. It is out of scope for this research to pursue this kind of analysis.

The experiments in this research have been repeated many times. Only one of each type is documented in this dissertation. In the starting stage, many experiments failed due to the lack of experience in the choice of a perceptron neural network topology and the adjustment of its variables. Initially, a four-layer perpectron neural network was implemented as the classifier. As detailed in Chapter 3, this type of network has the ability to separate a very complicated set of features, such as a set of features belonging to one object embraced by another set of features belonging to different objects. Mathematically, it can learn all possible features with whatever complexities as long as no features belonging to different objects are identical. The disadvantages of this network are its tremendous training and recognition time, and the difficulties of controlling its variables.

Every experiment conducted in this research has failed to obtain a successful four-layer

perceptron neural network. Since a network with four layers has complex geometric representation, slice changes in its parameters will result in unpredictable output variations. The main reason for the failure in the four layer experiments is the unstable weights or the unconvergent output nodes values of a four-layer perceptron neural network.

Two-layer perceptron neural networks were also implemented in this research during the initial stages. Experimentally and theoretically this network is not suitable for this reseach due to its learning ability and the complexity of the proposed features. Although it is relatively fast for this type of network to achieve a stable state, its output values never meet an accuracy requirement, i.e., it misclassifies more than half of its training features. Through many preliminary examinations, the three-layer peceptron neural network has been chosen as the classifier for this research. It provides better recognition and training results with reasonable computational effort. Some of the successful experiments are listed in Table A.1 to A.25.

In Section 5.1 and 5.2, two other features have been proposed but their implementation and and experimental results are not included in this thesis. In fact, feature (SVDO) proposed in Section 5.2 was first implemented. It has only five training features and each of them corresponds to one object. In the recognition process, it requires calculation of the distances between the recognition vector and each projection vector. The projection vector with the minimum distance to the recognition vector is replaced by the recognition vector to construct the so called *recognition matrix*. Five objects will bring in five recognition matrices and five SVD (Singular Valued Decomposition) operations. Another aspect of this feature is the extra use of the neural network. Since the calculation of minimum distances between features is the process of deriving the most similar features, it is not necessary to use a neural network to recalculate the similarity.

Feature (SVDI) described in Section 5.1 is the extension of feature (SVDO). It uses more than one projection vector in each image to construct training or recognition features rather than just one in the SOPI features. It needs $n \times k$ SDV operations, where n is number of objects and k is the number of orientations of each object, before the training is started. However, it only requires one SVD operation in the recognition process.

Originally, 36 projection vectors were used to construct the *projection matrix* since its idea was borrowed from feature SOPI. If there are more than 36 elements in each projection vector, the rank of a projection matrix will be 36. This requires extremely intensive computation effort in the training process. Even with one SVD operation, the recognition speed using this feature is still much slower than that using the feature detailed in Section 5.3 due to the long computational time spent in calculating a 36 rank matrix. Therefore, the experiment for this feature was not implemented

After the successful implementation of features (SOPI) described in Section 5.3, it suggests that features extracted from two projection vectors are sufficient to represent an object image. Thus, if a SDVI feature is calculated from a projection matrix with rank of 2, the computations in either the training or recognition process will be significantly reduced. Although it requires some extra computations compared with SOPI features, its inheret advantage can not be ignored. Since each training or recognition feature of this type has only two elements, a neural network will require less hidden nodes. Therefore it offsets some of the processing time spent in calculating the SVD. Another import advantage of the SVDI features is its feasibility of being analyzed. Unlike other feature types which have more than three elements in each feature and are difficult to be analyzed, the SDVI feature can be plotted in a simple XY diagram and its characteristics can be observed graphically. Based on its characteristics, the neural network topology can be chosen and the prediction model can be established much more easily. Due to time constraint of this research, it has not been experimentally implemented.

8 CONCLUSION

As stated in Chapter 1, the objective of this research was to develop a three-dimensional object recognition system that can be applied to automated assembly lines based on the demands in processing speed, intelligence, and integration with CAD systems. The purpose of this chapter is to summarize the conclusions drawn from the development of such a system and provide the recommendations for future investigation.

8.1 Summary

A complete three-dimensional object recognition system consists of image acquisition, image processing, feature extraction, and feature recognition. This work concentrated on the area of feature extraction - that of generating an effective feature to represent a three-dimensional object so that a non-traditional classifier, a perceptron neural network with back-propagation error training, can distinguish the object from others within desired time. Also, an analysis of inter-relations between features from five different three-dimensional objects was conducted based on the outputs of a perceptron neural network. Although there are a large number of different objects in the real world, the analysis provides a guide to other three-dimensional object recognition research that requires classification of similiar features. Another aspect of this research is the integration of a CAD systems with an object recognition system. It is obviously not a new concept to directly use CAD models in a recognition system. But there is very limited research or applications for object recognition systems that use neural network classifiers, especially for three-dimensional object recognition systems. This research demonstrates the feasibility of using CAD models to train neural network classifiers for recognizing three-dimensional objects. Several important contributions provided by this research can be

summarized as:

- Developed and implemented algorithms for deriving compact and representative features for a perceptron neural network classifier to recognize three-dimensional objects.
- Analyzed and displayed the inter-relationships between training features and intermediate (recognition) features representing different 3D objects.
- Successfully intergrated CAD models into the application of 3D object recognition system.

8.2 Future Investigation

Different aspects of the research that can be further investigated and explored based on this work are recommended in this section. They are based on the experience gained during the development of the 3D object recognition system. Such additional work will help in developing a truly flexible, as well as accurate, recognition system for applications in industrial automation. This will eventually help in eliminating the need for human intervention in hostile working environments.

8.2.1 Integrating the complete system

This research has integrated every essential component required in an object recognition system (see Figure 2.1). However, to apply it to a robotics application it is necessary to implement several other processes. One of them is to locate an object in a work cell. This can be accomplished by using the method presented in Section 6.3. Another process is to minimize the environmental effects on the recognition accuracy. These effects are from the environmental lighting or the object illumination. This can be implemented by adjusting the illumination of computer models to match real object images. The last necessary process is to establish the communication link between a recognition system and a robot so that the complete operation can be conducted in an orderly fashion.

8.2.2 Implementing the SVDI algorithm

Three proposed algorithms for feature extraction in this research have been implemented and experimented. The experiment for the SVDI algorithm failed to provide a satisfactory recognition result. The significant problem was the training and recongition time. It used 72 projection vectors to construct a projection matrix. For 10^o angular intervals used to obtain images from five objects, there were 5×36 72-rank matrices that required calculation of singular values. The training time was extremely long. In each classification process, a 72-rank matrix must be computed and thus, the recognition time was much longer those practical for automation in today's industry.

However, if only two or three projection vectors are used to construct the projection matrix, the computational time for obtaining singular values was reduced significantly. Therefore the training and recognition time was much faster. The main issue now becomes how well these two or three singular values can represent an image. From the experiments conducted in this research, there were 96.4 percent correct classifications when two projection vectors are used to construct feature vectors. Although some information may be lost during the transformation from projection matrix to singular values, some information can be gained by increasing the number of projection vectors used in the projection matrix. The primary advantage is the compact form of a neural network classifier required by the algorithm. The neural network needs only two input nodes if two projection vectors are used to form a projection matrix. Consequently, the training and classifying time is dramaticly reduced. Another advantage of this algorithm is its ease of analysis of the correlation between input features and output values due to the small number of elements in a feature vector.

8.2.3 Optimizing the training process

In the training process, the parameters of the perceptron neural network were manually adjusted so that its weights could reach the local or global minimun. The decision of the adjustment was based on the number of misclassifications and the output values in a sequence of training epoches. If a big jump in misclassification number was observed after a consecutive

training epoches, one or more parameters will be adjusted. In most case, the values of these parameters were decreased. By doing that, a neural network could be converged to a global or local minimum more quickly compared to random adjustment. The problem with this strategy is the need of human intervention. In order to achieve complete automation, an adjustment mechanism needs to be implemented. The implementation is trivial rather than difficult. It can be done by simply applying the "if-else" statements in the C or C++ programs.

8.2.4 Eliminating the step of image processing

During the recognition stage, an image is first acquired by a digital camera and then processes through a series of image processing operations in order to filter out noise and obtain a classification feature. The feature is fed into a trained neural network to finish the recognition. The most time comsuming step is the image processing task. It consists of image enhancement, edge detection, and object image-background separation. The image processing step can be eliminated from the classification process by applying the texturing technique in the training stage.

As described in Chapter 2, the objective of image processing in an object recognition system is to obtain a "clean" image. An object contained in a "clean" image should have the same appearance as a computer model which does not have any background noise. In a learning process, training features are extracted from these computer models, and a neural network only learns models without any background noise. Thus, the image processing techniques need to be applied to a noised image in order to have a good recognition result. However, if a neural network is trained by using computer model with added noise, it will be capable of recognizing an object in an image which comes directly from a digital camera. Therefore, the recognition time will be reduced significantly without any extra time spent in the training process.

To accomplish this elimination, an image of a platform which supports objects should be obtained by a camera. When the computer model is constructed, this image will be used as the background of the model so that the training features extracted from computer models will consist for background noise which is similar to that of object image from camera. The

need of object and background separation is eliminated.

8.2.5 Establishing an explicit statistical or mathematical model

Due to the time and equipment constraints, the experimental analysis in this work was carried out by only using the output node values of a perceptron neural network. Although the analysis provided some indications of correlations between training features and intermediate features, there was no further study of relationship between input features of different objects as well as the relationship between training features and intermediate features themselves. An inside view of the proposed features has not been explored. Since it is important to gain the inside view of these features for future optimization processes, an explicit statistical or mathematical model needs to be established. Notice that an explicit model does not mean that a statistical or mathematical formula exactly fits feature vectors. However, it should provide information about feature correlations. Based on the known information, the number of nodes and topology of a neural network as well as the required training features can be optimized.

APPENDIX TABLES OF EXPERIMENTAL DATA

Table A.1 Training process using features extracted from only the projection vector in X direction with 10° angular interval: X.

Gain Factor	Momuntum Factor	Flat Spot Handling Factor	Number of Training Epoches	Number of Errors
.5	.9	.1	50	117
.5	.8	.09	100	111
.5	.75	.075	225	74
.5	.7	.075	295	80
.4	.7	.075	330	72
.4	.6	.06	370	47
.4	.6	.05	410	53
.35	.6	.05	440	81
.35	.55	.05	475	49
.3	.55	.04	515	39
.3	.5	.04	545	53
.25	.5	.03	590	16
.25	.4	.025	630	25
.2	.4	.025	720	12
.2	.35	.0175	760	3
.15	.35	.0175	795	4
.15	.275	.0175	825	5
.1	.275	.0175	845	3
.1	.225	.0125	885	3
.075	.225	.01	1115	0

Table A.2 Training process using features extracted from only the projection vector in X direction with 6° angular interval: X.

Gain Factor	Momuntum Factor	Flat Spot Handling Factor	Number of Training Epoches	Number of Errors
.5	.9	.1	60	178
.4	.75	.08	120	154
.35	.6	.06	156	122
.25	.5	.04	192	99
.25	.45	.03	234	105
.2	.3	.02	294	74
.1	.3	.02	354	93
.1	.2	.01	444	75
.05	.1	.0075	504	77
.2	.35	.0075	564	90
.2	.2	.0075	624	76
.15	.2	.0075	690	88
.15	.15	.0075	750	127
.5	.75	.08	804	143
.5	.7	.08	870	166
.4	.7	.08	930	158
.4	.6	.06	984	105
.4	.6	.05	1008	930
.35	.6	.05	1026	89
.35	.55	.05	1050	111
.3	.55	.04	1080	75
.3	.5	.04	1110	51
.25	.5	.03	1146	41
.25	.4	.03	1182	67
.2	.4	.025	1218	43
.2	.35	.025	1248	39
.15	.35	.0175	1272	13
.15	.275	.0175	1290	9
.1	.225	.0125	1314	3
.075	.225	.01	1338	3
.075	.175	.01	1356	2
.05	.125	.0075	1386	2

Table A.3 Training process using features extracted from only the projection vector in X direction with 4° angular interval: X.

Gain Factor	Momuntum Factor	Flat Spot Handling Factor	Number of Training Epoches	Number of Errors
.5	.9	.1	100	304
.5	.8	.08	175	224
.4	.8	.08	260	211
.4	.7	.08	310	247
.35	.7	.07	360	194
.35	.6	.06	410	203
.3	.6	.05	485	165
.3	.55	.04	550	186
.275	.55	.03	610	130
.25	.5	.025	695	135
.25	.4	.0225	720	98
.225	.4	.0225	920	135
.225	.4	.02	1020	138
.2	.4	.02	1125	130
.2	.375	.0175	1225	123
.2	.425	.0175	1380	138
.175	.425	.0175	1520	158
.175	.425	.015	1570	144
.175	.4125	.015	1620	120
.2	.4125	.015	1770	113
.175	.4125	.015	1870	121
.175	.4	.015	1970	143
.15	.4	.015	2225	135
.15	.4	.0125	2325	153
.15	.375	.0125	2575	250

Table A.4 Training process using features extracted from the X and Y projection vectors with 10° angular interval: $(X + Y)/2$.

Gain Factor	Momuntum Factor	Flat Spot Handling Factor	Number of Training Epoches	Number of Errors
.5	.9	.1	25	116
.4	.8	.08	50	115
.35	.7	.06	75	48
.3	.6	.05	100	55
.25	.5	.35	130	13
.2	.4	.025	175	7
.175	.3	.0175	200	3
.15	.25	.0125	230	2
.125	.225	.0075	255	0

Table A.5 Training process using features extracted from the X and Y projection vectors with 6° angular interval: $(X + Y)/2$.

Gain Factor	Momuntum Factor	Flat Spot Handling Factor	Number of Training Epoches	Number of Errors
.5	.9	.1	6	156
.4	.85	.085	54	142
.35	.85	.07	72	62
.3	.7	.05	108	43
.25	.6	.04	120	19
.125	.4	.025	132	9
.075	.2	.0125	174	6
.075	.3	.0125	204	3
.1	.3	.0125	252	1
.1	.3	.01	264	1
.15	.3	.01	294	2
.125	.3	.0125	312	2
.125	.35	.0125	360	0

Table A.6 Training process using features extracted from the X and Y projection vectors with 4° angular interval: $(X + Y)/2$.

Gain Factor	Momuntum Factor	Flat Spot Handling Factor	Number of Training Epoches	Number of Errors
.5	.9	.1	70	243
.4	.9	.09	120	256
.4	.8	.08	170	118
.35	.75	.07	220	79
.35	.7	.06	270	118
.325	.65	.05	320	64
.3	.6	.04	370	34
.275	.5	.03	420	28
.25	.4	.02	470	17
.225	.35	.015	520	14
.2	.3	.01	560	7
.2	.225	.0075	660	1

Table A.7 Training process using features extracted from the X and Y projection vectors with 10° angular interval: (X, Y).

Gain Factor	Momuntum Factor	Flat Spot Handling Factor	Number of Training Epoches	Number of Errors
.5	.9	.1	30	83
.4	.8	.075	55	26
.3	.6	.05	85	4
.2	.3	.025	120	0

Table A.8 Training process using features extracted from the X and Y projection vectors with 6° angular interval: (X, Y).

Gain Factor	Momuntum Factor	Flat Spot Handling Factor	Number of Training Epoches	Number of Errors
.5	.9	.1	20	244
.4	.85	.085	100	198
.35	.85	.07	130	180
.3	.7	.05	180	122
.25	.6	.04	200	94
.125	.4	.025	220	36
.075	.2	.0125	290	22
.075	.3	.0125	340	22
.1	.3	.0125	420	54
.075	.3	.0125	535	27
.075	.3	.01	585	36
.1	.3	.01	675	37
.09	.25	.009	750	19
.08	.225	.007	900	33
.08	.225	.005	1005	41
.06	.2	.0025	1115	39
.15	.3	.01	1165	35
.15	.3	.005	1215	35
.125	.25	.0025	1385	40
.15	.3	.001	1505	33
.15	.3	.002	1515	33
.175	.35	.002	1640	24
.175	.35	.002	1760	1
.125	.25	.001	1770	1

Table A.9 Training process using features extracted from the X and Y projection vectors with 4° angular interval: (X, Y).

Gain Factor	Momuntum Factor	Flat Spot Handling Factor	Number of Training Epoches	Number of Errors
.5	.9	.1	10	95
.25	.5	.025	45	27
.2	.4	.0175	105	15
.175	.35	.0125	270	39
.2	.4	.0175	320	24
.175	.4	.0125	360	13
.15	.3	.0075	410	19
.125	.25	.005	460	12
.1	.225	.0025	565	22
.125	.25	.0025	580	9
.125	.25	.00125	650	19
.1	.2	.001	670	4

Table A.10 Examing the neural network which was trained using the features extracted from the projection vector in X direction with 10° angular interval: X.

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 1	0°	.890276	-.945713	-.897612	-.985586	-.999990	-.9
	10°	.999746	-.991037	-.923865	-.997581	-.996655	-.9
	20°	.911941	-.877238	-.983918	-.884757	-.983653	-.9
	30°	.984602	-.998165	-.926912	-.954480	-.998587	-.9
	40°	.997172	-.975274	-.840722	-.973297	-.999995	-.9
	50°	.855606	-.997023	-.940641	-.898672	-.995958	-.9
	60°	.979089	-.968480	-.925841	-.948633	-.992880	-.9
	70°	.898651	-.999525	-.880246	-1.00000	-.880898	-.9
	80°	.888783	-.998858	-.924818	-.906437	-.855017	-.9
	90°	.890276	-.945713	-.897612	-0985586	-.999990	-.9
# 2	0°	-.999980	.793570	-.948394	-.886909	-.959692	-.9
	10°	-.998903	.936016	-.829391	-.884906	-.999502	-.9
	20°	-.999765	.953875	-.913799	-.997644	-.918247	-.9
	30°	-.997764	.916315	-.887380	-.999139	-.862538	-.9
	40°	-.974891	.857348	-.942672	-.873573	-.867423	-.9
	50°	-.999583	.936037	-.904733	-.890251	-.993643	-.9
	60°	-.911846	.829601	-.931618	-.975607	-.873556	-.9
	70°	-1.000000	.959250	-.928341	-.999998	-.999190	-.9
	80°	-.999980	.868238	-.918182	-.893628	-.946793	-.9
	90°	-.999980	.793570	-.948394	-.886909	-.959692	-.9
# 3	0°	-.999770	-.926733	.912228	-.826704	-.996270	-.9
	10°	-.934504	-.918859	.971812	-.997792	-.999958	-.9
	20°	-.883317	-.848343	.909942	-.970650	-.999982	-.9
	30°	-.895169	-.982518	.899354	-.999894	-.999892	-.9
	40°	-.858631	-.885356	.850881	-.999995	-.917331	-.9
	50°	-.978732	-.859973	.851943	-.996337	-.999997	-.9
	60°	-.990007	-.910993	.881808	-.999610	-.864580	-.9
	70°	-.999911	-.991324	.862637	-.999103	-.890476	-.9
	80°	-.970321	-.908457	.962522	-.991384	-.999521	-.9
	90°	-.999770	-.926733	.912228	-.826704	-.996270	-.9

Table A.10 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node
		#1	#2	#3	#4	#5	Average
# 4	0°	-.936836	-.859098	-.961489	.920835	-.989078	-.9
	10°	-.999955	-.843214	-.931219	.988973	-.996196	-.9
	20°	-.901292	-.914354	-.978653	.880918	-.892224	-.9
	30°	-.976915	-.938496	-.906787	.999059	-.993967	-.9
	40°	-.880384	-.986238	-.915379	.867631	-.999964	-.9
	50°	-.883194	-.855504	-.881327	.922797	-.999671	-.9
	60°	-.998690	-.903070	-.864536	.915935	-.879947	-.9
	70°	-.999997	-.997690	-.938387	.903746	-.849124	-.9
	80°	-.864348	-.992455	-.901835	.873236	-.957648	-.9
	90°	-.936836	-.859098	-.961489	.920835	-.989078	-.9
# 5	0°	-.942973	-.998203	-.975376	-.999999	.971601	-.9
	10°	-.999969	-.876021	-.981324	-.933298	.937710	-.9
	20°	-.999701	-.996317	-.936642	-.998082	.999826	-.9
	30°	-.884756	-.828457	-.913461	-.999146	.952045	-.9
	40°	-1.000000	-1.000000	-.837390	-1.000000	1.000000	-.9
	50°	-.887032	-.955492	-.963844	-.853939	.999989	-.9
	60°	-1.000000	-.999949	-.967292	-.999977	.999980	-.9
	70°	-1.000000	-.999977	-.830706	-.824804	.889441	-.9
	80°	-.902823	-.857694	-.934663	-.936530	.871921	-.9
	90°	-.996605	-.792256	-.975031	-.991288	.969459	-.9
	100°	-1.000000	-.980047	-.987573	-.999475	.996538	-.9
	110°	-.999998	-.985479	-.987128	-.993607	.999378	-.9
	120°	-.993252	-.999948	-.937493	-1.000000	1.000000	-.9
	130°	-.869801	-.897670	-.785861	-1.000000	.875174	-.9
	140°	-.858628	-.998297	-.919242	-.991518	.976755	-.9
	150°	-.929773	-.971349	-.934216	-.898337	1.000000	-.9
	160°	-.913627	-.761614	-.962285	-.828245	.952858	-.9
	170°	-1.000000	-.999974	-.964459	-1.000000	1.000000	-.9
	180°	-.999975	-.999973	-.900862	-.999537	.850403	-.9

Table A.10 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 5	190°	-.993860	-.999978	-.975918	-1.000000	.996780	-.9
	200°	-.839486	-.994245	-.898522	-.989554	.809460	-.9
	210°	-.999999	-.999286	-.867581	-.999831	.999964	-.9
	220°	-1.000000	-.989503	-.947894	-1.000000	.999913	-.9
	230°	-1.000000	-.892904	-.883679	-.990412	.991129	-.9
	240°	-.821572	-.886828	-.799993	-.999074	.998960	-.9
	250°	-1.000000	-.994630	-.935022	-.849368	.867442	-.9
	260°	-.999996	-.915087	-.847138	-.916972	.993284	-.9
	270°	-.951991	-.862934	-.904693	-.999942	.970827	-.9
	280°	-1.000000	-.999974	-.989444	-1.000000	1.000000	-.9
	290°	-.999996	-.999795	-.949681	-.999921	1.000000	-.9
	300°	-.892392	-.830755	-.931805	-.999999	.999810	-.9
	310°	-.899227	-.819321	-.819497	-.841584	.819176	-.9
	320°	-.999993	-.998610	-.937199	-.999997	1.000000	-.9
	330°	-.888200	-.914192	-.964743	-.999499	.998595	-.9
	340°	-.969552	-.973508	-.856550	-.982714	.993927	-.9
	350°	.908226	-.997993	-.935093	-1.000000	.839508	-.9

Table A.11 Examing the neural network which was trained using the features extracted from the projection vector in X direction with 6° angular interval: X.

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node
		#1	#2	#3	#4	#5	Average
# 1	0°	.967160	-.894048	-.939316	-.708513	-.999996	-.900000
	6°	.841439	-.981876	-.936475	-.996378	-.874421	-.900000
	12°	.900848	-.995725	-.991287	-.915493	-.858734	-.900000
	18°	.837938	-.884432	-.951290	-.830019	-.999986	-.900000
	24°	.907500	-.965910	-.990394	-.971649	-.873874	-.900001
	30°	.898271	-.881468	-.932633	-.823749	-.998711	-.900000
	36°	.995711	-.913104	-.994382	-.825246	-.999976	-.900000
	42°	.857594	-.980084	-.905371	-.919784	-.840110	-.900000
	48°	.998652	-.943300	-.900611	-.987255	-.997099	-.900000
	54°	.771132	-.891309	-.920462	-.832052	-.850251	-.900000
	60°	.999545	-.917086	-.873546	-.999970	-.999869	-.900000
	66°	.862562	-.883168	-.956856	-.999993	-.999918	-.900000
	72°	.907668	-.948405	-.928963	-.912108	-.999169	-.900000
	78°	.872907	-.988859	-.937958	-.870597	-.881293	-.900000
	84°	.804574	-.968991	-.996674	-.999974	-.790652	-.900000
# 2	0°	-.996677	.955784	-.967129	-.983092	-1.000000	-.899999
	6°	-.999998	.875639	-.860631	-.999902	-.999999	-.900000
	12°	-.999999	.883765	-.884388	-.978315	-.999940	-.900000
	18°	-.999999	.932094	-.995709	-.999997	-.967363	-.900000
	24°	-.881753	.954287	-.978084	-.998407	-1.000000	-.900001
	30°	-.999266	.915635	-.862734	-.940199	-.999513	-.900001
	36°	-.878993	.855390	-.988636	-.943329	-.999970	-.900000
	42°	-1.000000	.882629	-.995185	-.999907	-.812903	-.900001
	48°	-1.000000	.866214	-.876125	-.999999	-.896245	-.900000
	54°	-1.000000	.895130	-.942263	-.863957	-.829116	-.900000
	60°	-1.000000	.877896	-.931049	-.989051	-.997352	-.900000
	66°	-1.000000	.877236	-.907652	-.951808	-.990302	-.900001
	72°	-.999994	.904591	-.996110	-.999999	-.998003	-.900000
	78°	-1.000000	.870864	-.956693	-.998010	-.863177	-.900000
	84°	-1.000000	.923904	-.920998	-.975868	-.999806	-.899999

Table A.11 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 3	0°	-.999989	-.999110	.856051	-.923714	-.821225	-.900000
	6°	-1.000000	-.953661	.988953	-.947886	-.908094	-.900000
	12°	-.999999	-.946959	.965993	-.896660	-.997496	-.900000
	18°	-1.000000	-.927963	.838501	-.999646	-.813348	-.900000
	24°	-.999859	-.848097	.877464	-.988710	-.985030	-.900000
	30°	-.999985	-.931716	.970591	-.999670	-.991523	-.900000
	36°	-.999863	-.905842	.876186	-.889984	-.996116	-.900000
	42°	-.999988	-.861641	.961115	-.999785	-.921978	-.900000
	48°	-.999971	-.901061	.970013	-.999427	-.982940	-.900000
	54°	-1.000000	-.893496	.874301	-.999998	-.909643	-.900000
	60°	-1.000000	-.897672	.862250	-1.000000	-.790883	-.900000
	66°	-.999849	-.956650	.992566	-.886559	-.996704	-.900000
	72°	-.999998	-.864391	.875052	-.987097	-.912452	-.900000
	78°	-.999999	-.947834	.853791	-.997729	-.963846	-.900000
	84°	-.999997	-.912476	.926344	-.999955	-.871978	-.900000
# 4	0°	-.998479	-.948607	-.913449	.981469	-.998135	-.900000
	6°	-.999956	-.903665	-.887014	.998254	-.999946	-.900000
	12°	-.866802	-.897031	-.929339	.895154	-.999642	-.899999
	18°	-.999901	-.992674	-.999216	.870216	-.799462	-.900000
	24°	-.855103	-.869344	-.867931	.901400	-.999920	-.900001
	30°	-.958577	-.864904	-.936446	.986600	-.868355	-.900001
	36°	-.956425	-.864944	-.910318	.982726	-.999981	-.900000
	42°	-.993496	-.930353	-.879976	.999966	-.995840	-.900000
	48°	-.943112	-.875804	-.988979	.960628	-.999361	-.900000
	54°	-.804057	-.939023	-.897474	.888543	-.814752	-.900000
	60°	-.851983	-.875276	-.900066	.871765	-.999995	-.900000
	66°	-.999394	-.956485	-.872572	.983994	-.999981	-.900001
	72°	-.942952	-.896976	-.883553	.929379	-.999975	-.900000
	78°	-.900454	-.997390	-.951572	.989555	-.897979	-.900001
	84°	-1.000000	-.999960	-.892208	.941303	-.848248	-.900001

Table A.11 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node
		#1	#2	#3	#4	#5	Average
# 5	0°	-.999979	-.999722	-.997416	-.999776	.999806	-.900000
	6°	-.996520	-.990790	-.993375	-.849773	.923139	-.900000
	12°	-.999658	-.886593	-.995213	-.880888	.977239	-.900001
	18°	-1.000000	-.996928	-.999994	-.999996	1.000000	-.900000
	24°	-1.000000	-.999456	-.993557	-1.000000	1.000000	-.900000
	30°	-.918930	-.899970	-.998661	-.997834	.997240	-.899999
	36°	-1.000000	-.998327	-.919914	-.999991	.999999	-.900000
	42°	-1.000000	-.999933	-.989820	-.999998	.999936	-.900000
	48°	-1.000000	-.745605	-.982125	-.998254	.988495	-.900000
	54°	-1.000000	-.997302	-.966330	-.967892	.999057	-.900000
	60°	-1.000000	-.999944	-.961852	-.999991	.999130	-.900000
	66°	-1.000000	-.998534	-.952280	-1.000000	.993426	-.900000
	72°	-.974082	-.997406	-.813632	-.999998	.880367	-.900000
	78°	-.765162	-.999469	-.984926	-.999999	.975022	-.900000
	84°	-.999962	-.789476	-.931206	-1.000000	.905147	-.900000
	*90°	-.769568	-.867177	-.999822	-.990185	-1.000000	-.900000
	96°	-.999972	-.862298	-.910819	-.999997	.976820	-.900000
	102°	-.999999	-.875490	-.829784	-.996491	.977574	-.900000
	108°	-.999983	-.996277	-.996624	-.978368	.769447	-.900000
	114°	-1.000000	-.999703	-.999650	-.650708	1.000000	-.900001
	120°	-1.000000	-.981770	-.986710	-1.000000	1.000000	-.900000
	126°	-.999184	-.782017	-.769815	-1.000000	.817520	-.900001
	132°	-.998528	-.999938	-.999831	-1.000000	1.000000	-.900000
	138°	-.999959	-.999646	-.992339	-.999317	.999999	-.900000
	144°	-1.000000	-.999333	-.999379	-1.000000	1.000000	-.900001
	150°	-1.000000	-.968760	-.914463	-.999989	.999996	-.900001
	156°	-1.000000	-.940005	-.896926	-.856991	.999325	-.900001
	162°	-1.000000	-.964596	-.969218	-.996418	.993143	-.900000
	168°	-1.000000	-.999952	-.999413	-.995251	1.000000	-.900001
	174°	-1.000000	-.996355	-.908358	-1.000000	.999974	-.900000

Table A.11 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 5	180°	-1.000000	-.999907	-.924417	-1.000000	.999977	-.900001
	186°	-1.000000	-.999849	-.952723	-.999996	.999500	-.900000
	192°	-1.000000	-.988577	-.995488	-.998837	.853448	-.900001
	198°	-1.000000	-.994642	-.827698	-.999925	.781472	-.900001
	204°	-1.000000	-.999834	-.961192	-1.000000	.998705	-.900001
	210°	-1.000000	-.999906	-.996239	-.999551	.988072	-.900000
	216°	-1.000000	-.999997	-.991363	-1.000000	.999989	-.900001
	222°	-1.000000	-.998999	-.996168	-.999998	.999999	-.900000
	228°	-1.000000	-.908509	-.999995	-1.000000	.999696	-.900000
	234°	-1.000000	-.996002	-.911356	-.999999	.992078	-.900000
	240°	-1.000000	-.999989	-.991694	-1.000000	1.000000	-.900000
	246°	-.991515	-.998163	-.910491	-.887420	.805982	-.900000
	252°	-.999998	-.999786	-.740454	-.999905	.773038	-.900000
	258°	-.999955	-.965468	-.956674	-.999994	.859704	-.900000
	264°	-1.000000	-.877928	-.775260	-.784608	.675004	-.900000
	*270°	-.999377	-.866964	-.912874	-.999921	-.999622	-.900000
	276°	-.804483	-.968403	-.993416	-1.000000	.985495	-.900000
	282°	-.999979	-.832262	-.973437	-1.000000	.999556	-.900001
	288°	-1.000000	-.860809	-.995806	-1.000000	.998815	-.900000
	294°	-1.000000	-.988742	-.998739	-.999994	1.000000	-.900000
	300°	-1.000000	-.991481	-.997133	-1.000000	1.000000	-.900000
	306°	-.916596	-.972110	-.971897	-1.000000	.793461	-.900001
	312°	-.814236	-.947356	-.994597	-1.000000	.999965	-.900000
	318°	-1.000000	-.999959	-.999260	-1.000000	1.000000	-.900001
	324°	-.999888	-.999751	-.984683	-1.000000	1.000000	-.900001
	330°	-.986201	-.909899	-.858280	-.973113	.812708	-.900000
	336°	-.733883	-.906586	-.795379	-.997677	.675518	-.900000
	342°	-.781164	-.980748	-.902735	-1.000000	.995228	-.900000
	348°	-.995085	-.998238	-.957702	-.999999	.999948	-.899999
	354°	-.754726	-.978310	-.859277	-.999984	.777017	-.900000

Table A.12 Examing the neural network which was trained using the features extracted from the projection vector in X and Y directions with 10° angular interval: $(X+Y)/2$.

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 1	0°	.851236	-.999990	-.848276	-.890447	-.871327	-.881423
	10°	.874751	-1.000000	-1.000000	-.985356	-.895899	-.897816
	20°	.993440	-1.000000	-1.000000	-.999961	-.966813	-.912325
	30°	.832154	-1.000000	-.999810	-.883930	-.864402	-.892484
	40°	.992971	-1.000000	-1.000000	-.999285	-.913624	-.888515
	50°	.999716	-1.000000	-.995519	-1.000000	-.999572	-.912008
	60°	.954175	-1.000000	-.938428	-.994489	-.938783	-.904136
	70°	.898325	-1.000000	-.999968	-.942204	-.886077	-.901493
	80°	.986234	-1.000000	-.999852	-.998406	-.889560	-.906553
	90°	.851236	-.999990	-.848276	-.890447	-.871327	-.881423
# 2	0°	-.995809	.999985	-.917939	-.997190	-.999749	-.899746
	10°	-.999999	.878880	-.943209	-.999992	-.996871	-.906311
	20°	-.999999	.999210	-.999769	-.910959	-.999497	-.899801
	30°	-1.000000	.884851	-.988422	-.877428	-.990448	-.910709
	40°	-.999998	.896394	-.999857	-.908411	-.871669	-.902632
	50°	-.982355	.874249	-.996344	-1.000000	-.907478	-.901234
	60°	-.950453	.997393	-.998112	-1.000000	-.998046	-.906283
	70°	-1.000000	1.000000	-1.000000	-.991843	-.909378	-.901460
	80°	-.999992	1.000000	-1.000000	-.960483	-.893445	-.896727
	90°	-.995809	.999985	-.917939	-.997190	-.999749	-.899746
# 3	0°	-.999757	-.887122	.887578	-.894892	-.890079	-.902539
	10°	-.838238	-.999969	.996639	-1.000000	-.926181	-.901278
	20°	-.999994	-.932987	.993248	-.900443	-.890779	-.896586
	30°	-.999999	-.999933	1.000000	-.885268	-.916391	-.894007
	40°	-1.000000	-1.000000	.934814	-.915825	-.908095	-.903351
	50°	-.999965	-.877204	.912099	-.999907	-.999596	-.893801
	60°	-.880167	-1.000000	.999175	-1.000000	-.921021	-.905939
	70°	-.927792	-1.000000	.892117	-1.000000	-.866094	-.900925
	80°	-.999757	-.887122	.887578	-.894892	-.890079	-.902539
	90°	-.999757	-.887122	.887578	-.894892	-.890079	-.902539

Table A.12 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node
		#1	#2	#3	#4	#5	Average
# 4	0°	-.927679	-.998697	-.915954	.967545	-.979599	-.904779
	10°	-.883043	-1.000000	-.999891	.896468	-.999201	-.908732
	20°	-.819114	-.999998	-.999897	.884783	-.991280	-.901753
	30°	-.999750	-.999839	-.982389	.999971	-.930444	-.898949
	40°	-.999999	-.999288	-.998093	.999999	-.999981	-.901955
	50°	-1.000000	-.916380	-.897405	1.000000	-.999250	-.891778
	60°	-.909156	-.966182	-.871616	.900029	-.876119	-.907275
	70°	-.883290	-.866725	-.957011	.996945	-.999999	-.900611
	80°	-.886061	-.999552	-.998246	.996082	-.956421	-.898005
	90°	-.927679	-.998697	-.915954	.967545	-.979599	-.904779
# 5	0°	-.978951	-1.000000	-.963943	-1.000000	.999872	-.883825
	10°	-1.000000	-1.000000	-.998948	-.802231	.999953	-.894940
	20°	-.988166	-.834123	-.908368	-1.000000	.999949	-.883241
	30°	-.999997	-.991382	-.999985	-1.000000	.999883	-.896375
	40°	-1.000000	-.751544	-.999174	-1.000000	.928600	-.908057
	50°	-.999977	-.999640	-1.000000	-1.000000	.999225	-.910064
	60°	-.999993	-.999578	-.999992	-1.000000	.997689	-.893837
	70°	-1.000000	-.999983	-.999865	-.987057	.949683	-.904551
	80°	-.999998	-1.000000	-.999997	-.999250	.999324	-.924594
	90°	-.999357	-.995023	-.990751	-.999068	.994380	-.901263
	100°	-.975573	-1.000000	-.797846	-1.000000	.780667	-.916320
	110°	-1.000000	-.999784	-1.000000	-1.000000	1.000000	-.902004
	120°	-.999996	-.999966	-.999999	-1.000000	1.000000	-.895320
	130°	-.999989	-.999961	-.999999	-1.000000	.999999	-.896918
	140°	-1.000000	-.999859	-1.000000	-1.000000	1.000000	-.915335
	150°	-.999941	-1.000000	-.998947	-1.000000	.999996	-.893585
	160°	-.999843	-.895908	-.999793	-1.000000	.999882	-.888990
	170°	-.999953	-.966214	-.999972	-1.000000	1.000000	-.863009
	180°	-.999995	-1.000000	-.999457	-1.000000	1.000000	-.907803

Table A.12 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 5	190°	-1.000000	-.999992	-.999956	-1.000000	1.000000	-.899745
	200°	-.999937	-.995033	-.973409	-.869776	.999996	-.900614
	210°	-.999839	-.828154	-.804971	-.991872	.989013	-.903615
	220°	-1.000000	-.999932	-1.000000	-.864043	1.000000	-.910400
	230°	-.999867	-1.000000	-.999999	-1.000000	1.000000	-.924345
	240°	-.999982	-1.000000	-.825770	-1.000000	1.000000	-.904043
	250°	-.998968	-.999531	-.999996	-1.000000	.999998	-.890978
	260°	-.934200	-1.000000	-.995886	-1.000000	.999983	-.891869
	270°	-.999868	-.853085	-1.000000	-.999999	.999994	-.902658
	280°	-.999980	-.995140	-.965704	-.999993	.991446	-.917344
	290°	-.920286	-1.000000	-.998401	-.999907	.999999	-.903315
	300°	-.999942	-1.000000	-.935080	-.922068	1.000000	-.872700
	310°	-1.000000	-1.000000	-1.000000	-1.000000	1.000000	-.907490
	320°	-1.000000	-1.000000	-.999999	-1.000000	1.000000	-.907527
	330°	-.999998	-1.000000	-1.000000	-1.000000	1.000000	-.918858
	340°	-.863170	-.891895	-1.000000	-1.000000	.899713	-.909915
	350°	-1.000000	-.999999	-1.000000	-1.000000	.999999	-.925250

Table A.13 Examing the neural network which was trained using the features extracted from the projection vector in X and Y directions with 6° angular interval: $(X+Y)/2$.

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 1	0°	.892749	-.975803	-.939212	-.907089	-1.000000	-.899972
	6°	.819098	-.990422	-.999864	-.872222	-.788430	-.900010
	12°	.908701	-.898972	-.983198	-.893879	-.999996	-.899997
	18°	.712566	-.954794	-.999823	-.980882	-.998137	-.900033
	24°	.848431	-.999902	-.995776	-.983939	-.996020	-.900007
	30°	.917534	-.999982	-.999999	-.994297	-.814714	-.900031
	36°	.884595	-.878200	-.995012	-.912034	-.999847	-.899998
	42°	.874497	-.917508	-.999888	-.999834	-.948151	-.899975
	48°	.860006	-.999953	-.872187	-.989052	-.987491	-.900031
	54°	.995826	-.999449	-.895715	-.992117	-.997162	-.899983
	60°	.867401	-.976160	-.932673	-.940145	-.999996	-.900016
	66°	.917968	-.991177	-.879402	-.881940	-.999456	-.899990
	72°	.994943	-.885678	-.999749	-.999633	-.814740	-.899996
	78°	.996193	-.999400	-.879977	-.946711	-.999978	-.899980
	84°	.921355	-.999155	-.921354	-.952360	-1.000000	-.900032
# 2	0°	-.972053	.996331	-.898350	-.998820	-.999981	-.900022
	6°	-.984942	.908840	-.999507	-.885298	-.896418	-.900003
	12°	-.980403	.883551	-.991708	-.878884	-.884155	-.899979
	18°	-.893253	.930347	-.923568	-.998865	-.837205	-.899989
	24°	-.897772	.867197	-.996803	-.992085	-.966590	-.900024
	30°	-.876737	.893659	-.974621	-.994249	-.999990	-.900016
	36°	-.892891	.827278	-.865700	-.999625	-.999103	-.899992
	42°	-.992412	.847915	-.829699	-.999977	-.998528	-.900029
	48°	-.999010	.997671	-.868054	-.870302	-.885716	-.899948
	54°	-.987984	.837679	-.998998	-.999977	-.817235	-.899997
	60°	-.881430	.886066	-.998729	-.875504	-.914623	-.900016
	66°	-.958188	.901694	-.995549	-.999998	-.997581	-.900021
	72°	-.949245	.985548	-.999972	-.882033	-.916095	-.900011
	78°	-.921604	.997507	-.990242	-.999601	-.987230	-.899977
	84°	-.906406	.941962	-.994983	-.998875	-.997846	-.900005

Table A.13 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 3	0°	-.959816	-.964363	.916578	-.949686	-.949496	-.899997
	6°	-.935076	-.979392	.884852	-.984005	-.912600	-.900005
	12°	-.955417	-.996740	.976544	-.913193	-.869316	-.899995
	18°	-.964454	-.996720	.988182	-.952758	-.998759	-.899998
	24°	-.872391	-.887447	.962413	-.867908	-.999840	-.899966
	30°	-.992918	-.907789	.989039	-.883828	-.999903	-.900002
	36°	-.883313	-.782335	.974200	-.997999	-.999857	-.899956
	42°	-.999706	-.914435	.881322	-.999759	-.998918	-.900100
	48°	-.998238	-.976347	.953666	-.974720	-.891615	-.899970
	54°	-.999440	-.998708	.844997	-1.000000	-.837810	-.900010
	60°	-.960077	-.998586	.965779	-.999990	-.999939	-.900023
	66°	-.838640	-.911939	.964034	-.905936	-.945373	-.899996
	72°	-.893835	-.858284	.992093	-.884205	-.999249	-.899955
	78°	-.935076	-.979392	.884852	-.984005	-.912600	-.900005
	84°	-.994951	-.986423	.838107	-.872515	-.817716	-.899998
# 4	0°	-.871527	-.984193	-.852575	.995326	-.999998	-.899993
	6°	-.914148	-.911439	-.902136	.999523	-.999983	-.900002
	12°	-.975351	-.926152	-.874457	.981020	-1.000000	-.899976
	18°	-.860749	-.968405	-.998504	.968099	-.999998	-.900024
	24°	-.921658	-.997365	-.995615	.897387	-.999651	-.899986
	30°	-.967086	-.888620	-.880690	.891112	-.999026	-.899982
	36°	-.859422	-.820090	-.998325	.852914	-1.000000	-.900016
	42°	-.919688	-.990118	-.999998	.900243	-.999718	-.900022
	48°	-.903160	-.991865	-.999994	.974685	-.999905	-.900002
	54°	-.891568	-.974266	-.997747	.996778	-.999892	-.899985
	60°	-.979639	-.994398	-.904596	.994650	-1.000000	-.900016
	66°	-.916061	-.907931	-.967807	.993348	-1.000000	-.899991
	72°	-.890976	-.924217	-.999970	.852482	-.874347	-.900015
	78°	-.999787	-.863046	-.999798	.956095	-.962072	-.900015
	84°	-.906233	-.999970	-.997615	.883914	-.904212	-.900002

Table A.13 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 5	0°	-.953076	-1.000000	-.999997	-.994167	.999884	-.900003
	6°	-.991159	-1.000000	-.999752	-.955728	.999443	-.900025
	12°	-.996826	-.999317	-.998249	-.938049	.999773	-.899981
	18°	-.818549	-.737266	-.958076	-.999915	.784665	-.899991
	24°	-.972545	-.995851	-.911026	-1.000000	.999997	-.900007
	30°	-.960042	-.756035	-.973308	-1.000000	1.000000	-.899986
	36°	-.999461	-.998061	-.913420	-1.000000	.999999	-.900013
	42°	-.999878	-.902796	-.989595	-.999829	.804795	-.900044
	48°	-.999960	-.890044	-.999366	-.999818	.999998	-.900048
	54°	-.998290	-.986844	-.995617	-1.000000	1.000000	-.899975
	60°	-.944569	-.999994	-.960611	-1.000000	.774470	-.900045
	66°	-.999968	-1.000000	-.999036	-.999706	.968067	-.900064
	72°	-.999957	-1.000000	-.999994	-.908355	.999994	-.900016
	78°	-.950361	-.999978	-.903058	-.814853	.731126	-.900021
	84°	-.999769	-.995850	-.908763	-.959574	.889110	-.900025
	90°	-.997637	-.956203	-.713304	-.986235	.712246	-.899985
	96°	-.999411	-.994888	-.985574	-.956798	.999308	-.900000
	102°	-.987570	-.910497	-.912860	-.998834	.861588	-.899969
	108°	-.999996	-.998867	-.992396	-.999837	1.000000	-.900028
	114°	-.996662	-.998964	-.796977	-.999986	.818257	-.900029
	120°	-.999580	-.999997	-.997584	-.999999	1.000000	-.900072
	126°	-.990036	-.997975	-.903577	-1.000000	.999720	-.900020
	132°	-.999527	-.776665	-.867535	-.999991	.944213	-.900003
	138°	-.985194	-.986948	-.999994	-.994756	.999675	-.899985
	144°	-.999367	-.863467	-.999973	-.957589	.993254	-.900015
	150°	-.793948	-.912006	-.987557	-.999998	.800697	-.900012
	156°	-.984224	-.704436	-.997684	-.984531	.600504	-.899992
	162°	-.907224	-.883937	-.978422	-.990331	.982491	-.899978
	168°	-.999994	-.971026	-.822005	-.806765	.964997	-.899917
	174°	-.999866	-.999973	-.999579	-.877989	.999999	-.900002

Table A.13 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 5	180°	-.999803	-.999845	-.880786	-.869699	.826403	-.899974
	186°	-.998339	-.978390	-.996103	-.997067	.999956	-.899981
	192°	-.822264	-.794447	-.999924	-.950124	.999990	-.899934
	198°	-.883891	-.814115	-.974972	-.999366	.828552	-.899994
	204°	-.996921	-.994544	-.766718	-.999979	.930169	-.900048
	210°	-.982876	-.986910	-.848433	-.999937	.993358	-.900024
	216°	-.984844	-.960194	-.992429	-.884697	.797532	-.900043
	222°	-.850128	-.906515	-.977166	-.999994	.997854	-.899958
	228°	-.999997	-.999557	-1.000000	-1.000000	.999983	-.900106
	234°	-1.000000	-.999967	-1.000000	-1.000000	.999984	-.900109
	240°	-1.000000	-1.000000	-1.000000	-.999908	.999176	-.900130
	246°	-.999848	-.998908	-.999597	-.999949	.999159	-.900032
	252°	-.979280	-.967177	-.995310	-.985086	.923286	-.899980
	258°	-.957827	-.999859	-.843027	-.805701	.965380	-.900006
	264°	-.993299	-.998965	-.999884	-.985445	.999815	-.900014
	270°	-.999999	-.977094	-1.000000	-.999242	.990708	-.900111
	276°	-.978097	-.992104	-.991456	-.967311	.898147	-.900009
	282°	-.823965	-.820329	-.842875	-.998792	.807828	-.900011
	288°	-.981022	-.987936	-.992840	-.829661	.940259	-.900024
	294°	-.801006	-.814389	-.924361	-1.000000	.790855	-.900008
	300°	-.882692	-.845354	-.914884	-1.000000	.802924	-.899980
	306°	-.996988	-.970310	-.993336	-.999898	.994925	-.900004
	312°	-.996262	-.848470	-.997114	-.999990	.999111	-.899958
	318°	-.999979	-.999995	-.994187	-.983034	.999695	-.900000
	324°	-.999942	-.999999	-.831783	-.998189	.983515	-.900012
	330°	-.990043	-.999993	-.993220	-.999953	.857063	-.900059
	336°	-.966054	-.999310	-.995427	-.999789	.974917	-.899979
	342°	-.998426	-.999136	-.983373	-.955222	.999116	-.899933
	348°	-.895273	-.999991	-.999999	-.974133	.999875	-.899990
	354°	-.827856	-.999997	-.999998	-.998594	.999860	-.899960

Table A.14 Examing the neural network which was trained using the features extracted from the projection vector in X and Y directions with 4° angular interval: $(X+Y)/2$.

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 1	0°	.686221	-.998458	-.808917	-.928907	-1.000000	-.900000
	4°	.914357	-.999390	-.932912	-.802372	-.995980	-.900000
	8°	.951099	-.999809	-.992847	-.977987	-.985562	-.900000
	12°	.905337	-.999414	-.877757	-.987306	-.999713	-.900000
	16°	.998861	-.999961	-.937404	-.999293	-.937341	-.900000
	20°	.771929	-.999946	-.948429	-.991621	-.943913	-.900000
	24°	.911634	-.998675	-.857343	-.995959	-1.000000	-.900000
	28°	.646995	-.999996	-.992941	-.999118	-.766075	-.900000
	32°	.991010	-.984619	-.978413	-.924516	-.999906	-.899999
	36°	.964552	-.999713	-.999614	-.999998	-.850974	-.900000
	40°	.985298	-.996594	-.957449	-.989646	-.999959	-.900000
	44°	.990782	-.989567	-.999413	-.999461	-.998314	-.900000
	48°	.992873	-.992679	-.987991	-.997632	-.999986	-.900000
	52°	.975542	-.955581	-.986908	-.794166	-1.000000	-.900000
	56°	.803370	-.874276	-.970696	-.879118	-.999956	-.900000
	60°	.796136	-.985827	-.931979	-.812264	-1.000000	-.900000
	64°	.967303	-.999701	-.925759	-.999362	-.999451	-.900000
	68°	.981095	-.999690	-.950545	-.987007	-.999954	-.900000
	72°	.938105	-.997449	-.936753	-.999442	-.999998	-.900000
	76°	.939515	-.999908	-.926607	-.858285	-1.000000	-.900000
	80°	.986706	-.998235	-.999329	-.999449	-.991871	-.900000
	84°	.997270	-.999937	-.996133	-.999997	-.780189	-.900000
	88°	.999762	-.999946	-.991642	-.999990	-.917301	-.900000
	92°	.928858	-.998989	-.936996	-.972775	-.999993	-.900000
	96°	.975798	-.999796	-.982521	-.992280	-.841295	-.900000
	100°	.811664	-.975636	-.981403	-.841149	-1.000000	-.900000
	104°	.981095	-.999690	-.950545	-.987007	-.999954	-.900000
	108°	.987487	-.998779	-.927770	-.999990	-.887879	-.900000
	112°	.776151	-.999993	-.878504	-.847367	-.999183	-.900000
	116°	.983780	-.997494	-.966404	-.998830	-1.000000	-.900000
	120°	.957724	-.999118	-.988472	-.914745	-.988517	-.899999
	124°	.951272	-.997264	-.968982	-.990003	-.999004	-.899999
	128°	.918669	-.988745	-.993889	-.983381	-.774239	-.900000
	132°	.997389	-.998823	-.926816	-.999755	-.999889	-.900000

Table A.14 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
#1	136°	.801921	-.873862	-.997910	-.798281	-1.000000	-.900000
	140°	.994282	-.876922	-.956349	-.965755	-1.000000	-.900000
	144°	.803370	-.874276	-.970696	-.879118	-.999956	-.900000
	148°	.918954	-.998804	-.846669	-.949755	-.999998	-.900000
	152°	.933364	-.999779	-.869990	-.832217	-.999965	-.900000
	156°	.981095	-.999690	-.950545	-.987007	-.999954	-.900000
	160°	.973553	-.999167	-.992470	-.903555	-.999993	-.900000
	164°	.984492	-.999982	-.803537	-.983643	-1.000000	-.900000
	168°	.978546	-.999987	-.912627	-.987197	-.999981	-.900000
	172°	.811414	-.999151	-.986929	-.952538	-.997555	-.900000
	176°	.999313	-.999977	-.988125	-.999997	-.889841	-.900000
# 2	0°	-.997356	.981651	-.998491	-.996812	-.998437	-.900000
	4°	-.908189	.812279	-.999852	-.949429	-.986628	-.900000
	8°	-.997407	.784567	-.993683	-.998821	-.883797	-.900000
	12°	-.999945	.990952	-.999172	-.988341	-.801713	-.900000
	16°	-.931600	.999359	-.862343	-.999999	-.999997	-.900000
	20°	-.989597	.989785	-.989250	-.999998	-.999487	-.900000
	24°	-.987593	.999466	-.993093	-1.000000	-.999997	-.900000
	28°	-.968981	.866011	-.999272	-.999994	-.963834	-.900000
	32°	-.953248	.993542	-.986679	-.999461	-.873362	-.900000
	36°	-.728093	.979585	-.817696	-.999994	-1.000000	-.900000
	40°	-.999188	.991137	-.975934	-.837376	-1.000000	-.900000
	44°	-.903955	.896978	-.894922	-.993780	-1.000000	-.900001
	48°	-.986679	.862163	-.966593	-.999500	-.997433	-.900000
	52°	-.999672	.992416	-.992754	-.914675	-1.000000	-.900000
	56°	-.999257	.829558	-.999887	-.983402	-.993207	-.900000
	60°	-.999697	.813876	-.999970	-.951232	-.965108	-.900000
	64°	-.999508	.992933	-.999995	-.999831	-.796568	-.900000
	68°	-.999796	.982527	-.999986	-.999874	-.999207	-.900000
	72°	-.999379	.992205	-.999969	-.999830	-.999830	-.900000
	76°	-.999583	.963565	-.999985	-.999317	-.999132	-.900000
	80°	-.986461	.791116	-.999923	-.986282	-.998805	-.900000
	84°	-.962787	.993184	-.999997	-.830771	-.999439	-.900000

Table A.14 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 2	88°	-.964819	.996615	-.825033	-.821653	-.996728	-.900000
	92°	-.992802	.990443	-.999723	-.997877	-.997283	-.900000
	96°	-.997078	.817989	-.915205	-.994845	-.838653	-.900000
	100°	-.971630	.797050	-.756709	-.999337	-.994652	-.900000
	104°	-.999387	.825298	-.807362	-.999642	-.975507	-.900000
	108°	-.991646	.999646	-.977790	-.999999	-.999993	-.900000
	112°	-.955870	.997681	-.920137	-1.000000	-.999997	-.900001
	116°	-.918271	.995830	-.996747	-.999998	-.999880	-.900000
	120°	-.987983	.937573	-.997842	-.999773	-.996875	-.900000
	124°	-.916350	.791377	-.779447	-.999538	-.999944	-.900000
	128°	-.851254	.970037	-.893282	-.998703	-1.000000	-.900001
	132°	-.984327	.862283	-.974040	-.946263	-1.000000	-.900001
	136°	-.996537	.960025	-.965161	-.998185	-.999986	-.900000
	140°	-.997711	.733823	-.996747	-.993954	-.870074	-.900000
	144°	-.999910	.966288	-.999612	-.994575	-.999979	-.900000
	148°	-.999635	.866920	-.999982	-.982103	-.722205	-.900000
	152°	-.978631	.868007	-.999995	-.999893	-.720144	-.900000
	156°	-.999796	.995340	-.999996	-.999336	-.998177	-.900000
	160°	-.999804	.893200	-.999999	-.999163	-.773223	-.900000
	164°	-.999229	.909962	-.999998	-.999249	-.874571	-.900000
	168°	-.999789	.867122	-.999811	-.999501	-.999559	-.900000
	172°	-.967441	.997295	-.999933	-.997766	-1.000000	-.900000
	*176°	-.994780	-.154221	-.998834	-.364469	.065985	-.900000
# 3	0°	-.993711	-.966545	.954937	-.984891	-.964367	-.900000
	4°	-.949169	-.894571	.985177	-.858723	-.999878	-.900000
	8°	-.995432	-.998864	.980728	-.999117	-.997815	-.900000
	12°	-.999526	-.953580	.978998	-.999553	-1.000000	-.900000
	16°	-.999175	-.986229	.974141	-.926220	-.999940	-.900000
	20°	-.920629	-.888982	.927316	-.999998	-.855462	-.900000
	24°	-.873386	-.869876	.918791	-.911480	-1.000000	-.900000
	28°	-.996202	-.957621	.999982	-.872098	-1.000000	-.900000
	32°	-.894986	-.971292	.851265	-.995139	-.999969	-.900000
	36°	-.998907	-.998450	.894208	-.957590	-.963393	-.900000
	40°	-.942946	-.957238	.927419	-.999526	-.999917	-.900000
	44°	-.995970	-.883966	.987860	-.854174	-.999999	-.900000

Table A.14 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 3	48°	-.963857	-.957616	.933696	-.969679	-.999665	-.900000
	52°	-.982931	-.938921	.994388	-1.000000	-.999919	-.900000
	56°	-.987835	-.987496	.817162	-1.000000	-.777029	-.900000
	60°	-.903831	-.987434	.806937	-1.000000	-.991813	-.900000
	64°	-.997360	-.844043	.968387	-.999952	-.999999	-.900000
	68°	-.996506	-.996191	.997680	-.994966	-.999933	-.900000
	72°	-.939654	-.999975	.962694	-.987856	-.998849	-.900000
	76°	-.862264	-.999003	.840023	-.997725	-.995236	-.900000
	80°	-.999808	-.998641	.823639	-.999396	-.829039	-.900000
	84°	-.877963	-.882189	.951321	-.999704	-.999840	-.900000
	88°	-.968231	-.919174	.997716	-.999306	-.999996	-.900000
	92°	-.974600	-.823618	.976822	-.843574	-.999995	-.900000
	96°	-.998261	-.998931	.979501	-.994677	-.997535	-.900000
	100°	-.995006	-.994805	.985379	-.999636	-.999997	-.900000
	104°	-.999918	-.998550	.991597	-.997446	-.999994	-.900000
	108°	-.999816	-.994972	.987015	-.987984	-.999801	-.900000
	112°	-.961766	-.787514	.745706	-.999340	-.999996	-.900000
	116°	-.967549	-.857575	.997377	-.822259	-1.000000	-.900000
	120°	-.990898	-.859410	.996976	-.999450	-.999652	-.900000
	124°	-.953101	-.996284	.883456	-.990623	-.883695	-.900000
	128°	-.987346	-.812129	.844650	-.998909	-.995743	-.900000
	132°	-.981845	-.930379	.860393	-.987504	-.999979	-.900000
	136°	-.999955	-.967476	.983427	-.982338	-.999998	-.900000
	140°	-.999156	-.997630	.932240	-.999329	-.985038	-.900000
	144°	-.997817	-.844110	.855307	-1.000000	-.992067	-.900000
	148°	-.842085	-.990639	.947362	-1.000000	-.989331	-.900000
	152°	-.825113	-.999909	.999570	-.999999	-.999997	-.900000
	156°	-.982462	-.992096	.986742	-.977314	-1.000000	-.900000
	160°	-.999564	-.999855	.891000	-.822704	-.992812	-.900000
	164°	-.909708	-1.000000	.876102	-.976754	-.804482	-.900000
	168°	-.999542	-.999975	.939408	-.997918	-.943027	-.900000
	172°	-.924713	-.997202	.883745	-.963108	-.999473	-.900000
	176°	-.978304	-.946952	.822331	-.998225	-.978707	-.900000

Table A.14 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 4	0°	-.965005	-.866493	-.997550	.995571	-.999956	-.900000
	4°	-.970573	-.918058	-.996192	.996737	-.999839	-.900000
	8°	-.980719	-.998918	-.800395	.900476	-.964393	-.900000
	12°	-.999068	-.961385	-.996241	.997125	-.999991	-.900000
	16°	-.996201	-.999699	-.999440	.998776	-.999888	-.900000
	20°	-.972516	-.836301	-.853276	.979448	-1.000000	-.900000
	24°	-.812024	-.966634	-.913598	.977643	-1.000000	-.900000
	28°	-.919356	-.995196	-.950961	.994870	-.999992	-.899999
	32°	-.836837	-.874210	-.996918	.796461	-.999967	-.900000
	36°	-.904161	-.873914	-.996956	.972052	-.999999	-.900000
	40°	-.999912	-.993838	-.998752	.999165	-.898417	-.900000
	44°	-.975698	-.881384	-.999984	.835316	-.950660	-.900000
	48°	-.974769	-.896679	-.999616	.960656	-.998024	-.900000
	52°	-.982311	-.994472	-.954181	.993883	-.999944	-.900000
	56°	-.907784	-.829526	-.953103	.765817	-1.000000	-.900000
	60°	-.967978	-.999312	-.806034	.945452	-.999999	-.900000
	64°	-.999093	-.994584	-.968038	.957650	-.999982	-.900000
	68°	-.849107	-.959104	-.979578	.791881	-1.000000	-.900000
	72°	-.845607	-.996751	-.995861	.996003	-1.000000	-.900000
	76°	-.978675	-.986389	-.994513	.996927	-.999966	-.900000
	80°	-.998514	-.968388	-.999529	.998120	-.994032	-.900000
	84°	-.760473	-.999116	-.945535	.774425	-.999966	-.900000
	88°	-.887559	-.999484	-.999015	.891855	-.999910	-.900000
	92°	-.996949	-.989252	-.999929	.999700	-.950408	-.899999
	96°	-.828391	-.995324	-.997641	.763427	-.838505	-.900000
	100°	-.944452	-.976511	-.999131	.955739	-.926384	-.900000
	104°	-.997367	-.805530	-.999335	.997822	-.999997	-.900000
	108°	-.925217	-.999715	-.963322	.996157	-.999925	-.900000
	112°	-.809822	-.989329	-.956204	.847339	-.999999	-.900000
	116°	-.875641	-.999988	-.863852	.969752	-.999853	-.900000
	120°	-.765179	-.890192	-.997246	.975032	-.999780	-.899999
	124°	-.767643	-.992265	-.999640	.982134	-.999992	-.900000
	128°	-.999963	-.999647	-.991364	.993921	-.856984	-.900000
	132°	-.902223	-.940762	-.999280	.735583	-.819877	-.900000
	136°	-.924836	-.870659	-.999442	.970625	-.985726	-.900000

Table A.14 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 4	140°	-.989429	-.994887	-.964916	.996443	-.999994	-.900000
	144°	-.907506	-.993402	-.955577	.997586	-1.000000	-.900000
	148°	-.996544	-.859246	-.888070	.997421	-1.000000	-.900000
	152°	-.996439	-.995093	-.820970	.996299	-.999590	-.900000
	156°	-.991084	-.912267	-.994343	.826873	-1.000000	-.900000
	160°	-.974434	-.999916	-.998905	.994131	-.999045	-.900000
	164°	-.990956	-.999979	-.952277	.998029	-.999305	-.900000
	168°	-.997941	-.869524	-.999423	.997182	-.999970	-.900000
	172°	-.946329	-.987441	-.941570	.879264	-.996234	-.900000
	176°	-.849122	-.999774	-.954849	.978696	-.999992	-.900000
# 5	0°	-.999894	-.995892	-.994100	-.981343	.872539	-.900000
	4°	-.999983	-.997060	-.935646	-.851766	.748007	-.900000
	8°	-.999560	-.998458	-.982352	-.984761	.995433	-.900000
	12°	-.993294	-.926488	-.996945	-.793150	.998634	-.899999
	16°	-.919858	-.999633	-.890317	-.997702	.999557	-.900000
	20°	-.976704	-.895593	-.797737	-.973704	.702774	-.900000
	24°	-.807049	-.999721	-.991009	-.999996	.999498	-.900000
	28°	-.997253	-.999998	-.938308	-.999999	.999999	-.899999
	32°	-.837522	-.999988	-.974206	-1.000000	.999998	-.900000
	36°	-.921667	-.999982	-.977253	-1.000000	.998931	-.900000
	40°	-.848317	-.997826	-.794162	-1.000000	.740403	-.900000
	44°	-.946536	-.998803	-.953369	-1.000000	.995049	-.900000
	48°	-.927335	-.972722	-.745126	-1.000000	.999885	-.900000
	52°	-.997671	-.999998	-.999060	-1.000000	1.000000	-.900000
	56°	-.929707	-.999991	-.993426	-1.000000	1.000000	-.900000
	60°	-.999441	-.999999	-.994288	-.999518	1.000000	-.900000
	64°	-.998682	-1.000000	-.864579	-.999837	.998241	-.900000
	68°	-.998738	-.999887	-.937392	-.928081	.489725	-.899999
	72°	-.975319	-.999776	-.993119	-.977613	.997107	-.900000
	76°	-.999877	-.999970	-.999761	-.984490	.996204	-.900000
	80°	-.961079	-.999159	-.904589	-.999685	.975740	-.900000
	84°	-.951807	-.998196	-.818152	-.989984	.709280	-.900000
	88°	-.903064	-.998809	-.725300	-.961255	.681058	-.900000

Table A.14 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 5	92°	-.998252	-.791264	-.999950	-.993595	.999956	-.900000
	96°	-.887415	-.984111	-.779594	-.998655	.638881	-.900000
	100°	-.964378	-.718000	-.756438	-.999863	.742058	-.900000
	104°	-.988434	-.744855	-.997376	-.999205	.389254	-.900000
	108°	-.999086	-.950035	-.836092	-.999974	.957845	-.900000
	112°	-.999499	-.999955	-.626033	-1.000000	.735342	-.900000
	116°	-.993253	-.999959	-.916240	-1.000000	1.000000	-.900000
	120°	-.989327	-.999921	-.936453	-1.000000	1.000000	-.900000
	124°	-.993056	-.999307	-.958710	-1.000000	1.000000	-.900000
	128°	-.883073	-.976968	-.757812	-1.000000	.993961	-.900000
	132°	-.879956	-.999965	-.918397	-.999804	.966917	-.899999
	136°	-.999672	-.999874	-.995431	-.920137	.853532	-.900000
	140°	-.999848	-.999999	-.998811	-.818345	.999890	-.900000
	144°	-.999947	-.999938	-.649687	-.971602	.996412	-.899999
	148°	-.980684	-.999846	-.928130	-.780665	.990791	-.899999
	152°	-.983231	-.999142	-.999824	-.998503	.999989	-.900000
	156°	-.965067	-.829097	-.999997	-.938759	.784579	-.900000
	160°	-.959498	-.732960	-1.000000	-.992826	.810995	-.900000
	164°	-.777632	-.978944	-1.000000	-.905129	.999844	-.900000
	168°	-.995489	-.842787	-.999990	-.815191	.994524	-.900001
	172°	-.998782	-.879274	-.999790	-.969329	.999258	-.900000
	176°	-.999702	-.999937	-.983318	-.970773	.980253	-.900000
	180°	-.999908	-.999998	-.998432	-.989644	1.000000	-.900000
	184°	-.996327	-1.000000	-.952812	-.999623	1.000000	-.900000
	188°	-.995189	-1.000000	-.847296	-.999726	1.000000	-.900000
	192°	-.994721	-.999629	-.991399	-.999797	1.000000	-.899999
	196°	-.985495	-.999694	-.758430	-.992901	.999709	-.900000
	200°	-.994362	-.996435	-.995540	-.938393	.904311	-.900000
	204°	-.998961	-.999523	-.987959	-.734279	.864056	-.900000
	208°	-.966402	-.999104	-.991301	-.999752	.987355	-.900000
	212°	-.998143	-.999569	-.971815	-.999599	.999999	-.900000
	216°	-.999355	-.999997	-.996099	-.999998	1.000000	-.900000
	220°	-.999751	-.999968	-.999386	-.999999	1.000000	-.900000
	224°	-.998514	-.999981	-.999539	-1.000000	1.000000	-.900000

Table A.14 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 5	228°	-.999657	-.999580	-.973299	-1.000000	1.000000	-.900001
	232°	-.997661	-.999946	-.987360	-1.000000	1.000000	-.900001
	236°	-.993688	-.998157	-.944689	-1.000000	.999999	-.900001
	240°	-.993688	-.998157	-.944689	-1.000000	.999999	-.900001
	244°	-.909758	-.915863	-.997450	-1.000000	.999995	-.900000
	248°	-.936402	-.997122	-.997742	-1.000000	.999998	-.900000
	252°	-.830171	-.919366	-.823416	-1.000000	.884040	-.900001
	256°	-.953354	-.837714	-.993048	-1.000000	.999999	-.900000
	260°	-.998394	-.998182	-.934683	-1.000000	.999620	-.900001
	264°	-.962925	-.946135	-.999863	-1.000000	.999998	-.900001
	268°	-.875562	-.999480	-.999476	-1.000000	.999999	-.900000
	272°	-.982127	-.998328	-.998552	-1.000000	.999995	-.900000
	276°	-.985714	-.999678	-.999840	-1.000000	.998156	-.900000
	280°	-.859396	-.999869	-.965011	-1.000000	.842449	-.900000
	284°	-.859396	-.999869	-.965011	-1.000000	.842449	-.900000
	288°	-.968350	-.999927	-.999518	-1.000000	.999394	-.900001
	292°	-.757316	-.999991	-.995281	-.999950	.798787	-.900000
	296°	-.941860	-.999982	-.999984	-.999997	1.000000	-.900000
	300°	-.999634	-1.000000	-.999349	-1.000000	1.000000	-.900000
	304°	-.999630	-1.000000	-.999370	-1.000000	1.000000	-.900001
	308°	-.999946	-1.000000	-.999897	-1.000000	1.000000	-.900001
	312°	-.999887	-.999989	-.999984	-1.000000	1.000000	-.900001
	316°	-.998819	-.999999	-.992033	-1.000000	1.000000	-.900000
	320°	-.998626	-1.000000	-.997819	-1.000000	1.000000	-.900001
	324°	-.999529	-.999971	-.994176	-1.000000	1.000000	-.900001
	328°	-.999968	-.999977	-.999886	-1.000000	1.000000	-.900001
	332°	-.997900	-.999842	-.989195	-1.000000	.999993	-.900000
	336°	-.973205	-.832839	-.999919	-.965066	.868350	-.900000
	340°	-.999514	-.793410	-.999277	-.864757	.775230	-.900000
	344°	-.960347	-.998692	-.986867	-.999802	.999996	-.900000
	348°	-.918233	-.977977	-.941763	-.949894	.999863	-.900000
	352°	-.807823	-.883582	-.995046	-.830870	.960662	-.900000
	356°	-.999076	-.998913	-.929447	-.818478	.999833	-.900000

Table A.15 Examing the neural network which was trained using the features extracted from the projection vector in X and Y directions with 10° angular interval: (X, Y).

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 1	0°	-.043078	-1.999424	-1.909837	-1.832049	-1.747054	-1.800364
	10°	1.705124	-1.871759	-1.889378	-1.712255	-1.963542	-1.801273
	20°	-.099392	-1.943952	-1.880409	-1.833044	-1.876617	-1.799566
	30°	1.779176	-1.898422	-1.897103	-1.631109	-1.824667	-1.801123
	40°	1.374390	-1.924180	-1.992610	-1.931957	-1.747985	-1.806522
	50°	1.980372	-1.892911	-1.939307	-1.808107	-1.995559	-1.807242
	60°	1.246856	-1.897536	-1.876758	-1.568106	-1.851622	-1.806809
	70°	1.887936	-1.992877	-1.878582	-1.994908	-1.936648	-1.796071
	80°	1.862170	-1.888024	-1.861594	-1.775838	-1.899914	-1.797768
	90°	-.043078	-1.999424	-1.909837	-1.832049	-1.747054	-1.800364
# 2	0°	-1.980128	1.856337	-1.940809	-1.889768	-1.871813	-1.803496
	10°	-1.902658	1.799501	-1.929023	-1.935438	-1.969144	-1.789779
	20°	-1.999657	1.873788	-1.910788	-1.884173	-1.929164	-1.803664
	30°	-1.939706	1.829634	-1.892588	-1.895372	-1.963158	-1.801555
	40°	-1.876449	1.994495	-1.989347	-1.999988	-1.977653	-1.799411
	50°	-1.921385	1.888204	-1.970877	-1.998125	-1.888057	-1.810283
	60°	-1.898287	1.997380	-1.882783	-1.999965	-1.856787	-1.794194
	70°	-1.898339	1.855784	-1.891781	-1.907837	-1.702320	-1.796743
	80°	-1.877568	1.912030	-1.889726	-1.926687	-1.746005	-1.791546
	90°	-1.980128	1.856337	-1.940809	-1.889768	-1.871813	-1.803496
# 3	0°	-1.910539	-1.992085	1.977150	-1.964127	-1.873188	-1.804505
	10°	-1.900383	-1.866146	1.888488	-1.996868	-1.985388	-1.802524
	20°	-1.880691	-1.999465	1.894545	-1.892140	-1.713025	-1.797446
	30°	-1.896871	-1.781362	1.878700	-1.997910	-1.795664	-1.801236
	40°	-1.993908	-1.922273	1.818886	-2.000000	-1.913930	-1.799978
	50°	-1.917962	-1.961584	1.914162	-1.998467	-1.766515	-1.795605
	60°	-1.914239	-1.905178	1.823874	-1.508324	-1.921199	-1.800817
	70°	-1.917888	-1.801736	1.980067	-1.999996	-1.999994	-1.806409
	80°	-1.889108	-1.909491	1.896842	-1.854571	-1.905482	-1.799673
	90°	-1.910539	-1.992085	1.977150	-1.964127	-1.873188	-1.804505

Table A.15 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 4	0°	-1.930401	-1.994263	-1.974249	-.020247	-1.770589	-1.800424
	10°	-1.899024	-1.695137	-1.908257	1.919677	-1.897172	-1.801481
	20°	-1.887315	-1.884511	-1.913241	1.966048	-1.770764	-1.796645
	30°	-1.858433	-1.980308	-1.955586	1.841495	-1.827583	-1.804063
	40°	-.501262	-1.851473	-1.980872	.054532	-1.820711	-1.798046
	50°	-1.819742	-1.918391	-1.904220	1.998371	-1.999933	-1.800058
	60°	-1.124479	-1.953714	-1.928599	1.813019	-1.995004	-1.806097
	70°	-1.897040	-1.890485	-1.955427	1.844030	-1.939352	-1.802973
	80°	-1.902908	-1.999531	-1.884237	1.879175	-1.814482	-1.795173
	90°	-1.930401	-1.994263	-1.974249	-.020247	-1.770589	-1.800424
# 5	0°	.131501	-1.814299	-1.988530	-1.909211	1.915721	-1.806095
	10°	-1.381353	-1.976555	-1.977773	-1.978144	1.889571	-1.799659
	20°	-1.999909	-1.997017	-1.999923	-1.982889	1.967635	-1.858973
	30°	-.067517	-1.777008	-1.998639	-1.996745	-.001285	-1.783093
	40°	-1.949851	-1.983199	-1.977921	-1.995187	1.993352	-1.801015
	50°	-1.995062	-1.992499	-1.996934	-1.841031	1.570120	-1.803080
	60°	-1.991721	-1.999997	-1.999999	-1.952970	1.810525	-1.898969
	70°	-1.999807	-1.998493	-1.997628	-1.344327	-.020531	-1.799907
	80°	-1.958194	-1.999929	-1.822082	-1.998768	1.858944	-1.803293
	90°	-1.954330	-1.986768	-1.990037	-1.999703	1.854447	-1.804898
	100°	-1.925931	-1.997598	-1.841340	-1.950475	-.000137	-1.798075
	110°	-1.999333	-1.812097	-1.999986	-2.000000	1.967409	-1.800681
	120°	-1.999851	-1.995244	-1.999665	-1.999994	1.870896	-1.829375
	130°	-1.975782	-1.998006	-1.998378	-1.999998	1.728418	-1.800211
	140°	-1.870243	-1.979129	-1.999529	-1.999844	1.981785	-1.797200
	150°	-1.862800	-1.906839	-1.884773	-1.965478	.000010	-1.818307
	160°	-1.950639	-1.953945	-1.964105	-1.956905	1.884425	-1.801155
	170°	-1.907440	-1.884628	-1.985619	-1.522756	1.926989	-1.799214
	180°	-1.838539	-1.818121	-1.957816	-1.630493	1.883574	-1.794516

Table A.15 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 5	190°	-1.987044	-1.970890	-1.965610	-1.836260	1.917763	-1.805545
	200°	-1.999328	-1.984463	-1.849231	-1.866259	1.999764	-1.800673
	210°	-1.995802	-1.733373	-1.866914	-1.998831	1.998923	-1.790075
	220°	-1.993880	-1.760852	-1.993209	-1.901226	1.813298	-1.815291
	230°	-1.771958	-1.999920	-1.956748	-1.999820	1.992907	-1.788423
	240°	-1.996559	-1.995740	-1.948577	-2.000000	1.994256	-1.814982
	250°	-1.896086	-1.972916	-1.924883	-1.999933	1.849346	-1.810112
	260°	-1.923306	-1.969088	-1.634677	-1.999814	1.890525	-1.806953
	270°	-1.920869	-1.864860	-1.824518	-1.999780	-.003630	-1.809967
	280°	-1.903401	-1.744621	-1.828073	-1.999487	-.005801	-1.814910
	290°	-1.999187	-1.938479	-1.995015	-1.998005	-.015246	-1.811691
	300°	-1.866318	-1.993884	-1.994794	-1.999999	-.003718	-1.775113
	310°	-1.967394	-1.989643	-1.867835	-1.999995	1.964957	-1.806270
	320°	-1.917153	-1.999905	-1.967602	-1.999999	-.005746	-1.797353
	330°	-1.774629	-1.698291	-1.907766	-1.999977	1.919421	-1.789663
	340°	.004122	-1.963123	-1.897679	-1.911628	1.964449	-1.813094
	350°	-1.987495	-1.978315	-1.999815	-1.863707	-.000145	-1.825223

Table A.16 Examing the neural network which was trained using the features extracted from the projection vector in X and Y directions with 6° angular interval: (X, Y).

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 1	0°	-.144081	-1.998958	-1.923976	-1.839804	-1.881063	-1.806989
	6°	1.673981	-1.901752	-1.918345	-1.899953	-1.984564	-1.799207
	12°	-.110553	-1.889781	-1.877593	-1.900821	-1.988704	-1.801192
	18°	1.869023	-1.939310	-1.938999	-1.996118	-1.903277	-1.796889
	24°	1.695011	-1.906152	-1.996580	-1.907937	-1.861689	-1.799032
	30°	-.114399	-1.882112	-1.903097	-1.814254	-1.972840	-1.797681
	36°	-.030140	-1.868422	-1.896931	-1.902204	-1.937705	-1.797173
	42°	1.855161	-1.904577	-1.921821	-1.879517	-1.909266	-1.800699
	48°	1.929719	-1.915287	-1.998006	-1.929892	-1.877266	-1.796327
	54°	1.468680	-1.884834	-1.935499	-1.743532	-1.889021	-1.800077
	60°	.611172	-1.914234	-1.781485	-1.894018	-1.895813	-1.799779
	66°	.712376	-1.895384	-1.892164	-1.498502	-1.916285	-1.800533
	72°	1.856525	-1.908857	-1.983346	-1.896245	-1.914493	-1.806038
	78°	1.777081	-1.979377	-1.900887	-1.818285	-1.904005	-1.797180
	84°	1.654707	-1.896411	-1.875920	-1.881670	-1.913875	-1.800813
# 2	0°	-1.877998	1.900424	-1.906087	-1.888654	-1.897030	-1.796570
	6°	-1.920253	1.857120	-1.894682	-1.991386	-1.755265	-1.798275
	12°	-1.994672	1.878312	-1.909044	-1.864845	-1.855264	-1.803924
	18°	-1.884110	1.911069	-1.884431	-1.891227	-1.922882	-1.800950
	24°	-1.891428	1.847925	-1.890630	-1.872920	-1.999302	-1.798420
	30°	-1.896647	1.890836	-1.903091	-1.937809	-1.998984	-1.796246
	36°	-1.988106	1.851971	-1.861255	-1.866020	-1.896449	-1.803439
	42°	-1.917946	1.895320	-1.868843	-1.885459	-1.988362	-1.800326
	48°	-1.922316	1.772302	-1.900013	-1.999893	-1.884193	-1.799672
	54°	-1.938362	1.877217	-1.918264	-1.999407	-1.899456	-1.802204
	60°	-1.869553	1.905812	-1.956144	-1.999980	-1.907204	-1.798500
	66°	-1.890505	1.915151	-1.963317	-1.999960	-1.869611	-1.798951
	72°	-1.880176	1.894499	-1.894751	-1.928176	-1.912130	-1.799431
	78°	-1.949472	1.894282	-1.903835	-1.925164	-1.949419	-1.803233
	84°	-1.997853	1.913941	-1.895328	-1.928869	-1.947203	-1.798358

Table A.16 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 3	0°	-1.896756	-1.940071	1.905440	-1.933367	-1.870460	-1.799449
	6°	-1.978889	-1.898884	1.865798	-1.905303	-1.960643	-1.797681
	12°	-1.999737	-1.900368	1.784526	-1.896273	-1.950111	-1.801564
	18°	-1.991574	-1.897770	1.940115	-1.831815	-1.756354	-1.800100
	24°	-1.905712	-1.900869	1.887418	-1.898109	-1.891912	-1.807121
	30°	-1.899680	-1.897171	1.770131	-1.919461	-1.735397	-1.798968
	36°	-1.996274	-1.763416	1.859309	-1.880561	-1.876916	-1.804228
	42°	-1.887974	-1.896106	1.774410	-1.934254	-1.873541	-1.801647
	48°	-1.881067	-1.884916	1.858655	-1.919644	-1.974482	-1.799342
	54°	-1.886061	-1.892997	1.921942	-1.921147	-1.972780	-1.799489
	60°	-1.874693	-1.897934	1.857931	-1.997445	-1.895354	-1.799282
	66°	-1.966888	-1.908963	1.884151	-1.651672	-1.944755	-1.796492
	72°	-1.895147	-1.905169	1.889607	-1.903836	-1.855991	-1.798494
	78°	-1.923537	-1.886147	1.906772	-1.890842	-1.970185	-1.801519
	84°	-1.967756	-1.904218	1.808687	-1.871182	-1.932019	-1.802756
# 4	0°	-1.923710	-1.999871	-1.858951	1.862082	-1.866618	-1.794184
	6°	-1.878338	-1.930807	-1.901026	1.881508	-1.888983	-1.800139
	12°	-1.878338	-1.998733	-1.900866	1.878684	-1.888983	-1.801385
	18°	-1.875803	-1.886492	-1.927525	1.892135	-1.886447	-1.796825
	24°	-1.843441	-1.905947	-1.960356	1.779284	-1.929641	-1.798216
	30°	-1.915586	-1.884733	-1.909869	-.100147	-1.740642	-1.796957
	36°	-1.887279	-1.770070	-1.895913	1.879992	-1.906772	-1.808361
	42°	-1.853253	-1.795041	-1.900048	1.791306	-1.904970	-1.801185
	48°	-1.777923	-1.895983	-1.900252	1.582687	-1.904970	-1.802159
	54°	-1.907776	-1.906741	-1.902771	1.888896	-1.916643	-1.802656
	60°	-1.216510	-1.883127	-1.812517	-.113385	-1.871138	-1.799779
	66°	-1.080700	-1.921642	-1.893212	.472595	-1.981098	-1.800532
	72°	-1.884829	-1.899863	-1.892373	1.899962	-1.868235	-1.800334
	78°	-1.892410	-1.999792	-1.942260	1.801260	-1.746456	-1.798076
	84°	-1.899123	-1.999824	-1.903010	1.918670	.126259	-1.797859

Table A.16 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 5	0°	-1.902712	-1.904865	-1.799204	-1.171319	-.103593	-1.806683
	6°	-1.980972	-1.949203	-1.909205	-1.976166	-.051180	-1.810720
	12°	-1.991527	-1.886798	-1.857584	-1.932908	-.143977	-1.796710
	18°	-1.996690	-1.906105	-1.906360	-1.908145	1.819950	-1.809163
	24°	-1.880698	-1.995724	-1.998462	-1.961944	1.970265	-1.841752
	30°	-1.891745	-1.998838	-1.999862	-1.898332	1.941641	-1.807762
	36°	-1.873088	-1.997197	-1.994475	-1.991275	1.869605	-1.805623
	42°	-1.826341	-1.857691	-1.999853	-1.999679	1.877848	-1.848701
	48°	-1.866430	-1.794492	-1.859772	-1.999753	-.000003	-1.807707
	*54°	-1.982420	-1.999987	-1.997266	.000721	.000000	-1.794497
	60°	-1.885646	-1.999988	-1.856787	-1.893863	1.841663	-1.846628
	66°	-1.937719	-1.999983	-1.966596	-1.986658	1.990657	-1.842873
	72°	-1.980707	-1.999831	-1.981540	-1.874288	1.995952	-1.797504
	78°	-1.866011	-1.997509	-1.818047	-1.816902	1.864664	-1.792592
	84°	-1.988125	-1.999024	-1.677186	-1.927880	1.996560	-1.802916
	90°	-1.882204	-1.886004	-1.999261	-1.985098	-.020790	-1.800819
	96°	-1.966770	-1.856941	-1.818407	-1.892773	1.687044	-1.799738
	102°	-1.915189	-1.979780	-1.993701	-1.997334	1.984573	-1.803632
	108°	-1.922893	-1.853403	-1.998230	-1.902646	1.850254	-1.800032
	114°	-1.922895	-1.999433	-1.998230	-1.902646	1.904593	-1.842553
	120°	-1.933106	-1.998753	-1.998355	-1.915185	1.774696	-1.814474
	126°	-1.892884	-1.998821	-1.999862	-1.898332	1.941647	-1.793770
	132°	-1.994537	-1.999987	-1.999870	-1.998732	1.999998	-1.793433
	138°	-1.801535	-1.997975	-1.999339	-1.999994	1.964183	-1.796825
	144°	-1.928927	-1.993653	-1.998624	-1.999882	1.999758	-1.795450
	150°	-1.995747	-.000091	-1.988137	-1.999943	1.999305	-1.797294
	156°	-1.812341	-1.999679	-1.922527	-1.961754	1.999963	-1.799787
	162°	-1.863888	-1.977022	-1.901558	-1.988768	1.858934	-1.802051
	168°	-1.998368	-1.999921	-1.887973	-1.884717	1.999874	-1.806066
	174°	-1.998597	-1.873982	-1.992976	.062879	1.809721	-1.823828

Table A.16 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 5	180°	-1.965322	-1.859529	-1.995865	-1.884133	1.880046	-1.815093
	186°	-1.981802	-1.999882	-1.929409	-1.900375	1.954062	-1.813191
	192°	-1.982098	-1.999676	-1.995994	-1.906758	2.000000	-1.806583
	198°	-1.999564	-1.998880	-1.983334	-1.898639	1.999532	-1.836154
	204°	-1.973269	-1.880704	-1.918705	-1.935738	1.996067	-1.839356
	210°	-1.872943	-1.903411	-1.809882	-1.996000	1.850484	-1.849502
	216°	-1.976439	-1.999939	-1.999181	-1.918415	2.000000	-1.798113
	222°	-1.945685	-1.999995	-1.999490	-1.942935	2.000000	-1.792972
	228°	-1.912537	-1.999691	-1.999988	-1.883947	1.884778	-1.791646
	234°	-1.859674	-1.999447	-1.981163	-1.975069	1.994160	-1.791074
	240°	-1.900234	-1.999785	-1.998391	-1.999596	1.998862	-1.797084
	246°	-1.940117	-1.845906	-1.998218	-1.998106	1.927885	-1.796884
	252°	-1.984733	-1.899472	-1.877874	-1.999867	1.916448	-1.795895
	258°	-1.868368	-1.999047	.134767	-1.997711	1.629261	-1.798538
	264°	-1.990096	-1.999438	-1.930292	-1.998081	1.991879	-1.801139
	270°	-1.918118	-1.869950	-1.856421	-1.998785	1.737463	-1.798000
	276°	-1.892701	-1.999898	-1.866132	-1.999496	1.998116	-1.800496
	282°	-1.908808	-1.989018	-1.992102	-1.999852	1.999806	-1.794769
	288°	-1.862578	-1.999982	-1.996236	-1.999731	1.999999	-1.801720
	294°	-1.843668	-1.999860	-1.997462	-1.999782	1.999990	-1.798262
	300°	-1.904245	-1.932166	-1.983487	-1.959096	1.999917	-1.798422
	306°	-1.871315	-1.999952	-1.998478	-1.987473	1.999997	-1.796667
	312°	-1.985766	-1.999997	-1.999073	-1.998608	1.999993	-1.798025
	318°	-1.961909	-1.999999	-1.999860	-1.847586	2.000000	-1.881374
	324°	-1.915190	-1.999997	-1.997988	-1.941146	1.997929	-1.796571
	330°	-1.976486	-1.988607	-1.947866	-1.986290	1.736921	-1.809413
	336°	-1.744787	-1.964865	-1.997922	-1.907168	1.946584	-1.806638
	342°	-1.890525	-1.996771	-1.813838	-1.890713	1.889338	-1.799419
	348°	-1.997847	-1.999944	-1.993579	-1.666070	-.000002	-1.799626
	354°	-1.804764	-1.960389	-1.879274	-1.973906	-.095016	-1.802029

Table A.17 Examing the neural network which used the features extracted from the projection vector in X and Y directions with 4° angular interval: (X, Y).

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 1	0°	-.108325	-1.904208	-1.933812	-1.866593	-1.977612	-1.799999
	4°	-.121984	-1.923412	-1.990500	-1.989067	-1.937609	-1.800000
	8°	-.156617	-1.958574	-1.992691	-1.983618	-1.906989	-1.800000
	12°	.325033	-1.874012	-1.982436	-1.997235	-1.869733	-1.800000
	16°	.077261	-1.853286	-1.952807	-1.896368	-1.988734	-1.800000
	20°	-.142517	-1.877492	-1.987836	-1.880539	-1.768987	-1.800001
	24°	1.179489	-1.975009	-1.819546	.149088	-1.999808	-1.800000
	28°	-.056051	-1.916381	-1.913132	-1.878629	-1.998996	-1.800000
	32°	-.133834	-1.848640	-1.989992	-1.979352	-1.983628	-1.799999
	36°	1.172162	-1.998687	-1.992107	-1.891068	-1.992279	-1.799999
	40°	1.504805	-1.995396	-1.997100	-1.873608	-1.976814	-1.800000
	44°	1.909050	-1.779196	-1.910748	-1.890740	-1.999398	-1.799999
	48°	-.114266	-1.957519	-1.898678	-1.943251	-1.824135	-1.799999
	52°	.866453	-1.988969	-1.920448	-1.953723	-1.910758	-1.800000
	56°	-.004921	-1.874457	-1.986370	-1.965557	-1.999259	-1.800000
	60°	-.055907	-1.999835	-1.967422	-1.953765	-1.947706	-1.800000
	64°	-.088384	-1.982835	-1.937474	-1.990306	-1.909097	-1.800000
	68°	1.384534	-1.998921	-1.822209	-1.837867	-1.990486	-1.800000
	72°	.256119	-1.997975	-1.922156	-1.866241	-1.954030	-1.800000
	76°	.286498	-1.946169	-1.967836	-1.884726	-1.990086	-1.800000
	80°	-.071186	-1.998216	-1.865446	-1.996227	-1.811728	-1.800000
	84°	-.146689	-1.986628	-1.854245	-.503307	-1.994143	-1.800000
	88°	-.108325	-1.904208	-1.933808	-1.996433	-1.973914	-1.799999
	92°	-.156882	-1.844559	-1.995037	-1.992989	-1.824345	-1.800000
	96°	-.156775	-1.959004	-1.992815	-1.981709	-1.906989	-1.800000
	100°	-.040672	-1.980470	-1.994932	-1.986958	-1.923898	-1.800000
	104°	-.051394	-1.940801	-1.861515	-1.985280	-1.936409	-1.800000
	108°	-.013556	-1.947583	-1.932102	-1.923693	-.688396	-1.800000
	112°	-.142517	-1.877492	-1.987836	-1.880538	-1.945447	-1.800001
	116°	-.070180	-1.878083	-1.974689	-1.870037	-1.999346	-1.800000
	120°	-.125618	-1.986975	-1.992468	-1.872783	-1.983762	-1.799999
	124°	1.900231	-1.994244	-1.993168	-1.998455	-1.983850	-1.799999
	128°	1.793924	-1.999315	-1.993303	-1.876663	-1.982410	-1.800000
	132°	-.115293	-1.999207	-1.997043	-1.897331	-1.844059	-1.799999

Table A.17 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 1	136°	1.922814	-1.978685	-1.900064	-1.893435	-1.999822	-1.799999
	140°	1.650906	-1.850583	-1.913484	-1.851920	-1.999704	-1.800000
	144°	-.004908	-1.874458	-1.984926	-1.965557	-1.999812	-1.800000
	148°	-.053786	-1.940320	-1.959519	-1.906274	-1.997402	-1.800000
	152°	-.160537	-1.995113	-1.991706	-1.936071	-1.834427	-1.800000
	156°	1.014536	-1.998643	-1.895306	.031914	-1.966505	-1.800000
	160°	-.021250	-1.999855	-1.959004	-1.963579	-1.910477	-1.800000
	164°	.280676	-1.891898	-1.983325	-1.880204	-1.990802	-1.800000
	168°	-.073435	-1.989433	-1.940950	-1.973054	-1.814346	-1.800000
	172°	1.854522	-1.999292	-1.887562	-1.961274	-1.991878	-1.800000
	176°	-.146846	-1.953659	-1.895468	-1.972393	-1.988471	-1.800000
# 2	0°	-1.962804	1.490433	-1.737720	-1.997968	-1.985728	-1.799999
	4°	-1.986210	1.683423	-1.926081	-1.996811	-1.827019	-1.800000
	8°	-1.922605	1.870405	-1.978581	-1.999400	-1.846688	-1.800000
	12°	-1.956523	1.929747	-1.999270	-1.997472	-1.824904	-1.800000
	16°	-1.985013	1.868680	-1.972942	-1.985553	-1.992726	-1.800000
	20°	-1.996682	1.725682	-1.933825	-1.999653	-1.840156	-1.800000
	24°	-1.958792	1.913215	-1.923914	-1.853915	-1.999774	-1.800000
	28°	-1.662637	-.043106	-1.661227	-1.993924	-1.999881	-1.799999
	32°	-1.968884	1.814332	-1.847644	-1.867530	-1.827368	-1.799999
	36°	-1.962073	1.636259	-1.863061	-1.941705	-1.989898	-1.800000
	40°	-1.852371	1.766224	-1.941370	-1.992270	-1.911731	-1.800000
	44°	-1.812864	-.002712	-1.958323	-1.996300	-1.914553	-1.800000
	48°	-1.931036	1.999303	-1.985036	-1.998154	-1.970771	-1.800000
	52°	-1.789726	1.863362	-1.944423	-1.974905	-1.974695	-1.800000
	56°	-1.909196	1.722838	-1.757404	-1.969471	-1.985236	-1.800000
	60°	-1.879215	1.882815	-1.956757	-1.999888	-1.475337	-1.800000
	64°	-1.860981	1.941485	-1.971977	-1.999940	-1.480910	-1.800000
	68°	-1.884509	1.887127	-1.932896	-1.999975	-1.436926	-1.800000
	72°	-1.843031	1.872131	-1.863450	-1.996479	-1.641719	-1.800000
	76°	-1.918800	1.906982	-1.967180	-1.944273	-1.666340	-1.800000
	80°	-1.993396	1.845025	-1.990527	-1.929309	-1.290363	-1.800000
	84°	-1.884268	1.712394	-1.979213	-1.922564	-1.918800	-1.800000

Table A.17 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 2	88°	-1.933677	1.765430	-1.889308	-1.997969	-1.985728	-1.800000
	92°	-1.986210	1.683423	-1.926081	-1.996811	-1.827019	-1.800000
	96°	-1.985235	1.428563	-1.816672	-1.999059	-1.844568	-1.800000
	100°	-1.922605	1.883919	-1.999050	-1.999416	-1.846688	-1.800000
	104°	-1.888716	1.860710	-1.999521	-1.997980	-1.855605	-1.800000
	108°	-1.817018	1.938296	-1.918280	-1.907182	-1.999999	-1.800000
	112°	-1.994147	1.895213	-1.880242	-1.998184	-1.913276	-1.799999
	116°	-1.993164	1.782074	-1.787521	-1.939864	-1.999511	-1.799999
	120°	-1.975625	1.823738	-1.780181	-1.913617	-1.998366	-1.799999
	124°	-1.983055	1.995907	-1.948427	-1.844243	-1.995034	-1.800000
	128°	-1.947294	1.774825	-1.980793	-1.999683	-1.529290	-1.800000
	132°	-1.988865	1.998692	-1.927561	-1.833965	-1.999517	-1.800000
	136°	-1.895423	1.999891	-1.992730	-1.999504	-1.847500	-1.800000
	140°	-1.927303	1.999991	-1.881087	-1.995346	-1.999093	-1.800000
	144°	-1.879248	1.999284	-1.844127	-1.951830	-1.129495	-1.800000
	148°	-1.975845	1.999961	-1.830825	-1.996626	-1.687783	-1.800000
	152°	-1.879215	1.833620	-1.960894	-1.999888	-1.927948	-1.800000
	156°	-1.921750	1.931940	-1.896887	-1.999975	-1.862719	-1.799999
	160°	-1.865478	1.863475	-1.986524	-1.999978	-1.569854	-1.800000
	164°	-1.976006	1.920125	-1.875199	-1.876041	-1.666339	-1.800000
	168°	-1.985413	1.988572	-1.947420	-1.943721	-1.956527	-1.800000
	172°	-1.998199	1.969527	-1.969687	-1.999456	-1.900338	-1.800000
	176°	-1.884267	-.150726	-1.933309	-1.922564	-1.919083	-1.800000
# 3	0°	-1.959581	-1.990721	1.961060	-1.992552	-1.999007	-1.800000
	4°	-1.959234	-1.945094	1.829574	-1.999244	-1.994679	-1.800000
	8°	-1.999027	-1.947207	1.754004	-1.943254	-1.999430	-1.800000
	12°	-1.999147	-1.881313	1.879013	-1.891894	-1.999945	-1.799999
	16°	-1.968372	-1.984190	1.673553	-1.885128	-1.999806	-1.799999
	20°	-1.961986	-1.825567	-.161054	-1.866827	-1.994536	-1.799999
	24°	-1.996831	-1.975506	1.760533	-1.905118	-1.948800	-1.800000
	28°	-1.993084	-1.945461	1.722574	-1.996393	-1.975054	-1.799999
	32°	-1.915313	-1.999753	1.901699	-1.994915	-1.999262	-1.799999
	36°	-1.991929	-1.863978	1.846026	-1.999541	-1.997362	-1.799999
	40°	-1.996843	-1.996504	1.985414	-1.976040	-1.993973	-1.799999
	44°	-1.999961	-1.994748	1.944024	-1.890229	-1.890973	-1.800000

Table A.17 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 3	48°	-1.999367	-1.909687	1.981610	-1.999812	-1.939571	-1.799999
	52°	-1.999732	-1.846266	1.873830	-1.858037	-1.996241	-1.800000
	56°	-1.947039	-1.848666	1.828434	-1.996761	-1.930711	-1.800000
	60°	-1.875726	-1.879583	1.914463	-1.967225	-1.998695	-1.800000
	64°	-1.875855	-1.997182	1.935898	-1.884725	-1.999471	-1.800000
	68°	-1.898631	-1.584275	1.806657	-1.960720	-1.999894	-1.799999
	72°	-1.893547	-1.982918	1.821540	-1.899186	-1.992165	-1.800000
	76°	-1.959639	-1.989787	1.731951	-1.859210	-1.901485	-1.800000
	80°	-1.869510	-1.999833	1.946797	-1.971390	-1.871277	-1.800000
	84°	-1.923152	-1.999798	1.991472	-1.983237	-1.999048	-1.800000
	88°	-1.886190	-1.999822	1.972649	-1.999863	-1.946391	-1.800000
	92°	-1.892516	-1.909473	1.890582	-1.999470	-1.999295	-1.799999
	96°	-1.997792	-1.928311	1.797999	-1.937674	-1.999928	-1.800000
	100°	-1.999147	-1.881313	1.879013	-1.891894	-1.999945	-1.799999
	104°	-1.999301	-1.781818	1.802205	-1.930747	-1.994140	-1.800000
	108°	-1.934025	-1.736010	1.766874	-1.854614	-1.999576	-1.800000
	112°	-1.955568	-1.979312	1.781231	-1.907628	-1.901149	-1.800000
	116°	-1.999134	-1.944723	1.809640	-1.884327	-1.977778	-1.800000
	120°	-1.915313	-1.999753	1.708577	-1.994915	-1.998957	-1.799999
	124°	-1.999320	-1.901237	1.869827	-1.998655	-1.962635	-1.800000
	128°	-1.992376	-1.993884	1.822220	-1.996432	-1.739046	-1.799999
	132°	-1.995507	-1.938301	1.985437	-1.998769	-1.904828	-1.800000
	136°	-1.999972	-1.878790	1.876094	-1.908389	-1.872481	-1.800000
	140°	-1.999972	-1.999442	1.834954	-1.984458	-1.923207	-1.800000
	144°	-1.984822	-1.806696	1.702898	-1.999383	-1.448670	-1.800000
	148°	-1.875726	-1.879583	1.915448	-1.967225	-1.998597	-1.800000
	152°	-1.895652	-1.867250	1.822387	-1.864242	-1.999653	-1.800000
	156°	-1.881447	-1.999960	1.991124	-1.994825	-1.995057	-1.800000
	160°	-1.898631	-1.999954	1.994507	-1.960720	-1.999894	-1.800000
	164°	-1.893545	-1.982918	1.833928	-1.899185	-1.992165	-1.800000
	168°	-1.951672	-1.999796	1.949968	-1.880404	-1.842844	-1.799999
	172°	-1.888960	-1.999100	1.991693	-1.999487	-1.998221	-1.800000
	176°	-1.888960	-1.999129	1.991585	-1.999487	-1.998221	-1.800000

Table A.17 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 4	0°	-1.903028	-1.928429	-1.895471	.114685	-1.997710	-1.800000
	4°	-1.872042	-1.886567	-1.957448	-.019516	-1.996409	-1.800000
	8°	-1.933154	-1.998995	-1.952216	.191901	-1.827386	-1.800000
	12°	-1.972073	-1.980229	-1.989690	-.010584	-1.981986	-1.800000
	16°	-1.948313	-1.845574	-1.999306	-.002626	-1.940297	-1.800000
	20°	-1.823766	-1.856101	-1.965627	-.131075	-1.768993	-1.800000
	24°	-1.830990	-1.994311	-1.869261	1.079803	-1.995090	-1.800000
	28°	-1.942137	-1.945095	-1.917916	-.062946	-1.925404	-1.800000
	32°	-1.838390	-1.841739	-1.994253	-.160108	-1.992555	-1.799999
	36°	-1.984405	-1.884766	-1.846713	.385450	-1.995082	-1.799999
	40°	-1.927097	-1.983353	-1.999527	.425710	-1.936022	-1.800000
	44°	.048358	-1.983112	-1.991753	.163868	-1.921261	-1.800000
	48°	-1.765731	-1.787860	-1.999163	-.027324	-1.948465	-1.800000
	52°	-1.755708	-1.993829	-1.998475	.232536	-1.957631	-1.800000
	56°	-1.966967	-1.927036	-1.994962	-.112186	-1.947674	-1.800000
	60°	-1.799150	-1.992977	-1.963060	-.118175	-1.995367	-1.800000
	64°	-1.941428	-1.914892	-1.995619	.612470	-1.901278	-1.800000
	68°	-1.940740	-1.898758	-1.920873	1.904100	-1.979222	-1.800000
	72°	-1.520525	-1.890469	-1.899691	-.088559	-1.980355	-1.800000
	76°	-1.912996	-1.999100	-1.934720	1.727275	-1.723001	-1.800000
	80°	-1.946845	-1.999979	-1.899789	1.869404	-1.466109	-1.800000
	84°	-1.841818	-1.999968	-1.894499	.471064	-1.916377	-1.800000
	88°	-1.898176	-1.999783	-1.951076	-.161163	-1.867366	-1.800000
	92°	-1.913746	-1.911808	-1.943129	-.171073	-1.917772	-1.799999
	96°	-1.871829	-1.904795	-1.957327	-.023854	-1.996416	-1.800000
	100°	-1.855353	-1.968512	-1.998419	-.011993	-1.959460	-1.800001
	104°	-1.903438	-1.984940	-1.865339	.887998	-1.999996	-1.800000
	108°	-1.995070	-1.903755	-1.997041	.933950	-1.119159	-1.800000
	112°	-1.897202	-1.996454	-1.915886	1.512288	-1.998325	-1.800001
	116°	-1.917549	-1.985331	-1.997766	1.952702	-1.990859	-1.800000
	120°	-1.934367	-1.883410	-1.949586	-.063806	-1.807019	-1.800000
	124°	-1.966590	-1.992717	-1.993259	1.432181	-1.814849	-1.799999
	128°	-1.980421	-1.964958	-1.999314	1.340956	-1.921697	-1.800000
	132°	-1.915347	-1.969455	-1.999617	1.829104	-1.991948	-1.799999
	136°	-1.899399	-1.982996	-1.991771	-.107628	-1.921040	-1.800000

Table A.17 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 4	140°	-1.903002	-1.993828	-1.998476	1.923492	-1.958481	-1.799999
	144°	-1.846320	-1.998516	-1.914147	-.156641	-1.984457	-1.800000
	148°	-1.966967	-1.927036	-1.994962	-.112186	-1.947674	-1.800000
	152°	-1.977024	-1.836283	-1.998112	-.014128	-1.869652	-1.800000
	156° '	-1.860369	-1.992188	-1.975823	1.495913	-1.991256	-1.800000
	160°	-1.988516	-1.950322	-1.932426	-.008140	-1.447112	-1.800000
	164°	-1.581596	-1.991241	-1.880694	-.127148	-1.948938	-1.800000
	168°	-1.936970	-1.999991	-1.941811	1.032130	-1.822722	-1.800000
	172°	-1.831294	-1.999959	-1.830633	1.988520	-1.925636	-1.800000
	176°	-1.940582	-1.999864	-1.924326	-.031897	-1.985663	-1.800000
# 5	0°	-1.750232	-1.996787	-1.999893	-1.996817	1.031958	-1.800000
	4°	-1.844803	-1.874707	-1.999716	-1.992011	-.005514	-1.800000
	8°	-1.992852	-1.874884	-1.997359	-1.997018	-.008257	-1.800000
	12°	-1.992780	-1.975842	-1.990752	-1.940598	1.219258	-1.800001
	16°	-1.995547	-1.997234	-1.994303	-1.994895	1.992877	-1.800001
	20°	-1.995641	-1.954839	-1.994432	-1.997945	1.994826	-1.800000
	24°	-1.997230	-1.998214	-1.999276	-1.997015	1.907892	-1.800000
	28°	-1.998956	-1.974383	-1.892645	-1.999728	1.874073	-1.800000
	32°	-1.967385	-1.998262	-1.999672	-1.984911	1.550050	-1.800000
	36°	-1.980175	-1.996577	-1.999603	-1.741685	1.996410	-1.800000
	40°	-1.880185	-1.809982	-1.999161	-1.999241	-.001499	-1.800000
	*44°	-1.995081	-1.969431	-1.997869	.004317	-.001085	-1.799999
	48°	-1.988313	-1.989347	-1.996989	-1.841279	-.006201	-1.800000
	52°	-1.881726	-1.998174	-1.998528	-1.790022	.008274	-1.800001
	56°	-1.973363	-1.999269	-1.986501	-1.999584	1.728431	-1.800001
	60°	-1.991063	-1.999835	-1.844675	-1.988217	.989020	-1.800001
	64°	-1.940601	-1.992928	-1.938522	-1.998858	1.988287	-1.800001
	68°	-1.939036	-1.999964	-1.799355	-1.964548	1.983174	-1.800000
	72°	-1.946374	-1.999495	-1.776911	-1.874932	-.054989	-1.800000
	76°	-1.905616	-1.991442	-1.960541	-1.926070	-.189623	-1.800000
	80°	-1.949602	-1.997813	-1.989341	-1.979829	-.021958	-1.800000
	84°	-1.841071	-1.257899	-1.807984	-1.996102	-.016584	-1.800000
	88°	-1.683515	-1.984446	-1.852125	-1.996465	-.214118	-1.800000

Table A.17 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 5	92°	-1.934968	-1.935626	-1.911624	-1.993247	-.000078	-1.800000
	*96°	-1.985072	.019178	-1.804402	-1.889977	-.119394	-1.800000
	100°	-1.952716	-1.356223	-1.996927	-1.999238	-.011217	-1.800000
	104°	-1.906746	-1.855863	-1.958162	-1.999641	-.000281	-1.800000
	108°	-1.999492	-1.528148	-1.997426	-1.996384	1.999791	-1.800001
	112°	-1.998648	-1.999803	-1.999298	-1.987844	1.999993	-1.800001
	116°	-1.998682	-1.999174	-1.997366	-1.955064	1.987751	-1.800000
	120°	-1.986285	-1.999327	-1.819584	-1.998196	1.989210	-1.800000
	124°	-1.996588	-1.997990	-1.996288	-1.997542	1.989927	-1.800000
	128°	-1.999651	-1.991939	-1.999633	-1.948355	1.999765	-1.800000
	132°	-1.998001	-1.999635	-1.999867	-1.928337	1.999942	-1.800000
	136°	-1.995081	-1.969299	-1.997869	-1.995683	1.974222	-1.799999
	140°	-1.996579	-1.999858	-1.995535	-1.996835	.000005	-1.800000
	144°	-1.987999	-1.985338	-1.974740	-1.928836	-.137253	-1.800000
	148°	-1.997996	-1.495605	-1.969884	-1.960274	.002984	-1.800000
	152°	-1.981360	-1.944880	-1.762837	-1.999992	1.998914	-1.799999
	156°	-1.921336	-1.999878	-1.828137	-1.840238	1.953740	-1.800000
	160°	-1.938244	-1.966608	-1.912351	-1.982636	1.886122	-1.800000
	164°	-1.989523	-1.972446	-1.844329	-1.984228	1.601749	-1.800000
	168°	-1.999161	-1.964930	-1.902146	-1.869923	1.846317	-1.800000
	172°	-1.993422	-1.855546	-1.921447	-1.952427	-.202402	-1.799999
	176°	-1.977172	-1.817855	-1.882852	-1.848223	1.849918	-1.800000
	180°	-1.921325	-1.996230	-1.998414	-1.995789	1.993279	-1.800000
	184°	-1.999772	-1.981877	-1.993173	-1.899802	1.984098	-1.800000
	188°	-1.971893	-1.767460	-1.882686	-1.999319	1.808229	-1.800000
	192°	-1.999720	-1.999832	-1.922907	-1.819218	1.627086	-1.800000
	196°	-1.997668	-1.992105	-1.944634	-1.904131	1.971418	-1.800000
	200°	-1.999917	-1.702371	-1.936751	-1.883649	1.911498	-1.800000
	204°	-1.983876	-1.999056	-1.832759	-1.912302	1.778089	-1.800000
	208°	-1.982078	-1.999372	-1.988245	-1.871403	1.993192	-1.800000
	212°	-1.952007	-1.991231	-1.991308	-1.985831	1.829325	-1.800000
	216°	-1.968968	-1.999340	-1.946446	-1.999056	1.989214	-1.800000
	220°	-1.983445	-1.994860	-1.984330	-1.999026	-.000404	-1.800000
	224°	-1.986834	-1.999163	-1.996491	-1.970350	-.000187	-1.800000

Table A.17 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 5	228°	-1.969351	-1.447977	-1.946616	-1.998921	-.110732	-1.800000
	232°	-1.925543	-1.998567	-1.741428	-1.999722	-.090059	-1.799999
	*236°	-1.966015	-1.997865	.070944	-1.999934	-.007583	-1.800000
	240°	-1.936288	-1.842669	-1.837780	-1.999991	1.952574	-1.800000
	244°	-1.946799	-1.869964	-1.789049	-1.999986	1.686255	-1.800000
	248°	-1.961267	-1.936503	-1.984744	-1.999870	-.051053	-1.800000
	252°	-1.984877	-1.876498	-1.857656	-1.999617	-.060415	-1.800000
	256°	-1.965920	-1.811399	-1.997651	-1.999764	-.175733	-1.800000
	260°	-1.920346	-1.994003	-1.753159	-1.999097	-.200096	-1.800000
	264°	-1.984538	-1.996897	-1.994357	-1.988923	-.157315	-1.800000
	268°	-1.920346	-1.970104	-1.975762	-1.999097	-.200096	-1.800000
	*272°	-1.978584	.000065	-1.786458	-1.999953	-.047541	-1.800000
	276°	-1.996932	-1.889714	-1.821374	-1.999970	-.020692	-1.800000
	280°	-1.995886	-1.980533	-1.899255	-1.999993	-.054576	-1.800000
	284°	-1.970744	-1.884470	-1.881728	-1.999997	.014342	-1.800000
	288°	-1.815556	-1.883425	-1.910215	-1.999980	1.861760	-1.800000
	292°	-1.867945	-1.993340	-1.981778	-1.999986	1.973234	-1.800000
	296°	-1.988839	-1.998145	-1.979066	-1.998126	1.999674	-1.800000
	300°	-1.995615	-1.998472	-1.983651	-1.843573	1.936079	-1.800000
	304°	-1.994905	-1.996826	-1.988335	-1.912926	1.979571	-1.800000
	308°	-1.790191	-1.998263	-1.987658	-1.999972	1.999188	-1.800000
	312°	-1.879636	-1.999968	-1.997748	-1.999924	1.999974	-1.800000
	316°	-1.977329	-1.994945	-1.945549	-1.999146	1.901203	-1.800000
	320°	-1.969760	-1.988422	-1.982801	-1.999720	1.978627	-1.800000
	324°	-1.988247	-1.976808	-1.992560	-1.999174	1.296144	-1.800000
	328°	-1.995453	-1.972046	-1.992942	-1.999692	1.998674	-1.800000
	332°	-1.963221	-1.998336	-1.831776	-1.999849	1.985816	-1.800000
	336°	-1.989138	-1.935327	-1.993429	-1.999774	1.988779	-1.800000
	340°	-1.986578	-1.904126	-1.935227	-1.999868	1.983757	-1.800000
	344°	-1.990475	-1.992738	-1.887970	-1.998354	1.795591	-1.800000
	348°	-1.994796	-1.994568	-1.933502	-1.847519	-.003423	-1.800000
	352°	-1.978574	-1.816591	-1.813996	-1.837008	-.194244	-1.800000
	356°	-1.758541	-1.833553	-1.983261	-1.986124	-.000670	-1.800000

Table A.18 Recognition process used the neural network which trained using the features extracted from the projection vector in X direction with 10° angular interval: X.

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node
		#1	#2	#3	#4	#5	Average
# 1	5°	.741807	-.946223	-.979080	-.999971	.998785	-.900000
	*15°	-.951186	-.402787	-.989583	-.996468	.996285	-.900000
	*25°	-.999704	-.999998	-.854941	-1.000000	1.000000	-.900000
	*35°	-.998431	.996356	-.816688	-.953953	-.999926	-.900000
	*45°	-1.000000	-.999989	-.845339	-1.000000	.999998	-.900000
	*55°	-.514600	-.663460	-.951340	-.994949	-.143640	-.900000
	*65°	-.506043	-.999852	-.164685	-.999998	.999905	-.900000
	75°	.957482	-.977390	-.877622	-.706593	-.989212	-.900000
	85°	.726261	-.999553	-.517201	-.997453	-.934028	-.900000
# 2	5°	-.988824	.925491	-.857667	-.999977	-.999565	-.900000
	15°	-.998634	.964498	-.904221	-.993154	-.988368	-.900000
	*25°	-.998979	-.989560	-.768929	-.996059	.925158	-.900000
	35°	-.998938	.999028	-.961896	.952788	-.999960	-.900000
	*45°	-1.000000	-.999088	-.920939	-1.000000	1.000000	-.900000
	55°	-.956397	.719977	-.928927	-.882501	-.261792	-.900000
	65°	-.756781	.433165	-.880856	-.895272	-.882361	-.900000
	*75°	.532287	-.875606	-.975483	-.999447	.985783	-.900000
	*85°	-.635180	-.865642	.185491	-.997489	-.999986	-.900000
# 3	5°	-.755527	-.865093	.931812	-.992421	-.999998	-.900000
	15°	-.896721	-.918714	.522349	-.999396	-.997816	-.900000
	*25°	-.999853	-.987548	.555188	-.999981	.917257	-.900000
	35°	-.934126	-.921982	.744057	-.997362	-.959519	-.900000
	*45°	-.060853	-.995065	-.212616	-.999993	-.076084	-.900000
	55°	-.993976	-.937250	.968128	-.981769	-.999637	-.900000
	*65°	-.868131	-.999918	.015542	-.999999	.999990	-.900000
	75°	.870614	-.999371	.878420	-.999568	.362244	-.900000
	85°	-.970321	-.908457	.962522	-.991384	-.999521	-.900000

Table A.18 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 4	*5°	.771422	-.570432	-.958470	-.999803	.825038	-.900000
	15°	-.998799	.749946	-.995572	.894067	.722698	-.900000
	*25°	-.999704	-.999998	-.854941	-1.000000	1.000000	-.900000
	*35°	-.998431	.996356	-.816688	-.953953	-.999926	-.900000
	*45°	-1.000000	-.999989	-.845339	-1.000000	.999998	-.900000
	*55°	-.167138	-.762409	-.973114	-.998866	.427858	-.900000
	*65°	.185743	-.999784	-.289760	-1.000000	.999861	-.900000
	75°	.613691	-.950003	-.913272	.903448	-.999865	-.900000
	*85°	.946554	-.994541	-.914341	-.999908	.584638	-.900000
# 5	*5°	-.999497	-.518667	-.985119	-.986875	-.934977	-.900000
	15°	-.999991	-.969806	-.997323	-.999961	.999800	-.900000
	25°	-.935725	-.999830	-.346988	-.999252	.992518	-.900000
	*35°	-.998675	.275899	-.973504	.998672	-.934872	-.900000
	45°	-.999798	-.576167	-.878408	-.998018	.999984	-.900000
	55°	-.999804	-.998959	-.999340	-.970654	.999950	-.900000
	65°	-.999997	-.999999	-.974916	-1.000000	1.000000	-.900000
	*75°	.731650	-.999055	.775042	-.999199	-.552299	-.900000
	*85°	.994852	-.996656	-.942410	-.999999	.702359	-.900000
	*95°	-.996743	.979328	-.880989	-.999999	-.999992	-.900000
	105°	.598698	-.926192	-.998660	-.999993	1.000000	-.900000
	115°	-1.000000	-1.000000	-.929806	-.999965	1.000000	-.899999
	*125°	-.994957	.653011	-.265125	.995300	-.993112	-.900000
	135°	-.873591	-.761635	-.857044	-.999994	.991522	-.900000
	145°	-.997815	-.785534	-.964886	-.821517	.984524	-.900000
	*155°	.955515	-.569380	-.950243	-.999992	-.748519	-.900000
	*165°	-.999874	.915261	-.964809	-.999959	-.910157	-.900000
	175°	-1.000000	-.999996	-.982834	-.999931	1.000000	-.900000
	*185°	-.999133	-.709241	-.382732	.999859	-.996258	-.900000

Table A.18 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 5	195°	-.999998	-.859326	-.988091	-.996584	.999227	-.900000
	205°	-1.000000	-.999998	-.912866	-1.000000	1.000000	-.900000
	*215°	-1.000000	.976433	-.911937	-.999655	-.999814	-.900000
	225°	-1.000000	-.999053	-.717254	-1.000000	1.000000	-.900000
	*235°	-.999652	-.965363	.497253	-.993413	-.995789	-.900000
	245°	.591104	-.999847	-.751925	-.999925	.999997	-.900000
	*255°	-.997307	-.907102	.938531	-.966465	-.999951	-.900000
	*265°	-.915323	-.771828	.670290	-.987448	-1.000000	-.900000
	*275°	.979053	-.988835	.529804	-1.000000	-.999456	-.900000
	*285°	-.015948	.721113	-.911571	-.996060	-.675070	-.900000
	295°	-1.000000	-.999915	-.638899	-.999837	.999983	-.900000
	*305°	-.999642	.990713	-.516416	.096713	-.559644	-.900000
	315°	-1.000000	-.999785	-.961353	-.999978	1.000000	-.900000
	325°	.626757	-.996467	-.887079	-.999932	.989968	-.900000
	335°	.163404	-.994241	-.053270	-.625941	.963830	-.900000
	345°	-.296501	-.996460	-.041337	-.975020	.995527	-.900000
	355°	.792517	-.997997	-.924476	-1.000000	.932695	-.900000

Table A.19 Recognition process used the neural network which trained using the features extracted from the projection vector in X direction with 6° angular interval: X.

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 1	3°	.869487	-.875082	-.730354	-.720693	-.999998	-.900000
	*9°	-.737597	-.879053	-.971598	-.999257	-.999954	-.900000
	15°	.999020	-.562986	-.979791	-.967706	-.999979	-.900000
	*21°	-.999970	-.997218	-.999567	-.663796	.987586	-.900000
	*27°	-.999601	-.998723	-.466817	-1.000000	.869183	-.900001
	*33°	-1.000000	-.315586	-.595414	-.996776	.971016	-.900000
	39°	.947681	-.996751	-.977337	-.999972	.973470	-.900001
	*45°	-1.000000	-.999991	-.995583	-.999995	1.000000	-.900000
	*51°	-1.000000	-.991312	-.726839	.999386	.998666	-.900001
	*57°	-.999999	-.956231	-.998351	.752151	-.091926	-.900000
	*63°	-.999961	-.998977	.476774	.324267	-.778813	-.900001
	69°	.912901	-.652379	-.998660	-.999795	-.999989	-.900000
	*75°	-1.000000	-.995822	-.998107	.999981	-.486609	-.900000
	81°	.752015	-.988564	-.817281	.516378	-.983977	-.900000
	*87°	-.999833	-.999345	.843740	-.317539	.956373	-.900000
# 2	*3°	-1.000000	-.071198	-.681154	-.999995	-.999526	-.900000
	9°	-1.000000	.452079	-.735676	-.999615	-.999039	-.900000
	15°	-1.000000	.936379	-.554558	.686926	-.999795	-.900000
	*21°	-1.000000	-.298894	-.999678	-1.000000	.999986	-.900000
	*27°	.902772	-.990440	-.928112	-1.000000	.992137	-.900000
	*33°	-1.000000	-.747785	-.987283	-.999939	.999985	-.900000
	*39°	-.999856	-.990812	-.998072	-.999987	.999984	-.900001
	*45°	-1.000000	-.952381	-.998127	-1.000000	1.000000	-.900001
	51°	-.999999	.961173	-.838704	-.999898	-.811284	-.900001
	57°	-1.000000	.927874	-.983152	-.939458	-.994531	-.900001
	63°	-.997635	.572916	-.535362	-.999963	-.996455	-.900000
	*69°	-1.000000	-.081462	-.997550	-1.000000	.999997	-.900000
	75°	-1.000000	.970872	-.990009	-.999997	-.548253	-.900000
	81°	-.999995	.838397	-.858122	-.996234	-.999270	-.900000
	*87°	-1.000000	-.091135	.599008	-.999995	-.634148	-.900000

Table A.19 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 3	3°	-.999993	-.799265	.993518	-.700128	-.999715	-.900000
	9°	-.999991	-.859424	.991340	-.989729	-.994487	-.900000
	15°	-.999999	-.946959	.965993	-.896660	-.997496	-.900000
	*21°	-.999799	.661820	-.968428	-.999954	-.999999	-.900000
	27°	-1.000000	-.906007	.976229	-1.000000	-.592639	-.900000
	*33°	-1.000000	-.991990	.595170	-.997042	.680501	-.900000
	39°	-1.000000	-.921724	.984855	-1.000000	-.817011	-.900000
	45°	-1.000000	-.927140	.895465	-.999492	.600322	-.900000
	51°	-.999937	-.951776	.855205	-.999999	-.862400	-.900000
	57°	-.999816	-.951246	.761882	-.999900	-.984401	-.900000
	63°	-1.000000	-.998393	.850745	-1.000000	.647652	-.900000
	*69°	-1.000000	-.991877	.738218	-.999962	.998515	-.900000
	75°	-1.000000	-.997465	.623686	-.999999	-.809988	-.900000
	81°	-1.000000	-.982600	.891860	.212749	-.974408	-.900000
	87°	-1.000000	-.887520	.970472	-.994807	-.891438	-.900000
# 4	3°	-.999913	-.487656	.498165	.999981	-1.000000	-.900000
	9°	-1.000000	.233033	-.709662	.999998	-.999969	-.900001
	15°	-.181140	-.576129	-.998982	.988613	-.976984	-.900000
	21°	-1.000000	-.997523	-.999296	.999411	.878631	-.900001
	*27°	-.998626	-.997805	.408673	-.999952	-.822208	-.900001
	*33°	-.999999	-.019172	.383624	-.994958	-.072128	-.900000
	*39°	-.915856	-.999912	-.936625	-.991727	.999454	-.900001
	*45°	-1.000000	-.999995	-.989825	-.998760	.999963	-.900000
	51°	-.999972	-.991422	-.717360	1.000000	-.987999	-.900000
	*57°	-1.000000	-.966826	-.993968	.802922	.961760	-.900000
	*63°	-.998855	-.999453	-.022627	-.859548	-.372518	-.900001
	*69°	.999973	-.934771	-.999069	-.999999	-.999931	-.900000
	75°	-1.000000	-.997166	-.847889	1.000000	-.994755	-.900000
	*81°	-.999979	-.999524	-.994374	-.879914	.999791	-.900000
	87°	-.999840	-.987686	-.738006	.999999	-.999993	-.900000

Table A.19 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 5	3°	-.986237	-.998697	-.992974	-.998768	.972512	-.900000
	*9°	-.999999	-.922137	-.612954	.999959	-.994511	-.900000
	15°	-.999970	-.945593	-.999814	-.404891	.853472	-.900000
	*21°	-.996465	-.872669	-.999236	.459255	-.514005	-.900000
	27°	-1.000000	-.999359	-.987050	-1.000000	1.000000	-.900001
	33°	-1.000000	-.999801	-.990676	-.999991	1.000000	-.900000
	39°	-1.000000	-.999967	-.993529	-1.000000	.999991	-.900000
	*45°	-1.000000	.344544	-.042893	.995817	-.997893	-.900000
	51°	-1.000000	-.999575	-.966736	-.984515	1.000000	-.900000
	57°	-.978723	-.984274	-.981846	-.999974	.869792	-.900000
	63°	-.999993	-.999680	-.935088	-1.000000	.999820	-.900001
	69°	-1.000000	-.998105	-.998014	-1.000000	.999999	-.900000
	75°	-1.000000	-.999285	.261220	-.999996	.987288	-.900000
	81°	-.950503	-.838815	-.823714	-1.000000	.748178	-.900001
	*87°	-1.000000	.987400	.032992	-.035012	-.999985	-.899999
	*93°	-.941745	.402474	-.901760	-.999999	-.999967	-.899999
	*99°	-1.000000	.381176	-.997685	-1.000000	.032190	-.900000
	*105°	-.944728	.339380	-.963756	-.999644	-.876236	-.900000
	111°	-1.000000	-.999903	-.999927	-.999521	1.000000	-.900000
	117°	-1.000000	-.999940	-.994076	-1.000000	1.000000	-.900001
	123°	-1.000000	-.269089	-.994121	-1.000000	1.000000	-.900000
	129°	-.999999	-.994456	-.992407	-1.000000	.999997	-.900001
	135°	.992300	-.997369	-.999683	-1.000000	.999914	-.900000
	141°	-.999880	-.999918	-.696700	-.993286	.999363	-.900000
	147°	-1.000000	-.935383	-.998397	-.999997	1.000000	-.900001
	153°	-1.000000	-.968760	-.914463	-.999989	.999996	-.900001
	159°	-1.000000	.436084	-.889401	-.887842	.995352	-.900000
	165°	-1.000000	-.996875	-.993087	-.692258	.999945	-.900001
	171°	-1.000000	-.999904	-.998421	-.999997	.999862	-.900001
	177°	-1.000000	-.999751	-.965192	-1.000000	.995508	-.900001

Table A.19 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 5	183°	-1.000000	-.999478	-.452141	-.999310	.959771	-.900000
	189°	-1.000000	-.999798	-.988736	-.999789	.861002	-.900000
	*195°	-1.000000	-.992181	-.826448	-.999997	-.131877	-.900001
	201°	-1.000000	-.999840	-.994161	-1.000000	.999807	-.900000
	207°	-1.000000	-.999997	-.948877	-1.000000	1.000000	-.900001
	213°	-1.000000	-.721729	-.652845	-.999994	.999743	-.900001
	219°	-1.000000	-.999972	-.997941	-1.000000	1.000000	-.900001
	225°	-1.000000	-.999458	-.999811	-1.000000	1.000000	-.900001
	231°	-1.000000	-.999931	-.239759	-.999991	1.000000	-.900000
	237°	-1.000000	-.999869	.829619	.986036	.991483	-.900000
	243°	-.338634	-.999282	-.600308	-1.000000	.885989	-.900001
	249°	-1.000000	-.880464	-.970932	-.994358	.958305	-.900000
	*255°	-1.000000	-.854537	.953452	-.999996	-.980150	-.900000
	*261°	-.999982	-.945527	.604221	-.999992	-.003090	-.900000
	267°	-.999909	-.757352	-.434513	-.999985	.948034	-.900000
	*273°	-.999994	.797773	-.989154	-.180455	-1.000000	-.900000
	*279°	-.332857	.721932	-.885723	-.999896	-.999998	-.900000
	285°	-.999962	-.947718	-.994312	-1.000000	.999709	-.900000
	291°	-1.000000	-.774233	-.999666	-.999985	1.000000	-.900000
	297°	-1.000000	-.996777	-.975695	-1.000000	1.000000	-.900001
	303°	-1.000000	-.940331	-.997663	-1.000000	.999995	-.900000
	309°	-1.000000	-.995103	-.996332	-.999999	.999985	-.900001
	315°	-1.000000	-.980658	-.995581	-.999996	1.000000	-.900000
	321°	-1.000000	-.999954	-.999267	-1.000000	1.000000	-.900000
	327°	-.992700	-.999953	-.992444	-.999939	1.000000	-.900001
	333°	-.999999	-.998943	-.984489	-.824763	.999999	-.900000
	339°	-.999993	-.341704	-.861555	-.996748	.403642	-.900001
	345°	-.999671	-.998907	.148918	-1.000000	.999839	-.900000
	351°	.962071	-.998820	-.995324	-1.000000	.999622	-.900000
	357°	-.999740	-.999544	-.857957	-.967537	.992905	-.900000

Table A.20 Recognition process used the neural network which trained using the features extracted from the projection vector in X and Y directions with 10° angular interval: $(X+Y)/2$.

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 1	*5°	-.663713	-1.000000	-.999561	.702704	-.764145	-.893423
	15°	.602152	-.999999	-1.000000	-.984467	-.999982	-.920255
	25°	.876768	-1.000000	-.999999	-.995856	-.998556	-.925439
	35°	.998473	-1.000000	-.999953	-.978705	.461781	-.884835
	45°	.998076	-1.000000	-.999968	-.999999	-.999018	-.908642
	55°	.996661	-.998048	-.999583	-.999536	-.999993	-.901472
	65°	.999320	-.999989	-.979468	-.997280	-.891388	-.889576
	75°	.999280	-1.000000	-1.000000	-.999998	.890204	-.889627
	85°	.758800	-.999933	-.996238	.060135	-.351903	-.881202
# 2	5°	-.990370	.999589	.967923	-.995539	-.999999	-.895881
	15°	-1.000000	1.000000	-1.000000	-.918891	-.999617	-.878989
	25°	-.999961	.988387	-.987822	-.998663	-.997020	-.899921
	*35°	-1.000000	-.289040	.704693	-.872629	-.999958	-.925452
	*45°	-.999712	-.999408	-.984366	-.998441	.999804	-.892487
	*55°	-.979130	-.627854	-.999999	-1.000000	.999596	-.890227
	65°	-.999245	.974315	-.891735	-1.000000	-.146164	-.909230
	*75°	-1.000000	-.886271	-.999953	-1.000000	.359341	-.870050
	*85°	-.533272	.195450	-.981983	-.999598	.406757	-.897294
# 3	5°	-.999060	-.906792	.989668	-.986062	-.996714	-.890325
	*15°	-.997709	.757252	-.994584	-1.000000	.089965	-.914445
	25°	-.999911	-.993653	1.000000	-.805728	-1.000000	-.881846
	35°	-1.000000	-.999650	.999994	-.765658	-.999994	-.918238
	*45°	-.999997	-.999999	-.536347	-.998752	.998719	-.865061
	*55°	-1.000000	-.996805	-.994214	-1.000000	.978951	-.917373
	65°	-.981695	-1.000000	.971614	-.999873	-.992775	-.897994
	*75°	-.974531	.916755	-.999981	-.999988	-.044688	-.904034
	85°	-.974843	-.968081	.997843	-.999555	-.995993	-.884107

Table A.20 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node
		#1	#2	#3	#4	#5	Average
# 4	*5°	.886166	-1.000000	-.742803	-.817923	.746398	-.862384
	15°	-.998800	-.997718	-1.000000	.999998	-.985112	-.897187
	25°	-.818866	-1.000000	-.927393	.877517	-.998947	-.900847
	35°	-.999987	-.999951	-.948526	.999972	-.997567	-.903713
	45°	-.999557	-.867205	-1.000000	.999972	-.998879	-.903055
	55°	-.996541	-.909260	-.969388	.683602	-.999165	-.904878
	*65°	.522543	-.999989	-.992543	-.427349	.425143	-.890910
	*75°	-.355354	-1.000000	-.999105	-.453003	-.678798	-.877078
	*85°	.791035	-.999999	-.988107	-.943172	-.978579	-.875741
# 5	5°	-1.000000	-.999981	-.989915	-1.000000	.999579	-.875550
	15°	-.994968	.997942	-.987632	-.999996	.962433	-.898450
	25°	-.999992	-.999854	-.999985	-1.000000	.999996	-.887791
	35°	-1.000000	-.573629	-.998327	-.999536	.999792	-.905127
	45°	-1.000000	-.992403	-.999970	-1.000000	.999945	-.927218
	55°	-1.000000	-1.000000	-1.000000	-1.000000	1.000000	-.920658
	65°	-.999999	-.999827	-.893170	-.999997	.677605	-.919982
	*75°	-1.000000	-1.000000	-.510413	-.503382	.196241	-.929427
	85°	-.998782	-1.000000	-.678539	-.999756	1.000000	-.903037
	95°	-.999861	-.998724	-.999976	-.999978	.999642	-.907994
	105°	-.999999	-1.000000	-.698281	-.999999	.999984	-.938862
	115°	-.999995	-1.000000	-1.000000	-1.000000	.999998	-.941705
	125°	-1.000000	-.998425	-.999946	-.999996	1.000000	-.920218
	135°	-.999998	-.997368	-.999948	-1.000000	.999975	-.893886
	145°	-.999998	-.999999	-1.000000	-1.000000	.999923	-.900304
	*155°	-.393536	-.999985	.956334	-1.000000	-.999960	-.890371
	165°	-.999998	-.999649	-.283750	-1.000000	.975833	-.907793
	175°	-.764082	-.974297	-.999939	-1.000000	1.000000	-.856520
	185°	-1.000000	-1.000000	-.999256	-1.000000	1.000000	-.915572

Table A.20 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 5	195°	-.931414	.132034	-.949282	-.999793	.999499	-.846416
	205°	-.999988	-.406096	-.759219	-.839499	.999982	-.891136
	215°	-.999972	-.999688	-.999999	.298654	.999933	-.900447
	225°	-1.000000	-.999994	-1.000000	-.999992	1.000000	-.914922
	235°	-.999989	-1.000000	-.999991	-1.000000	1.000000	-.919064
	245°	-.996710	-.999998	-.988409	-1.000000	1.000000	-.922805
	255°	-.750668	-.999981	-.998088	-1.000000	1.000000	-.912037
	265°	-.995296	.117208	-.999957	-1.000000	1.000000	-.894409
	275°	-.999370	-.998571	-.999999	-.999999	.999996	-.893580
	285°	-.997950	-.998189	-1.000000	-.994658	1.000000	-.888413
	295°	-.983466	-1.000000	-.566622	-.999887	1.000000	-.886979
	305°	-1.000000	-1.000000	-1.000000	-1.000000	1.000000	-.907734
	315°	-1.000000	-1.000000	-1.000000	-1.000000	1.000000	-.907490
	325°	-1.000000	-1.000000	-.999993	-1.000000	1.000000	-.920351
	335°	-.999914	-.999995	-1.000000	-1.000000	.999996	-.929164
	345°	-.999993	-.999880	-1.000000	-1.000000	.999648	-.928678
	355°	-.978951	-1.000000	-.963943	-1.000000	.999872	-.883825

Table A.21 Recognition process used the neural network which trained using the features extracted from the projection vector in X and Y directions with 6° angular interval: $(X+Y)/2$.

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 1	3°	.933509	-.682352	-.998935	-.723613	-.998163	-.899971
	*9°	.366604	.506610	-.942071	.521360	-1.000000	-.899981
	15°	.978734	-.338716	-.999489	-.967104	-.999644	-.899978
	*21°	-.031362	-.999789	-.999529	-.643210	-.932151	-.899972
	*27°	-.722727	-.999994	-.999784	-.981317	-.741422	-.900035
	33°	.961144	-.999238	-.999881	-.913873	-.966691	-.899950
	*39°	.357173	-.999937	-.989887	-.989491	.444393	-.899980
	45°	.990877	-.958577	-.999113	-.896209	-.999992	-.900043
	51°	.987825	-.999998	-.907956	-.999554	.433020	-.900014
	57°	.774494	-.931177	-.806994	-.705373	-1.000000	-.900008
	*63°	-.047799	-.998897	-.944630	-.183607	-.999900	-.899986
	69°	.992378	-.884614	-.985915	-.993402	-.987151	-.899989
	75°	.986130	-.999998	-.934121	-.968528	.805032	-.900033
	81°	.996696	-.995281	-.917824	-.960316	-.999981	-.899977
	*87°	-.608332	-.993224	-.980375	.739549	-.999998	-.900020
# 2	3°	.028455	.993832	-.653733	-.762968	-.999984	-.899978
	9°	-.941472	.994147	-.131548	-.830323	-.999836	-.899960
	*15°	-.928589	.086987	.590135	-.806912	.220030	-.899992
	21°	-.923546	.546618	-.999930	-.996792	-.294074	-.900023
	27°	-.708667	.820999	-.997659	-.998234	-.988640	-.899996
	33°	-.771074	.640397	-.998802	-.997963	-.997320	-.900009
	*39°	-.981293	-.755810	-.998061	-.999999	-.583903	-.900107
	45°	-.635319	.983725	.852174	.972465	-.999007	-.899861
	*51°	-.994523	-.898816	-.999071	-.999999	.777875	-.900088
	57°	-.978562	.605375	-.983687	-.998214	-.988654	-.899993
	63°	-.994140	.594143	-.998147	-.999998	-.964812	-.900040
	69°	-.374723	.924080	-.901627	-.999980	-.999925	-.900013
	75°	-.200175	.907351	-.999986	-1.000000	.999991	-.900009
	81°	.294461	.997930	-.551183	-.976124	-.999993	-.899899
	*87°	-.943286	-.870208	-.859408	-.930181	-.912780	-.899993

Table A.21 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 3	3°	-.934985	-.979376	.884957	-.983966	-.912422	-.900005
	9°	-.955417	-.996740	.976544	-.913193	-.869316	-.899995
	15°	-.964454	-.996720	.988182	-.952758	-.998759	-.899998
	*21°	-.548185	.567918	-.033324	-.996472	.146890	-.899965
	27°	-.991404	-.982774	.998921	-.939477	-.999899	-.899976
	*33°	-.991578	-.896997	-.988229	-.993829	-.999036	-.899973
	39°	-.995686	-.277070	.921287	-.999915	-1.000000	-.900087
	*45°	-.999980	-.995880	-.962378	-.999321	.975358	-.900145
	51°	-.999998	-.997502	.590534	-.999980	-.320593	-.899997
	*57°	-.997458	-.991600	.225260	-1.000000	.998897	-.900005
	63°	-.983807	-.992254	.989472	-.999844	-.993428	-.900019
	*69°	-.999455	-.993291	-.976370	-.370737	.981993	-.900007
	75°	-.921934	-.982942	.955236	-.938114	-.968820	-.899985
	81°	-.872458	-.688406	.970012	-.992786	-.999878	-.900035
	*87°	-.993989	-.999120	.179060	-.977407	.758045	-.900036
# 4	*3°	-.994640	-.962213	-.779564	.999973	-.999984	-.899992
	9°	-.770221	-.973667	-.885367	.994328	-1.000000	-.899969
	*15°	.459151	-.880852	-.999913	-.671049	-.999999	-.900016
	*21°	-.638886	-.999366	-.999823	-.581734	-.970737	-.900000
	27°	-.837656	-.999968	-.992869	.520661	-.998496	-.900048
	*33°	-.282521	.656965	-.999706	-.051968	-.983046	-.899923
	39°	-.787724	-.996426	-.930889	.999688	-.999999	-.899953
	*45°	-.953992	-.994461	-.999998	-.230626	-1.000000	-.900185
	51°	-.537220	-.992262	-.998634	.999823	-.999984	-.899955
	57°	-.698170	-.988792	-.999965	.397614	-.997836	-.900015
	63°	-.821386	-.999854	-.779617	.875238	-.999999	-.899973
	69°	.199358	-.902570	-.802669	.994413	-1.000000	-.899991
	*75°	-.992255	-.999997	-.999999	-.791591	.999984	-.900024
	81°	-.986134	-.999634	-.946801	.987143	-.999835	-.899993
	87°	-.925620	-.999513	-.998683	.927623	-.986899	-.899967

Table A.21 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 5	3°	-.943969	-1.000000	-.999903	-.995635	.999941	-.900030
	9°	-.995073	-.998166	-.992801	-.983105	.999962	-.899952
	15°	-.999671	-.994738	-.999241	.606683	.999398	-.899954
	21°	-.999995	-.999367	-.996988	.938067	1.000000	-.899957
	27°	-.972089	-.999980	.918916	-.999995	.980949	-.900009
	33°	-.997934	-.980572	-.833200	-1.000000	.999986	-.900001
	39°	-.992753	-.614553	-.784390	-.998811	.999620	-.899893
	45°	-.998393	.848367	-.998935	-.984214	.997047	-.899941
	51°	-.998684	-.992685	-.999914	-1.000000	1.000000	-.900047
	57°	-.979754	-.999979	-.999858	-1.000000	1.000000	-.900032
	63°	-.999708	-1.000000	-.999901	-.999986	.975392	-.900136
	69°	-.999894	-.999999	-1.000000	-.999891	.999184	-.900130
	75°	-.955547	-.999994	-.999983	-.997236	.999588	-.900075
	*81°	-.997535	-.948803	.846582	-.977018	-.756814	-.900002
	*87°	-.985439	-.869394	-.934972	-.991628	-.960174	-.900016
	*93°	-.985396	-.816653	.564545	-.986648	-.710140	-.899964
	99°	-.999037	-.997063	-.989700	-.986300	.999608	-.900009
	105°	-.968228	-.894903	-.900446	-.987593	.950708	-.899980
	111°	-.999999	-.999898	-.998887	-.999998	.999996	-.900089
	117°	-.999738	-.999978	-.863510	-.999994	.999611	-.900046
	123°	-.997394	-.999800	-.950858	-1.000000	.999978	-.900032
	129°	-.999808	-.999349	-.991013	-.999999	.999993	-.900028
	135°	-.996623	-.379443	-.999999	-.867235	.945372	-.899966
	141°	-.993261	-.990133	-1.000000	-.974724	.999999	-.900041
	*147°	-.999688	-.955986	-.999958	-.730439	-.300112	-.900045
	153°	-.999992	-.976264	-.999970	-.999999	.995482	-.900191
	*159°	-.883733	.932350	-.997324	-.998225	-.951546	-.900021
	165°	-.998043	-.980945	-.991480	-.998323	.999959	-.900024
	171°	-1.000000	-.999995	-.999949	-.778282	1.000000	-.900032
	177°	-.999965	-.998999	-.999605	-.998737	.999860	-.899959
	183°	-.998854	-.953851	-.398043	-.997494	.971892	-.899965

Table A.21 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 5	189°	-.999993	-.999999	-.999986	-.955147	1.000000	-.900014
	195°	-.912755	.351001	-.999986	-.999998	.987138	-.900003
	201°	-.349678	-.991022	-.986564	-.999999	.866266	-.900028
	207°	-.990731	-.998759	.017976	-.999897	.999939	-.899996
	*213°	-.999921	-.981717	-.749366	.859594	-.403677	-.899996
	219°	-.996566	-.997560	-.946396	-.999938	.998095	-.900006
	225°	-.999154	-.364956	-.999915	-.999970	.995744	-.900003
	231°	-1.000000	-.999990	-1.000000	-1.000000	1.000000	-.900086
	237°	-1.000000	-1.000000	-1.000000	-.999991	.999991	-.900121
	243°	-.999925	-1.000000	-.999682	-.998393	.999414	-.900093
	249°	-.999999	-.999997	-1.000000	-1.000000	.970705	-.900209
	255°	-.948520	-.999860	-.831376	-.996645	.997895	-.899988
	261°	-.990101	-.998552	-.969317	-.547970	.874393	-.899991
	267°	-.992954	-.866030	-.999998	-.997639	.997895	-.900029
	*273°	-.984617	.743329	-.999873	-.974877	.654580	-.899985
	279°	-.978705	-.919613	-.958206	-.768269	.834116	-.899983
	*285°	-.794769	-.178288	-.671038	-.999989	-.994444	-.900043
	*291°	-.999662	-.888538	-.966124	-.996643	-.965179	-.900017
	*297°	-.929874	-.965505	.880344	-.999824	-.999996	-.900017
	303°	-.996743	-.991733	-.974825	-1.000000	.996695	-.900044
	309°	-.999996	-.999533	-.999879	-.999954	.999983	-.900034
	315°	-.999398	-.964280	-.999865	-.999989	.999975	-.899979
	*321°	-.999660	-.998792	-.503942	-.945342	-.974371	-.899955
	327°	-.998246	-.999968	-.998659	-.999980	.999957	-.899986
	333°	-.990043	-.999993	-.993220	-.999953	.857063	-.900059
	*339°	-.859840	-.995890	-.994104	-.918051	-.827869	-.899998
	345°	-.947406	-.999996	-.999997	-.975210	.999967	-.899978
	351°	-.919014	-1.000000	-.999988	-.974187	.999870	-.899980
	357°	-.953334	-.999999	-.999635	-.948940	.998589	-.899951

Table A.22 Recognition process used the neural network which trained using the features extracted from the projection vector in X and Y directions with 4° angular interval: $(X + Y)/2$.

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 1	2°	.928858	-.998989	-.936996	-.972775	-.999993	-.900000
	6°	.975798	-.999796	-.982521	-.992280	-.841295	-.900000
	10°	.811664	-.975636	-.981403	-.841149	-1.000000	-.900000
	14°	.981095	-.999690	-.950545	-.987007	-.999954	-.900000
	18°	.987487	-.998779	-.927770	-.999990	-.887879	-.900000
	22°	.776151	-.999993	-.878504	-.847367	-.999183	-.900000
	26°	.983780	-.997494	-.966404	-.998830	-1.000000	-.900000
	30°	.957724	-.999118	-.988472	-.914745	-.988517	-.899999
	34°	.951272	-.997264	-.968982	-.990003	-.999004	-.899999
	38°	.918669	-.988745	-.993889	-.983381	-.774239	-.900000
	42°	.997389	-.998823	-.926816	-.999755	-.999889	-.900000
	46°	.801921	-.873862	-.997910	-.798281	-1.000000	-.900000
	50°	.994282	-.876922	-.956349	-.965755	-1.000000	-.900000
	54°	.803370	-.874276	-.970696	-.879118	-.999956	-.900000
	58°	.918954	-.998804	-.846669	-.949755	-.999998	-.900000
	62°	.933364	-.999779	-.869990	-.832217	-.999965	-.900000
	66°	.981095	-.999690	-.950545	-.987007	-.999954	-.900000
	70°	.973553	-.999167	-.992470	-.903555	-.999993	-.900000
	74°	.984492	-.999982	-.803537	-.983643	-1.000000	-.900000
	78°	.978546	-.999987	-.912627	-.987197	-.999981	-.900000
	82°	.811414	-.999151	-.986929	-.952538	-.997555	-.900000
	86°	.999313	-.999977	-.988125	-.999997	-.889841	-.900000
	90°	.686221	-.998458	-.808917	-.928907	-1.000000	-.900000
# 2	2°	-.992802	.990443	-.999723	-.997877	-.997283	-.900000
	6°	-.997078	.817989	-.915205	-.994845	-.838653	-.900000
	10°	-.971630	.797050	-.756709	-.999337	-.994652	-.900000
	14°	-.999387	.825298	-.807362	-.999642	-.975507	-.900000
	18°	-.991646	.999646	-.977790	-.999999	-.999993	-.900000
	22°	-.955870	.997681	-.920137	-1.000000	-.999997	-.900001
	26°	-.918271	.995830	-.996747	-.999998	-.999880	-.900000
	30°	-.987983	.937573	-.997842	-.999773	-.996875	-.900000
	34°	-.916350	.791377	-.779447	-.999538	-.999944	-.900000
	38°	-.851254	.970037	-.893282	-.998703	-1.000000	-.900001
	42°	-.984327	.862283	-.974040	-.946263	-1.000000	-.900001

Table A.22 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 2	46°	-.996537	.960025	-.965161	-.998185	-.999986	-.900000
	50°	-.997711	.733823	-.996747	-.993954	-.870074	-.900000
	54°	-.999910	.966288	-.999612	-.994575	-.999979	-.900000
	58°	-.999635	.866920	-.999982	-.982103	-.722205	-.900000
	62°	-.978631	.868007	-.999995	-.999893	-.720144	-.900000
	66°	-.999796	.995340	-.999996	-.999336	-.998177	-.900000
	70°	-.999873	.987252	-.999999	-.999384	-.844463	-.900000
	74°	-.999229	.909962	-.999998	-.999249	-.874571	-.900000
	78°	-.999789	.867122	-.999811	-.999501	-.999559	-.900000
	82°	-.967441	.997295	-.999933	-.997766	-1.000000	-.900000
	86°	-.993022	.953175	-.999605	-.826956	-.867735	-.900000
	90°	-.997356	.981651	-.998491	-.996812	-.998437	-.900000
# 3	2°	-.974600	-.823618	.976822	-.843574	-.999995	-.900000
	6°	-.998261	-.998931	.979501	-.994677	-.997535	-.900000
	10°	-.995006	-.994805	.985379	-.999636	-.999997	-.900000
	14°	-.999918	-.998550	.991597	-.997446	-.999994	-.900000
	18°	-.999816	-.994972	.987015	-.987984	-.999801	-.900000
	22°	-.961766	-.787514	.745706	-.999340	-.999996	-.900000
	26°	-.967549	-.857575	.997377	-.822259	-1.000000	-.900000
	30°	-.990898	-.859410	.996976	-.999450	-.999652	-.900000
	34°	-.953101	-.996284	.883456	-.990623	-.883695	-.900000
	38°	-.987346	-.812129	.844650	-.998909	-.995743	-.900000
	42°	-.981845	-.930379	.860393	-.987504	-.999979	-.900000
	46°	-.999955	-.967476	.983427	-.982338	-.999998	-.900000
	50°	-.999156	-.997630	.932240	-.999329	-.985038	-.900000
	54°	-.997817	-.844110	.855307	-1.000000	-.992067	-.900000
	58°	-.842085	-.990639	.947362	-1.000000	-.989331	-.900000
	62°	-.825113	-.999909	.999570	-.999999	-.999997	-.900000
	66°	-.982462	-.992096	.986742	-.977314	-1.000000	-.900000
	70°	-.995902	-.999961	.924119	-.972354	-.768931	-.900000
	74°	-.909708	-1.000000	.876102	-.976754	-.804482	-.900000
	78°	-.999542	-.999975	.939408	-.997918	-.943027	-.900000
	82°	-.924713	-.997202	.883745	-.963108	-.999473	-.900000
	86°	-.978304	-.946952	.822331	-.998225	-.978707	-.900000
	90°	-.993711	-.966545	.954937	-.984891	-.964367	-.900000
# 4	2°	-.996949	-.989252	-.999929	.999700	-.950408	-.899999
	6°	-.828391	-.995324	-.997641	.763427	-.838505	-.900000
	10°	-.944452	-.976511	-.999131	.955739	-.926384	-.900000

Table A.22 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 4	14°	-.997367	-.805530	-.999335	.997822	-.999997	-.900000
	18°	-.925217	-.999715	-.963322	.996157	-.999925	-.900000
	22°	-.809822	-.989329	-.956204	.847339	-.999999	-.900000
	26°	-.875641	-.999988	-.863852	.969752	-.999853	-.900000
	30°	-.765179	-.890192	-.997246	.975032	-.999780	-.899999
	34°	-.767643	-.992265	-.999640	.982134	-.999992	-.900000
	38°	-.999963	-.999647	-.991364	.993921	-.856984	-.900000
	42°	-.902223	-.940762	-.999280	.735583	-.819877	-.900000
	46°	-.924836	-.870659	-.999442	.970625	-.985726	-.900000
	50°	-.989429	-.994887	-.964916	.996443	-.999994	-.900000
	54°	-.907506	-.993402	-.955577	.997586	-1.000000	-.900000
	58°	-.996544	-.859246	-.888070	.997421	-1.000000	-.900000
	62°	-.996439	-.995093	-.820970	.996299	-.999590	-.900000
	66°	-.991084	-.912267	-.994343	.826873	-1.000000	-.900000
	70°	-.974434	-.999916	-.998905	.994131	-.999045	-.900000
	74°	-.990956	-.999979	-.952277	.998029	-.999305	-.900000
	78°	-.997941	-.869524	-.999423	.997182	-.999970	-.900000
	82°	-.946329	-.987441	-.941570	.879264	-.996234	-.900000
	86°	-.849122	-.999774	-.954849	.978696	-.999992	-.900000
	90°	-.965005	-.866493	-.997550	.995571	-.999956	-.900000
# 5	2°	-.997154	-.958741	-.957543	-.452367	.260357	-.900000
	6°	-.999691	-.999106	-.912216	-.942250	.834564	-.900000
	10°	-.988557	-.976656	-.921912	-.904895	.988140	-.899999
	14°	-.995058	-.348516	-.999865	-.993243	.999078	-.900000
	18°	-.996028	-.991948	-.999991	-.999890	1.000000	-.900000
	22°	-.999168	-.993110	-.999993	-.998207	.999955	-.900001
	26°	-.998372	-.999909	-.974529	-1.000000	.998004	-.900000
	30°	-.761203	-.998649	-.993376	-.999996	.999989	-.900000
	34°	-.861947	-.999993	-.960162	-1.000000	.999999	-.900000
	38°	-.515372	-.994302	-.314703	-1.000000	.999239	-.900000
	42°	-.991863	-.998683	-.980904	-1.000000	.990933	-.900000
	*46°	-.999471	-.990610	-.992614	-.999996	.246751	-.900000
	50°	-.993532	-.997525	-.999411	-1.000000	1.000000	-.900000
	54°	-.861947	-.999993	-.960162	-1.000000	.999999	-.900000
	58°	-.998057	-.999998	-.999397	-1.000000	1.000000	-.900000
	62°	-.975507	-1.000000	.852386	-.999912	.969439	-.899999
	66°	-.998682	-1.000000	-.864579	-.999837	.998241	-.900000

Table A.22 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 5	70°	-.998952	-.999829	-.997735	-.976935	.999733	-.900000
	74°	-.999869	-.999992	-.988259	-.989886	.848886	-.900000
	78°	-.992540	-.999774	-.999147	-.895174	.973432	-.900000
	82°	-.983028	-.945592	-.999586	-.794227	.997874	-.900000
	*86°	-.998165	-.998968	-.906293	-.987298	-.967800	-.900001
	90°	-.998038	-.978303	-.999916	-.445248	.996515	-.900000
	94°	-.993731	-.997369	-.974286	-.999365	.839372	-.900000
	*98°	-.815431	-.995058	.533856	-.999421	-.307859	-.900000
	*102°	-.856885	-.995118	-.806852	-.999928	-.190097	-.900000
	106°	-.999995	-.991540	-.050976	-.999994	.986465	-.900000
	110°	-.999575	-.971793	-.559429	-.999919	.421465	-.900000
	114°	-.999023	-.999957	.138262	-1.000000	.999941	-.900000
	118°	-.975845	-.999943	-.968619	-1.000000	1.000000	-.900000
	122°	-.929695	-.999932	-.665322	-1.000000	1.000000	-.900000
	126°	-.972701	-.986047	-.937379	-1.000000	.999997	-.900000
	130°	-.565788	-.999988	-.137355	-1.000000	.996945	-.900000
	134°	-.993364	-.997942	-.949135	-.999995	.928810	-.900000
	138°	-.999925	-.999985	-.999537	-.426461	.999687	-.900000
	142°	-.999984	-.999968	-.994439	-.979896	.999997	-.900000
	146°	-.999509	-.999998	-.967931	-.953809	.999139	-.900000
	150°	-.999954	-.999986	-.996744	-.852211	.999996	-.899999
	154°	-.996387	-.962984	-.999994	.000606	.999766	-.900000
	158°	-.974868	-.738521	-1.000000	-.574708	.630586	-.900001
	162°	-.980194	-.986764	-1.000000	-.060468	.587155	-.900001
	166°	-.995489	-.842787	-.999990	-.815191	.994524	-.900001
	170°	-.997033	-.995727	-.998668	-.961027	.999162	-.900000
	174°	-.998125	-.985496	-.999672	-.958567	.999511	-.900000
	178°	-.971859	-1.000000	.996642	-.995972	.999860	-.899999
	182°	-.999908	-.999998	-.998432	-.989644	1.000000	-.900000
	186°	-.998667	-.999999	-.986319	-.999106	1.000000	-.900000
	190°	-.997627	-.999944	-.948568	-.999999	1.000000	-.900000
	194°	-.853167	-.996648	-.611821	-.999999	.999999	-.900000
	198°	-.999540	-.998467	-.992042	.751635	.999997	-.900000
	*202°	-.999576	-.995395	-.998483	.034242	.089563	-.900000
	*206°	-.998674	-.995452	-.999774	-.983330	.245123	-.900001
	210°	-.992167	-.997433	-.999373	-.999115	1.000000	-.900000

Table A.22 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 5	214°	-.999953	-.999993	-.994667	-.999756	1.000000	-.900000
	218°	-.999751	-.999968	-.999386	-.999999	1.000000	-.900000
	222°	-.999743	-.999995	-.999593	-1.000000	1.000000	-.900001
	226°	-.998819	-.999999	-.992033	-1.000000	1.000000	-.900000
	230°	-.997661	-.999946	-.987360	-1.000000	1.000000	-.900001
	234°	-.967569	-.999025	-.978088	-1.000000	.999996	-.900001
	238°	-.995163	-.947417	-.996422	-1.000000	.999987	-.900001
	242°	-.989763	-.996716	-.992095	-1.000000	.999996	-.900001
	246°	-.778840	-.910808	-.997287	-1.000000	.999994	-.900000
	250°	-.976430	.075629	-.999682	-1.000000	.999993	-.900001
	254°	-.999417	-.978802	-.869459	-1.000000	.997211	-.900001
	258°	-.941097	-.947786	-.876997	-1.000000	.999959	-.900000
	262°	-.933277	-.991904	-.993084	-1.000000	.999997	-.900000
	266°	-.463719	-.960676	-.986301	-1.000000	.996732	-.899999
	270°	-.982127	-.998328	-.998552	-1.000000	.999995	-.900000
	274°	-.996542	-.996839	-.996287	-1.000000	.999999	-.900000
	*278°	-.996242	-.997359	-.997528	-1.000000	-.870795	-.900000
	282°	-.859396	-.999869	-.965011	-1.000000	.842449	-.900000
	286°	-.916652	-.997492	-.997844	-1.000000	.830058	-.900001
	290°	-.842791	-.999963	-.997425	-.999994	.996560	-.900000
	294°	-.958834	-.999686	-.999934	-.999997	.999940	-.900001
	298°	-.999968	-1.000000	-.999919	-.999999	1.000000	-.900001
	302°	-.969279	-.999978	-.999975	-.999995	1.000000	-.900001
	306°	-.999559	-.999999	-.999595	-1.000000	1.000000	-.900001
	310°	-.999827	-.999999	-.999974	-1.000000	1.000000	-.900001
	314°	-.998754	-.999989	-.999283	-1.000000	1.000000	-.900000
	318°	-.998362	-.999999	-.998597	-1.000000	1.000000	-.900001
	322°	-.999192	-1.000000	-.998372	-1.000000	1.000000	-.900001
	326°	-.999656	-.999961	-.999474	-1.000000	1.000000	-.900001
	330°	-.998268	-.999182	-.988653	-.999998	.999989	-.900000
	334°	-.953813	-.999034	-.883351	-.999991	.999984	-.900000
	338°	-.997769	-.908052	-.999906	-.789747	.809522	-.900000
	342°	.643719	-.999666	-.026296	-.999234	.999998	-.900000
	346°	-.905112	-.999955	-.960425	-.999936	1.000000	-.900000
	350°	-.985565	-.998514	-.166773	-.732529	.999601	-.900000
	354°	-.949738	-.987284	-.941845	-.754298	.462953	-.899999
	*358°	-.997321	-.503617	-.984166	.592808	-.948542	-.900000

Table A.23 Recognition process used the neural network which trained using the features extracted from the projection vector in X and Y directions with 10° angular interval: (X, Y).

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 1	*5°	-.126324	-1.997550	-.121194	-1.557918	-1.924335	-1.806861
	*15°	-.190415	-1.479923	-1.810482	1.848007	-1.999985	-1.810271
	*25°	-1.678612	-1.911367	-1.916884	1.780425	.000685	-1.795529
	*35°	-.056122	.000694	-1.963676	-1.906210	-1.387410	-1.780206
	*45°	.169965	.081854	-1.972239	-1.998466	.325032	-1.786158
	55°	1.429423	-1.583598	-1.869093	-.553187	-1.852042	-1.765338
	*65°	-1.653313	-1.101888	-1.978525	.017665	-1.998561	-1.838903
	*75°	-.155197	-1.754443	-1.890379	1.533977	-1.999984	-1.810667
	85°	.437216	-.274903	.063645	-.039056	-1.797316	-1.793724
# 2	5°	-1.930848	1.987226	-1.941507	-1.103176	-1.837393	-1.784648
	15°	-1.998690	1.999639	-1.985143	-.103634	-1.999943	-1.806965
	25°	-1.943284	1.813597	-.131568	.020466	-1.931737	-1.789494
	35°	-1.781687	.533805	-1.456687	-.294219	-.912298	-1.780897
	*45°	-1.905149	-.001678	-1.977818	-.050265	.981762	-1.846451
	55°	-1.088685	1.952355	-1.379849	-1.999999	-.736709	-1.809539
	65°	-1.898287	1.997614	-1.882852	-1.999965	-1.991738	-1.798326
	75°	-.397565	1.356396	-.064868	-1.971457	-1.999084	-1.787718
	85°	-1.994720	1.865260	-.540059	-1.999758	-1.636024	-1.809633
# 3	5°	-1.944440	-1.996238	1.981173	-1.868461	-1.994260	-1.791506
	15°	-1.936403	-1.842888	1.905719	-1.867042	-1.999108	-1.792532
	25°	-1.982440	-1.993435	.883983	-1.959330	.013445	-1.787850
	35°	-1.993908	-1.922180	1.814954	-2.000000	-1.917038	-1.811349
	*45°	-1.999066	-.004901	-1.996331	-1.999910	-1.998702	-1.830426
	*55°	-1.796150	-.014072	-.819947	-1.996822	-1.996332	-1.832257
	65°	-1.970755	-1.969938	1.091444	-1.999894	-.069685	-1.811021
	75°	-1.815210	-1.965620	1.966996	-1.988695	-1.857323	-1.790511
	85°	-1.889108	-1.909515	1.970098	-1.853644	-1.927880	-1.795416

Table A.23 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node
		#1	#2	#3	#4	#5	Average
# 4	*5°	-1.982262	-1.999838	1.376641	-1.520546	-.372201	-1.796587
	15°	-1.703615	-.096086	-.862544	.005276	-.023839	-1.775567
	25°	-1.887539	-.140934	-1.823976	.246527	.392481	-1.713107
	*35°	-1.942465	.017136	-1.802776	-.185788	-1.983409	-1.790431
	45°	-1.949285	-1.910356	-1.603744	.000334	-.780116	-1.789948
	*55°	1.168708	-.127698	-1.997830	-1.447724	-1.525916	-1.767798
	65°	-1.730209	-.130850	-1.935591	-.001774	-.048701	-1.805418
	75°	.087460	-1.749989	-1.274163	1.527882	-1.949993	-1.730298
	85°	1.722976	-1.999467	-.399425	.231074	-1.657187	-1.817679
# 5	5°	.129196	-1.962875	-1.911876	-1.960037	1.976703	-1.814682
	15°	-1.999197	-1.988768	-1.992258	-1.942411	1.532007	-1.822121
	25°	-1.999823	-1.941091	-1.999880	-.043448	.631640	-1.827642
	*35°	-1.996686	-1.974245	-1.999351	-.000008	-.134317	-1.796051
	45°	-1.971473	-1.967519	-1.999886	-1.999812	.195591	-1.806786
	55°	-1.989793	-1.997879	-1.958741	-.452976	1.928581	-1.754556
	65°	-1.999272	-1.999998	-2.000000	-1.999992	1.997534	-1.910897
	75°	.003108	-1.999650	-1.967277	-1.999843	1.026845	-1.803101
	85°	-1.993015	-1.987723	-1.909279	-1.995886	-.003278	-1.789390
	95°	-1.999120	-.700139	-1.999856	-1.974354	-.054038	-1.827383
	*105°	-1.988923	.008025	-1.998576	-2.000000	-.023818	-1.793822
	115°	-1.999787	-1.984900	-.018148	-2.000000	1.987828	-1.813514
	125°	-1.996684	-1.971870	-1.998246	-2.000000	-.002426	-1.810305
	135°	-1.910618	-1.990524	-1.999763	-1.999966	1.986234	-1.800709
	145°	-1.954720	-1.751346	-1.994166	-1.998759	-.016060	-1.832738
	*155°	-1.246534	1.634718	-1.963591	-1.950825	-.000017	-1.728674
	165°	-1.505946	-1.996065	-.237297	-1.608585	.068132	-1.805970
	175°	-1.660562	-1.975091	-.477338	-1.952833	1.993339	-1.791910
	185°	-1.986295	-1.975024	-1.987335	-1.994501	1.678629	-1.798459

Table A.23 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 5	195°	-1.986708	-1.998484	-1.914314	-1.329383	1.949438	-1.723187
	205°	-1.622303	-.152834	-1.677059	-1.999886	1.999936	-1.824199
	215°	-1.997364	-1.860222	-1.931344	-1.948625	1.990564	-1.797467
	*225°	-1.983373	.356302	-1.999053	-2.000000	.214664	-1.858685
	235°	-1.995255	-1.982906	-1.998878	-1.999958	1.947937	-1.808141
	245°	-1.986368	-1.999958	-1.977039	-1.999968	1.993226	-1.803886
	255°	-1.939369	-1.973625	-.214044	-1.998621	1.890764	-1.778576
	265°	-1.965300	-1.784479	-.233148	-1.998943	-.060335	-1.810699
	275°	-1.921428	-1.566946	-1.947225	-1.999771	-.004119	-1.808294
	285°	-1.965172	-1.976043	-1.989762	-1.999972	-.007644	-1.807879
	295°	-1.956248	-1.992269	-1.972589	-1.999860	-.008936	-1.818191
	305°	-1.756044	-1.997360	-1.931232	-1.999998	1.243629	-1.790634
	315°	-1.983372	-1.632321	-1.998958	-2.000000	1.549094	-1.816961
	*325°	.059870	-1.957371	-1.957762	-.675203	.000541	-1.763459
	*335°	.023927	-1.995219	-.272217	-.732753	-.000160	-1.839813
	345°	-1.306231	-1.967557	-1.989285	-1.999120	.027990	-1.789447
	355°	.029929	-1.999537	-1.999960	-1.999010	.161869	-1.827428

Table A.24 Recognition process used the neural network which trained using the features extracted from the projection vector in X and Y directions with 6° angular interval: (X, Y).

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 1	3°	1.855878	-1.999099	-1.879420	-1.733209	-1.881069	-1.799059
	9°	1.829847	-1.959710	-1.812692	-1.957702	-1.900460	-1.800838
	15°	.211908	-1.966269	-1.885869	-1.965913	-1.931854	-1.798137
	21°	1.036394	-1.879109	-1.932745	-1.630157	-1.907036	-1.794726
	*27°	-1.332039	-1.618360	-1.814777	-1.922970	.012832	-1.798813
	*33°	-1.181405	-1.785379	-1.972348	-1.462763	-.053332	-1.792316
	*39°	-.101139	-.867319	-1.999457	-1.958686	.808075	-1.797274
	*45°	-.032263	-1.906587	-1.999009	-.755493	.003045	-1.793575
	*51°	-.065550	-1.897695	-1.993055	.016789	-1.844859	-1.803253
	57°	1.764583	-1.870890	-1.967579	-1.976060	-1.838830	-1.800515
	63°	-.198380	-1.963778	-1.929148	-1.943004	-.230806	-1.797370
	69°	1.905675	-1.910967	-1.957494	-1.960138	-1.275961	-1.806094
	75°	-.047786	-1.920738	-1.940371	-1.860346	-1.924426	-1.788208
	81°	.008149	-1.969818	-1.940187	-1.445141	-1.943220	-1.801549
	87°	-.070606	-1.999398	-1.565346	-1.790647	-1.994768	-1.802963
# 2	*3°	.013548	-.499294	-1.969037	-1.773247	-1.998919	-1.797142
	9°	-1.834768	1.898750	-1.948170	-1.828047	-1.912413	-1.803653
	15°	-.809565	1.916823	-1.975105	-1.632368	-1.314062	-1.796669
	21°	-1.997882	1.717006	-1.970800	-1.551180	-1.832895	-1.807095
	27°	-1.955929	1.865258	-1.840963	-1.781049	-1.663853	-1.794253
	33°	-1.835696	1.924891	-1.957375	-1.922622	-1.936227	-1.790992
	39°	-1.728661	1.902498	-1.437739	-1.979417	-1.999943	-1.799890
	45°	-1.243294	1.784993	-.378305	-1.996526	-.014077	-1.801037
	51°	-1.779056	1.436211	-1.987813	-1.999934	-.305079	-1.805653
	57°	-1.869545	1.905831	-1.956144	-1.999980	-1.907205	-1.798278
	63°	-1.869553	1.905808	-1.956144	-1.999980	-1.483035	-1.796471
	69°	-1.894277	.103228	-1.775707	-1.961751	.035517	-1.885020
	75°	-1.878511	1.907065	-1.760681	-1.956155	-1.968145	-1.799020
	81°	-1.996690	.204066	-1.995063	-1.378939	.014325	-1.806886
	87°	-1.910705	-.196801	-1.832539	-1.903178	-.076981	-1.802169

Table A.24 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 3	3°	-1.841888	-1.929135	.759306	-1.961839	-1.583542	-1.797510
	9°	-1.983707	-1.846947	-.108306	-1.941939	-1.943938	-1.799886
	15°	-1.719578	-1.895379	1.858527	-1.779761	-.409696	-1.801079
	*21°	-.000605	-1.948295	-1.640552	-1.786061	.987804	-1.796300
	27°	-.939003	-1.746891	1.397310	-1.983532	.078925	-1.810227
	33°	-1.999926	-1.200498	.034201	-1.624621	-.768063	-1.799509
	39°	-1.997218	-1.829935	1.828551	-1.991524	-1.442290	-1.804197
	*45°	-1.998751	-1.913842	-1.097611	-1.938686	-.671975	-1.795675
	51°	-1.980434	-1.975872	1.962640	-1.747715	-1.984879	-1.799744
	57°	-1.913610	-1.436345	.093894	-1.999452	-.020718	-1.798949
	63°	-1.276376	-1.707830	1.795509	-1.992932	-1.951801	-1.808970
	69°	-1.750105	-1.736589	1.765795	-1.584135	-1.854634	-1.808192
	75°	-1.954935	-1.846399	1.905509	-1.934467	-1.829708	-1.800420
	81°	-1.949617	-1.887135	.997285	-1.925131	-1.764221	-1.805946
	87°	-1.896366	-1.838244	.739159	-1.919162	-1.853794	-1.814280
# 4	3°	-1.853424	-1.999259	-1.974619	1.886643	-1.406065	-1.799622
	*9°	-1.623865	-1.589221	-1.956269	-.622159	-.930575	-1.801285
	15°	-.067042	-1.860023	-1.950481	.112684	-1.929598	-1.799300
	21°	.016380	-1.826562	-1.895186	1.662363	-1.853116	-1.793920
	*27°	-1.258227	-1.655492	-1.890505	-1.587777	.100169	-1.798813
	*33°	-.974032	-1.363427	-1.693399	-1.693513	.000787	-1.794354
	39°	-1.613444	-1.823115	-1.965439	-.064394	-1.829345	-1.797567
	45°	-1.990239	-1.837309	-1.809184	.040489	.002676	-1.799118
	51°	-1.449416	-1.890750	-1.990385	.979966	-1.255930	-1.803254
	*57°	1.001121	-1.798219	-1.996872	-1.736373	-.440679	-1.795182
	*63°	-1.705154	-1.968375	-1.989522	-1.400809	.820332	-1.797370
	69°	-1.513779	-1.922439	-1.985961	-.015405	-1.990062	-1.811786
	75°	-.609327	-1.996519	-1.948748	1.986482	-1.994115	-1.789967
	81°	-1.183017	-1.999608	-1.961426	1.572375	-1.834180	-1.802046
	87°	-1.923710	-1.999871	-1.858951	1.972114	-1.934455	-1.804324

Table A.24 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 5	*3°	-1.880512	-1.997098	-.316519	-1.038057	-.929331	-1.803012
	9°	-1.981565	-1.887816	-1.668639	-1.959809	-.107468	-1.807754
	*15°	.016058	-1.905533	-1.941192	-1.953902	-.015554	-1.803983
	21°	-1.879885	-1.997405	-1.994420	-1.948330	-.045319	-1.793621
	27°	-1.817456	-1.996844	-1.999016	-1.988007	1.895709	-1.845622
	33°	-1.769723	-1.999192	-1.999472	.081079	1.954580	-1.795583
	39°	-1.771427	-1.999247	-1.998912	-1.994318	1.978475	-1.790212
	*45°	-.174672	-1.998319	-1.805456	-.000013	-.000315	-1.808425
	51°	-1.998764	-1.997094	-1.993249	-.001885	-.000003	-1.811754
	57°	-1.998503	-1.999998	-1.999134	-1.960262	.034189	-1.787746
	63°	-1.987318	-1.999998	-1.980450	-1.967393	2.000000	-1.850785
	69°	-1.901212	-1.999924	-1.915381	-1.980420	1.999996	-1.894372
	75°	-1.980633	-1.999831	-.076820	-1.874295	1.995950	-1.802648
	81°	-1.994101	-1.999519	-1.999660	-1.624084	2.000000	-1.802240
	87°	-1.977765	-.266458	-1.983208	-1.974176	-.130888	-1.798992
	*93°	.005623	-1.992257	-1.927511	-1.814838	-.018763	-1.799058
	99°	-1.808304	-1.563420	-1.999198	-1.996512	.044878	-1.804703
	*105°	.227319	.001202	-1.999021	-1.976266	-.026618	-1.800043
	111°	-1.770428	-1.998780	-1.999021	-1.976279	1.973382	-1.894629
	117°	-1.263769	-1.998131	-1.991868	-1.973385	1.697343	-1.846286
	123°	-1.922895	-1.806779	-1.998230	-1.902646	1.904592	-1.792260
	129°	-1.893731	-1.998917	-1.999716	-1.992614	1.972239	-1.796513
	135°	-1.978955	-1.983699	-1.997325	-1.999990	.008073	-1.799073
	141°	-1.897345	-1.997907	-1.999023	-1.999994	1.999918	-1.799306
	*147°	-1.996711	.000057	-1.972560	-1.999923	-.000002	-1.799270
	153°	-1.774523	-1.999944	-1.990283	-1.999770	1.999957	-1.800127
	*159°	-1.999548	.000810	-1.998481	-1.991681	-.000055	-1.800950
	165°	-.117459	-1.998669	-1.896754	-1.932366	-.095295	-1.797062
	171°	-1.934497	-1.980436	-1.997083	.046744	1.982538	-1.850393
	177°	-1.991987	-1.999221	-1.953074	.148083	1.894740	-1.812767

Table A.24 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 5	183°	-1.927071	-1.999561	-1.848743	-1.997098	1.869972	-1.813354
	189°	-1.971019	-1.985697	-1.996776	-1.903055	1.999976	-1.813023
	195°	-1.999029	-1.982430	-1.930269	-1.738753	1.125990	-1.785768
	201°	-1.999191	-1.912167	-1.837368	-1.973683	1.999048	-1.852221
	207°	-1.542679	-1.999272	-1.995039	-1.987085	2.000000	-1.841288
	213°	-1.998072	-1.995289	-1.998205	-1.849669	1.999995	-1.851275
	219°	-1.994354	-1.999979	-1.999303	-1.983380	2.000000	-1.794700
	225°	-1.899172	-1.999582	-1.999933	-1.997640	1.999989	-1.792712
	231°	-1.899172	-1.999582	-1.999933	-1.997640	1.999989	-1.788872
	237°	-.097395	-1.999507	-1.744259	-1.998973	1.999497	-1.799763
	243°	-1.912711	-1.031697	-1.269821	-1.986465	1.348254	-1.810762
	249°	-1.990473	-1.787907	-1.999666	-1.999948	1.975862	-1.820793
	255°	-1.992529	-1.999518	.019536	-1.998314	.047049	-1.801673
	261°	-1.869300	-1.999345	-.425085	-1.987061	1.884948	-1.796405
	267°	-1.179186	-1.999182	-1.385233	-1.998388	1.995955	-1.798026
	273°	-1.917519	-1.868599	-1.892804	-1.999787	1.733114	-1.798000
	279°	-1.891341	-1.999954	-1.666302	-1.999897	1.999103	-1.795886
	285°	-1.863306	.000048	-1.991966	-1.999771	.002185	-1.793666
	291°	-1.862578	-1.999982	-1.996236	-1.999732	1.999999	-1.783224
	297°	-1.641432	-1.995490	-1.994709	.016281	1.999956	-1.790282
	303°	-1.675588	-1.999713	-1.975756	-1.975920	1.999991	-1.802944
	309°	-1.991190	-1.997492	-1.999865	-1.989089	1.999997	-1.798445
	315°	-1.899172	-1.999542	-1.999933	-1.997640	1.999989	-1.882293
	321°	-1.845097	-1.999922	-1.999087	-1.888265	1.919912	-1.792422
	*327°	-1.957138	-1.998632	.036755	.010167	-.047909	-1.804894
	333°	-1.803132	-1.999983	-1.998531	-1.988715	1.983312	-1.814939
	339°	.010335	-1.869330	-1.997678	-1.978335	1.943573	-1.791889
	345°	-1.994324	-1.976815	-.017262	-1.981604	1.682207	-1.800785
	351°	-1.999125	-1.999825	-1.915016	-1.443273	-.083695	-1.799626
	*357°	.152868	-1.999027	-1.753153	-1.995893	-.118765	-1.800943

Table A.25 Recognition process used the neural network which trained using the features extracted from the projection vector in X and Y directions with 4^o angular interval: (X, Y).

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 1	2°	-.156882	-1.844559	-1.995037	-1.992989	-1.824345	-1.800000
	6°	-.156775	-1.959004	-1.992815	-1.981709	-1.906989	-1.800000
	10°	-.040672	-1.980470	-1.994932	-1.986958	-1.923898	-1.800000
	14°	-.051394	-1.940801	-1.861515	-1.985280	-1.936409	-1.800000
	18°	-.013556	-1.947583	-1.932102	-1.923693	-.688396	-1.800000
	22°	-.142517	-1.877492	-1.987836	-1.880538	-1.945447	-1.800001
	26°	-.070180	-1.878083	-1.974689	-1.870037	-1.999346	-1.800000
	30°	-.125618	-1.986975	-1.992468	-1.872783	-1.983762	-1.799999
	34°	1.900231	-1.994244	-1.993168	-1.998455	-1.983850	-1.799999
	38°	1.793924	-1.999315	-1.993303	-1.876663	-1.982410	-1.800000
	42°	-.115293	-1.999207	-1.997043	-1.897331	-1.844059	-1.799999
	46°	1.922814	-1.978685	-1.900064	-1.893435	-1.999822	-1.799999
	50°	1.650906	-1.850583	-1.913484	-1.851920	-1.999704	-1.800000
	54°	-.004908	-1.874458	-1.984926	-1.965557	-1.999812	-1.800000
	58°	-.053786	-1.940320	-1.959519	-1.906274	-1.997402	-1.800000
	62°	-.160537	-1.995113	-1.991706	-1.936071	-1.834427	-1.800000
	66°	1.014536	-1.998643	-1.895306	.031914	-1.966505	-1.800000
	70°	-.021250	-1.999855	-1.959004	-1.963579	-1.910477	-1.800000
	74°	.280676	-1.891898	-1.983325	-1.880204	-1.990802	-1.800000
	78°	-.073435	-1.989433	-1.940950	-1.973054	-1.814346	-1.800000
	82°	1.854522	-1.999292	-1.887562	-1.961274	-1.991878	-1.800000
	86°	-.146846	-1.953659	-1.895468	-1.972393	-1.988471	-1.800000
	90°	-.108325	-1.904208	-1.933812	-1.866593	-1.977612	-1.799999
# 2	2°	-1.986210	1.683423	-1.926081	-1.996811	-1.827019	-1.800000
	6°	-1.985235	1.428563	-1.816672	-1.999059	-1.844568	-1.800000
	10°	-1.922605	1.883919	-1.999050	-1.999416	-1.846688	-1.800000
	14°	-1.888716	1.860710	-1.999521	-1.997980	-1.855605	-1.800000
	18°	-1.817018	1.938296	-1.918280	-1.907182	-1.999999	-1.800000
	22°	-1.994147	1.895213	-1.880242	-1.998184	-1.913276	-1.799999
	26°	-1.993164	1.782074	-1.787521	-1.939864	-1.999511	-1.799999
	30°	-1.975625	1.823738	-1.780181	-1.913617	-1.998366	-1.799999
	34°	-1.983055	1.995907	-1.948427	-1.844243	-1.995034	-1.800000
	38°	-1.947294	1.774825	-1.980793	-1.999683	-1.529290	-1.800000
	42°	-1.988865	1.998692	-1.927561	-1.833965	-1.999517	-1.800000

Table A.25 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node
		#1	#2	#3	#4	#5	Average
# 2	46°	-1.895423	1.999891	-1.992730	-1.999504	-1.847500	-1.800000
	50°	-1.927303	1.999991	-1.881087	-1.995346	-1.999093	-1.800000
	54°	-1.879248	1.999284	-1.844127	-1.951830	-1.129495	-1.800000
	58°	-1.975845	1.999961	-1.830825	-1.996626	-1.687783	-1.800000
	62°	-1.879215	1.833620	-1.960894	-1.999888	-1.927948	-1.800000
	66°	-1.921750	1.931940	-1.896887	-1.999975	-1.862719	-1.799999
	70°	-1.865478	1.863475	-1.986524	-1.999978	-1.569854	-1.800000
	74°	-1.976006	1.920125	-1.875199	-1.876041	-1.666339	-1.800000
	78°	-1.985413	1.988572	-1.947420	-1.943721	-1.956527	-1.800000
	82°	-1.998208	1.969545	-1.969471	-1.999464	-1.901032	-1.800000
	86°	-1.884268	-.151026	-1.978765	-1.922564	-1.919083	-1.800000
	90°	-1.962804	1.490433	-1.737720	-1.997968	-1.985728	-1.799999
# 3	2°	-1.892516	-1.909473	1.890582	-1.999470	-1.999295	-1.799999
	6°	-1.997792	-1.928311	1.797999	-1.937674	-1.999928	-1.800000
	10°	-1.999147	-1.881313	1.879013	-1.891894	-1.999945	-1.799999
	14°	-1.999301	-1.781818	1.802205	-1.930747	-1.994140	-1.800000
	18°	-1.934025	-1.736010	1.766874	-1.854614	-1.999576	-1.800000
	22°	-1.955568	-1.979312	1.781231	-1.907628	-1.901149	-1.800000
	26°	-1.999134	-1.944723	1.809640	-1.884327	-1.977778	-1.800000
	30°	-1.915313	-1.999753	1.708577	-1.994915	-1.998957	-1.799999
	34°	-1.999320	-1.901237	1.869827	-1.998655	-1.962635	-1.800000
	38°	-1.992376	-1.993884	1.822220	-1.996432	-1.739046	-1.799999
	42°	-1.995507	-1.938301	1.985437	-1.998769	-1.904828	-1.800000
	46°	-1.999972	-1.878790	1.876094	-1.908389	-1.872481	-1.800000
	50°	-1.999972	-1.999442	1.834954	-1.984458	-1.923207	-1.800000
	54°	-1.984822	-1.806696	1.702898	-1.999383	-1.448670	-1.800000
	58°	-1.875726	-1.879583	1.915448	-1.967225	-1.998597	-1.800000
	62°	-1.895652	-1.867250	1.822387	-1.864242	-1.999653	-1.800000
	66°	-1.881447	-1.999960	1.991124	-1.994825	-1.995057	-1.800000
	70°	-1.898631	-1.999954	1.994507	-1.960720	-1.999894	-1.800000
	74°	-1.893545	-1.982918	1.833928	-1.899185	-1.992165	-1.800000
	78°	-1.951672	-1.999796	1.949968	-1.880404	-1.842844	-1.799999
	82°	-1.888960	-1.999100	1.991693	-1.999487	-1.998221	-1.800000
	86°	-1.888960	-1.999129	1.991585	-1.999487	-1.998221	-1.800000
	90°	-1.959581	-1.990721	1.961060	-1.992552	-1.999007	-1.800000
# 4	2°	-1.913746	-1.911808	-1.943129	-.171073	-1.917772	-1.799999
	6°	-1.871829	-1.904795	-1.957327	-.023854	-1.996416	-1.800000
	10°	-1.855353	-1.968512	-1.998419	-.011993	-1.959460	-1.800001

Table A.25 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node
		#1	#2	#3	#4	#5	Average
# 4	14°	-1.903438	-1.984940	-1.865339	.887998	-1.999996	-1.800000
	18°	-1.995070	-1.903755	-1.997041	.933950	-1.119159	-1.800000
	22°	-1.897202	-1.996454	-1.915886	1.512288	-1.998325	-1.800001
	26°	-1.917549	-1.985331	-1.997766	1.952702	-1.990859	-1.800000
	30°	-1.934367	-1.883410	-1.949586	-.063806	-1.807019	-1.800000
	34°	-1.966590	-1.992717	-1.993259	1.432181	-1.814849	-1.799999
	38°	-1.980421	-1.964958	-1.999314	1.340956	-1.921697	-1.800000
	42°	-1.915347	-1.969455	-1.999617	1.829104	-1.991948	-1.799999
	46°	-1.899399	-1.982996	-1.991771	-.107628	-1.921040	-1.800000
	50°	-1.903002	-1.993828	-1.998476	1.923492	-1.958481	-1.799999
	54°	-1.846320	-1.998516	-1.914147	-.156641	-1.984457	-1.800000
	58°	-1.966967	-1.927036	-1.994962	-.112186	-1.947674	-1.800000
	62°	-1.977024	-1.836283	-1.998112	-.014128	-1.869652	-1.800000
	66°	-1.860369	-1.992188	-1.975823	1.495913	-1.991256	-1.800000
	70°	-1.988516	-1.950322	-1.932426	-.008140	-1.447112	-1.800000
	74°	-1.581596	-1.991241	-1.880694	-.127148	-1.948938	-1.800000
	78°	-1.936970	-1.999991	-1.941811	1.032130	-1.822722	-1.800000
	82°	-1.831294	-1.999959	-1.830633	1.988520	-1.925636	-1.800000
	86°	-1.940582	-1.999864	-1.924326	-.031897	-1.985663	-1.800000
	90°	-1.903028	-1.928429	-1.895471	.114685	-1.997710	-1.800000
# 5	2°	-1.833067	-1.872426	-1.998889	-1.998112	-.004883	-1.800000
	6°	-1.998212	-1.994465	-1.994145	-1.904067	-.002387	-1.800000
	*10°	-1.982165	.018541	-1.916027	-1.983501	-.009964	-1.800000
	14°	-1.995224	-1.978716	-1.980122	-1.940598	.578535	-1.800001
	18°	-1.998918	-1.996089	-1.997112	-1.965482	1.998736	-1.800001
	22°	-1.996492	-1.949247	-1.998660	-1.975554	1.805631	-1.800001
	26°	-1.700371	-1.999942	-1.991506	-1.945496	.000155	-1.800000
	30°	-1.944193	-1.998198	-1.995475	-1.439052	-.000007	-1.800000
	34°	-1.989732	-1.998214	-1.999277	-1.982837	1.550056	-1.800000
	38°	-1.471536	-1.997491	-1.999004	-1.999484	1.201222	-1.800000
	42°	-1.994917	-.349001	-1.998837	-1.998593	-.000102	-1.800000
	46°	-1.994921	-1.983579	-1.998837	-1.335334	-.000102	-1.800000
	*50°	-1.998578	-1.999990	-1.999759	.004153	.000000	-1.800151
	54°	-1.999628	-1.996971	-1.997841	-1.946054	1.986437	-1.800000
	58°	-1.999629	-1.992739	-1.781713	-1.902339	1.176747	-1.800001
	62°	-1.899293	-1.998518	-1.976163	-1.985821	1.996427	-1.800001
	66°	-1.939056	-1.999954	-1.799372	-1.959666	.051626	-1.800000

Table A.25 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 5	70°	-1.879580	-1.997628	-1.832624	-1.549764	.052350	-1.800000
	*74°	.102097	-1.996804	-1.744262	.070790	-.437228	-1.800000
	*78°	.049195	-1.997775	-1.886830	-1.979829	-.021497	-1.800000
	82°	-1.867269	-1.998299	-1.989705	-1.995361	-.027152	-1.800000
	*86°	-1.793758	.013080	-1.846463	-1.996581	-.214118	-1.800000
	90°	-1.777519	-.945215	-1.436538	-1.988903	-.021750	-1.800000
	94°	-1.968935	-1.764291	-1.966427	-1.985223	-.027779	-1.800000
	*98°	-1.987217	.005835	-1.943040	-1.661844	-.029989	-1.799999
	102°	-1.973011	-1.956996	-1.976037	-1.998774	-.086468	-1.800000
	106°	-1.998738	-1.544978	-1.997512	-1.998335	1.182240	-1.800000
	110°	-1.999532	-1.900904	-1.889804	-1.999781	1.997906	-1.800001
	114°	-1.998682	-1.999174	-1.997366	-1.955063	.000414	-1.800000
	118°	-1.998465	-1.991955	-1.026289	-1.998101	1.768907	-1.800000
	122°	-1.995187	-1.999655	-1.983976	-1.994738	1.995889	-1.800000
	126°	-1.932611	-1.984637	-1.999865	-1.999688	.000003	-1.800000
	130°	-1.954373	-1.985349	-1.999472	-1.999571	1.999885	-1.799999
	134°	-1.986691	-1.982831	-1.997896	-1.999011	1.999886	-1.799999
	138°	-1.995081	-1.969431	-1.997869	-1.995683	1.998915	-1.799999
	142°	-1.996579	-1.999867	-1.993772	-1.996835	.004068	-1.800000
	146°	-1.993951	-1.991953	-1.963873	-1.895169	.225127	-1.800000
	150°	-1.994756	-1.999458	-1.972157	-1.989982	.003384	-1.800000
	154°	-1.994309	-1.999639	-1.817728	-1.553769	1.970029	-1.800000
	158°	-1.995385	-.183343	-1.836959	-1.995551	.173417	-1.800000
	162°	-1.938243	-1.969263	-1.912351	-1.982634	.002333	-1.800000
	166°	-1.995832	-1.974978	-1.762122	-1.969521	1.054693	-1.800000
	170°	-1.995471	-1.999901	-1.960612	-1.976644	-.002166	-1.800000
	174°	-1.995666	-1.991172	-1.940808	-1.419515	-.030992	-1.800000
	178°	-1.559553	-1.937522	-1.925494	-1.999210	1.993224	-1.800000
	182°	-1.784351	-1.996514	-1.998743	-1.998847	1.993176	-1.800000
	186°	-1.999479	-1.242835	-1.998090	-1.968154	1.979101	-1.800000
	190°	-1.992919	-1.880500	-1.821915	-1.998590	1.494325	-1.800000
	194°	-1.997720	-1.921816	-1.982654	-1.969694	1.994663	-1.800000
	198°	-1.999320	-1.666859	-1.981887	-1.886889	1.953379	-1.800000
	202°	-1.999765	-1.114660	-1.779406	-1.636317	.088775	-1.800000
	206°	-1.989534	-1.993855	-1.301267	-1.708211	1.180215	-1.800000
	210°	-1.992555	-1.996540	-1.898040	-1.757632	1.919014	-1.800000

Table A.25 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 5	214°	-1.978833	-1.996198	-1.987538	-1.988190	1.897784	-1.800000
	218°	-1.999141	-1.983112	-1.995721	-1.948324	1.978019	-1.800000
	222°	-1.813090	-.994036	-1.997826	-1.992683	-.000005	-1.800000
	226°	-1.974458	-.011227	-1.993489	-1.999884	-.053774	-1.800000
	*230°	-1.925564	-1.998567	.049214	-1.999718	-.090059	-1.799999
	234°	-1.966015	-1.997865	-1.927937	-1.999934	-.008430	-1.800000
	238°	-1.847334	-1.388421	-1.389173	-1.999962	1.748090	-1.800000
	242°	-1.936288	-1.842669	-1.837780	-1.999991	1.952574	-1.800000
	246°	-1.961268	-1.936503	-1.986617	-1.999870	1.932446	-1.800000
	*250°	-1.961268	-1.930726	.000373	-1.999674	-.053233	-1.800000
	254°	-1.984906	-1.876512	-.070843	-1.999618	-.060415	-1.800000
	258°	-1.813190	-1.961804	-1.907282	-1.999596	-.164159	-1.799999
	262°	-1.920346	-.673189	-1.993367	-1.999097	-.199279	-1.800000
	266°	-1.920346	-1.994003	-1.993172	-1.999097	-.199746	-1.800000
	270°	-1.978584	-1.999812	-.115473	-1.999953	-.047541	-1.800000
	274°	-1.995059	-1.359549	-1.717190	-1.999994	-.042694	-1.800000
	278°	-1.995059	-1.998299	-1.717202	-1.999994	-.042694	-1.800000
	282°	-1.953256	-1.999315	-1.808949	-1.999999	1.909842	-1.799999
	286°	-1.960967	-.230899	-1.934417	-1.999999	1.018285	-1.800000
	290°	-1.868117	-1.994011	-1.981778	-1.999986	1.996927	-1.800000
	294°	-1.795766	-1.999867	-1.974897	-1.999975	1.999883	-1.800000
	298°	-1.988659	-1.997318	-1.973644	-1.990817	1.975162	-1.800000
	302°	-1.964020	-1.996738	-1.943581	-1.994576	1.995117	-1.800000
	306°	-1.768610	-1.999832	-1.971551	-1.999409	1.995377	-1.800000
	310°	-1.790191	-1.998263	-1.987658	-1.999972	1.999188	-1.800000
	314°	-1.984725	-1.999995	-1.999071	-1.999937	1.999980	-1.800000
	318°	-1.969760	-1.988422	-1.982801	-1.999720	1.978688	-1.800000
	322°	-1.969760	.011474	-1.982405	-1.999720	.716178	-1.800000
	326°	-1.970502	-1.965786	-1.986074	-1.999801	.063492	-1.800001
	330°	-.671250	-.836741	-1.534068	-1.999994	-.007227	-1.800000
	334°	-1.989060	-1.935327	-1.993423	-1.999774	1.997868	-1.800000
	338°	-1.986226	-1.976258	-1.982345	-1.999965	1.990074	-1.800000
	342°	-1.948535	-1.405764	-1.985335	-1.152981	.044284	-1.800000
	346°	-1.951530	-1.999396	-1.717506	-1.999549	1.695613	-1.800000
	350°	-1.999965	-1.487859	-1.792535	-1.634776	-.088957	-1.800000
	354°	-1.987694	-1.811615	-1.994420	-1.544021	-.002706	-1.800000
	358°	.205086	-1.997982	-1.999868	-1.999734	1.853425	-1.800000

Table A.26 Classifying features using the network obtained from the experiment shown in Table A.5.

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 1	*2°	-.298411	-.197025	-.996856	-.014062	-.999999	-.900007
	4°	.951156	.254187	-.994015	-.540233	-.999955	-.899941
	*8°	-.797833	-.986246	-.990554	.770695	-.999987	-.899964
	*10°	-.377737	-.982961	-.998880	-.164386	-.999304	-.899971
	14°	.978734	-.338716	-.999489	-.967104	-.999644	-.899978
	16°	.907006	-.857789	-.999784	-.996804	-.998374	-.900020
	*20°	-.125100	-.998865	-.999983	-.869217	-.998444	-.900002
	22°	.347230	-.984859	-.998378	-.794927	-.999920	-.900024
	*26°	.056071	-.999853	-.979719	-.991458	-.999802	-.900039
	*28°	-.774157	-.999455	-.981174	-.635991	-.999901	-.900028
	32°	.932061	-.999818	-.999721	-.884950	-.999863	-.899999
	34°	.988797	-.808312	-.999744	-.826167	-.999686	-.899964
	38°	.900429	-.936377	-.990575	-.085002	-.992471	-.899961
	40°	.439441	-.999948	-.972606	-.957654	-.064901	-.899960
	44°	.970422	-.000214	-.999200	-.901330	-.999986	-.899992
	46°	.984248	-.998742	-.415835	-.463223	-.999997	-.899951
	50°	.576576	-.999884	-.915600	-.988913	.016665	-.900021
	52°	.987001	-.999628	-.910097	-.997662	.323173	-.899976
	56°	.855867	-.959917	-.858561	-.897583	-1.000000	-.900017
	58°	.999706	-.999916	-.507923	-.999338	-.999935	-.900039
	62°	.762723	-.956664	-.809161	-.882651	-.999995	-.899996
	64°	.748619	-.999794	-.112327	-.509124	-1.000000	-.899999
	68°	.910940	.719758	-.960938	-.698172	-.999998	-.899988
	70°	.581889	-.975165	-.997774	-.804292	-.999993	-.900043
	74°	.931867	-.999836	-.868071	-.975933	-.745686	-.900031
	76°	.879137	-.999897	-.600710	-.796029	-.860309	-.900027
	80°	.980644	-.996772	-.961383	-.913120	-.999995	-.900020
	82°	.969163	-.997945	-.952527	-.970870	-.999997	-.900004
	*86°	.006748	-.999464	-.995270	-.856065	-.999857	-.899993
	*88°	-.640130	-.999942	-.999073	.226649	-.999906	-.900068

Table A.26 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 2	2°	-.911113	.910200	-.690037	-.559378	-.999150	-.899988
	4°	-.292070	.999147	-.427800	-.752372	-.999999	-.899964
	8°	-.935069	.994515	-.811257	-.661109	-.999680	-.899972
	10°	-.955186	.428183	.068570	-.275989	-.999027	-.899971
	14°	-.864036	.727015	.775384	-.910313	-.997170	-.899986
	16°	-.893253	.930347	-.923568	-.998865	-.837205	-.899989
	20°	-.923546	.546618	-.999930	-.996792	-.294074	-.900023
	22°	-.948745	.869931	-.979240	-.988156	-.597796	-.899991
	26°	-.708667	.820999	-.997659	-.998234	-.988640	-.899996
	28°	.594131	.987802	.013900	-.981304	-.999998	-.899978
	32°	-.867629	.482397	-.995776	-.981166	-.998618	-.900006
	34°	-.919215	.952121	-.975177	-.994843	-.999985	-.900006
	*38°	-.998801	.196626	-.998603	-.999975	-.959817	-.900110
	*40°	-.988837	-.911376	-.984962	-.999995	-.501573	-.900074
	44°	-.882106	.999826	.262307	-.998946	-.999995	-.899914
	46°	-.995414	.998180	-.884155	-.954622	-.459771	-.899948
	50°	-.019054	.437545	-.951036	-.999943	.154366	-.900000
	*52°	-.983622	.259271	-.970394	-.999974	.984926	-.899986
	*56°	-.181244	.057136	-.955617	-.999951	.984012	-.899974
	*58°	-.999506	-.692552	-.996754	-.998999	-.991048	-.900027
	*62°	-.959688	-.055261	-.998774	-.999997	.625711	-.900053
	64°	-.924809	.532142	-.948249	-.999995	-.999424	-.900015
	68°	-.976876	.923392	-.999897	.048326	-.932248	-.899992
	*70°	-.990302	-.149844	-.994463	-.991366	-.819155	-.900010
	*74°	-.992923	.895787	-.999985	-.999992	.992600	-.900016
	76°	-.818311	.976684	-.999652	-.998495	-.989845	-.900000
	80°	-.944759	.999676	.627016	-.997552	-1.000000	-.899945
	82°	-.965546	.981577	-.985962	-.996516	-.999359	-.899989
	*86°	-.943286	-.870208	-.859408	-.930181	-.912780	-.899993
	*88°	-.993564	-.989627	.832413	-.930121	-.967733	-.900012

Table A.26 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 3	2°	-.963199	-.996037	.937181	-.978083	-.657042	-.900016
	4°	-.927708	-.981540	.933306	-.964698	-.951876	-.899994
	8°	-.955417	-.996740	.976544	-.913193	-.869316	-.899995
	10°	-.955417	-.996740	.976544	-.913193	-.869316	-.899995
	14°	-.977422	-.976928	.987553	-.908486	-.988686	-.899981
	16°	-.964511	-.941629	.981211	-.780938	-.999156	-.899946
	*20°	-.910348	-.108048	-.916498	-.996362	.958503	-.899986
	22°	-.579007	.138661	.817401	-.843280	-.999995	-.899982
	26°	-.789315	-.974297	.999482	-.925577	-.999986	-.899964
	28°	-.933910	-.821298	.997046	-.655539	-.999429	-.899985
	32°	-.984666	-.912851	.997968	-.955597	-.999978	-.899965
	*34°	-.999431	-.965463	-.909231	-.996087	-.966018	-.899997
	*38°	-.998674	-.985286	-.989042	-.999991	-.948232	-.900120
	40°	-.995686	-.277070	.921287	-.999915	-1.000000	-.900087
	*44°	-.998171	-.993177	-.159943	-.986341	.828362	-.899984
	*46°	-.998473	-.961242	.543703	-.940781	.979092	-.899985
	50°	-.999812	-.902886	.848174	-.999423	-.901998	-.899949
	52°	-.999965	-.999984	.998984	-.999974	-.969975	-.900013
	56°	-.999668	-.999967	.999727	-.999911	-.834803	-.899996
	*58°	.357392	.798215	.461481	-1.000000	-.885066	-.899990
	62°	-.970641	-.999658	.893027	-.999870	-.512507	-.900019
	*64°	-.982486	-.921098	-.122824	-.398517	-.549959	-.899986
	68°	-.797130	-.959508	.998283	-.461909	-.999817	-.899989
	*70°	-.999739	-.995511	-.985474	-.768465	.948076	-.900019
	74°	-.921586	-.983020	.956264	-.936084	.969598	-.899985
	76°	-.921586	-.983020	.956264	-.936084	-.969598	-.899985
	80°	-.872458	-.688406	.970012	-.992786	-.999878	-.900035
	82°	-.977348	-.993105	.954835	-.931495	-.789977	-.900007
	86°	-.939261	-.971332	.995795	-.844229	-.996602	-.899965
	88°	-.959816	-.964363	.916578	-.949686	-.949496	-.899997

Table A.26 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 4	2°	-.974241	-.974404	-.964956	.998174	-.999999	-.899964
	4°	-.994640	-.962213	-.779564	.999973	-.999984	-.899992
	8°	-.724292	-.853926	-.972594	.994238	-.999960	-.899958
	10°	-.881904	-.984209	-.993451	.997921	-.999999	-.899957
	*14°	.753090	-.993507	-.999690	.202094	-.998789	-.899975
	*16°	-.478784	-.492735	-.999997	-.411002	-.999562	-.900003
	*20°	-.125100	-.998865	-.999983	-.869217	-.998444	-.900002
	*22°	-.987412	-.999824	-.999739	.136426	-.990557	-.900001
	26°	-.995103	-.962404	-.967008	.980273	-.999958	-.900003
	*28°	-.947707	-.999996	-.999833	-.328795	.211385	-.900027
	32°	-.828475	.568325	-.999951	.841527	-.999537	-.899952
	34°	-.992908	.640564	-.576227	.993287	-1.000000	-.899946
	38°	-.956706	-.969816	-.999834	.999983	-.999999	-.900008
	40°	-.947315	-.998860	-.924498	.999893	-.999996	-.899973
	44°	.100104	-.931804	-.999971	.932841	-.999997	-.899989
	46°	-.989395	-.993861	-1.000000	.266990	-.990307	-.900028
	50°	-.992867	-.999094	-.999969	.999174	-.984620	-.900018
	52°	-.998976	-.973408	-.999661	.999875	-.999955	-.899993
	56°	-.698170	-.988792	-.999965	.397614	-.997836	-.900015
	*58°	-.826062	-.972945	-.999813	.060941	-.999972	-.900049
	62°	-.989143	-.940313	-.264480	.999437	-1.000000	-.899962
	64°	-.985594	-.938725	-.996017	.999166	-1.000000	-.899989
	68°	-.389647	-.887426	-.828136	.993882	-1.000000	-.899994
	70°	-.995050	-.978909	-.998880	.960741	-1.000000	-.900056
	*74°	-.933815	-1.000000	-.999994	-.773408	1.000000	-.900031
	76°	-.983212	-.999892	-.993991	.935085	-.326384	-.900012
	80°	-.970653	-.999998	-.928228	.984355	-.620280	-.899979
	*82°	.072480	-.999601	-.999836	-.151541	.603048	-.899992
	*86°	.327618	-.999067	-.997497	.289409	-.999993	-.899956
	88°	-.999278	-.999991	-.999846	.532291	-.999968	-.900197

Table A.26 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node
		#1	#2	#3	#4	#5	Average
# 5	2°	-.939569	-.999990	-.999998	-.928780	.997602	-.900034
	4°	-.903050	-1.000000	-.999501	-.994417	.931060	-.900028
	*8°	-.997472	-.999619	-.043696	-.927163	-.519292	-.899963
	10°	-.994924	-.999945	-.602475	-.771560	.999793	-.899965
	14°	-.999915	-.995131	-.997655	-.294051	.999892	-.899957
	16°	-.999729	-.991131	-.985080	.502362	.999066	-.899967
	20°	-.999995	-.999367	-.996988	.938067	1.000000	-.899957
	22°	-.922383	-.990324	-.582422	-1.000000	.999964	-.899987
	26°	-.981991	-.999783	.615851	-1.000000	1.000000	-.899996
	*28°	-.997095	-.999466	.619352	-.999993	-.393203	-.899984
	32°	-.980546	-.955175	-.207831	-1.000000	.999964	-.899943
	34°	-.999799	-.996370	.401040	-.999999	.999961	-.899983
	38°	-.875295	-.975112	-.196063	-1.000000	.999981	-.899901
	40°	-.998481	.243407	-.933486	-.984515	.999643	-.899932
	44°	-.998830	.155636	-.999811	-.991489	.999928	-.899971
	46°	-.999799	-.117788	-.999103	-.993015	.999897	-.899948
	50°	-.998819	-.997358	-.999783	-1.000000	1.000000	-.900052
	52°	-.998034	-.999949	-.999969	-.999315	1.000000	-.900037
	56°	-.996081	-.997813	-.920255	-1.000000	.999999	-.900006
	58°	-.990854	-.999996	-.999485	-1.000000	1.000000	-.900013
	62°	-.835058	-1.000000	-.999982	-1.000000	.999909	-.900110
	64°	-.999459	-1.000000	-.996371	-.999984	.705700	-.900099
	68°	-.999986	-1.000000	-.999829	-.945998	.977600	-.900066
	70°	-.999340	-.999996	-.999993	-.999894	.971731	-.900110
	74°	-.998233	-.999997	-.999708	-.995555	.990207	-.900045
	76°	-.973028	-.999999	-.999606	-.956638	.999765	-.900049
	80°	-.973028	-.999999	-.999606	-.956638	.999765	-.900049
	82°	-.950361	-.999978	-.903058	-.814853	.731126	-.900021
	*86°	-.985439	-.869394	-.934972	-.991628	-.960174	-.900016
	88°	-.997637	-.956203	-.713304	-.986235	.712246	-.899985

Table A.26 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 5	*92°	-.976954	-.971984	.582043	-.993241	-.963681	-.899980
	94°	-.998904	-.828049	.305525	-.815486	.373245	-.899972
	98°	-.999786	-.999658	-.839630	-.982619	.998036	-.900026
	100°	-.999322	-.999050	.268985	-.670015	.999624	-.899958
	104°	-.991730	-.990137	-.885934	-.992370	.906571	-.900000
	106°	-.999033	-.998577	-.954093	-.981584	.999998	-.900022
	110°	-.999997	-.999864	-.999983	-1.000000	1.000000	-.900138
	112°	-.998495	-.999950	-.715613	-.999999	.731401	-.900061
	116°	-.998200	-.999997	.868783	-.999819	.880594	-.900038
	118°	-.999784	-.999973	-.947791	-.999999	.999869	-.900066
	122°	-.999591	-.999957	-.996663	-1.000000	1.000000	-.900024
	124°	-.997394	-.999800	-.950858	-1.000000	.999978	-.900032
	128°	-.999023	-.762561	-.991387	-.999974	.996840	-.899971
	130°	-.999184	-.999784	-.976311	-.999999	.999997	-.899999
	*134°	-.986124	-.997804	-.997711	.937806	.759681	-.899973
	136°	-.994599	-.929541	-.999794	-.554207	.949937	-.899950
	140°	-.999548	-.988638	-1.000000	-.856471	.999999	-.900026
	142°	-.996559	-.966117	-.999855	-.999192	.998991	-.899973
	146°	-.998592	-.427627	-.999992	-.997538	.994595	-.900034
	148°	-.998649	-.995628	-.999771	-.891429	.343825	-.900061
	*152°	-.999960	-.999860	-.999977	-1.000000	-.980600	-.900221
	*154°	-.907247	-.965701	-.999969	-.999264	-.557615	-.900010
	*158°	-.992186	.921628	-.998991	-.998316	-.995244	-.900029
	*160°	-.735098	.676433	-.988538	-.996330	-.859439	-.900023
	164°	-.998520	-.944418	-.991853	-.991754	.999953	-.900015
	*166°	-.999977	.735034	-.145751	-.831908	.419103	-.899932
	170°	-.993915	-.996110	-.984751	.627775	.999762	-.899900
	172°	-.999981	-.999861	-.999708	-.741524	.999924	-.900001
	176°	-.999846	-.999980	-.669798	-.012103	.997803	-.899960
	178°	-.999666	-.999994	-.690567	-.999998	.256705	-.900053

Table A.26 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 5	182°	-.999992	-.941537	-.995348	-.952167	.997612	-.899974
	*184°	-.999360	-.869550	.137993	-.986045	.151216	-.899957
	188°	-.999330	-.999764	-.999497	-1.000000	.998897	-.900077
	190°	-.999993	-.999999	-.999986	-.955147	1.000000	-.900014
	194°	-.680915	-.020443	-.999969	-.999998	.997671	-.900008
	196°	-.734740	.217603	-.997231	-.999988	.872199	-.900007
	200°	-.927774	-.991599	-.999055	-1.000000	.997201	-.900033
	*202°	-.993477	-.989828	-.833334	-.999994	-.963599	-.900040
	206°	-.990731	-.998759	.017976	-.999897	.999939	-.899996
	208°	-.997793	-.980505	-.711474	-.999796	.999801	-.899984
	212°	-.999663	-.997072	-.538309	-.999573	.556190	-.900033
	214°	-.991245	-.926109	-.954727	-.937162	.651296	-.900000
	218°	-.991949	-.999496	-.997337	-.998978	.999992	-.900013
	220°	-.552278	-.988457	-.954836	-.999999	.999809	-.899994
	224°	-.957627	-.884593	-.990464	-.999997	.994163	-.900001
	226°	-1.000000	-.993163	-.999992	-.999999	.983840	-.900106
	230°	-1.000000	-.999556	-.999998	-1.000000	.994717	-.900082
	232°	-.999999	-.999853	-.999999	-1.000000	.999441	-.900088
	236°	-1.000000	-1.000000	-1.000000	-.999992	.999999	-.900117
	238°	-1.000000	-1.000000	-1.000000	-.999999	1.000000	-.900138
	242°	-.999976	-.999981	-.999701	-.996820	.998947	-.900024
	244°	-1.000000	-1.000000	-1.000000	-1.000000	.999998	-.900193
	248°	-.999386	-.999829	-.999663	-.999994	.999317	-.900058
	250°	-.998745	-.999579	-.999882	-.999993	.997447	-.900070
	254°	-.953687	-.999950	-.623755	-.990259	.997685	-.899992
	256°	-.979280	-.967177	-.995310	-.985086	.923286	-.899980
	260°	-.947743	-.979474	-.981698	-.995808	.651290	-.899968
	262°	-.993299	-.998965	-.999884	-.985445	.999815	-.900014
	266°	-.996550	-.943441	-.999994	-.997739	.999437	-.900017
	268°	-.996550	-.943441	-.999994	-.997739	.999437	-.900017

Table A.26 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 5	*272°	-.984617	.743329	-.999873	-.974877	.654580	-.899985
	*274°	-.984617	.743329	-.999873	-.974877	.654580	-.899985
	278°	-.980872	-.970273	-.898053	-.447456	.819050	-.899988
	280°	-.940758	-.996017	-.985440	-.976662	.968457	-.900007
	*284°	-.527181	.516739	-.741977	-.999997	-.997653	-.900041
	*286°	-.292930	.774137	-.323541	-.988528	-.980002	-.899998
	288°	-.995066	-.884829	-.989517	-.984104	.920740	-.899994
	*292°	-.999803	-.698363	-.231845	-.168527	-.999871	-.899975
	294°	-.455450	-.891333	-.996631	-1.000000	.965779	-.900025
	*298°	-.899883	-.877120	-.944286	-.999999	-.621617	-.900031
	302°	-.997935	-.997684	-.999466	-1.000000	.999168	-.900060
	304°	-.994132	-.999355	-.995848	-1.000000	.999007	-.900057
	308°	-.999446	-.933157	-.999978	-.999011	.999996	-.899989
	310°	-.998932	-.981061	-.999803	-.999994	.989571	-.900007
	314°	-.996262	-.848470	-.997114	-.999990	.999111	-.899958
	316°	-.998227	-.994759	-.964026	-.999580	.998886	-.900008
	320°	-.999808	-.999369	-.879110	-.957035	.880684	-.899958
	322°	-.995991	-.999980	-.939111	-.997261	.999851	-.899967
	326°	-.979907	-.999976	-.995562	-.999993	.999720	-.900029
	328°	-.999577	-.999990	-.986648	-.999787	.998424	-.899983
	*332°	-.691517	-.999825	-.906027	-.999349	-.454245	-.899988
	*334°	-.691517	-.999825	-.906027	-.999349	-.454245	-.899988
	*338°	-.903181	-.999329	-.999379	-.997556	-.737483	-.900020
	*340°	-.928950	-.998332	-.983243	-.921371	-.478997	-.899986
	344°	-.993478	-.997250	-.998328	-.995760	.999976	-.899936
	346°	-.947406	-.999996	-.999997	-.975210	.999967	-.899978
	350°	-.947406	-.999996	-.999997	-.975210	.999967	-.899978
	352°	-.831920	-1.000000	-.999936	-.990418	.999437	-.899943
	356°	-.953334	-.999999	-.999635	-.948940	.998589	-.899951
	358°	-.895535	-1.000000	-.999947	-.994974	.999498	-.900034

Table A.27 Classifying features using the network obtained from the experiment shown in Table A.9.

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 1	2°	.429216	-1.999100	-1.923702	-1.864607	-1.881069	-1.801549
	4°	-.144007	-1.999099	-1.923974	-.669304	-1.881069	-1.798395
	8°	-.078599	-1.933775	-1.985139	-1.996352	-1.552501	-1.801396
	10°	.028383	-1.954399	-1.808020	-1.957702	-1.900460	-1.802028
	14°	-.110552	-1.889759	-1.883465	-1.924485	-1.988670	-1.800441
	16°	1.849252	-1.966345	-1.883275	-1.965989	-1.932186	-1.796395
	*20°	-.340079	-1.959992	-1.993475	-1.977508	.107802	-1.795887
	22°	1.314062	-.510269	-1.990787	-1.959617	-1.755459	-1.794655
	*26°	-.508926	-1.992835	-1.994402	-1.866062	.680767	-1.802536
	28°	.905208	-1.861380	-1.647413	-1.934607	-1.781473	-1.796957
	*32°	-1.952908	-1.972357	-1.981374	-.220179	-.339349	-1.792662
	*34°	-1.685006	-1.860049	-1.994363	-.418766	.020392	-1.792206
	38°	-.124981	-1.454807	-1.981032	-.797118	-1.965187	-1.799037
	50°	-.159699	-1.740791	-1.990921	-1.651336	-1.741086	-1.797709
	44°	1.816392	-1.939613	-1.999860	-.368596	-.119231	-1.791191
	*46°	-.082492	-1.936107	-1.999460	-1.968799	.729714	-1.816500
	50°	-.174536	-1.905868	-1.984528	-1.897897	-1.850525	-1.805144
	*52°	-.047553	-1.907509	-1.971897	.169676	-1.939018	-1.802793
	56°	-.092993	-1.870929	-1.967538	-1.976060	-1.838811	-1.801174
	58°	.459006	-1.887416	-1.998245	-1.097217	-1.581425	-1.792410
	62°	1.791987	-1.963778	-1.935613	-1.944263	-1.175501	-1.797033
	64°	1.930214	-1.977926	-1.994580	-1.821883	-1.525615	-1.797146
	68°	1.935579	-1.871408	-1.995707	-1.980656	-1.775631	-1.806593
	70°	1.912515	-1.836339	-1.985814	-1.992541	-1.878899	-1.808020
	*74°	-.035892	-1.893287	-1.959551	.039890	.113542	-1.789902
	76°	-.020973	-1.920157	-1.958987	-1.008983	-1.978886	-1.788248
	*80°	-.027143	-1.941858	-1.939531	.155753	-1.989043	-1.802018
	82°	.782541	-1.987375	-1.885915	-1.965444	.050868	-1.802250
	86°	-.185606	-1.999050	-1.884072	-1.838204	-1.931772	-1.802200
	88°	-.140882	-1.999100	-1.923975	-1.688382	-1.881069	-1.806507

Table A.27 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 2	2°	-1.910705	-.196913	-1.912959	-1.903149	-1.985021	-1.799427
	4°	-1.867526	.495267	-1.817947	-1.862087	-1.985121	-1.797646
	8°	-.467730	1.614734	-1.817947	-1.862087	-1.896164	-1.797028
	10°	-1.956640	1.946273	-1.913599	-1.627907	-1.406679	-1.804954
	14°	-1.993496	1.935464	-1.905522	-1.841089	-1.588012	-1.797591
	16°	.001541	1.662827	-1.975261	-1.710558	-1.399353	-1.801265
	20°	-1.998284	1.885146	-1.965944	-1.053446	-1.632722	-1.797638
	22°	-1.995356	.084390	-1.945724	-1.689538	-.912055	-1.792586
	*26°	-1.957780	-.098697	-1.681369	-1.777589	-.020648	-1.795679
	28°	-1.913589	1.829624	-1.694143	-1.968192	-1.629301	-1.794253
	32°	-1.893180	.435534	-1.899835	-1.925222	.158872	-1.775461
	34°	-1.836157	1.890844	-1.977417	-1.975255	-1.910098	-1.796369
	*38°	-.825916	-.066984	-1.586006	-1.236226	.002544	-1.828706
	40°	-1.793208	1.982952	-1.537399	-1.999714	.002625	-1.807820
	*44°	-1.598858	-.021752	-.433053	-1.995858	.000009	-1.790348
	46°	-.077178	1.906248	-1.571126	-1.997386	-1.995977	-1.797779
	50°	-1.949251	1.871315	-1.952900	-1.999935	-1.371113	-1.799215
	52°	-1.954774	1.903960	-1.715378	-1.999773	-1.947419	-1.807222
	56°	-1.869553	-.094144	-1.956139	-1.999980	-1.733170	-1.802901
	58°	-1.869545	1.905831	-1.956144	-1.999980	-1.907205	-1.798278
	62°	-1.869553	.255377	-1.956144	-1.999980	.092795	-1.794286
	64°	-1.742840	.074770	-1.973915	-1.999974	-.586841	-1.800993
	68°	-1.739693	1.794650	-1.606930	-1.971084	.029606	-1.782406
	70°	-1.846268	1.522487	-1.421839	-1.932914	.022323	-1.884923
	74°	-1.818005	1.764675	-1.801397	-1.966178	-1.734303	-1.799431
	76°	-1.928008	1.844883	-1.892271	-1.955367	-1.678027	-1.805123
	80°	-1.998958	1.635619	-1.987740	-1.684440	.041007	-1.801213
	82°	-1.999101	1.511795	-1.988602	-1.721004	-1.949908	-1.804839
	86°	-1.995934	-.036236	-1.698611	-1.865720	-.068093	-1.802169
	*88°	-1.877998	-.099022	-1.905597	-1.888654	.059712	-1.796794

Table A.27 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node
		#1	#2	#3	#4	#5	Average
# 3	2°	-1.896756	-1.940064	1.589188	-1.933434	-1.870419	-1.797510
	4°	-1.841888	-1.929135	.759306	-1.961839	-1.583542	-1.797510
	*8°	-1.998259	-1.868434	-.952416	-1.996831	-.481298	-1.796195
	*10°	-1.999675	-1.882636	-.226730	-1.940043	-1.827666	-1.799886
	14°	-1.997830	-1.898268	1.419282	-1.761138	-1.918792	-1.801564
	16°	-.568697	-1.897770	1.942820	-1.912056	-.868402	-1.801323
	*20°	-.013308	-1.859512	-1.745970	-1.654025	-1.096653	-1.798230
	*22°	-1.997372	-1.937647	.480141	-1.941064	1.891133	-1.797181
	26°	-.552944	-1.806298	1.735717	-1.951804	-1.971649	-1.808791
	*28°	-1.243567	-1.857060	.673193	-1.993264	.760836	-1.800624
	32°	-1.886447	-1.916007	1.607310	-1.336043	-1.987387	-1.797832
	34°	-1.999923	-.614839	.049499	-1.675429	-1.072150	-1.800562
	*38°	-1.997674	.085628	-.379341	-1.941875	.882071	-1.824537
	40°	-1.816065	-1.770963	1.725276	-1.988692	-1.757099	-1.803513
	44°	-1.999852	-1.647760	.248452	-1.870326	-.345536	-1.797777
	46°	-1.999257	-1.622452	.007184	-1.954300	-.225754	-1.810762
	50°	-1.921340	-1.957597	1.916145	-1.954942	-1.787007	-1.797817
	52°	-1.954112	-1.947985	1.934084	-1.929304	-1.999345	-1.799745
	56°	-1.984078	-1.904726	1.215848	-1.998527	-1.986523	-1.804277
	58°	-1.887328	-1.930651	.488776	-1.976946	-1.764387	-1.799444
	62°	-1.951349	-1.842816	1.789944	-1.996890	-1.871245	-1.799282
	64°	-1.959916	-1.817936	1.961671	.002026	-1.981444	-1.802501
	68°	-1.750074	-1.736589	1.518146	-1.772569	-1.967279	-1.796658
	70°	-1.939799	-1.883859	1.984465	-1.115557	-.002736	-1.797121
	74°	-1.844913	-1.836931	1.846799	-1.882306	-1.983993	-1.800151
	76°	-1.923537	-1.886147	1.906761	-1.890554	-1.974323	-1.805644
	80°	-1.844913	-1.836931	1.846809	-1.882306	-1.985020	-1.805946
	82°	-1.967756	-1.904221	1.888438	-1.871198	-1.932000	-1.798656
	86°	-1.933016	-1.862183	1.820742	-1.861238	-1.959302	-1.814280
	88°	-1.844913	-1.836931	1.846809	-1.882306	-1.980401	-1.801519

Table A.27 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 4	2°	-1.821639	-1.999832	-1.974778	-.141361	-1.474199	-1.801731
	4°	-1.878338	-1.999139	-1.901026	1.876548	-1.888983	-1.799622
	8°	-1.623865	-1.997045	-1.956269	1.130139	-.930575	-1.801534
	10°	-1.370704	-.014018	-1.990780	-1.330514	-.119099	-1.800634
	14°	-1.458498	-1.102313	-1.841700	-.101507	-1.988591	-1.800276
	*16°	1.849850	-1.837764	-1.200710	1.541827	-1.989899	-1.796165
	20°	-1.750024	-1.995143	-1.995057	1.937781	-1.603299	-1.796518
	22°	-.256332	-1.963663	-1.971993	1.874791	-1.731013	-1.799159
	*26°	.321130	-1.619513	-1.975777	-1.716895	-.719660	-1.797116
	*28°	-.722671	-1.689557	-1.592256	-1.998822	-.899213	-1.796957
	*32°	-1.154817	-1.519386	-1.975335	-1.983332	.524665	-1.792987
	*34°	-1.633370	-1.929887	-1.997299	-.086686	.023653	-1.775651
	38°	-.856155	-1.737224	-1.980479	.397306	.002947	-1.800395
	40°	-1.767012	-1.885768	-1.871955	-.055991	-1.338933	-1.800447
	44°	-1.977442	-1.887582	-1.978262	1.987731	.035354	-1.791205
	46°	-1.983664	-1.889844	-1.973545	1.991869	-1.962943	-1.809704
	50°	-1.333503	-1.832956	-1.943927	1.948953	-1.370952	-1.805367
	52°	-.803757	-1.716450	-1.595754	.094189	-1.941500	-1.802761
	*56°	.963705	-1.740403	-1.993687	-1.967565	-.099824	-1.795342
	*58°	.606762	-1.844913	-1.995104	-1.672364	-.544336	-1.799154
	*62°	.446077	-1.870576	-1.988395	-1.807648	-.326265	-1.797146
	*64°	.262790	-1.944610	-1.994599	-1.869457	-.159654	-1.797146
	68°	-1.469050	-1.703957	-1.960528	-.094369	-1.968937	-1.811786
	*70°	.037480	-1.837098	-1.929703	-.023207	.008440	-1.808226
	74°	-.609327	-1.998138	-1.948748	1.986952	.005884	-1.794036
	76°	-1.770619	-1.999331	-1.924649	.265214	-1.985517	-1.797851
	80°	-.790613	-1.999821	-1.787868	1.993729	-1.995412	-1.802736
	82°	-.790619	-1.999821	-1.787869	1.993730	-.417833	-1.797186
	86°	-1.974966	-1.999527	-1.954298	1.741636	-1.556434	-1.804324
	88°	-1.953629	-1.999737	-1.957567	-.200125	-1.188468	-1.806506

Table A.27 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 5	*2°	-1.848697	-1.751009	-.365675	-1.098190	-.990758	-1.803012
	4°	-1.893459	-1.998021	-.369794	-1.289518	-.064946	-1.803011
	8°	-1.971081	-1.867998	-1.977353	-1.985191	-.029471	-1.807920
	10°	-1.977530	-1.996865	-1.969701	-1.380309	-.034902	-1.807371
	14°	-1.864819	-1.744519	-1.706710	-1.637738	-.479611	-1.791653
	16°	-1.983275	-1.754273	-1.862794	-1.944008	-.064479	-1.802977
	20°	-1.959981	-1.996755	-1.961266	-1.972609	-.413955	-1.804435
	22°	.118617	-1.997536	-1.998735	-1.997975	1.995627	-1.793899
	26°	-1.714465	-1.998775	-1.999007	-1.976278	-.026575	-1.797624
	28°	-1.762845	-1.995691	-1.999339	-.378234	-.053365	-1.810345
	32°	-1.769749	-1.999192	-1.999472	-.202123	1.951492	-1.802292
	34°	-1.906877	-1.999214	-1.999872	-.313489	.035214	-1.824067
	38°	-1.907519	-1.999266	-1.999736	-1.995987	1.977100	-1.789978
	40°	-1.473946	-1.548925	-1.989670	-.008945	.272869	-1.772420
	*44°	.191499	-1.886559	-1.997639	-1.990243	-.001455	-1.797211
	46°	-1.967290	-1.999996	-1.998486	-1.990231	1.999470	-1.805020
	*50°	-1.000614	-1.999992	-1.992535	.018135	-.000039	-1.802764
	52°	-1.996927	-1.999982	-1.881130	-1.480477	.000001	-1.793093
	56°	-1.984322	-1.999997	-1.998481	-.024450	.000000	-1.792062
	58°	-1.885641	-1.999987	-1.856783	-1.930476	-.157051	-1.785218
	62°	-1.784968	-1.999998	-1.980450	-1.994408	2.000000	-1.847675
	64°	-1.937719	-1.999983	-1.966596	-1.974097	1.990657	-1.892819
	68°	-1.933816	-1.999983	-1.966596	-1.986658	1.990657	-1.777085
	70°	.003165	-1.999976	-1.989230	-1.945507	2.000000	-1.853188
	74°	-1.980717	-1.999831	-1.958627	-1.872168	1.995952	-1.786148
	76°	-1.848750	-1.998594	-.348551	-.911760	1.318729	-1.803094
	80°	-1.955428	-1.998689	-1.922236	-1.379659	1.939644	-1.798831
	82°	-1.966779	-1.999347	-1.884316	-1.917669	1.999958	-1.803247
	86°	-1.980699	-.559736	-1.973478	-1.972488	-.313145	-1.798992
	88°	-1.986252	-1.416210	-1.955558	-1.976476	.250367	-1.796361

Table A.27 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 5	92°	-1.989226	-1.972658	-1.190850	-1.978086	-.030848	-1.799278
	*94°	.047458	-1.648265	-1.999490	-1.850991	-.072544	-1.798950
	*98°	.022076	-1.922341	-1.699979	-1.948384	-.008457	-1.800675
	100°	-1.515859	-1.961112	-1.997998	-1.996106	1.999723	-1.799395
	*104°	.088523	.001165	-1.999442	-1.907067	-.355272	-1.799566
	*106°	-.518417	.001134	-1.999021	-1.976279	-.026618	-1.799396
	110°	-1.922895	-1.999433	-1.998230	-1.902646	1.904593	-1.894842
	112°	-1.922850	-1.999433	-1.998230	-1.902646	1.904593	-1.892205
	116°	-1.922895	-1.999433	-1.998230	-1.498430	1.904593	-1.852317
	118°	-1.687748	-1.999388	-1.996701	-1.886482	1.857961	-1.844954
	122°	-1.944058	-1.998459	-1.999789	-1.881121	1.999992	-1.824322
	124°	-1.922895	-.724597	-1.998230	-1.902646	1.904593	-1.788036
	128°	-1.982966	-1.999157	-1.999967	-1.997530	1.999998	-1.799956
	130°	-1.546101	-1.790827	-1.999348	-1.999661	1.861757	-1.797925
	134°	-1.905875	-1.985828	-1.999321	-1.866572	1.999913	-1.881780
	136°	-1.967010	-1.743099	-1.999854	-1.999994	.000022	-1.799516
	140°	-1.901505	-1.998004	-1.998657	-1.999994	1.999905	-1.797395
	142°	-1.924402	-1.999994	-1.995892	-1.999841	1.999957	-1.800655
	146°	-1.900183	-1.978834	-1.988801	-1.999893	.069380	-1.806336
	*148°	-1.895942	.000021	-1.995150	-1.999938	.000000	-1.799270
	152°	.003578	-1.999668	-1.976500	-1.999705	1.999989	-1.799976
	154°	-1.998716	-1.999847	-1.996405	-1.913364	2.000000	-1.800162
	158°	-1.861051	-1.977395	-1.999439	-1.999581	1.864406	-1.802051
	160°	-1.863888	-1.977022	-1.901765	-1.988766	1.858951	-1.802051
	164°	-.119132	-1.998016	-1.188417	-1.979715	-.051941	-1.797062
	166°	-1.996858	-1.999818	-1.992969	.078179	1.999990	-1.786042
	170°	-1.994972	-1.999812	-1.836360	.003455	1.999957	-1.798597
	172°	-1.998597	-1.873986	-1.992976	.060632	1.809720	-1.822815
	176°	-1.999881	-1.997419	-1.612386	-1.732668	.741218	-1.811091
	178°	-1.936526	-1.751009	-1.981599	-1.737952	1.009242	-1.848772

Table A.27 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
	182°	-1.909662	-1.909076	-1.991114	-1.911110	1.851996	-1.818290
	184°	-1.883111	-1.999231	-1.712968	-1.995105	-.169089	-1.813157
	188°	-1.979025	-1.997531	-1.954661	-1.970140	-.002879	-1.822193
	190°	-1.990227	-1.999823	-1.995555	-1.912623	2.000000	-1.811374
	194°	-1.976826	-1.980866	-1.966588	-1.947323	.431112	-1.785026
	196°	-1.979580	-1.993663	-1.988755	.463322	1.838430	-1.777139
	200°	-1.999191	-1.912167	-1.837368	-1.973998	1.999050	-1.851358
	202°	-1.999461	-1.910097	-1.933815	-1.931559	1.998757	-1.844129
	206°	-1.174472	-1.937124	-1.986836	-1.993007	1.999994	-1.843756
	208°	-1.823502	-1.950390	-1.476568	-1.980104	1.371803	-1.840824
	212°	-1.997418	-1.993607	-1.998794	-1.947159	1.999997	-1.850415
	214°	-1.991668	-1.999937	-1.999873	-1.456358	2.000000	-1.851413
	218°	-1.999470	-1.999904	-1.999308	-1.474115	1.999999	-1.798310
	220°	-1.998251	-1.999990	-1.998739	-1.931020	1.999998	-1.794327
	224°	-1.938343	-1.999996	-1.807267	-1.995876	.965908	-1.794665
# 5	*226°	.094195	-1.979627	-1.999931	-1.997211	-.000011	-1.806593
	230°	-1.899171	-1.999582	-1.999933	-1.997640	1.999989	-1.790804
	232°	-1.959615	-1.999486	-1.999137	-1.116214	.026414	-1.790838
	236°	-1.859613	-1.999447	-1.995425	-1.975072	1.999362	-1.798963
	238°	.346005	-1.977015	-1.744381	-1.999828	1.999517	-1.799763
	242°	-1.912611	-1.031697	-.053886	-1.999652	1.348788	-1.802683
	244°	-1.833458	-1.463988	-1.999484	-1.362720	1.044387	-1.795178
	248°	-1.654445	-1.593965	-1.943487	-1.999790	1.090521	-1.797216
	250°	-1.891309	-1.642778	-1.992715	-1.999903	1.343819	-1.820793
	254°	-1.989391	-1.905860	-.001757	-1.999778	.346879	-1.799522
	256°	-1.995561	-1.999447	-.558595	-1.997481	1.849160	-1.796406
	*260°	-1.552660	-1.996243	.035011	-1.999928	.007462	-1.798808
	262°	-1.836389	-1.998252	-1.900633	-1.999726	1.413678	-1.802278
	266°	-1.814129	-1.998223	-1.403164	-1.998190	1.981771	-1.798026
	268°	-1.550702	-1.998614	-1.907523	-1.997923	1.987689	-1.797583

Table A.27 (Continued)

Object Number	Rotating Angle	Output Node Values					
		Object Nodes					Null Node Average
		#1	#2	#3	#4	#5	
# 5	272°	-1.918118	-1.997029	-1.855927	-1.998785	1.984820	-1.798402
	*274°	-1.503862	-.051688	-1.284380	-1.998432	-.081069	-1.798033
	278°	-1.388400	-1.991197	-1.930160	-1.999987	1.995442	-1.795812
	280°	-1.908808	-1.989018	-1.992102	-1.999852	1.999806	-1.794769
	*284°	-1.908808	.000006	-1.993708	-1.999852	.000000	-1.789714
	286°	-1.862577	-.000016	-1.996236	-1.999731	.000121	-1.795306
	290°	-1.887584	-1.979211	-1.807359	-1.996710	.688747	-1.787762
	292°	-1.887584	-1.979211	-1.807359	-1.996710	.688747	-1.796305
	296°	-1.959081	-1.999968	-1.989780	.008560	1.999980	-1.795612
	298°	-1.873446	-1.997236	-1.990448	.067690	1.999846	-1.790282
	302°	-1.536940	-1.934669	-1.999175	-1.717519	1.999974	-1.804777
	304°	-1.761032	-1.999944	-1.992754	-1.995150	1.999998	-1.803579
	308°	-1.995667	-1.999968	-1.999492	-1.734913	1.999997	-1.814830
	310°	-1.991190	-1.997492	-1.999865	-1.989137	1.999997	-1.799168
	314°	-1.968347	-.643867	-1.999964	-1.976679	1.999825	-1.882311
	316°	-1.938343	-1.992081	-1.999858	-1.996698	.000816	-1.801388
	320°	-1.915190	-1.999997	-1.951421	-.234347	1.999999	-1.799688
	322°	-1.972264	-.000501	-1.999915	-1.761684	.000001	-1.800983
	326°	-.447888	-1.999987	-1.993931	-1.980526	1.839707	-1.793529
	328°	-1.986820	-1.986756	.002196	-1.329540	.125851	-1.802998
	*332°	-1.950174	-1.996111	-1.667272	.082868	-.815979	-1.809236
	334°	.051755	-1.954583	-1.996572	-1.949455	1.991507	-1.812121
	*338°	.122762	-1.988314	-1.983822	-1.962615	-.001990	-1.796009
	340°	.010334	-1.869331	-1.982693	-1.978335	1.942053	-1.793496
	344°	-1.976611	-1.797451	.003803	-1.998346	1.733389	-1.800785
	*346°	-1.998271	-1.989156	.003540	-1.923846	-.210649	-1.799981
	350°	-.153900	-1.999950	-1.977151	-1.523386	-.000009	-1.800414
	352°	-1.998578	-1.939210	-1.938061	-1.799811	-.066625	-1.799626
	*356°	.008349	-1.728258	-1.495120	-1.996102	-1.371602	-1.800943
	358°	-1.851882	-1.909076	-1.927801	-1.872375	-.148003	-1.806257

BIBLIOGRAPHY

[1] J. Feldman, "The Use of Vision and Manipulation to Solve the Instant Insanity Puzzle." *Proceedings of the 2nd International Joint Conference on Artificial Intelligence*, pp. 359-364, London, England, 1971.

[2] H.P. Bao, "An Expert System for SMT Printed Circuit Board Design for Assembly." *Manufacturing Review*, vol. 1, no. 4, pp. 275-280, 1989.

[3] C.H. Dagli, & M. Vellanki, "A Hybrid Intelligent Architecture for Automated Printed Circuit Board Assembly." *Lecture Notes in Economic and Mathematical Systems*, A. Jones, T. Gulledge, & F. Fadel, Eds., Springer-Verlag, New York, 1992.

[4] L.F. Pau, *Computer Vision For Electronic Manufacturing*, Plenum Press, New York, 1990.

[5] M. Vellanki, & C.H. Dagli, "Artificial Neural Network Approach in Printed Circuit Board Assembly." *Journal of Intelligent Manufacturing*, vol. 4, pp. 109-119, 1993.

[6] M. Vellanki, & C.H. Dagli, "Automated Precision Assembly Through Neuro-vision ." *Proceedings of Applications of Neural Networks III*, SPIE, Orlando, Florida, pp. 49-504, 1992.

[7] J.L. Hwang, & M. Henderson, "Applying the Perceptron to 3D Feature Recognition." *Intelligent Engeineering Systems Through Artificial Neural Networks*, C.H. Dagli, S.R.T. Kumara, & Y.C. Shin, Eds., pp. 485-494, ASME Press, New York, 1991.

[8] A. Khotanzad, & J. Liou, "Recognition and Pose Estimation of 3D Objects from a Single 2D Perspective View by Banks of Neural Networks." *Intelligent Engineering Systems Through Artificial Neural Networks*, C.H. Dagli, S.R.T. Kumara, & Y.C. Shin, Eds., pp. 479-484, ASME Press, New York, 1991.

[9] M.C. Lu, C.H. Lo, & H.S. Don, "3D Object Identification and Pose Estimation." *Proc. of ANNIE'91*, Nov. 10-13, St. Louis, Missouri, pp. 473-478, 1991.

[10] B. Pravin, & G. Medioni, "A Constraint Satisfaction Network for Matching 3D Objects." *Int. Joint Conf. on Neural Networks*, Jun 18-22, Washington, DC, pp. II: 281-286.

[11] M. Seibert, &A.W. Waxman, "Learning and Recognizing 3D Objects from Multiple Views in a Neural System." *Neural Networks for Perception*, H. Wechsler, Ed., vol. 1, Academic Press, New York, pp. 426-444, 1992.

[12] S.A. Underwood, & Jr., C.L. Coates, "Visual Learning from Multiple Views." *IEEE Trans. on Computers*, C-24, pp. 651-661, 1975.

[13] P. Gilbert, "Iterative Methods for the Reconstruction of Three Dimensional Object from Their Projections." *J. Theoret. Biol.*, vol. 36, pp. 105-117, 1972.

[14] R. Kashyap, & M. Mittal, "Picture Reconstruction from Projections." *IEEE Trans. on Computers*, C-24, pp. 915-923, 1975.

[15] A. Jain, &S. Ansari, "Radon Transform Theory for Random Fields and Optimum Image Reconstruction from Noisy Projections." *Proc. ICASSP'84*, San Diego, 1984.

[16] G. Herman, *Image Reconstruction from Projections: The Fundamentals of Computerized Tomography*, Academic Press, New York, 1980.

[17] R.C. Gonzalez, & P. Wintz, *Digital Image Processing.* Addison-Wesley, Reading, Massachusetts, 1987.

[18] W. Niblack, *An Introduction to Digital Image Processing.* Prentice Hall, Englewood Cliffs, NJ, 1986.

[19] R.J. Schalkoff, *Digital Image Processing and Computer Vision.* John Wiley & Sons, New York, NY, 1989.

[20] J. Teuber, *Digital Image Processing.* Prentice Hall, New York, NY, 1993.

[21] W.K. Pratt, *Digital Image Processing.* Wiley, New York, NY, 1978.

[22] T.S. Huang, G.J. Yang, & G.Y. Tang, "A fast two-dimensional median filtering algorithm." *IEEE Trans. Acoust. Speech, Signal Process*, vol. 27, pp. 13-18, 1979.

[23] N.E. Nahi, "Nonlinear Adaptive Recursive Image Enhancement." USCEE Report 459, University of Southern California, Los Angeles, 1973.

[24] T. Peli, & J.S. Lim, "Adaptive filtering for image enhancement." *J. Opt. Eng.* vol. 21, pp. 108-112, 1982.

[25] L.G. Roberts, "Machine Perception of Three-Dimensional Solids." *Optical and Electro-Optical Information Processing*, J.T. Tippett, D.A. Berkowitz, L.C. Clapp, C.J. Koester, & A. Vanderburgh, Jr., Eds., MIT Press, Cambridge, Massachusetts, pp. 159-197, 1965.

[26] J.M.S. Prewitt, "Object Enhancement and Extraction." *Picture Processing and Psychopictorics*, B.S. Lipkin & A. Rosenfeld, Eds., Academic Press, New York, pp. 75-147, 1970.

[27] K.K. Pingle, "Visual Perception by a Computer." *Automatic Interpretation and Classification of Images*, A. Grasselli, Ed., Academic Press, New York, pp. 277-284, 1969.

[28] G. Robinson, "Edge detection by compass gradient mask." *CGIP*, vol. 6, pp. 492-501, 1977.

[29] R.A. Kirsch, "Computer determination of the constituent structure of biological images." *Computer and Biomedical Research*, vol. 4, No 3 (June), pp. 315-328, 1971.

[30] W. Frei, & C.C. Chen, "Fast Boundary Detection: A Generalization and a New Algorithm." *IEEE Trans. Comput.*, vol. 25, No. 12, pp. 1336-1346, 1977.

[31] E.L. Hall, *Computer Image Processing and Recognition.* Academic Press, New York, 1979.

[32] V.S. Nalwa, *A Guided Tour of Computer Vision.* Addison-Wesley, New York, 1993.

[33] A. Rosenfeld, & A.C. Kak, *Digital Picture Precessing.* 2nd ed., vol. 1 & 2, Academic Press, New York, 1982.

[34] G.R. Cross, & A.K. Jain, "Markov Random Field Texture Models." *Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-5, no. 4 (July), pp. 511-528, 1983.

[35] M.I. Shamos, *Computational Geometry.* Ph.D. Thesis, Yale University, New Haven, Conn., 1978.

[36] M.A. Fischler, "Fast Algorithms for Two Maximal Distance Problems with Applications to Image Analysis." *Rattern Recog.*, vol. 12, pp. 35-40, 1980.

[37] G.T. Toussaint, "Computational Geometric Problems in Pattern Recognition." *Pattern Recognition Theory and Applications*, J. Kittler, K.S. Fu, &L.F. Pau, Eds., Reidel, NY, pp. 73-91, 1982.

[38] M.K. Hu, "Visual Pattern Recognition by Moment Invariants." *IRE Trans. Info. Theory*, vol. IT-8, pp. 179-187, 1962.

[39] J.T. Tou, & R.C. Gonzalez, *Pattern Recognition Principles.* Addison-Wesley, Reading, Massachusetts, 1974.

[40] K.S. Fu, *Syntactic Methods in Pattern Recognition.* Springer-Verlag, New York, 1974.

[41] R.C. Gonzalez, & M.G. Thomason, *Syntactic Pattern Recognition: An Introduction.* Addison-Wesley, Reading, Massachusetts, 1978.

[42] M.M. Marefat, "Machine Vision for Computer Integrated Manufacturing." *Control and Dynamic Systems*, vol. 61, pp. 1-56, 1994.

[43] D.P. Huttenlocher & S.Ullman, "Recognizing Solid Objects by Alignmnet with an Image." *Int. J. Comput. Vision*, vol. 5, pp. 195-212, 1990.

[44] D. J Kriegman &J. Ponce, "On Recognition and Positioning Curved 3-D Objects from Image Contours." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol 12, pp. 1127-1137, 1990.

[45] "New Methods for Matching 3D Objects with Single Perspective Views." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 9, pp401-412, 1987.

[46] E.K. Wong, "Model Matching in Robot Vision by Subgraph Isomorphism." *Pattern Recognition*, vol. 25, pp. 287-303, 1992.

[47] W.E.L. Grimson & T. Lozano-Perez, "Localizing Overlapping Parts by Searching the Interpretation Tree." *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 9, pp. 469-482, 1987.

[48] D.W.Murray, "Model-based Recognition Using 3D Shape Alone." *Comput. Vision Graphics Image Process*, vol. 40, pp. 250-266, 1987.

[49] G. Marola & A. Vaccarelli, "An Algebraic Method for Detection and Recognition of Polyhedral Objects from a Single Image." *Pattern Recognition*, vol. 27, no. 10, pp 1407-1414, 1994.

[50] C.M. Lee, T.C. Pong, J.R. Slagle, & A. Esterline, "An Experimental Study of an Object Recognition System That Learns." *Pattern Recognition.* vol. 27, no. 1, pp. 65-89, 1994.

[51] D.A. Mitzias, & B.G. Mertzios, "Shape Recognition with a Neural Classifier Based on a Fast Polygon Approximation Technique." *Pattern Recognition.* vol. 27, no. 5, pp. 627-636, 1994.

[52] O.K. Al-Shaykh, *Radon Transform-based Invariant Image Recognition.* Master Thesis, Iowa State University, Ames, Iowa, 1992.

[53] G. Bradski, & S. Grossberg, "Recognition of 3-D Objects from Multiple 2-D Views by a Self-Organizing Neural Architecture." *From Statistics to Neural Network: Theory and Pattern Recognition Applications*, V. Cherkassky, J.H. Friedman, & H. Wechsler, eds, Springer-Verlag, New York, 1994.

[54] G. Carpenter, S. Grossberg, & C. Mehanian, "Invariant Recognition of Cluttered Scenes by a Self-organizing ART Architecture: CORT-X Boundary Segmentation." *Neural Networks*, vol. 2, pp. 169-181, 1989.

[55] S. Grossberg, & L. Wyse, "A Neural Network Architecture for Figure-ground Separation of Connected Scenic Figures." *Neural Networks*, vol. 4, pp. 723-742, 1991.

[56] M. Seibert, & A. Waxman, "Adaptive 3-D object Recognition from Multiple Views." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, pp. 107-124, 1992.

[57] G. Bradski, G. Carpenter, & S. Grossberg, "Working Memory Networks for Learning Multiple Groupings of Temporally Ordered Events: Application to 3-D Visual Object Recognition." In *Proceedings of the IJCNN-91, Seattle, Wa.*, vol. 1, pp. 723-728, Piscataway, NJ: IEEE Service Center, 1991.

[58] C. Lau, "Neural Networks Theoretica Foundations and Analysis." *IEEE Neural Networks Council*, New York, 1992.

[59] L. Fausett, *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1994.

[60] B. Widrow, & M.A. Lehr, "30 years of adaptive neural networks: perceptron madaline, and back-propagation." *IEEE Proceedings*, vol. 78, no. 9, pp. 1415-1442, September 1990.

[61] J.J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities." *Proceeding of the National Academy of Scientists*, vol. 79, pp. 2554-2558. Reprinted in Anderson and Rosenfeld, pp. 460-464, 1988.

[62] J.J. Hopfield, "Neurons with Graded Response Have Collective Computational Properties like Those of Two-state Neurons." *Proceedings of the National Academy of Sciences*, vol. 8, pp. 3088-3092. Reprinted in Anderson and Rosenfeld, pp.579-584, 1988.

[63] M. Cohen & S. Grossberg, "Absolute Stability of Global Pattern Formation and Parallel Memory Storage by Competitive Neural Networks." *IEEE Transactions on Systems, Man, and Cybernetics, SMC-13*, pp. 815-825, 1983.

[64] J.J. Hopfield, "Neurons with Graded Response Have Collective Computational Properties Like Those of Two-state Neurons." *Proceedings of the National Academy of Sciences*, vol. 81, pp. 3088-3092, 1984.

[65] R.P. Lippmann, "Introduction to Computing With Neural Nets." *IEEE ASSP Magazine*, pp. 4-22, April 1987.

[66] R.P. Lippmann, B.Gold, & M.L. Malpass, "A Comparison of Hamming and Hopfield Neural Nets for Pattern Classification." *MIT Lincoln Laboratory Technical Report*, TR-769.

[67] T. Kohonen, "Self-organization and Associative Memory." 3rd ed., Springer-Verlag, Berlin, 1989.

[68] G.A. Carpenter, & S. Grossberg, "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine." *Computer Vision, Graphics, and Image Processing*, vol. 37, pp. 54-115, 1987.

[69] G.A. Carpenter, & S. Grossberg, "ART2: Self-organization of Stable Category Recognition Codes for Analog Input Patterns." *Applied Optics*, vol. 26, pp. 4949-4930. Reprinted in Anderson, Pellionisz, & rosenfeld, pp. 151-162, 1990.

[70] S. Grossberg, "Adaptive Pattern Classification and Universal Recoding, I: Parallel Development and Codeing of Neural Feature Detectors." *Biolgical Cybernetics*, vol. 23, pp. 121-134, 1976 Reprinted in Anderson & Rosenfeld, pp. 245-258, 1988.

[71] P. Werbos, "Beyond Regression: New Tools For Prediction and Analysis in the Behavioral Sciences." Ph.D thesis, Cambridge, MA: Harvard U. Committee on Applied Mathematics, 1990.

[72] D.E. Rumelhart, G.E. Hinton, & R.J. Williams, "Learning Internal Representations by Error Backpropagation." In D.E. Rumelhart & J.L. McClelland, eds., *Parallel Distributed Processing*, vol. 1 chapter 8, 1986. Reprinted in Anderson & Rosenfeld, pp. 675-695, 1988.

[73] D.E. Rumelhart, G.E. Hinton, & R.J. Williams, "Learning Representations by Back-Propagating Error." *Nature*, vol. 323, pp. 533-536, 1986. Reprinted in Anderson & Rosenfeld, pp. 696-699, 1988.

[74] J.L. McClelland, & D.E. Rumelhart, "Explorations in Parallel Distributed Processing." Cambridge, MA: MIT Press, 1988.

[75] F. Rosenblatt, "The Perceptron: a Probabilistic Model for Information Stroage and Organization in the Brain." *Psychological Review*, vol. 65, pp.386-408, 1958. Reprinted in Anderson & Rosenfeld, pp. 92-114, 1988.

[76] F. Rosenblatt, "Two Theorems of Statistical Separability in the Perceptron." *Mechanization of Thought Processes: Proceeding of a Symposium Held at the National Physical Laboratory, November 1958.* London: HM Stationery Office, pp. 421-456.

[77] F. Rosenblatt, *Principles of Neurodynamics.* Spartan, New York, 1962.

[78] H.D. Block, "The Perceptron: A Model for Brain Functioning, I." *Review of Modern Physics*, vol. 34, pp. 123-135, 1969. Reprinted in Anderson & Rosenfeld, pp. 138-150, 1988.

[79] M.L. Minsky, & S.A. Papert, *Perceptrons, Expanded Edition*, 1988. Cambridge, MA: MIT Press. Original edition, 1969.

[80] J.A. Hertz, A. Krogh, & R.G. Palmer, *Introduction to the Theory of Neural Computation.* Addison-Wesley, Redwood City, CA, 1991.

[81] M.A. Arbib, *Brains, Machines, and Mathematics.* Second Edition, Springer-Verlag, New York, 1987.

[82] G. Cybenko, "Continuous Valued Neural Networks with Two Hidden Layers are Sufficient." Tech. Rep., Dept. of Computer Science, Tufts University, March 1988.

[83] B. Irie, & S. Miyake, "Capabilities of Three-layered Perceptrons." *Proc. 2d IEEE Intl. Conf. on Neural Networks*, vol. 1, pp. 641-647, San Diego, CA, July 1988.

[84] E.B. Baum, "On the Capabilities of Multilayer Perceptrons." *J. Complexity*, vol. 4, pp. 193-215, Sept. 1988.

[85] D.G. Bounds, P.J. Lloyd, B. Mathew, & G. Waddell, "A Multilayer Perceptron Network for the Diagnosis of Low Back Pain." *Proc. 2d IEEE Intl. Conf. on Neural Networks*, vol. 2, pp. 481-489, San Diego, CA, July 1988.

[86] J. Radon, *Translation of Radon's 1917 paper.* Translated by R. Lohner, School of Mathematics, Geogia Institute of Technology, The Radon transform and some of its applications by Stanley R. Deans, John Wiley & Sons, New York, 1983.

[87] K. Ma, & G. Kusic, "An Algorithm for Distortion Analysis in Two-dimensional Patterns Using its Projections." *Proc. Seventh New England Bioengineering Conf.*, pp. 177-180, Troy, NY, 1979.

[88] T. Pavlidis, "Algorithms for Shape Analysis of Contours and Waveforms." *Proc. Fourth Int. Conf. on Pattern Recogn.*, pp. 70-85, Kyoto, Japan, 1978.

[89] Z.Q. Wu, & A. Rosenfeld, "Filtered Projections as an Aid in Corner Detection." *Pattern Recognition*, vol. 16, 1983.

[90] A.K. Jain, *Fundamentals of Digital Image Processing.* Prentice Hall, Englewood Cliffs, NJ, 1989.

[91] J.L.C. Sanz, E.B. Hinkle, & A.K. Jain, *Radon and Projection Transform-based Computer Vision*, Springer-Verlag, Berlin, 1988.

[92] R.N. Bracewell, & S.J. Wernecke, "Image Reconstruction over a Finite Field of Views." *J. Opt. Soc. Am.*, vol. 65, pp. 1342-1346, 1967.

[93] R.M. Lewitt, & R.H.T. Bates, "Image Reconstruction from Projections." *Optik*, vol. 50, Part I: pp. 19-33; Part II: pp. 85-109; Part III: pp. 189-204; Part IV: pp. 269-278, 1978.

[94] R.M. Mersereau, & A.V. Oppenheim, "Digital Reconstruction of Multidimensional Signals from Their Projections." *Proc. IEEE*, vol. 62, pp. 1319-1338, 1974.

[95] J.L.C. Sanz, & A.K. Jain, "A New Approach to Computing Geometrical Features of Digital Objects for Machine Vision, Image Analysis and Image Processing: Algorithms in Pipeline Architectures." *Proc. ICASSP '85*; also *IEEE Trans. Pattern Anal. Mach. Intell.*, PAMI-9, vol. 1, pp. 160-167, 1987.

[96] H. Yee, & G. Flachs, "Structural Feature Extraction by Projections." *Region V IEEE Conf. Digest on Elec. Eng. for this Decade*, Austin, Texas, pp. 15-19, April 1976.

[97] P. Breuer, & M. Vajta, "Structural Character Recognition by Forming Projections." *Problems Control Inform. Theory* (Hungary), vol. 4, pp. 339-352, 1975.

[98] K. Yamamoto, & S. Mori, "Recognition of Handprinted Characters by Outermost Point Methods." *Proc. Fourth Int. Conf. on Pattern Recogn.*, pp. 794-796, Kyoto, Japan, 1978.

[99] H. Al-Yousofi, & S.S. Udpa, "Recognition of Arabic Characters." *IEEE Transactions on Pattern Analysis Machine Intelligence*, vol. 14, no. 8, pp. 853-857, July, 1992.

[100] P.R. Madhvapathy, *Pattern Recognition Using Simple Measures of Projections.* Master of Science Thesis, Colorado State University, Fall 1986.

[101] A.P. Reeves, & A. Rostampour, "Sahpe Analysis of Segmented Objects Using Moments." *Proceedings of the IEEE Computer Society Conference on Pattern Recognition and Image Processing*, pp. 171-174, Aug. 25-28, 1981.

[102] O.R. Mitchell, & S.M. Lutton, "Segmentation and Classification of Targets in FLIR Imagery." *SPIE*, vol. 155, Image Understanding Systems & Industrial Applications, pp. 83-90, 1978.

[103] S. Grossberg, "3-D Vision and Figure-ground Separation by Visual Cortex." *Tech. Rep. CAS/CNS-TR-92-019*, Boston University, Perception and Psychophysics, in press, 1993.

[104] V. Srinivasan, P. Bhatia, & S.H. Ong, "Edge Detection Using a Neural Network." *Pattern Recognition*, vol. 27, no. 12, pp. 1653-1662, 1994.

[105] J. Moh, & F.Y. Shih, "A General Purpose Model for Image Operations Based on Multilayer Perceptrons." *Pattern Recognition*, vol. 28, no. 7, pp. 1083-1090, 1995.

[106] C. Shang, & K. Brown, "Principal Features-based Texture Classification with Neural Networks." *Pattern Recognition*, vol. 27, no. 5, pp. 675-687, 1994.

[107] S. Ghosal, & R. Mehrotra, "Range Surface Characterization and Segmentation Using Neural Networks." *Pattern Recognition*, vol. 28, no. 5, pp. 711-727, 1995.

[108] C. Stewart, Y.C. Lu, & V. Larson, "A Neural Clustering Approach for High Resolution Radar Target Classification." *Pattern Recognition*. vol. 27, no. 4, pp. 503-513, 1994.

[109] H.J. Kim, & H. Yang, "A Neural Network Capable of Learning and Inference for Visual Pattern Recognition", *Pattern Recognition*, vol. 27, no. 10, pp. 1291-1302, 1994.

[110] L. Gupta, J. Wang, A. Charles, & P. Kisatsky, "Three-layer Perceptron Based Classifiers for the Partial Shape Classification Problem." *Pattern Recognition*, vol. 27, no. 1, pp. 91-97, 1994.

[111] M. Waite, S. Prata, & D. Martin, *C Primer Plus*, Howard W. Sams & Co., Indianapolis, Indiana, 1984.

[112] J. Neider, T. Davis, & M. Woo, *OpenGL Programming Guide*, Addison-Wesley, Menlo Park, California, 1993.

[113] D.E. Comer, & D.L. Stevens, *Internetworking With TCP/IP Vol 3: Client-Server Programming and Applications*, Prentice-Hall, Englewood Cliffs, New Jersey, 1994.

[114] D.E. Comer, *Internetworking With TCP/IP Volume 1: Principles, Protocols, and Architecture*, Prentice-Hall, Englewood Cliffs, New Jersey, 1995.

[115] M.E. Mortenson, *Geometric Modeling*, John Wiley & Sons, New York, 1985.

[116] L.A. Zadeh, "Fuzzy Sets." *Information and Control*, vol. 8, pp. 338-352, 1965.

ACKNOWLEDGEMENTS

I express my sincere thanks to Professor Jerry Lee Hall for being my major professor, advisor, and friend. Throughout the course of my study, he had extended freedom to pursue my own research interests and ideas, while at the same time always provides helpful hints and invaluable guidance. Without Dr. Hall and the R. A. Engel CIM Laboratory, Department of Mechanical Engineering for providing the necessary financial support, none of this would have been possible.

I gratefully acknowledge Dr. Oliver, Dr. Vardeman, Dr. Jones, and Dr. Molian for serving on the committee and taking their valuable time to read through my dissertation. I also thank Rosalie and ME department for helping me to overcome many difficulties while I was studying in ISU.

I thank Ran Chen's and Yu Chang's families for providing living places and entertainment in Ames for many weekends during my research period while I am working in Cedar Falls. Many thanks to Rick and Mary for correcting my English grammar and spelling in the dissertation and for preparing my final presentation.

I deeply thank my wife, Wendy for her unselfish support and giving. She provides the main energy mentally and physically to keep me going to accomplish today's achievement. Wendy, half of this achievement goes to you.

I owe most of what I have achieved until today to my parents and my brother's family. Mom and dad, all of your children are doctors now.