NOTE TO USERS

FEATURE RECOGNITION IN GEOMETRIC REVERSE ENGINEERING

by

Muhammad Arshad
B.E. (Mech), NED University of Eng. & Tech. 1993

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Applied Science

in the program of

Mechanical Engineering.

Toronto, Ontario, Canada, 2004

Borrower's Page:

Ryerson University requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Number	Print Name	Address	Signature	Date
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				

Supervisor: Dr. Vincent Chan

Abstract

Title : Feature Recognition in Geometric Reverse Engineering.
Name : Muhammad Arshad
Program : MASc., Mechanical Engineering, Ryerson University, 2004.

An artificial neural network based feature extraction system for finding three dimensional features from physical objects is presented. As part of a geometric reverse engineering system, the feed-forward neural network allows for the efficient implementation of feature recognition.

Reverse engineering of mechanical parts is the process of obtaining a geometric CAD model from the measurements of an existing artifact. Ideally, the reverse engineering system would automatically segment the cloud data into constituent surface patches and produce an accurate solid model. In order to accomplish this intent, a neural network is used to search and find the features in the initial scan data set.

In this work, feature extraction for geometric reverse engineering has been accomplished. Work has also been done to extract features from the multiple shapes. The technique developed will reduce the time and effort required to extract features from scanned data of a physical object.

Table of Contents

List of Figures

List of Tables

Nomenclature Table

A	parameter – learning rate of connectors between hidden and output layer
a	mean distance from center to object boundary point
B	parameter – learning rate of connectors between hidden and input layer
b	maximum distance from center to object boundary point
$dw_{(1,i),(2,j)}$	change of the weight between the i^{th} element of the input layer and the j^{th} element in the hidden layer.
$dv_{(2,i),(3,j)}$	change of weight between the j^{th} element of the hidden layer and the i^{th} element in the output layer.
$E_{output,I}$	error at the output layer, neuron location i
$E_{hidden,I}$	error at the hidden layer, neuron location i
k	number of iterations for the neural network training
R_i	reflected vector at the output layer
$w_{(1,i),(2,i)}$	connector weight between neurons at input and hidden layer.
$v_{(2,i),(3,i)}$	connector weight between neuron at hidden and output layer.
n	number of corners for geometric objects
$\propto$	mean angle between two lines of consecutive boundary points
$\lambda_{(1,i)}$	neuron value at input layer, neuron location i
$\lambda_{(2,i)}$	neuron value at hidden layer, neuron location i
$\lambda_{(3,i)}$	neuron value at output layer, neuron location i

Chapter 1- Introduction

1.1 The Geometric Reverse Engineering

Reverse engineering is the process of converting 3D surface data collected from a laser scanner or touch probe mounted on a coordinate measuring machine into a form compatible with CAD/CAM packages. The gathered data, normally huge in size and unstructured in nature are often called cloud data. Reverse engineering is used in industry for a number of reasons, such as modification of prototype parts after testing, or the custom fit of prosthesis for better comfort in the case of knee or hip replacements and the reproduction of broken machine parts whose drawings are not available. [1]

There are two main applications of reverse engineering:

1. To provide digital information for a product for which no CAD model is available.
2. To support the redesign of an existing product.

Either of these goals could be achieved by making sure that the 3D scanned data are complete and accurate. The dimensions of the part or its shape can then be derived from the digitized points. The fitting of one or multiple surfaces to the point data is then

necessary to generate a CAD model. Beyond the domain of prismatic and cylindrical objects, feature handling is still a major research area.

In an ideal reverse engineering system, the cloud data would automatically fragment into constituent surface patches and generate an exact solid model. In order to realize this objective, a neural network is employed to explore and find the features in the initial data set.

Besides the encouraging progression of several researchers, reverse engineering is a diverse and complex problem, to which a direct distinct solution has not been established. [2]

1.2 Traditional Geometric Reverse Engineering

Traditionally, the process of reverse engineering employed a touch probe, which was mounted on a coordinate measuring machine (CMM). [2] In order to accurately define the surface contours of an object, which needs to be reverse engineered, a CMM operator is required to manually guide the sensor to collect thousands of data points. This is a slow process which requires expensive equipment and takes a considerable amount of time.

On the other hand, advancements in machine vision technology provide a means to collect 3D data from the object surface with non-contact sensors like an active laser-based range finder. [2] CAD models are then created from this data for any computer-based design, analysis or manufacturing tasks. The adoption of machine vision-based reverse engineering in the last 15 years has been the result of demands for increased quality control and lower product cost coupled with ever increasing manufacturing throughput requirements. [3]

1.3 Feature Extraction in Reverse Engineering

Detection and localization of 3D objects in scenes represented by single or multiple 2D images has become a well-established technology. A related, but not so deeply investigated problem deals with the identification of 3D objects directly from 3D data. A number of engineering applications rely on robust and efficient shape feature recognition in 3D data, where these data can be either digitized points or synthetic data from a CAD modeling system.

The dimension of the part or its shape can be derived from the digitized points. The generation of a CAD model requires the fitting of one or multiple surfaces to the point data and to construct an appropriate surface or solid model. This step can only be fully automated for some special cases. The level of automation depends on the intended purpose of the CAD model.

Technical feature recognition systems are composed of consecutive blocks, each performing its predefined task in the processing. [4] This system can be described as a block diagram. In the simplest form, it is shown in Figure 1.

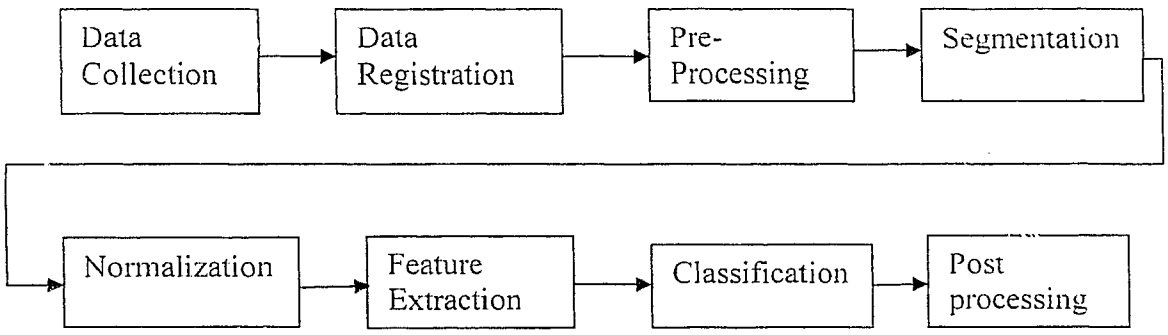


Figure 1. Block diagrams for feature recognition system.

1.3.1 Data Collection

Data collection is the first stage in any feature recognition system. Before an input vector is made up of a set of measurements, these measurements need to be taken. For example, video cameras and scanners are used in the case of character recognition and a microphone, in the case of speech recognition. Data collection devices must be able to record the object, ideally, with the highest reliability available. Noise is considered a disadvantage in order to perform the successful operation of any system.

1.3.2 Data Registration

Elementary model fitting can be performed in data registration. The objective could be achieved by somehow fixing the internal coordinates of the recognition system to the actual data acquired. [4] A priori knowledge surrounding the system is utilized in designing the registration stage. For example, in the case of optical recognition, the system must locate in the input image and the area of interest.

1.3.3 Pre - Processing

In the real world, especially in the case of reverse engineering, data always has some degree of noise and therefore requires a preprocessing stage. The term noise is used in broad sense, but can be simply defined as,

> "Anything that hinders a recognition system to fulfill its commission may be regarded as noise, no matter how inherent this 'noise' in the nature of the data." [4]

Also, preprocessing enhances some of the desirable properties in the data that are fed into the recognition system.

1.3.4 Data Segmentation

The data, which have already been registered and preprocessed, are split into subparts. This process is called data segmentation. In this work, data points which formed the boundary of the objects are segmented. This task is accomplished by developing MATLAB codes. The segmentation processes are outlined in detail in sections 4.5.1 and 4.5.2.

1.3.5 Normalization

A common characteristic of feature recognition systems is the inherent variance of the objects to be recognized. [4] The main problem in feature recognition is how these variances are accounted. There are many possibilities, one is to use feature extraction or a classification algorithm, which can deal with the variations in the outcomes of the object. The side effect of normalization is a loss of degrees of freedom, i.e., the dimension reduction in the intrinsic dimensionality of the data.

1.3.6 Feature Extraction

The dimensionality of data is reduced during the process of feature extraction. This is necessary as a result of limitations in memory and computation time. [4] A reliable feature extraction scheme can maintain and enhance those features of the input data which make distinct feature classes separate from each other. Also, the system must be restrained with respect to variation produced by both the humans and the measuring devices used in the data acquisition stage.

1.3.7 Data Classification

Classification is the most crucial step in the process of feature recognition. All the previous stages are designed and tuned with the aim to have success in the classification phase. In the simplest way, the operation of classification is the transformation of quantitative data to qualitative output information.

1.3.8 Post-processing

After the classification stage, some data processing is performed in most feature recognition systems [4]. The post processing subroutine carries forward some a *priori* information about the neighboring world into the system. This additional step helps in improving the overall classification accuracy. The post-processing phase is generally possible if the individual objects or segment make up meaningful entities such as bank account numbers or sentences.

1.4 Features Recognition for Geometric Reverse Engineering Data:

The "Feature" driven CAD modeling packages provide the vital link between design and manufacturing. In the same way, "Feature" driven reverse engineering would allow for more flawless application of the CAM software.

In this work, the features are extracted from geometric reverse engineering data. To test the proposed algorithms, seven different geometric objects are created: circle, ellipse, square, rectangle, triangle, diamond and wheel. These objects are shown in Figures 2, 3, 4, 5, 6, 7 and 8.

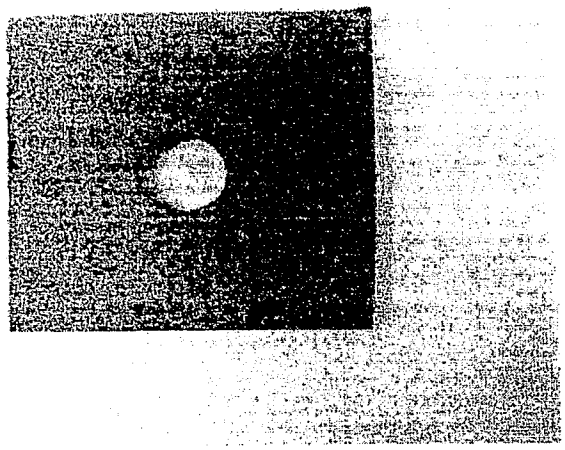


Figure 2. Picture of "circle" test object.

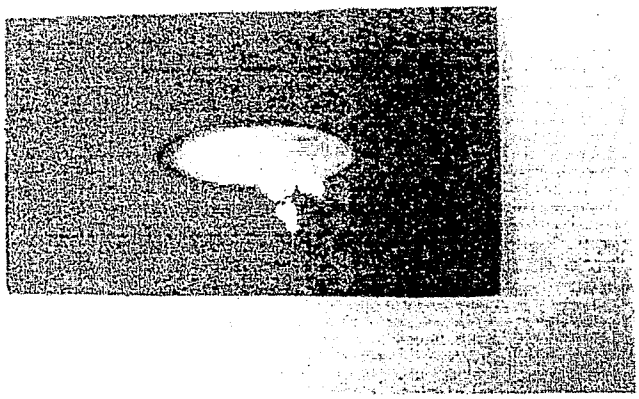


Figure 3. Picture of "ellipse" test object

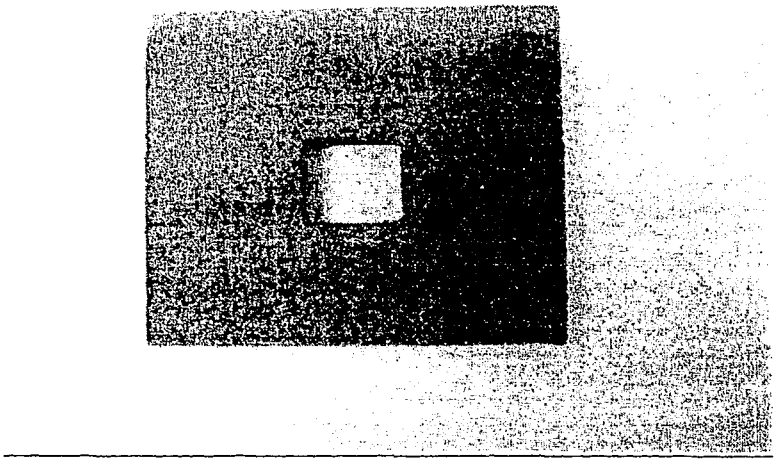


Figure 4. Picture of "square" test object

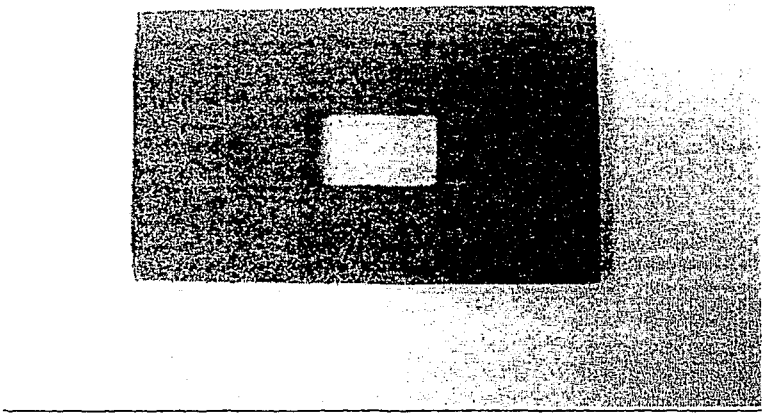


Figure 5. Picture of "rectangle" test object

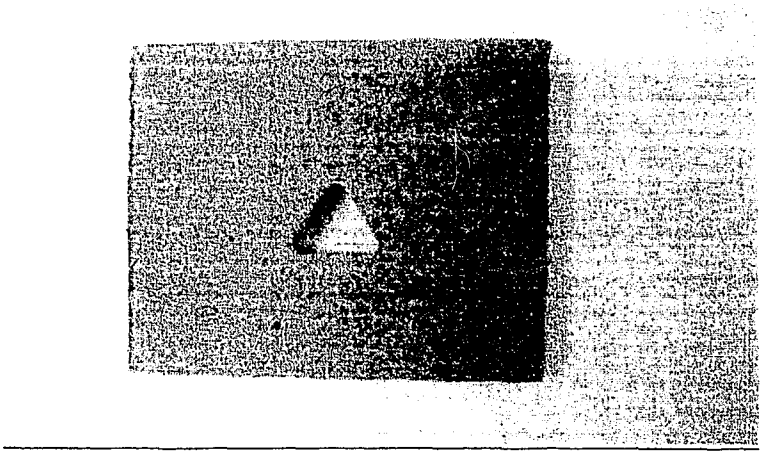


Figure 6. Picture of "triangle" test object

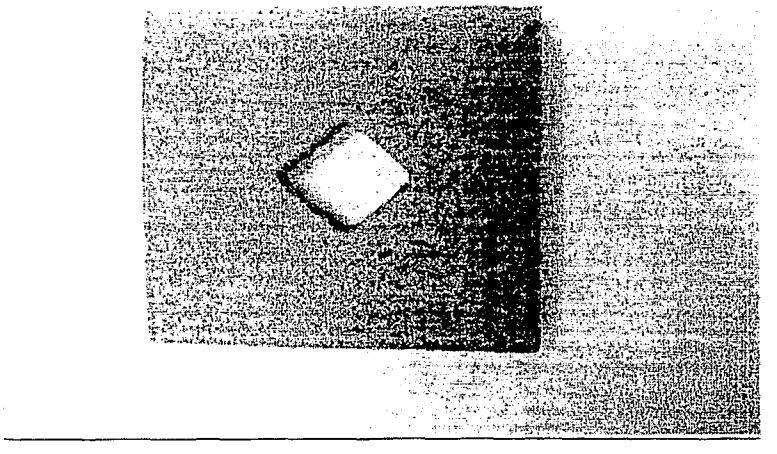


Figure 7. Picture of "diamond" test object

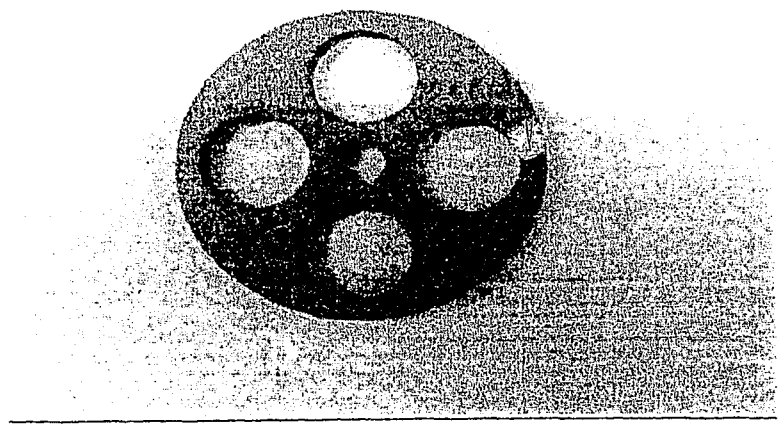


Figure 8. Picture of "wheel" test object

A Roland Dr. Picza PIX – 30 3D laser scanner is used to collect range data of different accuracy levels and densities. Based on a piezo sensor, the PIX – 30 is a contact scanner.

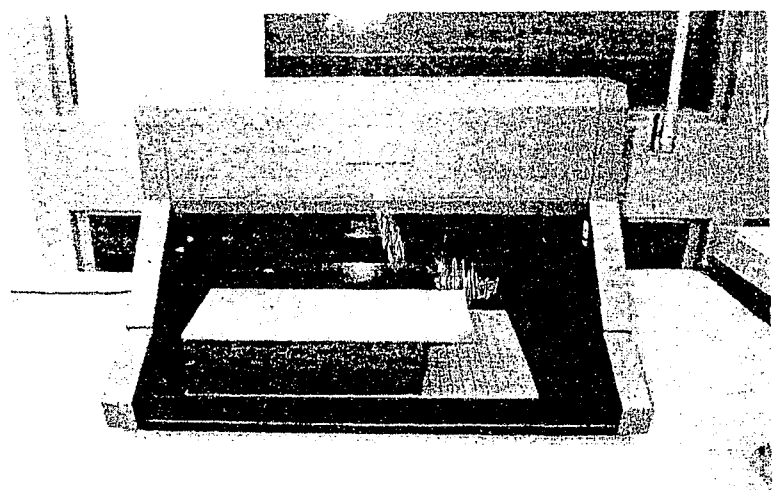


Figure 9. Roland PIX – 30 3D Scanner

This scanner gathers the data from the surfaces of the object. The collected measured data points are fitted with a suitable primitive geometric shape. A programming method base on the human brain architecture, Neural Network, is used for the recognition of important features on the object's surface. These features provide a more intuitive means for engineers to develop object definition.

MATLAB codes are developed first for data segmentation and then eventually for feature recognition. In a simple form, the process is outlined in Figure 10.

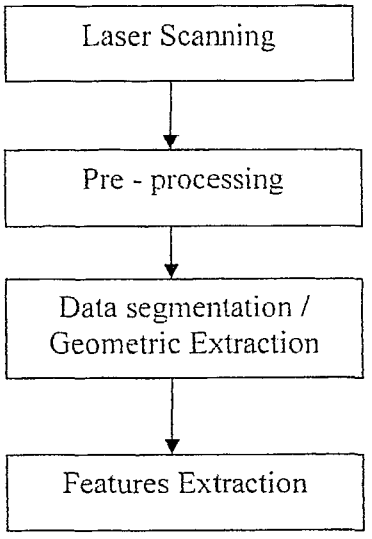


Figure10. Systematic diagrams for feature recognition system.

1.5 Legal Standing of Reverse Engineering

The legal standing of reverse engineering has long been an issue for the engineering discipline. Several U.S. Supreme Court rulings and congressional legislations are in place which allow the use of reverse engineering for development and innovative purposes. Reverse engineering has long been held as a lawful form of discovery in both

legislation and court opinions [5]. The Supreme Court of USA has confronted the issue of reverse engineering in mechanical technologies numerous times, upholding it under the principle that it is an important technique of dissemination of ideas and encourages innovation in the market place. The U.S Supreme Court addressed the first principle in Kewanee Oil v. Bicron, a case concerning trade secret protection over the manufacturing of synthetic crystals by defining reverse engineering as "a fair and honest means of starting with the known product and working backwards to divine the process which aided in its development or manufacture." [5]

There was another principle that encourages the innovative use of reverse engineering articulated in Bonito Boats. v. Thunder craft. This case involving laws forbidding the reverse engineering of the molding process of boat hulls. In this case, the U.S Supreme Court said "the competitive reality of reverse engineering may act as a spur to the inventor, creating an incentive to develop inventions that meet the rigorous requirements of patentability." [5]

1.6 Potential Benefits

This research looks at automating the collection of surface data points and the modeling of the surfaces in a computer aided design (CAD) program. The 3-D laser allows the gathering of preliminary surface information that could subsequently be used to locate important features on the object being examined.

Applications of this research range from the geometric reverse engineering of physical models to quality control. This research will allow manufacturers to reduce design cycles and to quickly bring products to market.

1.7 Scope of this work

This thesis is arranged in chronological fashion of the steps required to carry out feature extraction.

Chapter 1 discusses reverse engineering and the conventional reverse engineering techniques. This chapter describes the problem and proposes the methodology to extract features for geometric reverse engineering data. Potential benefits of this research are also discussed at the end of the chapter.

There is no universally agreed definition of a feature. The word "feature" has a different meaning for different researchers. The most commonly used feature definitions are described in Chapter 2. This Chapter also discusses types of feature recognition and the related literature review of features recognition techniques.

Chapter 3 describes different types of neural networks and discusses their structure. This chapter also looks at the related literature on artificial neural network techniques.

Chapter 4 discusses the selection of the artificial neural network method applied in this work to extract the important features, its configuration and potential benefits. This chapter also describes the algorithms that were developed for the segmentation of the boundaries and then calculation of the parameters that form the input vector to the neural network. Finally, the feed-forward neural network algorithm applied for feature recognition is described.

The testing of the neural network algorithm is a very important part of this technique. The algorithm is first tested with manually created synthetic data and then on the real reverse engineered data derived from the test samples. After the successful

training of the algorithm, the algorithm is tested on an unseen example. Chapter 5 presents the training and testing of the neural network algorithm.

Chapter 6 discusses the conclusion and future work.

Chapter 2: Features Extraction – Literature Review

2.1 Introduction

Many techniques have been developed for feature identification from CAD models. However, the literature is scarce in the area of extracting features from reverse engineered data. Most of the methods are based on matching algorithms, in which the data are compared with a predefined set of surfaces and edges. The features are usually defined generically before any matching process may be initiated, as a combination of topological entities.

The overall aim in feature recognition is to convert low level geometric information into a high level description in terms of form, functional, manufacturing or assembly features. This description could be for design, manufacturing, engineering analysis or even for administrative purposes. [6] It is well known that recognizing features that are required for machining may be considerably different from recognizing features useful for casting or for assembly purposes. In other words, features are context – dependent entities. [7]

"Feature" is general term and has been used to describe a number of different things. For example, Vandenbrande et. al. [8] define that a feature is a "region of an object that is meaningful for a specific activity or application". Schulte et. al.[9] considered a feature as "geometry associated with a specific operation". Another definition by Silver [10] describes a feature as "A region of interest consisting of voxels satisfying a set of pre-defined criteria". Shah [11] defines a feature as a physical constituent of a part that can be mapped to a generic shape and "represents the engineering meaning of the geometry of a part of assembly".

For this work, a feature will be defined as "a recognizable topological pattern of a set of edges".

2.2 Types of Feature Recognition

There have been various techniques developed to extract features from a geometric modeling database. By and large, feature recognition can be divided into three main categories. [12]

- Parameter matching
- Syntactic feature recognition
- Volume Decomposition

2.2.1 Parametric Matching

In parametric matching, the features are first characterized in term of their geometric and topological form. The algorithm searches the solid model data base, measures against topological type, connectivity and adjacency to decide if any of the characteristic fe tures are available.

2.2.2 Syntactic Feature Recognition

In syntactic pattern recognition, the geometry is represented in terms of a language grammar that describes the order of the lines and curves. The description of the object is then matched against grammar to recognize the features.

2.2.3 Volume Decomposition

In the third category, i.e., volume decomposition, the removed base stock material is identified and then broken down into distinct machining operations. This volume is decomposed into smaller volumes or "features", which conform to machining operations.

2.3 Feature Extraction - Literature Review

Free-form features are acquiring a great deal of attention since they are considered the important constituent in product styling, aesthetic design and shape conceptualization. Recently, some Computer-Aided Industrial Design (CAID) systems have surfaced, each of which is in some means based on surface features or free-form features. [13, 14] Various systems are dedicated to particular types of features, for example, protrusions and depressions. [15]

To make these type of systems truly flexible and useful, free-form features (shape patterns) are required to be extracted from existing objects, where these objects are either physical (and to be 3D scanned) or virtual (and to be sampled). [16] The main obstacle is the fitting of 3D shape patterns against 3D point sets. A necessary requirement of the fitting method is that it should not only acquire placement and scale parameters for the pattern, but also shape deformation parameters.

In a graph based approach for feature recognition, boundary representations are built upon a graph structure. Boundary representation model faces can be considered as nodes of a graph while face-face relationships form the arcs of the graph. As described by Wu and Liu [17], graph based approaches first represent an outline of the required topological and geometrical constraints for recognizing the feature. Once the graph which identifies a feature class has been defined, such a graph has to be searched in the object structure, which is a graph as well. The problem of recognizing a given sub-graph in a graph is fairly complex problem and its computing time in the worst case grows exponentially. Many authors proposed various search strategies to work out this problem. Some authors argued that the adjacency information available is usually not adequate for feature recognition. For this reason a number of augmented graphs have been recommended.

One of the main drawbacks of graph-based feature recognition techniques is the difficulty in recognizing interacting features. This is due to the fact that a feature characteristic pattern is changed when features intersect each other. Hint based reasoning approaches[18] have been developed to overcome this drawback. In the hint based approach, developed by Requicha et. al. [18, 19] those characteristic traces that features leave in the nominal geometry of the part are searched. These traces present hints for the potential existence of volumetric features even when features intersect. These hints are processed to generate the largest possible volumetric feature that is compatible with the hint and does not intrude into the feature.

In the pioneering work of Kyprianou [20], feature grammars are described for the extraction through syntactic feature recognition. Falcidieno et. al. [21] took allowance of

Kyprianou's method for the identification and extraction of feature information from a boundary representation of an object. The fundamental concept is to define a pattern description language where suitable rules are defined in order to create an applicable composition of primitives. The authors defined three basic primitives: convex edge, concave edge and smooth edge. This graph parsing based feature recognition scheme was focused on the identification of depression and protrusion features. Di Stefano [22] introduced the concept of semantema as the minimal element of meaning that defines the semantics of the representation. This approach requires the statement of the minimal set of semantema that identifies the feature clearly.

Volume decomposition methodologies operate more directly on the three-dimensional representation of volumes instead of working on the boundary representation graph of solid models. [23] Such approaches have been generally employed for the recognition of machining features. There are two main approaches for volume decomposition, [23] alternating sum of volumes (ASV), where an object is articulated in terms of a hierarchical structure of convex components, and delta volume decomposition, where the intent is to recognize the volume to be machined and then decompose it into a set of non-overlapping entities corresponding to different machining operations.

In the pioneering work of Woo, [24] ASV decomposition is applied to indicate a non-convex object by a hierarchical structure of convex components. This approach has been proven non-convergent in certain cases. Kim [25] proposed an enhanced convex decomposition approach to address this issue. Kim and Wang et. al. [26] proposed the alternating sum of volumes with partitioning (ASVP) approach. This is a convex decomposition method based on a convex hull, set difference and cutting operations. In

this scheme, the boundary faces of a part are arranged in an outside-in hierarchy and volumetric components are related with these faces. The ASVP decomposition is transformed, by means of combination operation between components, into a set of feature volumes (Form Feature Decomposition - FFD) corresponding to significant high level constituents of the product shape. FFD is then transformed into a Negative Feature Decomposition (NFD) by means of positive-to-negative conversion. The machined face information is obtained from negative feature stand for removal volumes.

Kailash et. al [27] described a method dedicated to machining feature extraction of casting and forging components. In this scheme, machining removal volumes were first obtained by subtracting the final part model from a row part model. Machined faces (M-faces) are then recognized and collected into groups (M-groups). Finally, M-groups are mapped into all possible machining process forms. This feature identification approach is process oriented as M-group is mapped to various processes.

2.4 Feature Recognition in Reverse Engineering

It is difficult to classify feature extraction methods into precise, organized groups as there is a considerable overlap between the various techniques. The majority of the methods, as discussed in Section 2.2, use a matching algorithm which compares data with predefined generic features. Features extraction algorithms may include the following specific tasks: [2]

i. Generic definition of a features topology.

ii. Searching the database to match topological patterns.

iii. Extracting the recognized features from the database (removing a portion of the model associated with the recognized feature).

iv. Determining the feature parameters (hole, diameter, number of corners).

The features must be generically defined by a combination of topological entities required to illustrate the features before any matching process may be initiated. For example, a hole could be described as combination of two circular edges surrounding a cylindrical surface. Secondly, the topological database would be searched for connectivity and adjacency to determine which of these features are present in the solid model.

The features will be further limited in scope to specific machining operations for the creation of hole. square, diamond, ellipse, triangle, rectangle and wheel. There are, of course, many reasons for limiting this definition. As this work is primarily concerned with reverse engineering, the emphasis is on the reconstruction of the solid parts, not on the design of parts. The reverse engineering algorithm is to model the part with some rudimentary editing features. It is not meant as a replacement for a CAD package.

Chapter 3 – Artificial Neural Network : An Overview

3.1 Introduction

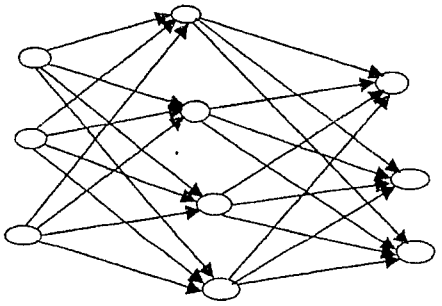
This chapter presents different types of neural networks. The related literature review on the artificial neural network is the main focus of this chapter.

3.2 Types of Neural Network

There are many types of neural networks. Each type has the characteristics of parallel processing from an interconnected network of computational elements. Several structures of neural networks are possible by connecting the elements together. There are two most commonly used structures from the neurons connection point of view: multi-layer neural network and fully connected neural network (Hopfield network). These networks differ from one another in architecture and training algorithm. The following three are the most commonly used networks. [28]

3.2.1 Multi Layer Feed Forward Neural Network

The multi layer feed-forward neural network has many successful applications and is the most commonly used neural network. As shown in Figure 11, the neurons are arranged in several layers. Any number of neurons and number of layers are possible. [28]



Input vectors Hidden layers Output vectors

Figure 11: Multi layer feed-forward neural network

The layers are classified into three types: input, hidden and outer layers. Any number of hidden layers is possible but the connections are allowed only in the feed-forward directions.

3.2.2 Hopfield networks

In the early 1980's, John Hopfield's pioneering work gave credibility to the fledging neural network field. Contrary to the multi layer feed-forward neural network, a Hopfield network is defined as a feed-back system with the output of one complete forward operation of the network serving as the input to the next network operation. The Hopfield network is also called the recurrent network as this network operates as a feed-back system. [29] This scheme is illustrated in Figure 12.

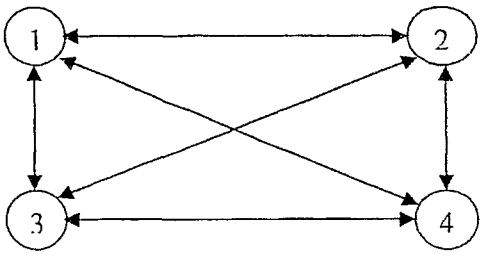


Figure 12. Simple Hopfield Network

Each forward operation of the network is called an iteration. The process is repeated until the output remains constant. Hopfield showed in his work that if the weight matrix is symmetrical with zero diagonal elements and the elements are updated asynchronously, the network will always converge.

3.2.3 Kohonen self organizing network

In the early 1980's, Teuve Kohonen developed an algorithm to mimic the brain's ability to organize itself in response to external stimuli. Kohonen called his algorithm a self organizing feature map. Kohonen's algorithm represents a type of neural network that is capable of learning without supervision. [29] In this technique, the weights strengthen themselves. The first layer is the input and the second layer, the output. It is a two dimensional grid where the self organizing takes place as illustrated in Figure 13.

In the winner take all competition, the output neuron with the highest value wins. The structure of the output vectors is laid out in a grid like pattern, this allows the concept of neighborhoods.

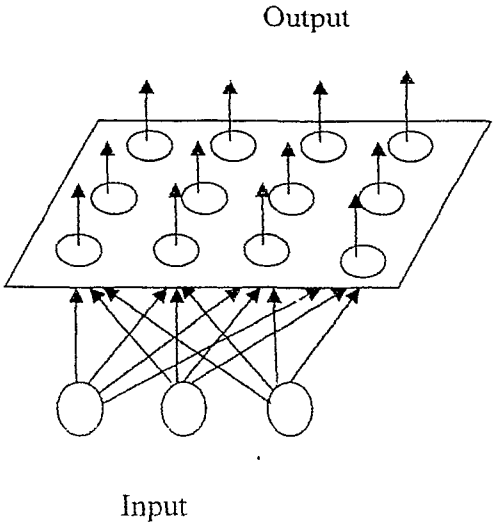


Figure 13. Kohonen self-organizing network

Once the winner element is found, the weights for that output element and its neighborhood are updated. The connectors are iterated until the value converges, i.e., no more winning neurons are declared.

3.3 Artificial Neural Network – Literature Review:

For classification problems, the neural network is able to give statistical information about the classification and is easy to train, but it is often not clear how the neural network has arrived at its answer. On the other hand, the operations of rule-based algorithms are traceable, but the set of rules chosen may be more difficult to train and may not generalize as well as a neural network. [30]

Recognition is one of the most complex problems in the computer and machine vision area. The major concern associated with the use of artificial neural networks for feature recognition is the formulation of an appropriate codification of the topological and geometrical entities in order to present a numerical input to the network.

In order to meet the neural network input requirements, Prabhakar and Henderson [31] defined a feature as a mathematical function, in which geometrical and topological data were variables, derived from the solid model of the part. In this method, these variables represented the net input and were arranged in a two dimensional matrix called the adjacency matrix. Each element (i,j) of the adjacency matrix represents the relationship of face j to face i. Non-adjacent faces i and j were represented by zeros whereas different integer values denote different types of edges. The sign of the adjacency matrix element indicates whether the edge is concave or not. The process of recognition is then reduced to row-by-row parsing of the adjacency matrix.

Zulkifli et. al. [32] proposed a method to recognize the interacting features. The authors used a B-rep solid model as input for the feature recognition system. This method is based on a layering technique to find interacting features. After selecting the principle direction, this technique searches for any volume that exists between two successive layers of the part. These volumes are then checked to find out if they represent the result of interacting features. This task is accomplished by means of a Kohonen self- organizing feature map (SOFM) neural network, which is used to create maximal rectangular regions which are then intersected with the resultant area. Primitive features are then obtained from resulting regions. Also, the second stage of SOFM was applied on the resulting regions to decompose them into primitive regions. As described by the authors, this technique is limited to apply only for the features that have identical thickness and a common bottom face.

Chan et. al. [1] present a method in which stereo pairs of images are used to plan the path for a co-ordinate measuring machine (CMM). The Kohonen self-organizing map

(SOM) network is used for the segmentation of the CCD images. The authors incorporate the charged coupled device (CCD) camera and a CMM touch probe digitizer together to accomplish this aspect of reverse engineering. In this reverse engineering system, an accurate solid data were obtained from the automatic digitization of the object using the CCD images.

Methods that require no explicit models, e.g., neural networks, case-based reasoning and inference have been developed, but their ability is saturated at a certain level. Two methods to evade the limits have been attempted, one is to build a more concrete model and the other is to fuse these methods together. [33] Yata's digit is a remarkable and promising success in the latter method. The main idea is to utilize many neural networks at the same time that construct total model "Multi-Model Neural Networks" (MNNs). Despite its very simple and easy implementation, the preliminary results showed that MNNs significantly increase sensitivity. In 2001, Yoshihara et. al. [33] applied a multi-model neural network to identify exon-intron boundaries (splice site) in DNA base sequences. The MNNs provide a higher identification rate of 95%, as compared to 83.4% with a single NN.

None of the reverse engineering packages address the automation of feature extraction due to the size, incomplete and unstructured nature of scanned data. The reverse engineering packages having provision for feature recognition rely on an interactive user interface [2]. Thompson et. al. [34] proposed a REFAB (Reverse Engineering – FeAture-Based) system. In the proposed system, the authors used an interactive graphics workstation that segments the reverse engineering data into features. REFAB allows a user to specify the types of manufacturing features and approximate

location of each feature in the object. The REFAB system then fits a feature to the scanned data by using an interactive refinement process.

Machine vision systems facilitate sophisticated industrial applications, such as classification and process control. Artificial neural networks (ANNs) and machine vision bonded together provide a new scheme for solving complex computational problems in many areas of science and engineering. Farhad et. al. [35] investigated several novel uses of machine vision and ANNs in the processing of single camera multi-positional images for 2D and 3D object recognition and classification. The authors used the boundary contour information as a method of representing the industrial component. A number of shortcomings were found most importantly the identification of unique start point, vital for rotation invariance.

Chapter 4 – Neural Network Based Feature Extraction

4.1 Introduction

The objective of this work is to recognize features in the scan data. This chapter will look at the artificial neural network in depth, its implementation on physical objects to extract features and potential benefits for this specific problem.

4.2 Selection of Artificial Neural Network

Current commercial reverse engineering software packages have not been addressing the automation of feature identification. Artificial neural networks are a good choice for feature extraction from the reverse engineered data due to the nature of scan data which have the following inherent traits:

- Noise in the scan data set.
- Unstructured and large in size.
- Incompleteness of the data.
- Defects in the scan object.

Due to the nature of reverse engineering data, a robust method is required for the implementation of a feature recognition algorithm. Rule based algorithms rely on searching through a set of rules, selecting the rules which will advance the search state from one state to the next until the final states are found. These algorithms need concise and accurate data to test the topology. As reverse engineering data is often incomplete and often carry a substantial amount of noise, a neural network based algorithm is considered to be more robust. Artificial Neural Networks (ANNs) have shown considerable promise in a wide variety of application areas and have been particularly useful in solving problems for which traditional techniques have failed or proved inefficient. Neural networks have seen many successful applications in machine vision feature recognition problems. The neural network technique is used in this work because:

- It has proven robustness in many 2-D machine vision problems.
- Has the ability to learn and work in many different situations.
- It is not susceptible to incomplete data sets as much as rule based algorithms.
- Higher computational ability because of massive parallelism
- Amenable to machine learning

4.3 Artificial Neural Network Configuration

The neural network in this work has an input layer, hidden layer, an output layer, weights, bias and a transfer function. The inputs are multiplied by weights, bias is added and the transfer function operates on the total to give the output. Generally, linear transfer functions are best suited to linear problems, and non-linear transfer functions are best

suited to non-linear problems. Graphically, the neural network configuration is shown in Figure 14. [36]

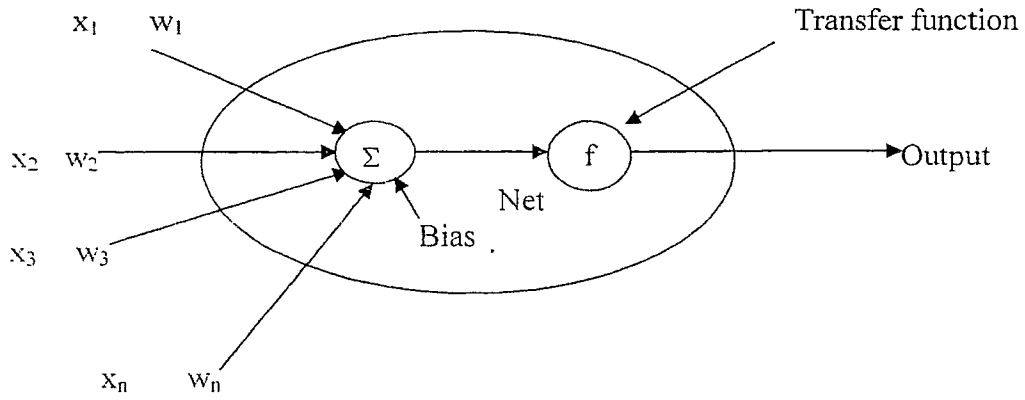


Figure 14. Neural network configuration.

In Figure 14:

$$Net = \Sigma (x_i w_i + bias) \qquad .....(4.1)$$

$$Output = \begin{cases} 1 & \text{if net} > \text{threshold} \\ 0 & \text{if net} < \text{threshold} \end{cases}$$

Weights are updated: $w_i (t+1) = w_i (t) + \varepsilon x_i$

Where,

ε = Error = desired output – cal. output

and

$(x_1, x2, x3, \ldots x_n)$ are input vectors.

The commonly used transfer functions for classification are step functions, linear functions, sigmoid functions, hyperbolic-tangent functions.

Due to the wide range of problems to which neural networks have been applied to, it is difficult to generalize which types of transfer functions are best suited to certain types of problems. For this work, a hyperbolic-tangent transfer function is used between the hidden and output layer to map the result at the output layer. The hyperbolic-tangent transfer function has the properties to vary from -1 to +1.

4.4 Feed-Forward Neural Network

The neural network selected for this work is a feed-forward (back propagated) based network with one hidden layer. The number of input elements should equal the number of parameters needed to define each feature, whereas the number of output elements should represent the number of different types of features which can be found as shown in Figure 15.

For this work, the neural network has four elements in the input vectors and seven elements in the output vector. The four elements define the geometry of the objects and form the input vector that was presented to the neural network. These parameters are discussed and calculated in Section 4.5. Also, each neuron at the output layer defines the feature type to recognize the target object from seven different objects.

The term back-propagation refers to the training of the algorithm rather than the network architecture. In a feed forward network, the networks feed the input forward, that is, towards the output. It is thought that adding an additional layer, a so-called hidden

layer, would soften the effects of input noise. However, whether or not to use one hidden layer or two or more hidden layers has still not been worked out. [36]

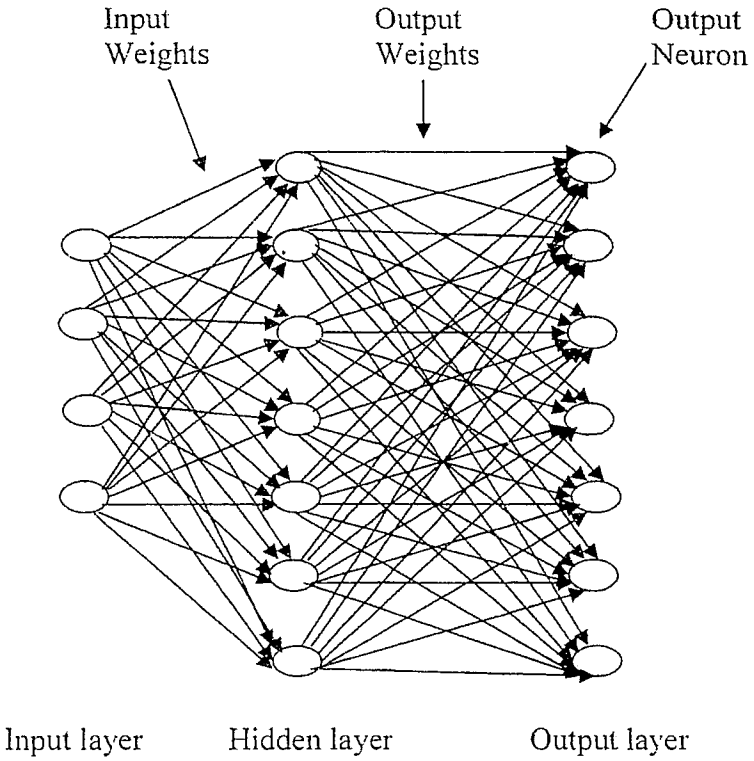


Figure 15. Neural network for feature recognition.

The number of elements in the hidden layer cannot be determined except through experimentation. Nezis and Vosniakos [37] found that increasing the number of hidden layers did not change the results, but did increase the training times significantly. They also found that increasing the number of elements in the hidden layer resulted in better mapping, however at the penalty of increased training times. For this work, one hidden layer was used to soften the effect of input noise. There are seven neurons employed in the hidden layer that was mapped on the output layer to generate the output values.

4.5 Input and Output Vectors

Before implementation, the neural network algorithm must receive the topology and its associated geometry in a form that can be presented to the input layer. Therefore, to search for features in the database. fragments of the topology geometry database must be coded into a format understandable to the neural network. [1] However, with reverse engineered data, the geometry may also prove to be an important indicator of the possible features, as the faces that make up a feature may not be fully defined.

Since the reverse engineering scanned data is huge and unstructured in nature, two different segmentation algorithms, one for a single object and the other for multiple objects, are developed for the cloud data to segment the boundary of the object and hence calculate the parameters that make the input vectors. The digitized wheel object and its isometric view are shown in Figure 16 and 17.

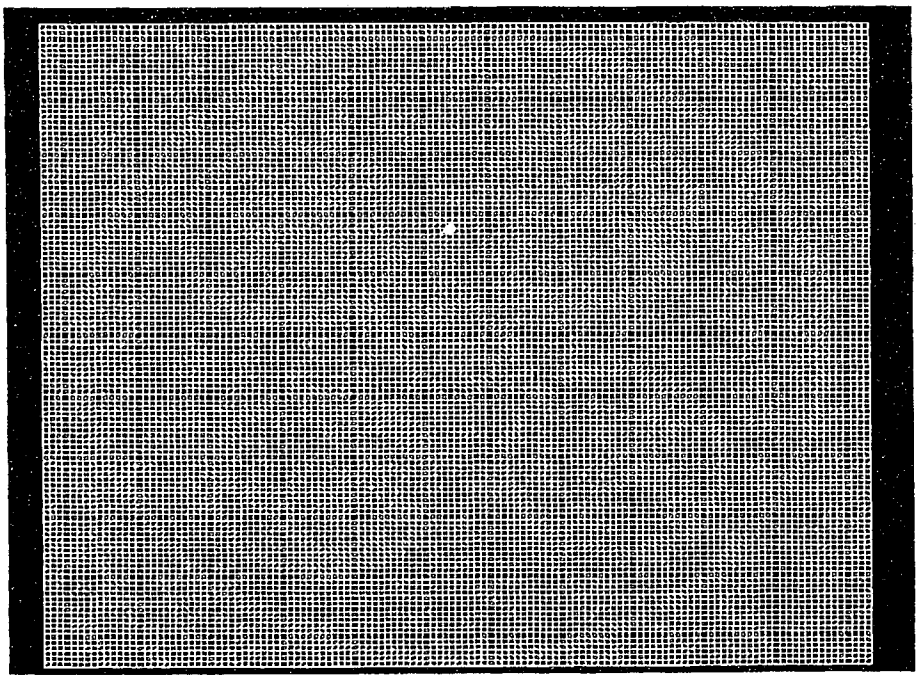


Figure 16. Digitized points from the physical object "wheel"

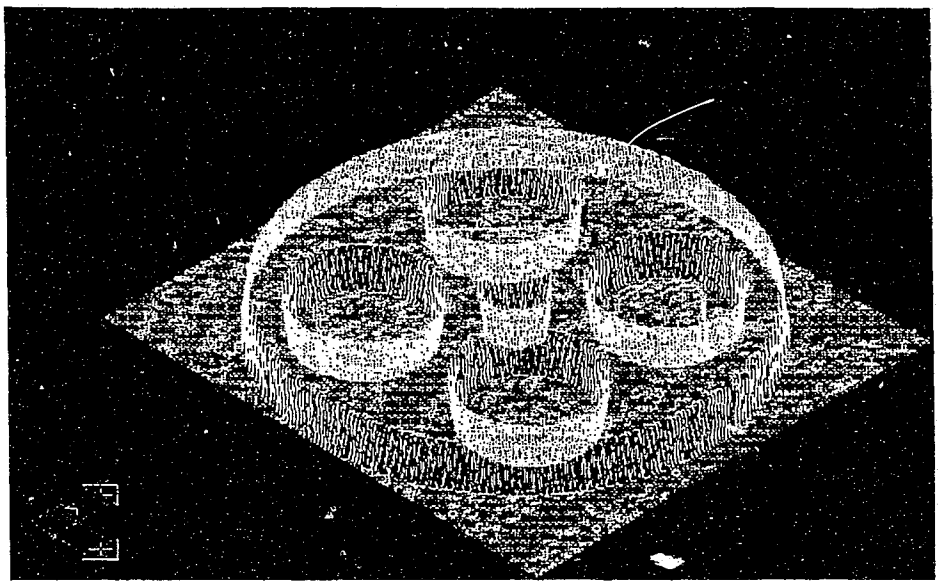


Figure 17. Isometric view of the digitized physical object "wheel".

4.5.1 Segmentation Algorithm for Single Shape Objects

This segmentation algorithm looks at the physical object with a single geometric shape, like a circle, an ellipse, etc. This algorithm first segments the data points from the cloud data that form the object's shape. For this purpose, this algorithm compares the data points that establish the depth of the physical object with those on the surface. These segmented data points are then further processed to find the boundary points of the object and hence calculate the important parameters that form the input vectors. This whole process was programmed in MATLAB. A flow chart for this segmentation process is shown in Figure 18.

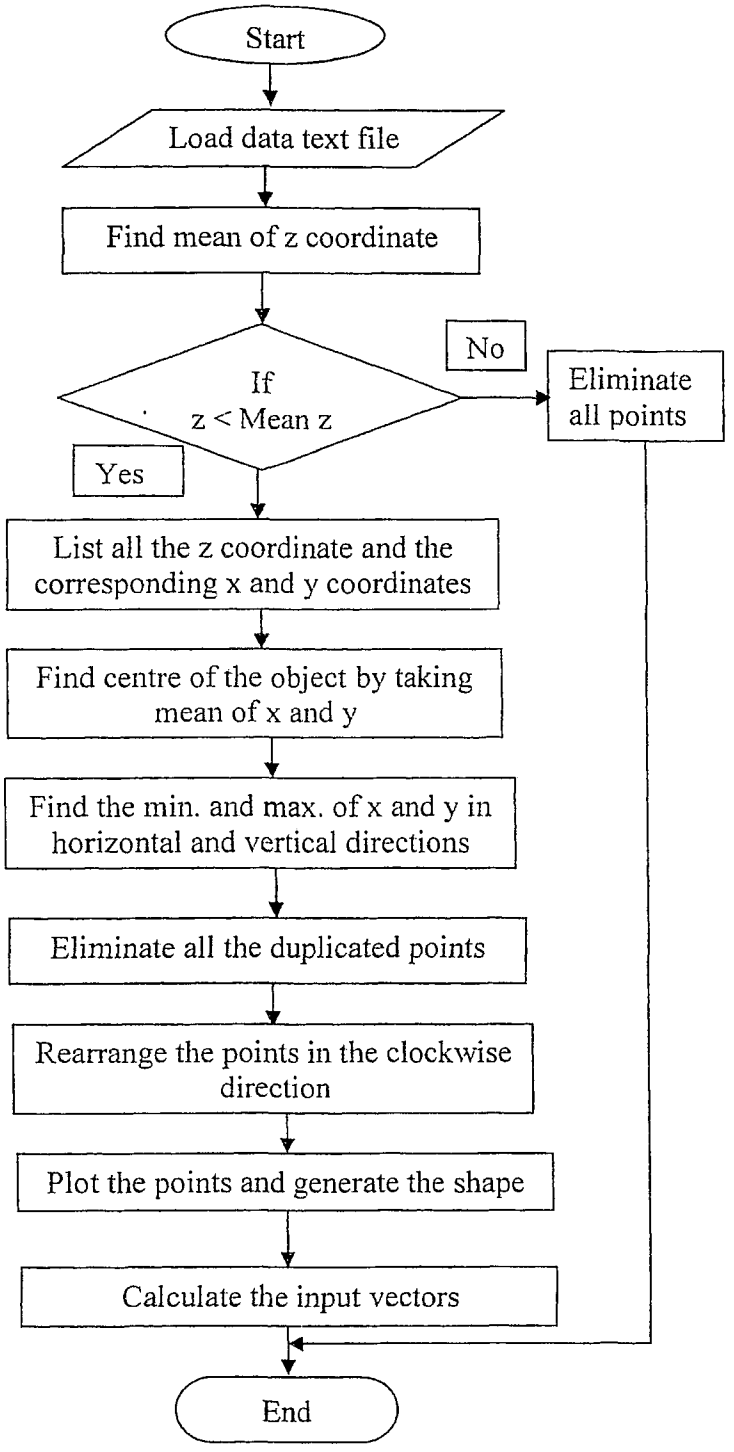


Figure 18. Flow chart for the segmentation of data for single object.

The following Figures 19, 20 and 21, show the segmented boundaries of the objects obtained from the reverse engineering cloud data by using the above segmentation scheme.

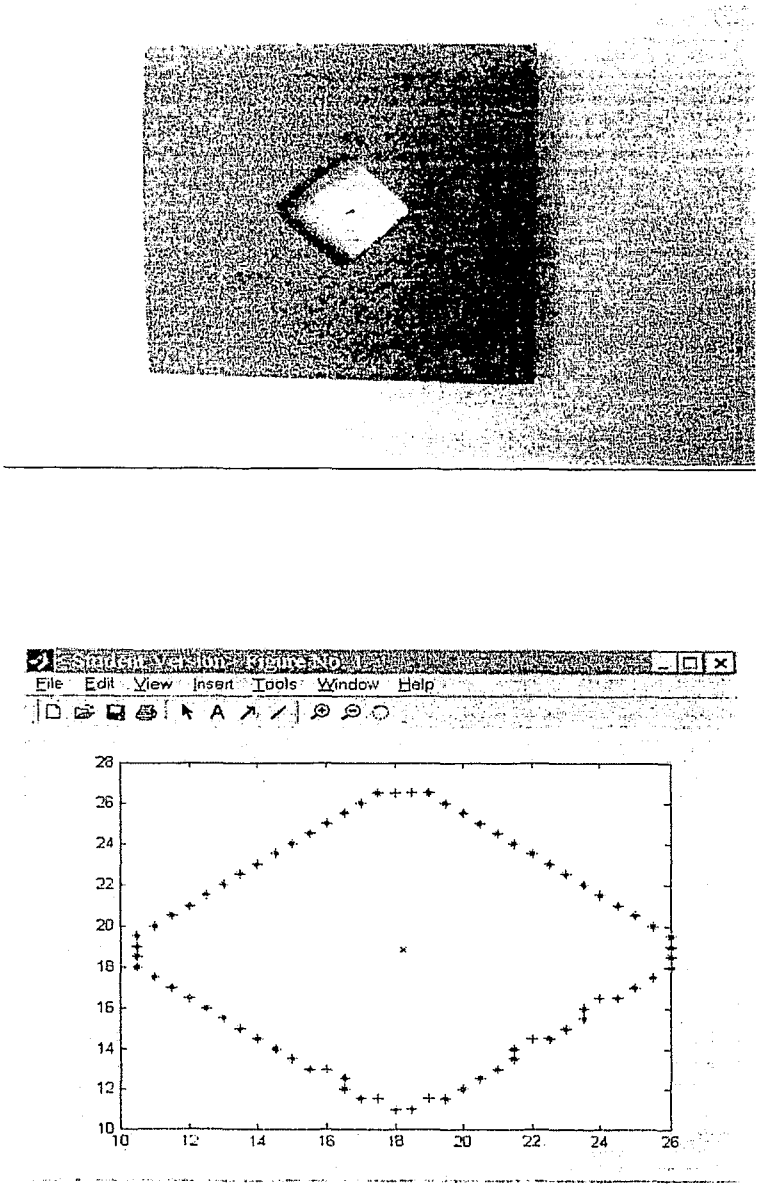


Figure 19. Test object "diamond" and its segmented boundaries.

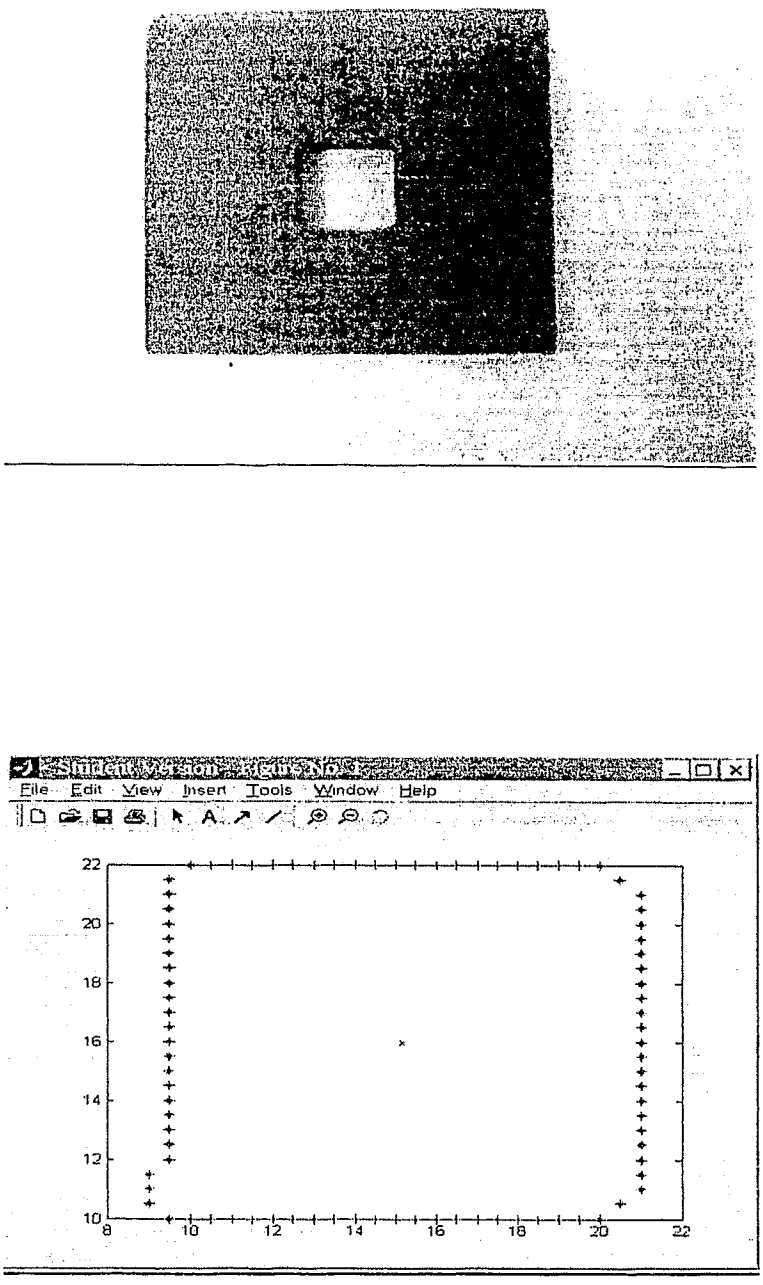
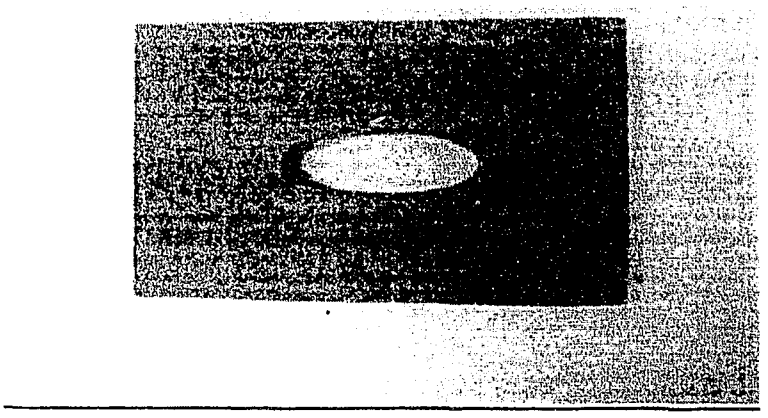


Figure 20. Test object "square" and its segmented boundaries.



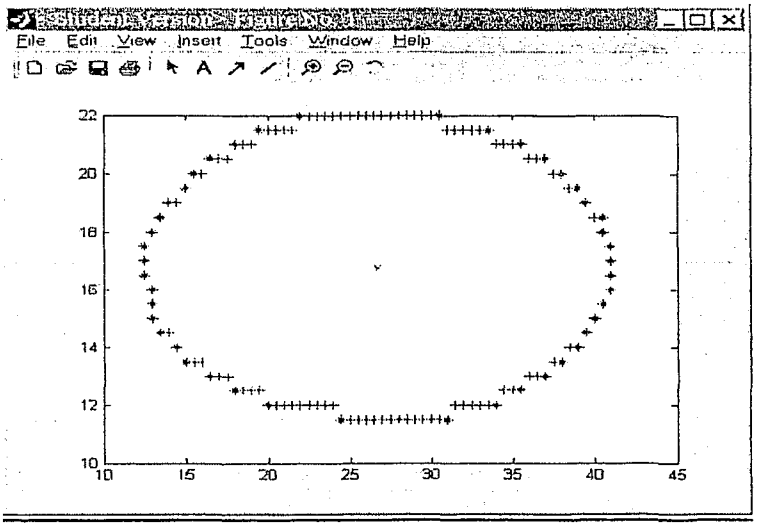


Figure 21. Test object "ellipse" and its segmented boundaries.

4.5.2 Segmentation Algorithm for Multiple Shape Objects

This segmentation algorithm looks at the physical object with multiple geometric shapes, like a wheel. Before initiating the segmentation algorithm, the object is divided into four quadrants. The segmentation algorithm then searches the object shape in each quadrant by comparing the data points that form the depth of the physical object with those on the surfaces. These segmented data are then further processed to find the boundary points of each object. Once the shapes are segmented, all four quadrants are then assembled together to calculate the important parameters that form the input vectors. A flow chart for the segmentation of multiple objects like wheel is shown in Figure 22.

The segmentation algorithm for single shape and multiple shape objects calculate the four important parameters. These four parameters (a, b, $\propto$, n) formed the input vectors. Due to the noise and incomplete data set, the parameter 'a' is calculated as the mean distance and 'b' as the maximum distance between the object boundary point and the center.

The angle $\propto$ is calculated by using the law of cosine between the two adjacent points at the object boundary. For this purpose, the distance formula is used to calculate the distance between the object boundary points and distance from centre. Also to mark the corner, the two adjacent angles that were calculated at the object boundary are added. This sum was subtracted from 180 degree. The output forms the corner if the difference varies more than 5 degree.

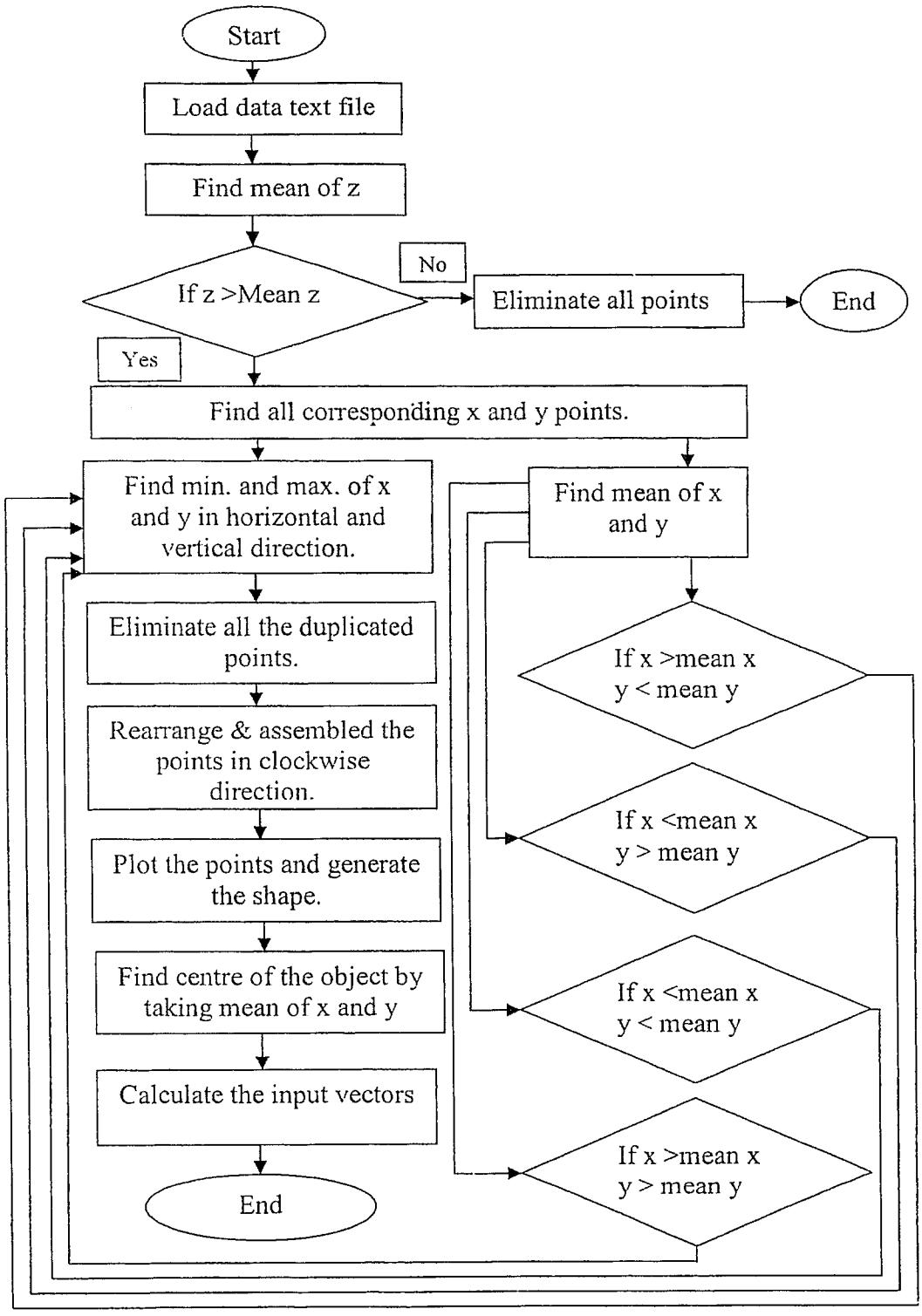


Figure 22. Flow chart for the data segmentation for multiple shapes object.

The segmented boundary of the wheel from the reverse engineering clouded data is shown in figure 23.

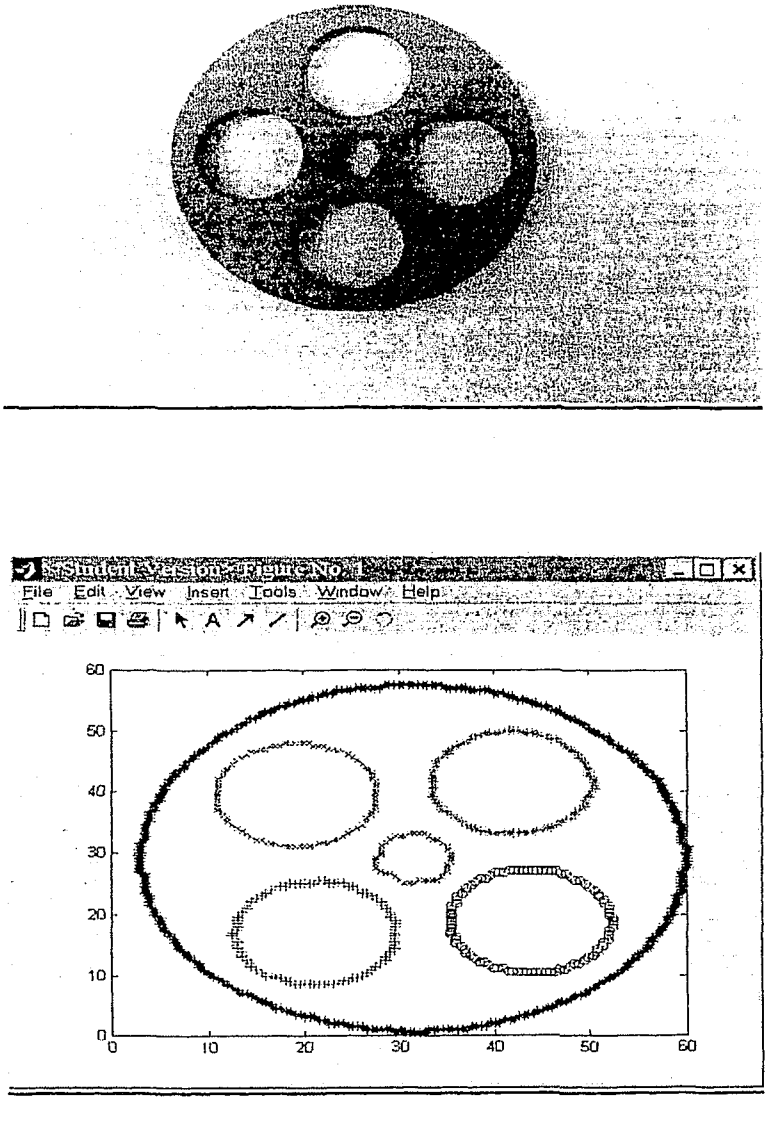


Figure 23. Test object "wheel" and its segmented boundaries.

The set of parameters (a, b, ∝, n) which are calculated from above scheme, forms the input vector. This input vector is presented to the input layer of the artificial neural network to recognize the features.

Input vector = (a, b, ∝, n) .....(4.2)

Where,

a = Mean distance from center

b = Maximum distance from center

∝ = Mean angle between two lines

n = Number of corners

To recognize a target object from the seven defined features, the desirable output can form one of seven vectors. The physical object which is associated with a circle will be mapped by the recognition system as vector (1 0 0 0 0 0 0) and a diamond as vector (0 1 0 0 0 0 0) and so on.

4.6 Back Propagation Training Algorithm:

In order to correctly recognize the feature, it is very important for the network to perform the correct mapping of the input parameters to produce the output classification. To obtain this, the weights are adjusted to the optimal values. This could be achieved by 'training' the network.

One of the problems of training a multi-layer network is how the weights of the connectors are updated. The most popular method for training a feed forward network is called back propagation.

The size of the neural network is dictated by the size of the input and the output vectors. Once the architecture has been fixed, the values of the connecting weights determine the behavior of a feed-forward network.

Let:

$\lambda_{(1,i)}$ = Input neuron value at neuron location i.

$\lambda_{(2,i)}$ = Hidden neuron value at neuron location i.

$\lambda_{(3,i)}$ = Output layer value at neuron location i.

$w_{(1,i),(2,i)}$ = Connector weight between neuron at the input and hidden layer.

$v_{(2,i),(3,i)}$ = Connector weight between neuron at the hidden and output layer.

Where i = 1 to n, depends on the respective layer as shown in the Figure 15. Therefore, the neuron value on the hidden layer can be calculated as:

$$\lambda_{(2,i)} = \sum_{m=1}^{n} w_{(1,m),(2,i)} \bullet \lambda_{(1,m)} \qquad \text{.....(4.3)}$$

Also, the output neuron values are calculated as the product of connector weights and the corresponding values at the hidden layer. A hyperbolic-tangent transfer function is used to bias the output neuron towards unity, [38] so that,

$$\lambda_{(3,i)} = (1 - \exp[-\sum_{m=1}^{n} v_{(2,m),(3,i)} \bullet \lambda_{(2,i)}]) / (1 + \exp[-\sum_{m=1}^{n} v_{(2,m),(3,i)} \bullet \lambda_{(2,i)}]) \qquad \text{..... (4.4)}$$

The errors in the output vector are used to adjust the connector weights between the output and the hidden layers and then between the input layer and the hidden layer, i.e., this error is back propagated to adjust the weights between the input and hidden layers.

Firstly, the output error is calculated. This error is the difference of the desired output and the calculated output and is given as:

$$E_{output,\,i} = \text{desired output}_{\,i} - \lambda_{(3,\,i)} \qquad \text{..... (4.5)}$$

The reflected vector is the product of the error vector, $E_{output,i}$ and the calculated output vector $\lambda_{(3,i)}$. This product is scaled by the complement of the output vector $\lambda_{(3,i)}$ for numerical stability. The reflected vector can be calculated as:

$$R_i = E_{output,i} \cdot \lambda_{(3,\,i)} \cdot (1 - \lambda_{(3,\,i)}) \qquad \text{.....(4.6)}$$

The reflected vector is used to calculate the adjustments to the connectors between the j^{th} neuron in the hidden layer and the i^{th} neuron in the output layer.

The adjustment of weights between the output and hidden layer can be calculated as:

$$dv_{(2,i),(3,j)} = A \cdot R_j \cdot \lambda_{(2,i)} \qquad \text{.....(4.7)}$$

Where,

- $dv_{(2,i),(3,j)}$ is the change of weight between the j^{th} element of the hidden layer 2 and the i^{th} element of on the output layer 3.
- $\lambda_{(2,i)}$ is the i^{th} neuron value on the hidden layer
- Constant A is the learning rate.

The error of the hidden layer is found by taking the product between the reflected vector and vector consisting of the connector weights between the hidden neuron $\lambda_{(2,\,j)}$ and the output. Again, this product is scaled by the product of the neuron and its complement for numerical stability.

$$E_{hidden,j} = \lambda_{(2,i)} \cdot (1 - \lambda_{(2,i)}) \cdot \sum_{m=1}^{n} R_m \cdot v_{(2,i),(3,m)} \qquad \text{.....(4.8)}$$

Finally, the adjustment of weights between the input layer and the hidden layer is calculated as:

$$dw_{(1,i),(2,j)} = B \cdot E_{(hidden,\,j)} \cdot \lambda_{(1,i)} \qquad .....(4.9)$$

Where,

- $dw_{(1,i),(2,j)}$ is the change of the weight between the i^{th} element of the input layer 1 and the j^{th} element in the hidden layer 2.
- $\lambda_{(1,i)}$ is the i^{th} neuron value at the input layer.
- Constant B is the learning rate.

Adjusting the weight sets between the layers and calculating the outputs is an iterative process and is repeated until the errors fall below a predetermined tolerance value. The allowable tolerance level and learning rates A and B are determined through experiment. Large error tolerances will result in a poorly performing neural network, while a very small allowable error will result in an excessively long training time.

There are two most commonly used techniques for setting up the tolerance level and learning rate for network training. [29]

i. Starting with relatively large error value and reduce it to a desired level, as training is achieved at each succeeding level.

ii. If the network fails to train the network at certain error tolerance value, then incrementally lower the learning rate.

The following is the flow of the feed forward (back-propagation) neural network procedure:

1. Initialize the weights.

2. Present the input data vectors.
3. Operate the neural network.
4. Compute the error between desirable and calculated output.

If the error is smaller than the preset tolerance, stop the algorithm (Current weights are the final weights).

5. Propagate the errors back to all the units towards the input layer.
6. Compute the adjusting value according to the error and adjust all the weights.
7. If number of iterations exceeds the predetermined number, stop (unsuccessful, adjust learning rates and try the procedure again).

Once the training is concluded and the weights are adjusted, the network is available to use for features identification.

A flow chart for the feature recognition from geometric reverse engineering data is shown in figure 24.

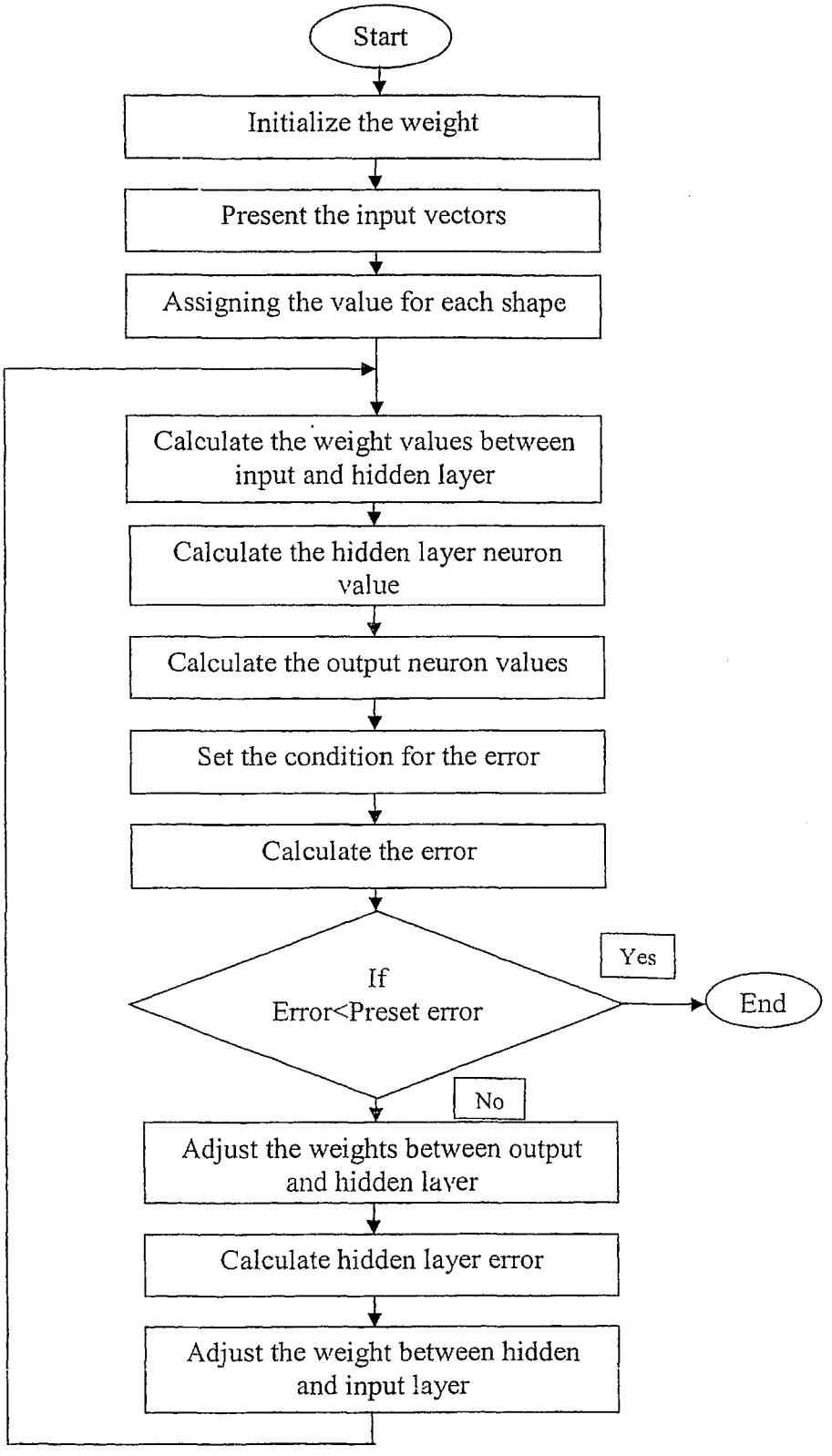


Figure 24: Flow chart for neural network-based feature recognition.

Chapter 5 – Testing of the Neural Network Algorithm

In this chapter, the training and the testing of the algorithm are discussed. The algorithm was first trained with manually created synthetic data and then tested with noisy data. The algorithm was subsequently tested with real reverse engineered data to fully test the robustness of the neural network.

5.1 Training of the Neural Network

The neural network recognition algorithm must first be trained to recognize features from the seven different shapes. Manually created synthetic data is presented to the neural network for training. The desired output for the seven shapes formed the seven dimensional vectors are as follows:

Circle: 1 0 0 0 0 0 0

Diamond : 0 1 0 0 0 0 0

Ellipse: 0 0 1 0 0 0 0

Rectangle: 0 0 0 1 0 0 0

Square: 0 0 0 0 1 0 0

Triangle: 0 0 0 0 0 1 0

Wheel: 0 0 0 0 0 0 1

Figure 25 represents the sample input vectors and corresponding desired outputs.

```
6.0000 6.0000 0.0936 0
1 0 0 0 0 0 0
7.0000 7.0000 0.0887 4
0 1 0 0 0 0 0
9.0000 14.0000 0.0516 0
0 0 1 0 0 0 0
9.0000 12.0000 0.0494 4
0 0 0 1 0 0 0
7.0000 7.0000 0.0704 4
0 0 0 0 1 0 0
5.0000 6.0000 0.1093 3
0 0 0 0 0 1 0
28.0000 28.0000 0.0195 0
0 0 0 0 0 0 1
```

Figure 25. Sample input vectors presented to neural network.

In Figure 25, the first line shows the input vectors while the second line represents the desired output vectors. The next two lines show the sets of input and output vectors.

The sample training vectors file were created by using random number generator. The first set of sample input vectors were the perfect dimensions for each shape. A random number generator is then used to create the rest of the input sample vectors. This input sample file was presented to the neural network for the training.

Since the solution to the input vectors during the training phase is known in advance, the neuron values are corrected by triggering the back - propagation training algorithm.

The transfer function performs the task to process the neuron values from the input layer to the hidden layer or from the hidden layer to the output layer to generate the desired output. For this work, the hyperbolic-tangent transfer function is used between the hidden and output layers to train the neural network algorithm.

Adjusting the error tolerance to obtain the desirable results played an important role in training the neural network. The error tolerance must be set before the neural network algorithm stops correcting the neuron connectors. One technique to train the neural network algorithm is to start with a large error value and then successively lower the value as the network starts learning. Through testing, an error value of 0.05 and learning rate values of 0.4 for A and 0.2 for B showed good agreement and gave reliable results.

Also, it is important to decide the number of training vectors presented to the neural network. [2, 12] It played an important role in the training of the neural network used in deciding the type of feature that was being presented. For this purpose, synthetic data are created to train the neural network algorithm. The neural network algorithm was also tested after adding 10% noise in the synthetic data. The results are mixed, as the neural network algorithm is able to recognize most of the desired features depending upon the weight values to initiate the training. It was found that a minimum of 90 vectors is required to train the neural network to recognize seven different types of features.

During training, it was also been found that the network was sensitive to the order the training vectors presented. For example, if the first vector pairs presented were to represent a circle, the network biased towards the circle feature. Table 1 shows the output for the learning rate values of 0.4 for A and 0.2 for B. The initial weights between the input and hidden layers and hidden to output layers are w = v = 0.005 for k = 400 iterations.

	Desired Classification							Network classification						
	Cir	Dia	Elps	Rec	Sqr	Tri	Wl	Circle	Diamond	Ellipse	Rectangle	Square	Triangle	Wheel
Circle	1	0	0	0	0	0	0	0.9776	0.0224	0.0224	0.0224	0.0224	0.0224	0.0224
Diamond	0	1	0	0	0	0	0	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Ellipse	0	0	1	0	0	0	0	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Rectangle	0	0	0	1	0	0	0	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Square	0	0	0	0	1	0	0	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Triangle	0	0	0	0	0	1	0	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Wheel	0	0	0	0	0	0	1	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table 1: Set of classification values with sigmoid transfer function.

Table 2 represents the outputs with the hyperbolic-tangent transfer function for the same parameters (A=0.4, B=0.2, w = v = 0.005 and k =400).

	Desired classification							Network classification						
	Cir	Dia	Elps	Rec	Sqr	Tri	Wl	Circle	Diamond	Ellipse	Rectangle	Square	Triangle	Wheel
Circle	1	0	0	0	0	0	0	0.0615	0.0033	0.0033	0.0033	0.0033	0.0033	0.0033
Diamond	0	1	0	0	0	0	0	0.0036	0.9687	0.0030	0.0030	0.0030	0.0030	0.0030
Ellipse	0	0	1	0	0	0	0	0.0025	0.0107	0.9722	0.0022	0.0022	0.0022	0.0022
Rectangle	0	0	0	1	0	0	0	0.0018	0.0040	0.0067	0.9676	0.0017	0.0017	0.0017
Square	0	0	0	0	1	0	0	0.0000	0.0079	0.0000	0.0078	0.9660	0.0000	0.0000
Triangle	0	0	0	0	0	1	0	0.0000	0.0001	0.0000	0.0001	0.0134	0.0000	0.0000
Wheel	0	0	0	0	0	0	1	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Table 2: Set of classification values with hyperbolic-tangent transfer function.

Where NaN in Table 2, is the IEEE arithmetic representation in MATLAB for Not-a-Number. A NaN is obtained as a result of mathematically undefined operations like 0.0/0.0 and Inf-inf.

During the training, it has also been found that the network recognized the circle feature by setting the initial weights w = v = 0.1. It has also been found that the network is not able to distinguish the diamond feature from rectangle and square features if the learning rate values are set as A=B=0.2. By setting A = 0.4 and B=0.2, the network clearly recognized the diamond, square and rectangle features. The neural network algorithm was tested on a Dell PIII computer with 256 RAM, and the total processing time to recognize the features is approximately four seconds.

The network trained for the geometric shapes recognized most of the shapes as shown in table 2, but failed to recognize the wheel. One of the possible reasons is that the wheel has multiple shapes (circle, hexagon, etc.) and the network may not be able to recognize all the shapes together.

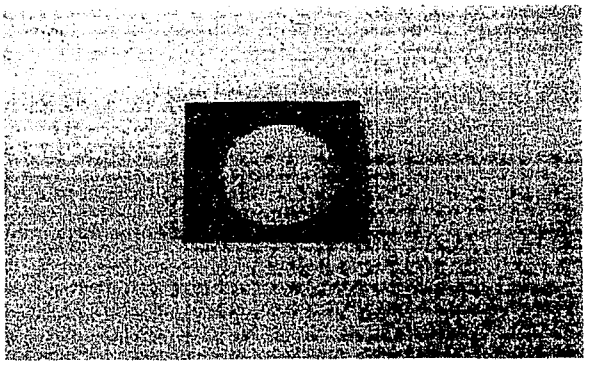
5.2 Testing of the Neural Network

After the training was concluded, the neural network should respond to the items not in the training set. One of the approaches to do this is to select the noise option. The usefulness of the neural network is measured from its response to noisy data, but at the same time the intention for the neural network is not to tolerate unlimited noise. Therefore, to fully test the ability of the network, the data corrupted with 10% noise was presented to the network. For this purpose, a separate sample input file was created by introducing 10% noise. This sample input file corrupted with noise was presented to the

neural network. The outputs are significant as the network was able recognize most of the geometric shapes.

5.3 Testing of the Algorithm with Real Reverse Engineering Data

To test the robustness of the segmentation algorithm developed in section 4.6 and the neural network algorithm to recognize features, real reverse engineering data derived from three test samples of each object feature at different orientations and dimensions were used. The results of segmentation algorithm are shown in Figures 26, 27, 28, 29, 30 and 31.



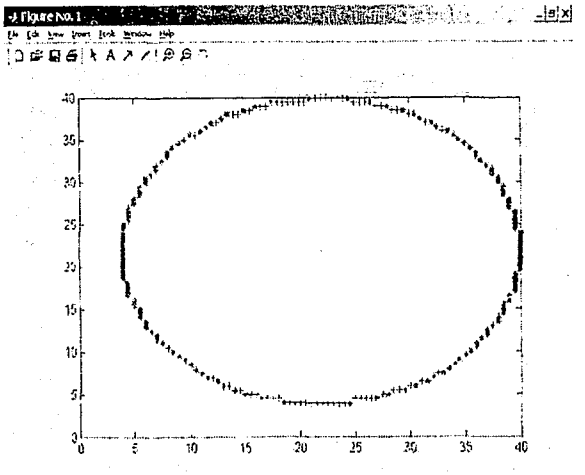


Figure 26. Test sample "circle" and its segmented boundary

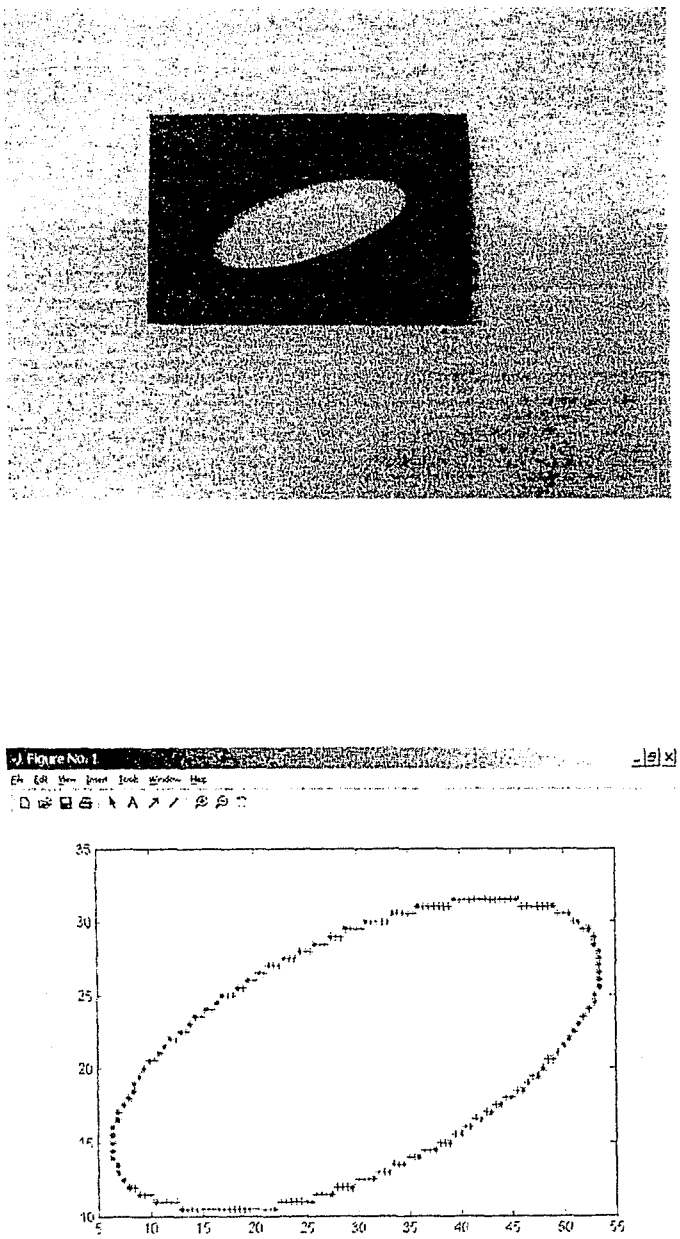
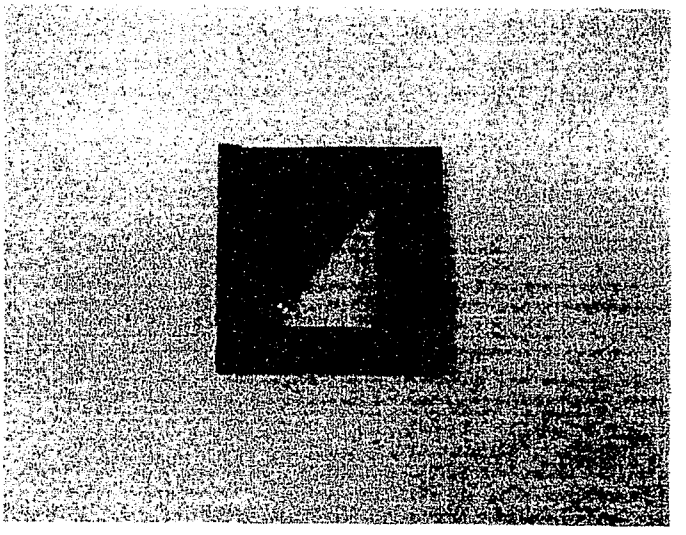


Figure 27. Test sample "ellipse" and its segmented boundary



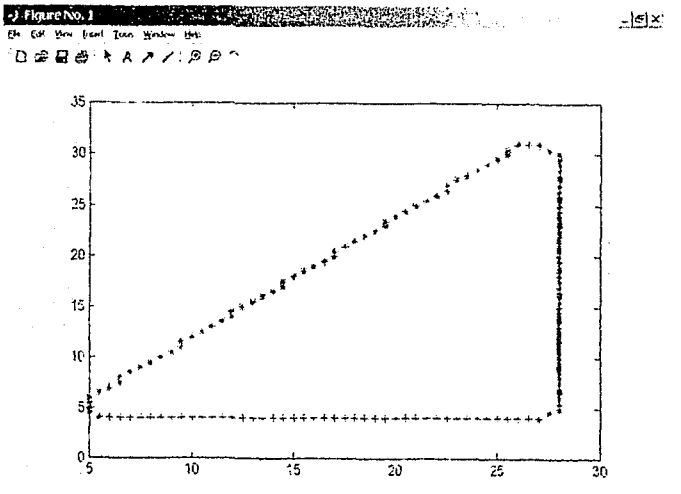


Figure 28. Test sample "triangle" and its segmented boundary

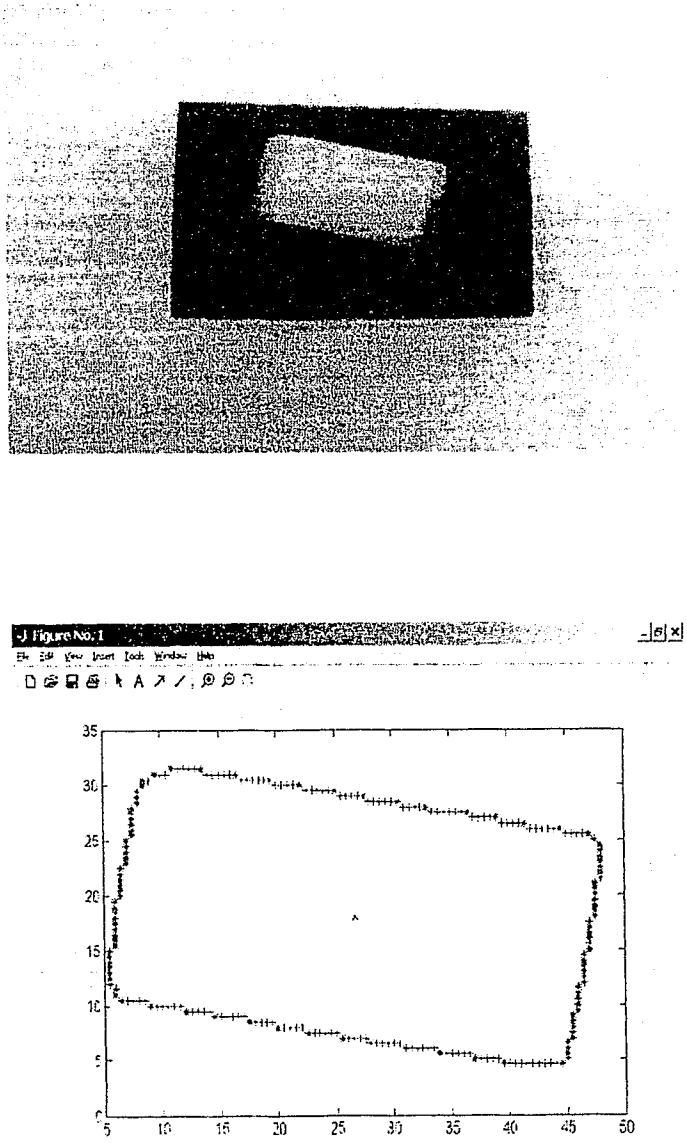
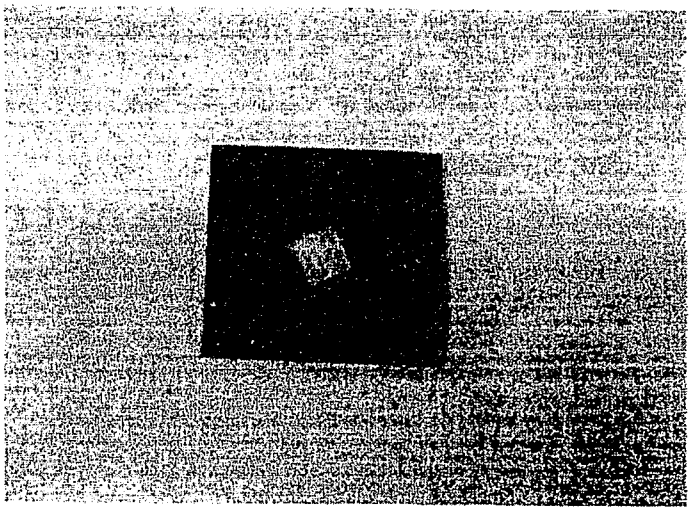


Figure 29. Test sample "rectangle" and its segmented boundary



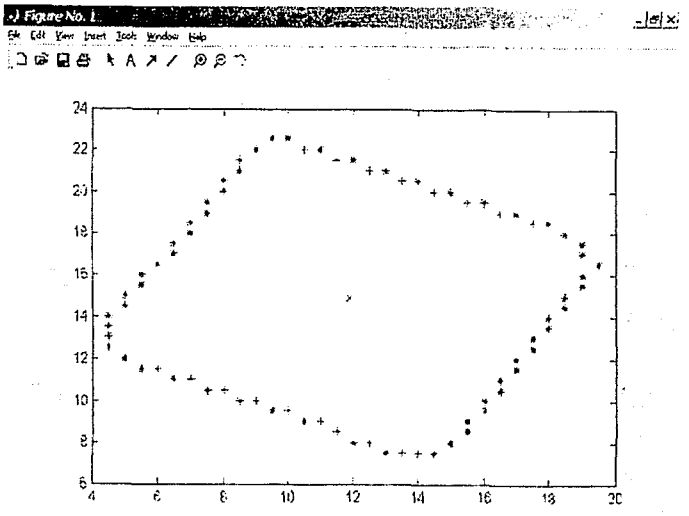
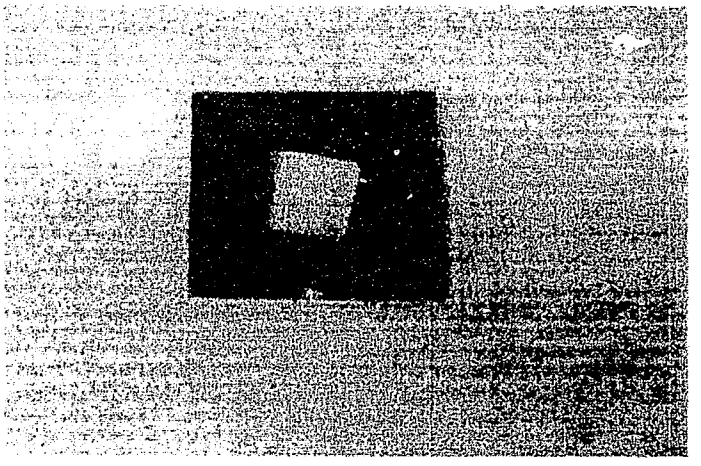


Figure 30. Test sample "diamond" and its segmented boundary



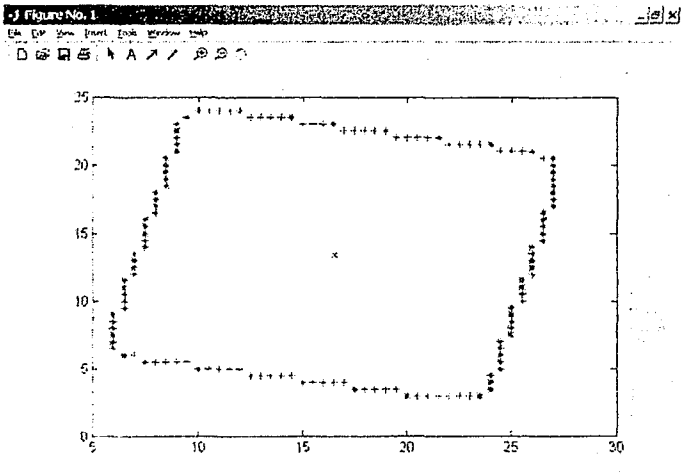


Figure 31. Test sample "square" and its segmented boundary

The results from both the segmentation and the neural network algorithms are promising. The rest of the test samples and segmentation algorithm results are shown in Appendix A.

The neural network was used to recognize the features of the object described in Section 1.4. The outputs are satisfactory as the network is able to recognized most of the geometric object features. Figure 32 represents the input vectors derived from the real reverse engineering data from the objects described in Section 1.4 and presented to the network for feature recognition.

```
a=5.8159; b=6.1433; alpha=5.3634*pi/180; nc=0;
Circle = [1 0 0 0 0 0 0];
a=6.7775; b=7.8032; alpha=5.0817*pi/180; nc = 4;
Diamond = [0 1 0 0 0 0 0];
a=9.1873; b=14.2755; alpha=2.9577*pi/180; nc = 0;
Ellipse = [0 0 1 0 0 0 0];
a=9.2376; b=11.5956; alpha=2.8298*pi/180; nc = 4;
Rectangle = [0 0 0 1 0 0 0];
a=7.6679; b=8.2436; alpha=4.0350*pi/180; nc = 4;
Square = [0 0 0 0 1 0 0];
a=4.5739; b=6.0755; alpha=6.2626*pi/180; nc = 3;
Triangle = [0 0 0 0 0 1 0];
a=28.3722; b=28.7355; alpha=1.1181*pi/180; nc = 0;
Wheel = [0 0 0 0 0 0 1];
```

Figure 32. Input vectors derived from real reverse engineering data.

The segmentation algorithms developed for this work are able to correctly identify the boundaries of the physical objects from the huge, noisy, incomplete and unstructured scanned data and then calculate the important parameters of each object that

form the input vectors for the neural network. The feed-forward neural network used for this work to recognize features from reverse engineered data are efficient in terms of:

1. Processing Time; The neural network algorithm took approximately four seconds for processing to correctly recognize the feature.
2. Efficiency; The neural network algorithm when used with the hyperbolic-tangent transfer function is more efficient in recognizing the features of the geometric object as compared to the sigmoid transfer function which was biased towards one shape.
3. Tolerance Level; The neural network algorithm works within the error tolerance level of 0.05 and learning rates of 0.4 for A and 0.2 for B.

The results of the technique (Feed-Forward Neural Network) used in this work for features recognition and the algorithms developed for segmentation to obtain important parameters are promising to recognize the feature, both in terms of efficiency and processing time.

Chapter 6 – Conclusion and Future Work

In this work, a feed-forward neural network based feature extraction system for geometric reverse engineering data is presented. Geometry and topology data associated with the object boundary are derived from methods described earlier in Section 4.5, and an appropriate input vector for the neural network algorithm was derived.

The main emphasis is to construct a CAD model from geometric reverse engineering data by applying a feature recognition technique. Neural network feature recognition from reverse engineering is promising. Its capability in handling the noisy and often incomplete data set confirms its desirable feature over conventional rule-based algorithms. Two segmentation algorithms, one for a single shape and the second for a set of multiple shapes (wheel), were developed to first segment the boundary of the object from huge and unstructured cloud data and then calculate the important parameters that form the input vector. A feed-forward neural network is used to recognize the features. Both, the segmentation and neural network algorithms are programmed in MATLAB.

The testing of the algorithm shows promising results, as the neural network was able to recognize the features with an error tolerance of 0.05.

Whether this technique can provide sufficient recognition capability to serve a universal set of features in any category of physical parts and how the artificial neural network should be structured for this purpose require further research. Although considerable work to extract features from scan data for geometric shapes has been realized. Expansion of the number of defined features, multiple shape objects as well adding to the number of surfaces of which the feature can be composed, for example, filleted corners and other types of curved surfaces and finer details will be a future extension.

Appendix A

Test samples and their segmented boundaries

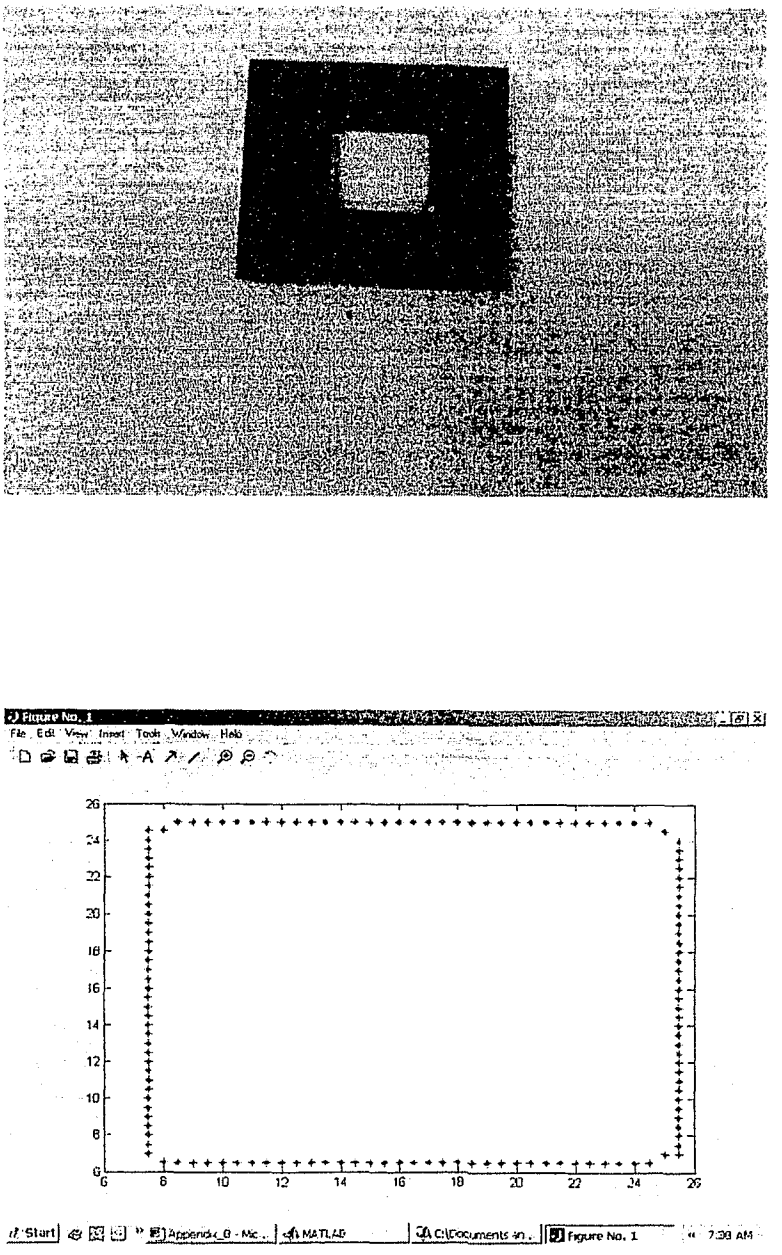


Figure 33. Test sample "square" and its segmented boundary

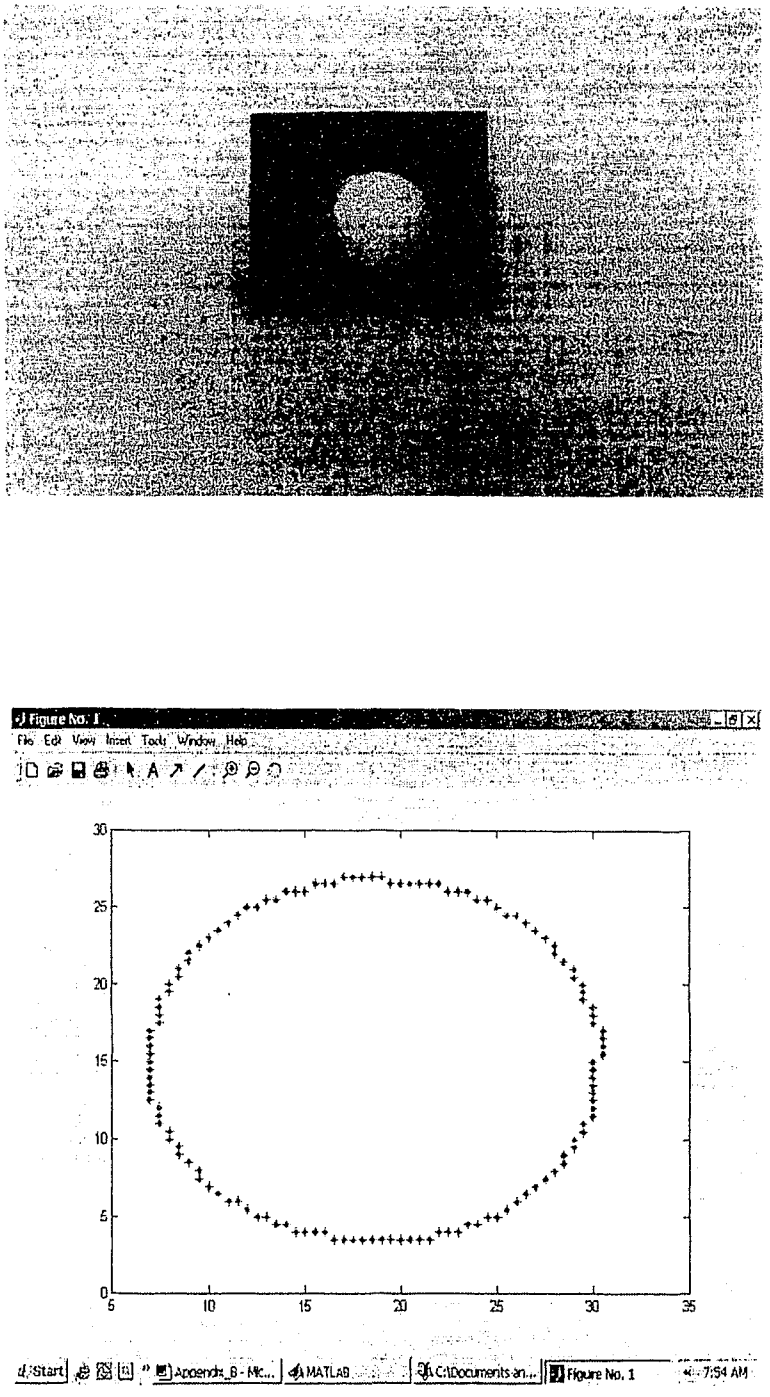
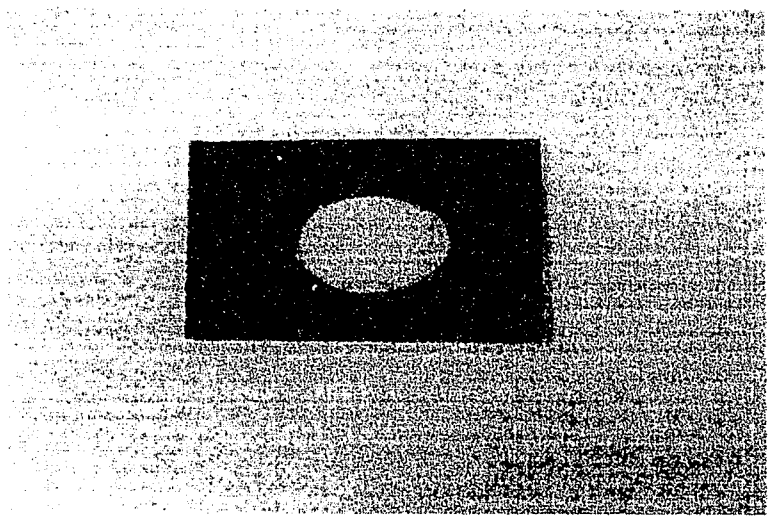


Figure 34. Test sample "circle" and its segmented boundary



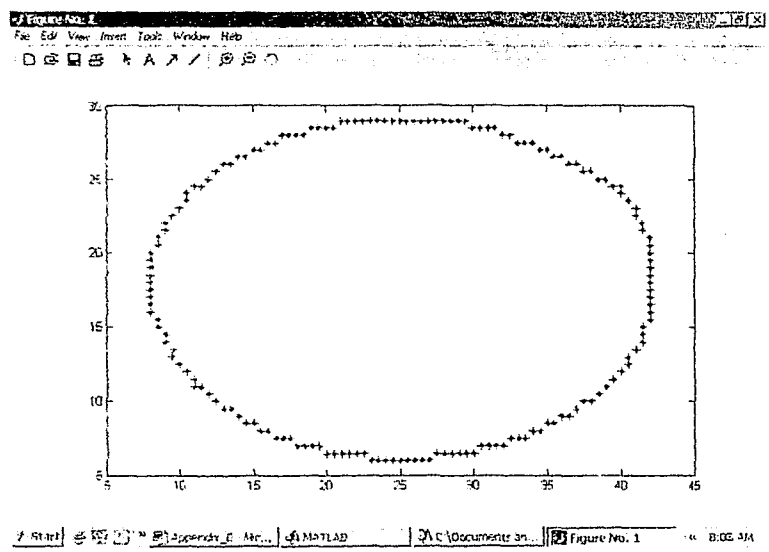
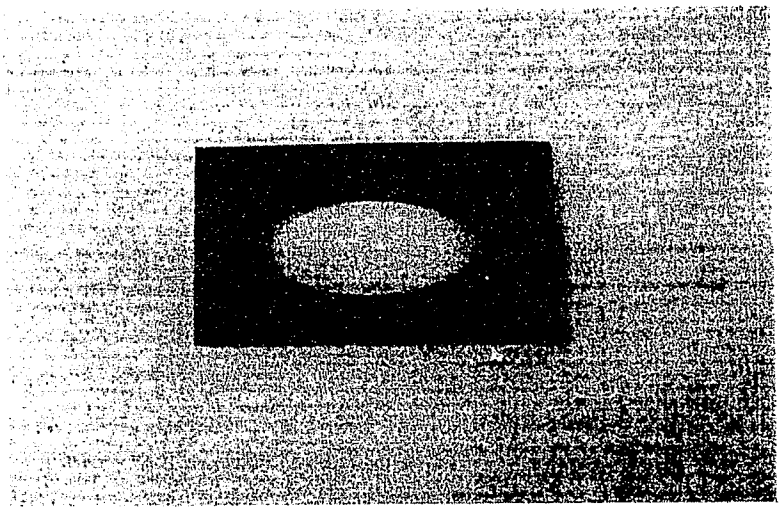


Figure 35. Test sample "ellipse" and its segmented boundary



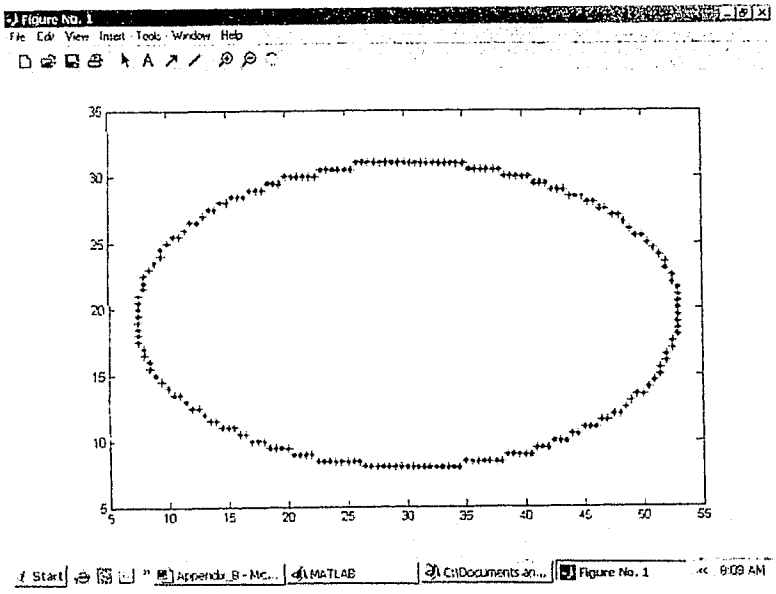


Figure 36. Test sample "ellipse" and its segmented boundary

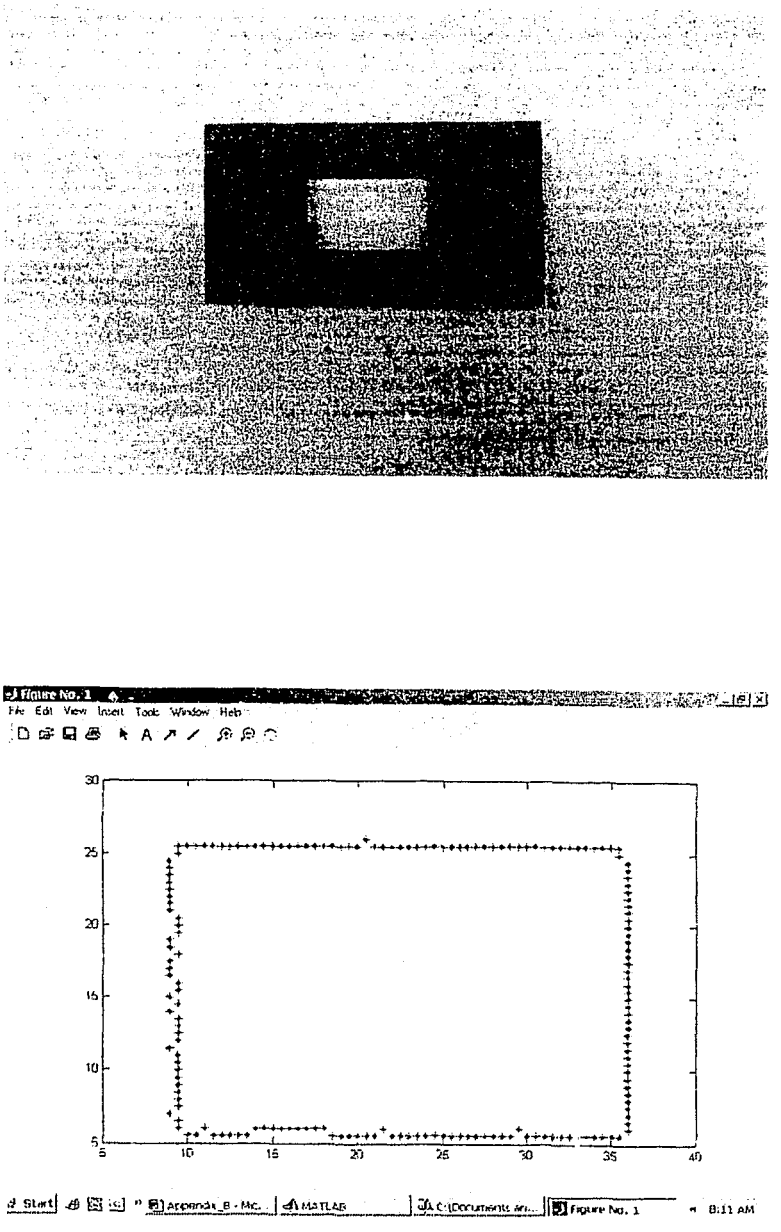


Figure 37. Test sample "rectangle" and its segmented boundary

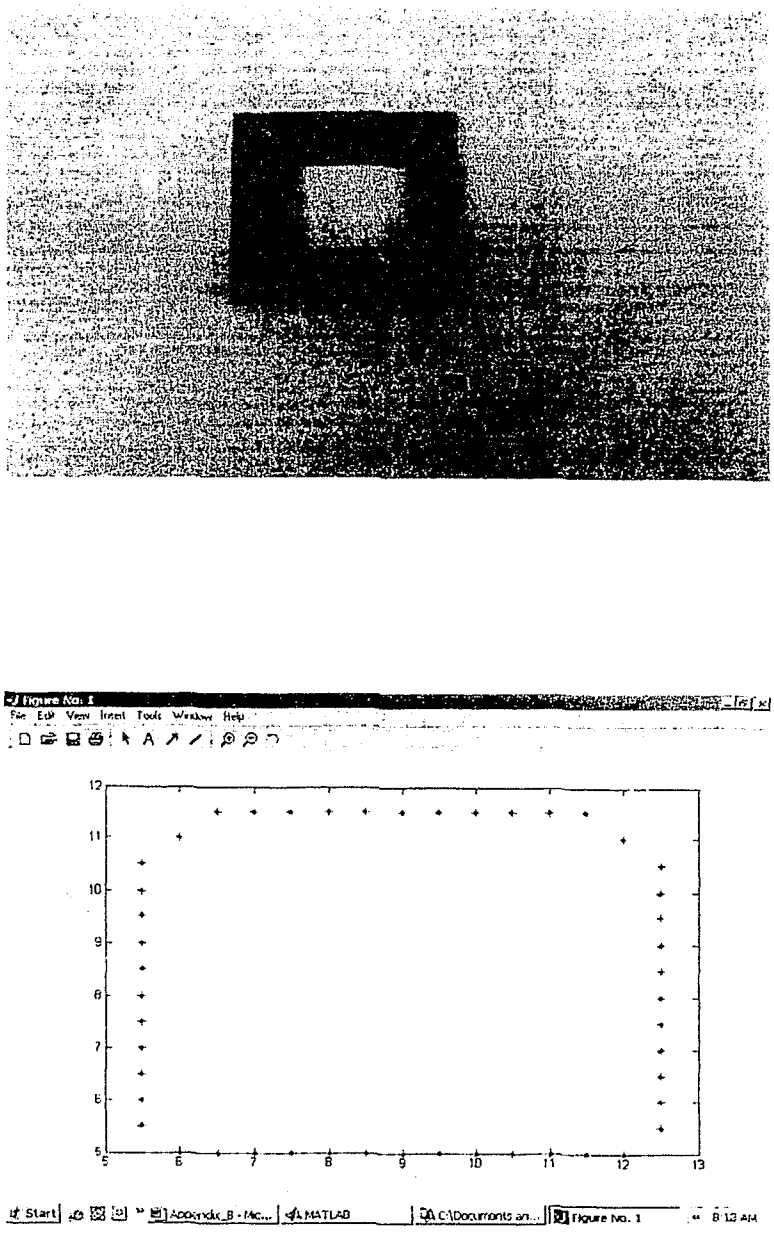


Figure 38. Test sample "square" and its segmented boundary

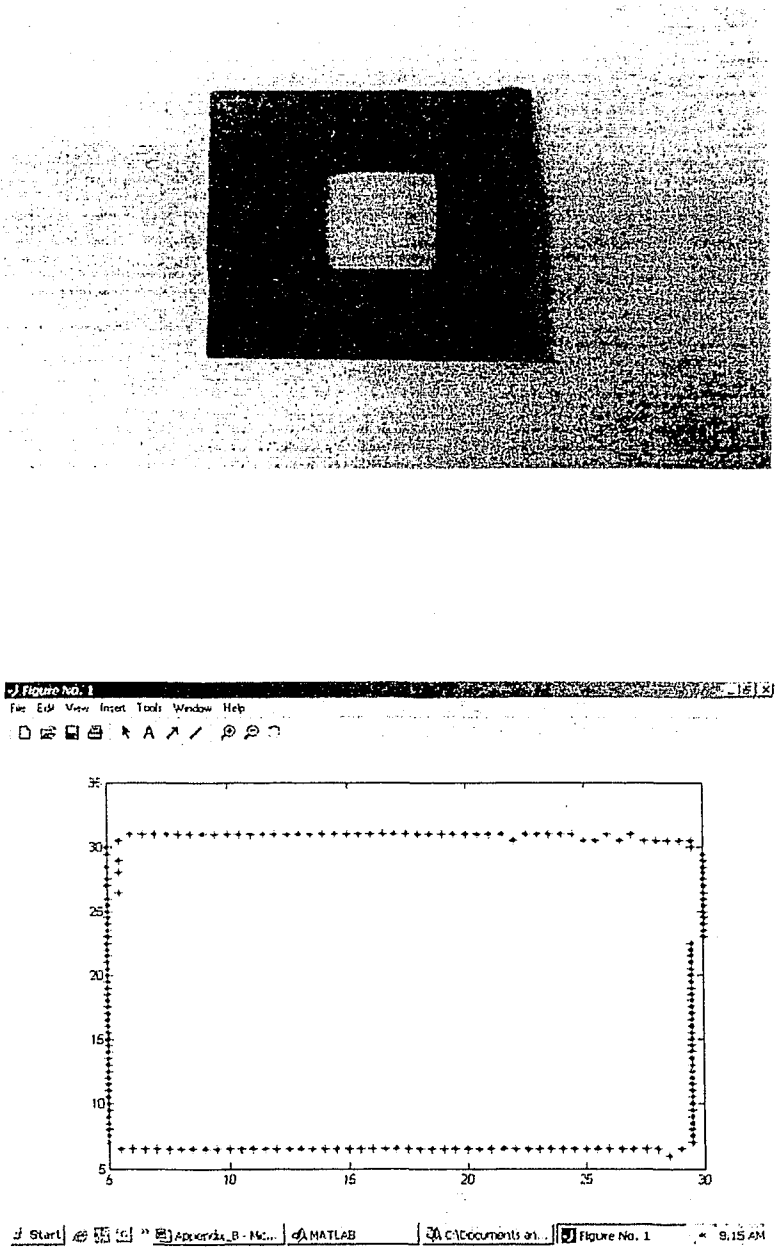


Figure 39. Test sample "square" and its segmented boundary

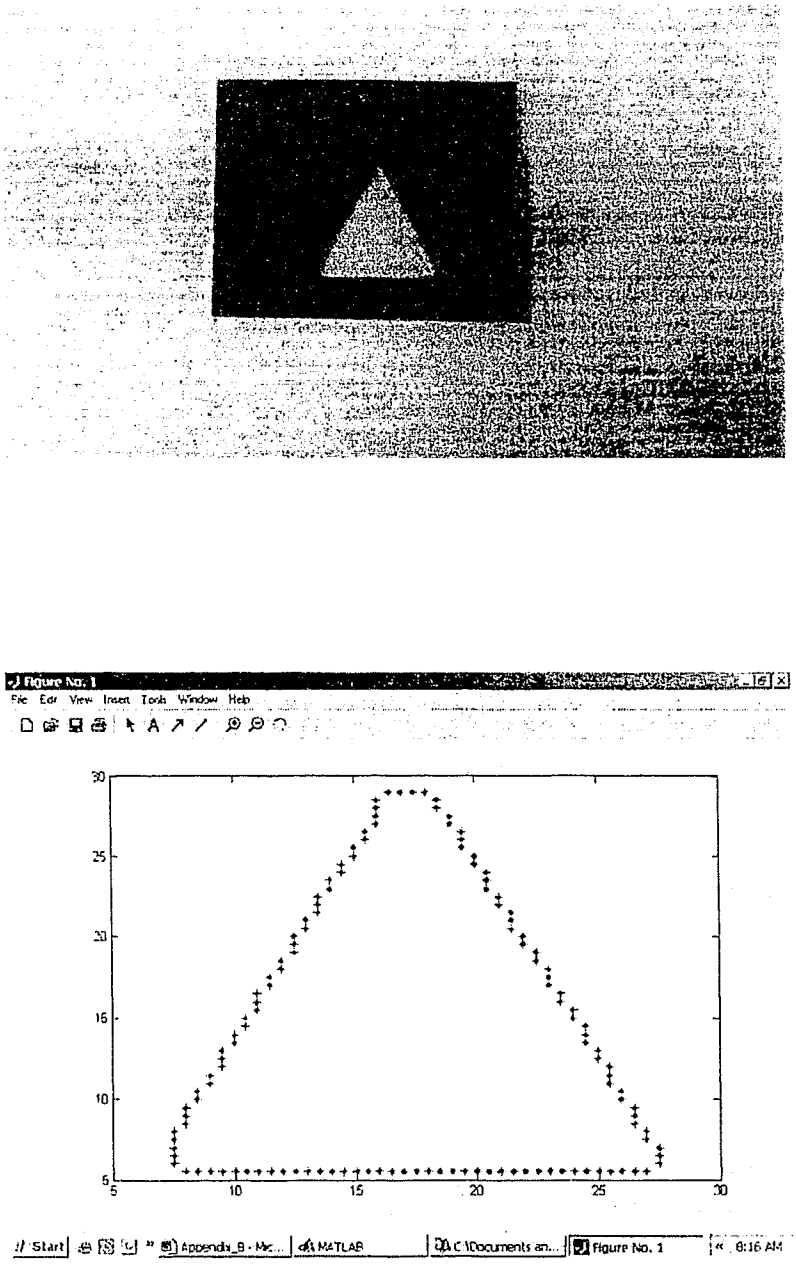
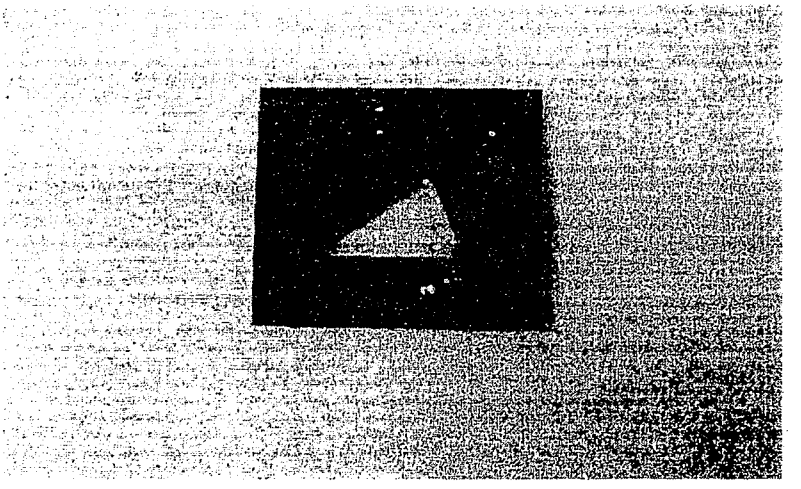


Figure 40. Test sample "triangle" and its segmented boundary



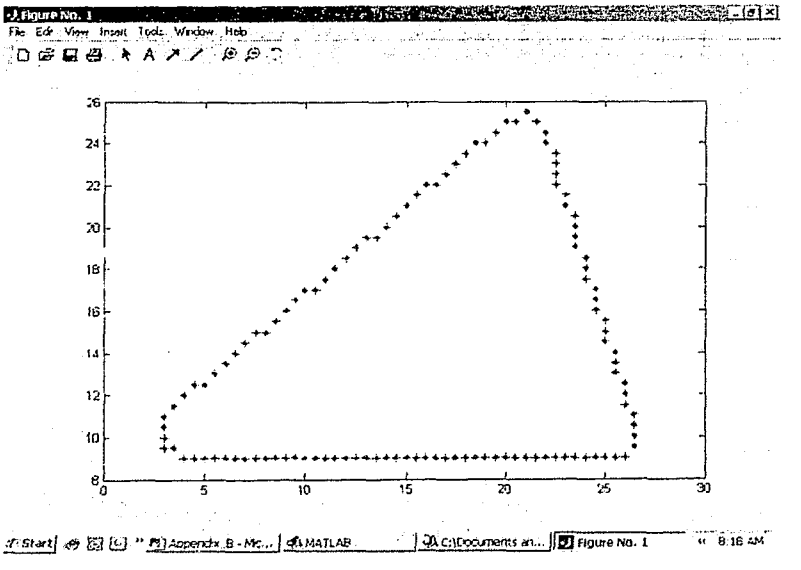


Figure 41. Test sample "triangle" and its segmented boundary

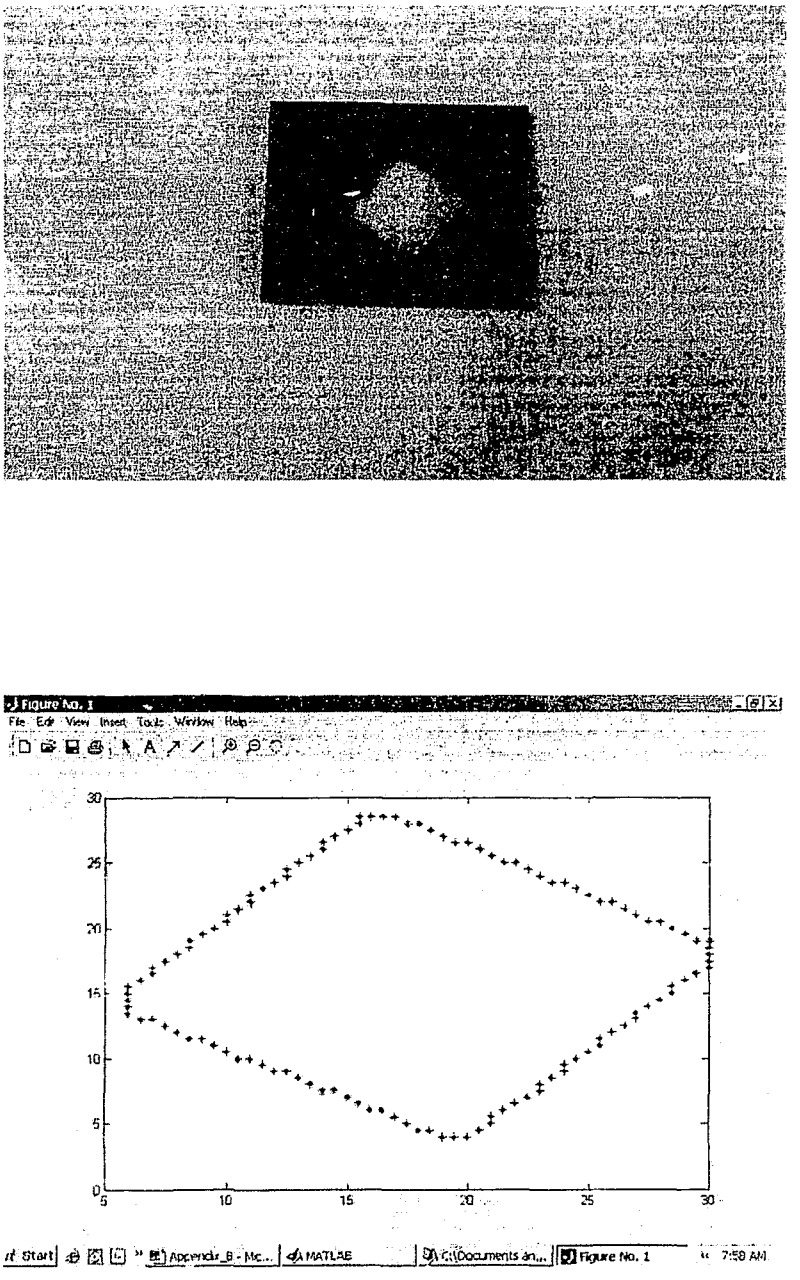


Figure 42. Test sample "diamond" and its segmented boundary

Appendix B

Glossary of Terms

Term	Definition
ANN	Artificial Neural Network – a computer algorithm based on the architecture of a biological brain.
A Priori	Information that was known beforehand.
B-rep	Boundary Representation – A method used by CAD Programs to model a solid with its boundaries.
Back-Propagation	Method to update connector weights in multi-layer neural networks based on the error.
CAD	Computer Aided Design – A computer program that allows for design on a computer.
CAM	Computer Aided Manufacturing - a computer program to aid in the planning of a manufacturing process.
CCD	Charged Coupled Device – a light sensitive microchip used to capture images in video camera.
Cloud Data	Term used to describe the cloud like structure of data collected by scanner or sensor.
CMM	Coordinate Measuring Machine – a precise machine used in industry to measure surface points
Competitive learning	Type of neural network where in training, there can only be one correct neuron.
Connectors	Weighted links between neurons.
Feature	A combination of geometric entities that together have a meaningful purpose.
Free-form surface	A surface not made of any geometric primitives.
Kohonen SOM	Self Organizing Map – A neural network based on competitive learning among neighboring neurons
Neuron	A node in a neural network.

Topology	The spatial relationship of different surfaces to each other.
Touch probe	A sensor used for making measurements.
Voxel bin	A cubic volume derived from a large volume.

Appendix C

MATLAB Program codes

```
%-------------------------------------------------------------------------
%  MATLAB code for segmentation of data for single shape object
%-------------------------------------------------------------------------

clear all
close all

load circle.txt;   %Loading Circle text file
x = circle(:,1);
y = circle(:,2);
z = circle(:,3);
mz = mean(z);      %Average (mean) of z ordinate
[n f]=size(z);
pitch = 0.5        % Scanning pitch

c = 0;
for m = 1:n
if z(m) < mz
   c = c+1;
   zx(c,1)=x(m);
   zy(c,1)=y(m);
end
end

% ----------------Centre of the geometric object----------------------

zxm = mean(zx);    % Mean of x ordinate
zym = mean(zy);    % Mean of y ordinate

%--------------------Boundary points for constant y--------------------

y_min = min(zy);   % Mininum values of y
y_max = max(zy);   % Maximum values of y
a=zy.';            % y values in rows.


%n1 -number of different points on y after eleminate the noise.

y1=y_min:pitch:y_max;
[f1,n1]=size(y1);

for j = 1:n1
   y_num = find(a==y1(j));
   x_val = zx(y_num);
```

```
  x_min = min(x_val);
  x_max = max(x_val);
        p(j) = x_min;
  q(j) = x_max;
end

%----------------------Boundary points for constant x--------------------

x_min1 = min(zx);   % Mininum values of y
x_max1 = max(zx);   % Maximum values of y
a1=zx.';            %x values in rows.

x1=x_min1:pitch:x_max1;
[f2,n2]=size(x1);   % size of the data

for u = 1:n2
  x_num = find(a1==x1(u));
  y_val = zy(x_num);
  y_min1 = min(y_val);
  y_max1 = max(y_val);
        pp(u) = y_min1;
  qq(u) = y_max1;
end
%-----------------Elimination of duplicate points----------------

s=0;
for k=1:n2
  for g=1:n1
 if (((x1(k)==p(g)|x1(k)==q(g))&(y1(g)==pp(k)|y1(g)==qq(k)))
|((x1(k)==p(g)|x1(k)==q(g))&(y1(g)~=pp(k)|y1(g)~=qq(k)))
|((x1(k)~=p(g)|x1(k)~=q(g))&(y1(g)==pp(k)|y1(g)==qq(k))))
  s=s+1;
  X(s)=x1(k);
  Y(s)=y1(g);
end
end
end

%------------Re-arrange the boundary points in clockwise order----------------

XC=X([1:6 9 10 13 14 16 17 19:2:51 54 53 57 63 62 61 67:-1:64 60 59 58 56 55 52:-2:18
15 11 12 7 8]);
YC=Y([1:6 9 10 13 14 16 17 19:2:51 54 53 57 63 62 61 67:-1:64 60 59 58 56 55 52: 2:18
15 11 12 7 8]);

%n3 -no.of boundary points after eleminate the same points.
```

```
[f3,n3]=size(XC);

%----------------Distance between boundary points and centre----------------

for w=1:n3
  d1(w) = sqrt((YC(w)-zym)^2 + (XC(w)-zxm)^2);     %distance from the center
end

%-----------------Calculating the angles between the lines----------------

%d2 distance between 2 adjacent point.
%ang1, angle between 2 lines by using cosine rule(from center).
%ang2, angle between 2 lines by using cosine rule (from 1st coner)
for w1=1:(n3-1)
  d2(w1) = sqrt((YC(w1)-YC(w1+1))^2 + (XC(w1)-XC(w1+1))^2);
  ang1(w1) = acos(((d1(w1))^2 +(d1(w1+1))^2-
(d2(w1))^2)/(2*d1(w1)*d1(w1+1)))*180/pi;
  ang2(w1) = acos(((d1(w1))^2 +(d2(w1))^2-
(d1(w1+1))^2)/(2*d1(w1)*d2(w1)))*180/pi;
end

%--------------------Calculating the corner formed--------------------------

%adding the 2 adjacent angles on the a boundary point to find the corner.

for w2=1:(n3-2)
  ang3(w2) = (ang2(w2+1)+(180-(ang2(w2)+ang1(w2))));
end
%ang3.'

num_of_coners=find(ang3==180);
[fc,nc]=size(num_of_coners);

disp('      Y      X')
[XC.',YC.']
disp('Distance from center')
d1.'
disp('ang(center)')
ang1.'

plot(XC,YC,'x')
hold on
plot(zxm,zym,'x')

disp('Mean distance from center')
mean(d1)
```

```
disp('Maximum distance from center')
max(d1)
disp('Mean angle between two lines')
mean(ang1)
disp('Number of coners')
nc-4

%fprintf('\n')
%fprintf(' %12.1f\t %12.1f\n %12.1f\t %12.1f\n',F0,F1,F2,F3)
```

```
%-------------------------------------------------------------------------------
% MATLAB codes for data segmentation for multiple shape object (wheel)
%-------------------------------------------------------------------------------

clear all
close all

load wheel.txt;    %Loading wheel text file
x = wheel(:,1);
y = wheel(:,2);
z = wheel(:,3);
mz = mean(z);      %Average (mean) of z ordinate
[n f]=size(z);
pitch = 0.5        % Scanning pitch

%mean of z < z values
c = 0;
for m = 1:n
if z(m) > mz
   c = c+1;
   zx(c,1)=x(m);
   zy(c,1)=y(m);
   zz(c,1)=z(m);
end
end

% -------------------- Centre of the wheel -------------------------

zxm = mean(zx);    % Mean of x ordinate
zym = mean(zy);    % Mean of y ordinate



%---------- Find the min & max y values of the data ----------------
y_min = min(zy);
y_max = max(zy);
ay=zy.';           % y values in rows.

y1=y_min:pitch:y_max;
[fy1,ny1]=size(y1);

%---- Boundary points on the outer circle & innner circles for const y ----
sy=0;
for jy = 1:ny1
```

```
 y_num = find(ay==y1(jy));
 x_val = zx(y_num);
 x_min(jy) = min(x_val);
 x_max(jy) = max(x_val);

 [ny2 fy2]=size(x_val);
 for iy=1:(ny2-1);
 if ((x_val(iy+1)-x_val(iy))>1)
  sy=sy+1;            % sy no.of pts inside the big circle
  Xb1(sy) = x_val(iy);
  Xb2(sy) = x_val(iy+1);
  Yb1(sy) = y1(jy);
 end
end
       px(jy) = x_min(jy);    %outer circle pts
 qx(jy) = x_max(jy);
end

%---- Seperate the RHS points belongs to 2 circles and groubed as 1st & 4the Quadrant --
--
ey1=0;
ey2=0;
ey3=0;
ey4=0;
ey5=0;
for ty=1:(sy-1)
  if ((Yb1(ty)>=33)&(Xb1(ty)>zxm))
    ey1=ey1+1;
    YCb1(ey1)=Yb1(ty);
    XCb1(ey1)=Xb1(ty);
    XCb12(ey1)=Xb2(ty);
  elseif ((Yb1(ty)>zym)&(Xb1(ty)<27.5))
    ey2=ey2+1;
    YCb2(ey2)=Yb1(ty);
    XCb2(ey2)=Xb1(ty);
    XCb22(ey2)=Xb2(ty);
  elseif ((Yb1(ty)<25.5)&(Xb1(ty)<zxm))
    ey3=ey3+1;
    YCb3(ey3)=Yb1(ty);
    XCb3(ey3)=Xb1(ty);
    XCb32(ey3)=Xb2(ty);
  elseif ((Yb1(ty)<zym)&(Xb1(ty)>35))
    ey4=ey4+1;
    YCb4(ey4)=Yb1(ty);
    XCb4(ey4)=Xb1(ty);
    XCb42(ey4)=Xb2(ty);
```

```
 else
          ey5=ey5+1;
    YMb(ey5)=Yb1(ty);
    XMb1(ey5)=Xb1(ty);
    XMb2(ey5)=Xb2(ty);
 end
end

%--------- Min. & max. x values of the data ----------
x_min = min(zx);
x_max = max(zx);
ax=zx.';          %x values in rows.

x1=x_min:pitch:x_max;
[fx1,nx1]=size(x1);

%------ Boundary points on the outer circle & innner circles ------
sx=0;
for jx = 1:nx1
   x_num = find(ax==x1(jx));
   y_val = zy(x_num);
   y_min(jx) = min(y_val);
   y_max(jx) = max(y_val);

   [nx2 fx2]=size(y_val);
   for ix=1:(nx2-1);
   if ((y_val(ix+1)-y_val(ix))>1)
     sx=sx+1;
     Y1(sx) = y_val(ix);
     Y2(sx) = y_val(ix+1);
     X1(sx) = x1(jx);
   end
 end
          py(jx) = y_min(jx);
    qy(jx) = y_max(jx);
 end

 ex1=0;
 ex2=0;
 ex3=0;
 ex4=0;
 ex5=0;
 for tx=1:(sx-1)
    if ((Y1(tx)>=33)&(X1(tx)>zxm))
      ex1=ex1+1;
      XC1(ex1)=X1(tx);
```

```
  YC1(ex1)=Y1(tx);
  YC12(ex1)=Y2(tx);
elseif ((Y1(tx)>zym)&(X1(tx)<27.5))
  ex2=ex2+1;
  XC2(ex2)=X1(tx);
  YC2(ex2)=Y1(tx);
  YC22(ex2)=Y2(tx);
elseif ((Y1(tx)<25)&(X1(tx)<zxm))
  ex3=ex3+1;
  XC3(ex3)=X1(tx);
  YC3(ex3)=Y1(tx);
  YC32(ex3)=Y2(tx);
elseif ((Y1(tx)<zym)&(X1(tx)>35.5))
  ex4=ex4+1;
  XC4(ex4)=X1(tx);
  YC4(ex4)=Y1(tx);
  YC42(ex4)=Y2(tx);
else
        ex5=ex5+1;
  XM(ex5)=X1(tx);
  YM1(ex5)=Y1(tx);
  YM2(ex5)=Y2(tx);
end
end

%-- Eleminating the duplicate points and Re-arranging the points in clockwise direction -
--
m3=0;
m4=0;
m5=0;
m6=0;
m7=0;
for m1=1:ex1
 for m2=1:ey1
       if
(((XC1(m1)==XCb1(m2)|XC1(m1)==XCb12(m2))&(YCb1(m2)==YC1(m1)|YCb1(m2)
==YC12(m1)))|((XC1(m1)==XCb1(m2)|XC1(m1)==XCb12(m2))&(YCb1(m2)~=YC1(
m1)|YCb1(m2)~=YC12(m1)))|((XC1(m1)~=XCb1(m2)|XC1(m1)~=XCb12(m2))&(YCb
1(m2)==YC1(m1)|YCb1(m2)==YC12(m1))))
 m3=m3+1;
 XA1(m3)=XC1(m1);
 YA1(m3)=YCb1(m2);
end
end
end
for m8=1:ex1
```

```
        if (YC1(m8)==min(YC1))
  m4=m4+1;
XA2(m4)=XC1(m8);
YA2(m4)=YC1(m8);
end
end
for m9=1:ex1
  if (YC12(m9)>=49.5)
  m5=m5+1;
XA3(m5)=XC1(m9);
YA3(m5)=YC12(m9);
end
end
for m10=1:ey1
        if (XCb1(m10)==min(XCb1))
  m6=m6+1;
XA4(m6)=XCb1(m10);
YA4(m6)=YCb1(m10);
end
end
for m11=1:ey1
        if (XCb12(m11)==max(XCb12))
  m7=m7+1;
XA5(m7)=XCb12(m11);
YA5(m7)=YCb1(m11);
end
end
XA6=[XA4 XA1 XA5 XA2 XA3];
YA6=[YA4 YA1 YA5 YA2 YA3];
XA=XA6([1:10 13:15 18 20 21 23:2:35 84:95 39:2:51 54 57 56 62 61 69:-1:66 73:-1:70
65 64 63 60 59 58 55 53 52 50:-2:38 37 83:-1:74 36:-2:22 19 16 17 11 12]);
YA=YA6([1:10 13:15 18 20 21 23:2:35 84:95 39:2:51 54 57 56 62 61 69:-1:66 73:-1:70
65 64 63 60 59 58 55 53 52 50:-2:38 37 83:-1:74 36:-2:22 19 16 17 11 12]);
[XA.',YA.'];
%-----------------------------------
n3=0;
n4=0;
n5=0;
n6=0;
n7=0;
for n1=1:ex2
  for n2=1:ey2
        if
(((XC2(n1)==XCb2(n2)|XC2(n1)==XCb22(n2))&(YCb2(n2)==YC2(n1)|YCb2(n2)==Y
C22(n1)))|((XC2(n1)==XCb2(n2)|XC2(n1)==XCb22(n2))&(YCb2(n2)~=YC2(n1)|YCb2
```

```
(n2)~=YC22(n1)))|((XC2(n1)~=XCb2(n2)|XC2(n1)~=XCb22(n2))&(YCb2(n2)==YC2(n
1)|YCb2(n2)==YC22(n1))))
 n3=n3+1;
 XB1(n3)=XC2(n1);
 YB1(n3)=YCb2(n2);
end
end
end
for n8=1:ex2
        if (YC2(n8)==min(YC2))
  n4=n4+1;
XB2(n4)=XC2(n8);
YB2(n4)=YC2(n8);
end
end
for n9=1:ex2
  if (YC22(n9)==max(YC22))
  n5=n5+1;
XB3(n5)=XC2(n9);
YB3(n5)=YC22(n9);
end
end
for n10=1:ey2
         if (XCb2(n10)==min(XCb2))
   n6=n6+1;
XB4(n6)=XCb2(n10);
YB4(n6)=YCb2(n10);
end
end
for n11=1:ey2
         if (XCb22(n11)==max(XCb22))
   n7=n7+1;
XB5(n7)=XCb22(n11);
YB5(n7)=YCb2(n11);
end
end
XB6=[XB4 XB1 XB5 XB2 XB3];
YB6=[YB4 YB1 YB5 YB2 YB3];
XB=XB6([1:11 14 15 18 19:2:39 40 89:94 41:2:61 65 64 70 69 68 80:-1:71 67 66 63 62
60:-2:42 88:-1:81 38:-2: 20 16 17 12 13]);
YB=YB6([1:11 14 15 18 19:2:39 40 89:94 41:2:61 65 64 70 69 68 80:-1:71 67 66 63 62
60:-2:42 88:-1:81 38:-2: 20 16 17 12 13]);
[XB.',YB.'];
%----------------------------------------
p3=0;
p4=0;
```

```
p5=0;
p6=0;
p7=0;
for p1=1:ex3
 for p2=1:ey3
        if
(((XC3(p1)==XCb3(p2)|XC3(p1)==XCb32(p2))&(YCb3(p2)==YC3(p1)|YCb3(p2)==Y
C32(p1)))|((XC3(p1)==XCb3(p2)|XC3(p1)==XCb32(p2))&(YCb3(p2)~=YC3(p1)|YCb3
(p2)~=YC32(p1)))|((XC3(p1)~=XCb3(p2)|XC3(p1)~=XCb32(p2))&(YCb3(p2)==YC3(p
1)|YCb3(p2)==YC32(p1))))
 p3=p3+1;
 XD1(p3)=XC3(p1);
 YD1(p3)=YCb3(p2);
end
end
end
 for p8=1:ex3
        if (YC3(p8)==min(YC3))
   p4=p4+1;
 XD2(p4)=XC3(p8);
 YD2(p4)=YC3(p8);
 end
 end
 for p9=1:ex3
   if (YC32(p9)==max(YC32))
   p5=p5+1;
 XD3(p5)=XC3(p9);
 YD3(p5)=YC32(p9);
 end
 end
 for p10=1:ey3
        if (XCb3(p10)==min(XCb3))
   p6=p6+1;
 XD4(p6)=XCb3(p10);
 YD4(p6)=YCb3(p10);
 end
 end
 for p11=1:ey3
        if (XCb32(p11)==max(XCb32))
   p7=p7+1;
 XD5(p7)=XCb32(p11);
 YD5(p7)=YCb3(p11);
 end
 end
 XD6=[XD4 XD1 XD5 XD2 XD3];
 YD6=[YD4 YD1 YD5 YD2 YD3];
```

```
XD=XD6([1 2 7:11 15 16 18 19 22:2:36 38:43 94 95 44:46 48:2:62 65 64 68 73 72 71
83:-1:74 70 69 67 66 63:-2:47 93:-1:84 37:-2:23 20 21 17 12:14 3:6]);
YD=YD6([1 2 7:11 15 16 18 19 22:2:36 38:43 94 95 44:46 48:2:62 65 64 68 73 72 71
83:-1:74 70 69 67 66 63:-2:47 93:-1:84 37:-2:23 20 21 17 12:14 3:6]);
[XD.',YD.'];
%-------------------------------

q3=0;
q4=0;
q5=0;
q6=0;
q7=0;
for q1=1:ex4
  for q2=1:ey4
        if
(((XC4(q1)==XCb4(q2)|XC4(q1)==XCb42(q2))&(YCb4(q2)==YC4(q1)|YCb4(q2)==Y
C42(q1)))|((XC4(q1)==XCb4(q2)|XC4(q1)==XCb42(q2))&(YCb4(q2)~=YC4(q1)|YCb4
(q2)~=YC42(q1)))|((XC4(q1)~=XCb4(q2)|XC4(q1)~=XCb42(q2))&(YCb4(q2)==YC4(q
1)|YCb4(q2)==YC42(q1))))
 q3=q3+1;
 XE1(q3)=XC4(q1);
 YE1(q3)=YCb4(q2);
end
end
end
for q8=1:ex4
        if (YC4(q8)==min(YC4))
  q4=q4+1;
XE2(q4)=XC4(q8);
YE2(q4)=YC4(q8);
end
end
for q9=1:ex4
  if (YC42(q9)==max(YC42))
  q5=q5+1;
XE3(q5)=XC4(q9);
YE3(q5)=YC42(q9);
end
end
for q10=1:ey4
        if (XCb4(q10)==min(XCb4))
  q6=q6+1;
XE4(q6)=XCb4(q10);
YE4(q6)=YCb4(q10);
end
end
```

```
for q11=1:ey4
        if (XCb42(q11)>=52)
  q7=q7+1;
XE5(q7)=XCb42(q11);
YE5(q7)=YCb4(q11);
end
end
XE6=[XE4 XE1 XE5 XE2 XE3];
YE6=[YE4 YE1 YE5 YE2 YE3];
XE=XE6([1:10 14:16 18 21 22:2:38 86:96 40:2:54 58 57 62 61 74:-1:67 66:-1:63 60 59
56 55:-2:39 85:-1:75 37:-2:23 19 20 17 11:13]);
YE=YE6([1:10 14:16 18 21 22:2:38 86:96 40:2:54 58 57 62 61 74:-1:67 66:-1:63 60 59
56 55:-2:39 85:-1:75 37:-2:23 19 20 17 11:13]);
[XE.',YE.'];

%----------------------------------------
r3=0;
r4=0;
r5=0;
r6=0;
r7=0;
for r1=1:ex5
  for r2=1:ey5
        if
(((XM(r1)==XMb1(r2)|XM(r1)==XMb2(r2))&(YMb(r2)==YM1(r1)|YMb(r2)==YM2(r1
)))|((XM(r1)==XMb1(r2)|XM(r1)==XMb2(r2))&(YMb(r2)~=YM1(r1)|YMb(r2)~=YM2(
r1)))|((XM(r1)~=XMb1(r2)|XM(r1)~=XMb2(r2))&(YMb(r2)==YM1(r1)|YMb(r2)==YM
2(r1))))
 r3=r3+1;
 XF1(r3)=XM(r1);
 YF1(r3)=YMb(r2);
end
end
end
for r8=1:ex5
        if (YM1(r8)==min(YM1))
  r4=r4+1;
XF2(r4)=XM(r8);
YF2(r4)=YM1(r8);
end
end
for r9=1:ex5
  if (YM2(r9)==max(YM2))
  r5=r5+1;
XF3(r5)=XM(r9);
YF3(r5)=YM2(r9);
```

```
end
end
for r10=1:ey5
        if (XMb1(r10)==min(XMb1))
  r6=r6+1;
XF4(r6)=XMb1(r10);
YF4(r6)=YMb(r10);
end
end
for r11=1:ey5
        if (XMb2(r11)==max(XMb2))
  r7=r7+1;
XF5(r7)=XMb2(r11);
YF5(r7)=YMb(r11);
end
end
XF6=[XF4 XF1 XF5 XF2 XF3];
YF6=[YF4 YF1 YF5 YF2 YF3];
XF=XF6([1:4 6:9 11:2:17 18 42:45 22:2:26 29 34 33 38:-1:35 32:-1:30 28 27:-2:21 20 19
41 40 39 16 14 12 10 5]);
YF=YF6([1:4 6:9 11:2:17 18 42:45 22:2:26 29 34 33 38:-1:35 32:-1:30 28 27:-2:21 20 19
41 40 39 16 14 12 10 5]);
[XF.',YF.'];

%----------------------------------
s3=0;
for s1=1:jx
  for s2=1:jy
        if
(((x1(s1)==px(s2)|x1(s1)==qx(s2))&(y1(s2)==py(s1)|y1(s2)==qy(s1)))|((x1(s1)==px(s2)|
x1(s1)==qx(s2))&(y1(s2)~=py(s1)|y1(s2)~=qy(s1)))|((x1(s1)~=px(s2)|x1(s1)~=qx(s2))&(
y1(s2)==py(s1)|y1(s2)==qy(s1))))
 s3=s3+1;
 XG1(s3)=x1(s1);
 YG1(s3)=y1(s2);
end
end
end
XG=XG1([1:10 18:23 28:31 35:37 41:43 46:48 51 52 55 56 59 61 62 65 67 68 71:2:79
80:2:242 245 244 247:2:251 254 257 256 260 263 262 267 266 270 274 273 279:-1:277
285:-1:283 291:-1:289 299:-1:296 314:-1:308 322:-1:315 307:-1:300 295:-1:292 288:-
1:286 282:-1:280 276 275 272 271 269 268 265 264 261 259 258 255 253 252:-2:246
243:-2:81 78:-2:72 69 70 66 63 64 60 57 58 53 54 49 50 44 45 38:40 32:34 24:27
11:17]);
YG=YG1([1:10 18:23 28:31 35:37 41:43 46:48 51 52 55 56 59 61 62 65 67 68 71:2:79
80:2:242 245 244 247:2:251 254 257 256 260 263 262 267 266 270 274 273 279:-1:277
```

```
285:-1:283 291:-1:289 299:-1:296 314:-1:308 322:-1:315 307:-1:300 295:-1:292 288:-
1:286 282:-1:280 276 275 272 271 269 268 265 264 261 259 258 255 253 252:-2:246
243:-2:81 78:-2:72 69 70 66 63 64 60 57 58 53 54 49 50 44 45 38:40 32:34 24:27
11:17]);
[XG.',YG.'];

%---------- Distance between center and a boundary point ----------
ma=(m3+m4+m5+m6+m7);
nb=(n3+n4+n5+n6+n7);
pd=(p3+p4+p5+p6+p7);
qe=(q3+q4+q5+q6+q7);
rf=(r3+r4+r5+r6+r7);

%---------- 1st Quard circle----------
for wa=1:ma
  da(wa) = sqrt((YA(wa)-mean(YA))^2 + (XA(wa)-mean(XA))^2);
end

%---------- 2nd Quaed circle ----------
for wb=1:nb
  db(wb) = sqrt((YB(wb)-mean(YB))^2 + (XB(wb)-mean(XB))^2);
end

%---------- 3rd Quaed circle ----------
for wd=1:pd
  dd(wd) = sqrt((YD(wd)-mean(YD))^2 + (XD(wd)-mean(XD))^2);
end

%---------- 4th Quaed circle ----------
for we=1:qe
  de(we) = sqrt((YE(we)-mean(YE))^2 + (XE(we)-mean(XE))^2);
end

%---------- middle circle ----------
for wf=1:rf
  df(wf) = sqrt((YF(wf)-mean(YF))^2 + (XF(wf)-mean(XF))^2);
end

%---------- outer circle ----------
for wg=1:s3
  dg(wg) = sqrt((YG(wg)-mean(YG))^2 + (XG(wg)-mean(XG))^2);
end
```

```
%------------------Calculating the angles between the lines----------------

%Finding the distance between 2 adjacent point.
%Finding the angle between 2 lines by using cosine rule(from center).
%Finding the angle between 2 lines by using cosine rule (from 1st coner of the triangle).

for wa1=1:(ma-1)
  da1(wa1) = sqrt((YA(wa1)-YA(wa1+1))^2 + (XA(wa1)-XA(wa1+1))^2);
  angA1(wa1) = acos(((da(wa1))^2 +(da(wa1+1))^2-
(da1(wa1))^2)/(2*da(wa1)*da(wa1+1)))*180/pi;
  angA2(wa1) = acos(((da(wa1))^2 +(da1(wa1))^2-
(da(wa1+1))^2)/(2*da(wa1)*da1(wa1)))*180/pi;
end

for wb1=1:(nb-1)
  db1(wb1) = sqrt((YB(wb1)-YB(wb1+1))^2 + (XB(wb1)-XB(wb1+1))^2);
  angB1(wb1) = acos(((db(wb1))^2 +(db(wb1+1))^2-
(db1(wb1))^2)/(2*db(wb1)*db(wb1+1)))*180/pi;
  angB2(wb1) = acos(((db(wb1))^2 +(db1(wb1))^2-
(db(wb1+1))^2)/(2*db(wb1)*db1(wb1)))*180/pi;
end

for wd1=1:(pd-1)
  dd1(wd1) = sqrt((YD(wd1)-YD(wd1+1))^2 + (XD(wd1)-XD(wd1+1))^2);
  angD1(wd1) = acos(((dd(wd1))^2 +(dd(wd1+1))^2-
(dd1(wd1))^2)/(2*dd(wd1)*dd(wd1+1)))*180/pi;
  angD2(wd1) = acos(((dd(wd1))^2 +(dd1(wd1))^2-
(dd(wd1+1))^2)/(2*dd(wd1)*dd1(wd1)))*180/pi;
end

for we1=1:(qe-1)
  de1(we1) = sqrt((YE(we1)-YE(we1+1))^2 + (XE(we1)-XE(we1+1))^2);
  angE1(we1) = acos(((de(we1))^2 +(de(we1+1))^2-
(de1(we1))^2)/(2*de(we1)*de(we1+1)))*180/pi;
  angE2(we1) = acos(((de(we1))^2 +(de1(we1))^2-
(de(we1+1))^2)/(2*de(we1)*de1(we1)))*180/pi;
end

for wf1=1:(rf-1)
  df1(wf1) = sqrt((YF(wf1)-YF(wf1+1))^2 + (XF(wf1)-XF(wf1+1))^2);
  angF1(wf1) = acos(((df(wf1))^2 +(df(wf1+1))^2-
(df1(wf1))^2)/(2*df(wf1)*df(wf1+1)))*180/pi;
  angF2(wf1) = acos(((df(wf1))^2 +(df1(wf1))^2-
(df(wf1+1))^2)/(2*df(wf1)*df1(wf1)))*180/pi;
end
```

```
for wg1=1:(s3-1)
  dg1(wg1) = sqrt((YG(wg1)-YG(wg1+1))^2 + (XG(wg1)-XG(wg1+1))^2);
  angG1(wg1) = acos(((dg(wg1))^2 +(dg(wg1+1))^2-
(dg1(wg1))^2)/(2*dg(wg1)*dg(wg1+1)))*180/pi;
  angG2(wg1) = acos(((dg(wg1))^2 +(dg1(wg1))^2-
(dg(wg1+1))^2)/(2*dg(wg1)*dg1(wg1)))*180/pi;
end
%-----------------------------------

disp('      YA      XA ')
[XA.',YA.']
disp('Distance from center (da)')
da.'
disp('angA1(center)')
angA1.'

disp('      YB      XB ')
[XB.',YB.']
disp('Distance from center (db)')
db.'
disp('angB1(center)')
angB1.'

disp('      YD      XD ')
[XD.',YD.']
disp('Distance from center (dd)')
dd.'
disp('angD1(center)')
angD1.'

disp('      YE      XE ')
[XE.',YE.']
disp('Distance from center (de)')
de.'
disp('angE1(center)')
angE1.'

disp('      YF      XF ')
[XF.',YF.']
disp('Distance from center (df)')
df.'
disp('angF1(center)')
angF1.'

disp('      YG      XG ')
[XG.',YG.']
```

```
disp('Distance from center (dg)')
dg.'
disp('angG1(center)')
angG1.'

%-----------------
% 1st Quard circle
for wa2=1:ma
  (da(wa2)-mean(da))
end

%plot the boundary points.
plot(XA,YA,'g*')
hold on
plot(XB,YB,'x')
hold on
plot(XD,YD,'m+')
hold on
plot(XE,YE,'ro')
hold on
plot(XF,YF,'*')
hold on
plot(XG,YG,'kx')
hold on
plot(mean(XA),mean(YA),'*')
plot(mean(XB),mean(YB),'*')
plot(mean(XD),mean(YD),'*')
plot(mean(XE),mean(YE),'*')
plot(mean(XF),mean(YF),'*')
plot(mean(XG),mean(YG),'kx')
```

```
%-------------------------------------------------------------------------------------------
% MATLAB codes for Feed-Forward Neural Network (Back-Propagated) Algorithm
%-------------------------------------------------------------------------------------------

% w      : Input-to-hidden layer weights
% v      : Hidden-to-output layer weights
% k      : # of iterations
% R      : Reflected vector
% A      : Learning rate b/w hidden and output connectors
% B      : Learning rate b/w input and hidden connectors
% hidden : Hidden neuron values
% output : Output neuron values


clear all
close all

tic    % Start a stopwatch timer

load input_data.txt
load desired_output.txt

%---------- Initialize the weights ----------

for i=1:4
  for j=1:7
    w(i,j)=0.005;
 end
end

for i=1:7
  for j=1:7
    v(i,j)=0.005;
  end
end

a = circle_data1(:,1);
b = circle_data1(:,2);
alpha = circle_data1(:,3);
nc = circle_data1(:,4);

p1=desired_output(:,1);
p2=desired_output(:,2);
p3=desired_output(:,3);
```

```
p4=desired_output(:,4);
p5=desired_output(:,5);
p6=desired_output(:,6);
p7=desired_output(:,7);

[n1 f]=size(a);

%Assigning the value to the shapes

for n=1:n1

Desired = [p1(n) p2(n) p3(n) p4(n) p5(n) p6(n) p7(n)];

input(1)=a(n);
input(2)=b(n);
input(3)=alpha(n);
input(4)=nc(n);


%------------------ Network Trainig ----------------------

for k=1:400

% calculate the hidden neuron values
for i=1:7
  hidden(i)=0;
  for j=1:4
    hidden(i)=hidden(i)+(w(j,i)*input(j));

  end
end

% calculate output neuron value
for i=1:7
  output(i)=0;

  for j=1:7
    output(i)=output(i)+(v(j,i)*hidden(j));
  end
  output(i)=(1-exp(-(output(i))))/(1+exp(-(output(i))));

end

%calculate total error
total_error=0;
for i=1:7
```

```
  total_error = total_error + abs(Desired(i)-output(i));
end

%-----------------Back propagation of the error -----------------

if total_error < 0.05;
  fprintf('1f\n', output)

else

% calculate reflected vector

  for i=1:7
  · R(i)=(Desired(i)-output(i))*output(i)*(1-output(i));
  end

%-------------------- Update the weights ----------------------

% update the weights b/w the hidden & the ouput layer
  A=0.2;
  for i=1:7
    for j=1:7
      v(i,j) = v(i,j) + (A*R(j)*hidden(i));
    end
  end

  % calculate hidden layer error
  for i=1:7
    E_hidden(i)=0;
    for j=1:7
      E_hidden(i) = E_hidden(i) + (R(j)*v(i,j));
    end
    E_hidden(i)=hidden(i)*(1-hidden(i))*E_hidden(i);
  end

  % update the weights b/w the input & hidden layer
  B=0.2;
  for i=1:4
    for j=1:7
      w(i,j) = w(i,j) +(B*E_hidden(j)*input(i));
    end
  end
end
end
```

```
disp('The weights b/w input & the hidden')
w
disp('The weights b/w the hidden & the output')
v
disp('The output')
output

end
toc    % Stop the stopwatch timer and display the elapsed time
```

References

[1]. V.H. Chan, C. Bradeley, G. W. Vickers, "A multi-sensor approach to automating co-ordinate measuring machine-based reverse engineering" computer in industry vol. 44, pp.105-115, 2001.

[2]. Vincent H. Chan, "Feature based reverse engineering employing automated multi-sensor scanning" Ph.D dissertation, Department of mechanical engineering. University of Victoria, 1999.

[3]. http://www.datx.co.uk dated 6/13/2003.

[4]. J. Lampinen, J. Laaksonen, E. Oja, "Neural network system, techniques and applications in pattern recognition", Research reports, Laboratory of computational engineering, Helsinki University of Technology, Miestentie 3, Finland, 1997.

[5] http://www.chillingeffects.org/reverse/ dated 09/26/2003.

[6]. Bhandarker M. P., Nagi, R, "STEP–based feature extraction from STEP geometry for Agile manufacturing", computers in industry, vol. 41, pp. 3-24, 2000.

[7]. Joshi S, Chang T. C., "Graph-based heuristics for recognition of machined features from a 3D solid model" Computer-Aided Design, vol. 20, no 2, pp. 58-66, 1988.

[8]. J.H. Vandenbrande, A.A.G. Requicha, "Spatial reasoning for the automatic recognition of machinable features in solid models" IEEE transactions on pattern analysis and machine intelligence, vol. 15, no. 12, pp1269-1285, 1993.

[9]. M. Schulte, C. Weber, S Rainer, "Functional features for design in mechanical engineering", Computer in industry, vol. 23, no. 1, pp.15-24, Nov'93.

[10]. D. Silver, "3D Feature Extraction for unstructured grids" http://www.caip.rutgers.edu/~silver/nasa/nasa2.html

[11]. J.J. Shah, "Assessment of features technology", Computer aided design, vol. 23, no. 5, pp. 331-343, June'91.

[12]. V.H. Chan, M. Arshad, "Recognition of Features in cloud data for Reverse Engineering", proceedings of CSME conference at Kingston, Canada, May 21-24, 2002.

[13]. J. Rowe, "Surface modeling", Computer graphics, vol. 20, no. 9, pp 47-52, 1997.

[14]. G. Wyvill, D. McRobie, M. Gigante, "Modelling with features", IEEE Computer graphics and applications, vol. 15, no. 5, pp. 40-46, 1997.

[15]. J. C. Cavendish, "Integrated feature based surface design with free form deformation", Computer aided design, vol. 27, no. 9, pp. 703-711, 1995.

[16]. J. S. M. Vergeest, I. Horvath, J. Jelier, Z. Rusak, "Free-form surface copy and paste techniques for shape synthesis", Proc. 3rd Int. symposium of competitive engineering, Delft University press, Delft, pp. 395-406, 2000.

[17]. Wu, M. C., Liu, C. R., "Analysis on machined feature recognition techniques based on B-rep" Computer-Aided Design, vol. 28, no.8, pp. 603-616, 1996.

[18]. J. Han, "Survey of feature research", Technical report IRIS-96-364, Institute for robotics and intelligence systems, USC, USA, 1996.

[19]. J. Han, A.A.G. Requicha, "Feature recognition from CAD models", IEEE computer graphics and applications, vol. 18, no. 2, 1998.

[20]. Kyprianou L. K., "Shape classification in Computer-Aided Design", PhD dissertation, Christ College, University Cambridge, Cambridge, U.K., July 1980.

[21]. B. Falcidieno, F. Giannini, "Automatic recognition and representation of shape based features in a geometric modeling system", Technical report no. 10/88, Instituto per la matematica applicata, 1988.

[22]. P. Di Stefano, "A feature based representation scheme for design", Proc. of the FEATS 2001 International conference, Valenciennes, France, June 2001.

[23]. J. J. Shah, M. Mantyla, "Parametric and feature based CAD/CAM", John Wiley & Sons, 1995.

[24]. T. Woo. "Feature extraction by volume decomposition", Proc. Conf. CAD/CAM technology in mechanical engineering, Cambridge, MA, USA, 1982.

[25]. Y. S. Kim, "Recognition of form features using convex decomposition", Computer-Aided Design, vol. 24, no. 9, pp. 461-476, 1992.

[26]. E. Wang, Y. S. Kim, "Form feature recognition using convex decomposition", results presented at the 1997 ASME CIE feature panel session, Computer-Aided Design, vol. 13, no. 9, pp. 983-989, 1998.

[27]. S. B. Kailash, Y. F. Zhang, J. Y. H. Fuh, "A volume decomposition approach to machining feature of casting and forging components", Computer-Aided Design, vol. 33, no. 8, 2001.

[28]. http://aitech.ac.jp/~kurokawa/kurokawa/neural_net/net_final.html

[29]. S. T. Welstead, "Neural Network and fuzzy logic Applications in C/C++", John wiley & sons, Inc, Toronto., 1994.

[30]. M. P. Carven, K. M. Curtis, B. R. hayes-Gill, C. D. Thursfield, "A Hybrid Neural Network/Rule-Based Technique for on-line gesture and hand-written character recognition", Proceedings of the fourth IEEE International Conference on Electronics circuits and systems. Cairo, Egypt, vol.2, pp. 850-853, Dec 15-18, 1997.

[31]. S. Prabhkar, M. R. Henderson, "Automatic form-feature recognition using neural network-based techniques .on boundary representations of solid models", Computer-Aided Design, vol. 24, no. 7, pp. 381-393, 1992.

[32]. A. H. Zulkifli, S. Meeran, "Decomposition of interacting features using a Kohonen self-organizing feature map neural network", Engineering applications of artificial intelligence, vol. 12, pp. 59-78, 1999.

[33]. I. Yoshihara, Y. Kamimai, M. Yasunaga, "Feature Extraction from genome sequence using Multi-Modal Network", Faculty of engineering, Miyazaki University, Japan, Genome Informatics 12: pp. 420-422, 2001.

[34]. William B. Thompson, J. C. Owen, H. J de St. Germain, S. R. Stark, T. C. Henderson., "Feature-based reverse engineering of mechanical parts", IEEE Transaction on Robotics and Automation, vol. 15, no. 1, 1999.

[35]. Farhad Nabhani, Terry Shaw, "Performance analysis and optimization of shape recognition and classification using ANN", Robotics and Computer Integrated Manufacturing 18 (2002), pp. 177-185.

[36]. V. H. Chan, "AI for Mechanical Engineers", Class Lecture notes, June2002, Ryerson University.

[37]. K. Nezis, G. Vosniakos, "Recognition 2 ½ D shape features using a neural network and heuristics", Computer-Aided Design, vol. 29, no. 7, pp. 523-539. 1997

[38]. A. D. Kulkarni, "Computer vision and fuzzy-neural systems", Prentice Hall PTR, Upper Saddle River, NJ, 2001.