INFORMATION TO USERS

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

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

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

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

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

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA

UMI®
800-521-0600

Feature Based Reverse Engineering Employing Automated Multi-Sensor Scanning

by

Vincent Harry Chan
M.Sc., Queen's University, 1994
B.A.Sc., University of Waterloo, 1992

A Dissertation Submitted in Partial Fulfilment of the
Requirements for the Degree of
DOCTOR OF PHILOSOPHY

in the Department of Mechanical Engineering

We accept this dissertation as conforming
to the required standard.

Dr. G.W. Vickers, Co-supervisor (Mechanical Engineering)

Dr. C. Bradley, Co-supervisor (Mechanical Engineering)

Dr. J. Wegner, Member (Mechanical Engineering)

Dr. Z. Dong, Member (Mechanical Engineering)

Dr. P. Driessen, Member (Electrical and Computer Engineering)

Dr. Y.F. Zhang, External Examiner (Mechanical and Production Engineering,
National University of Singapore)

Supervisors: Drs. G.W. Vickers, C. Bradley

Abstract

Reverse engineering of geometric models is the process of creating a computer aided model from an existing physical part so that subsequent manufacturing processes may be implemented. Applications of reverse engineering can range from the production of molds and dies from wood or clay models to the creation of replacement parts from worn existing machinery. In reverse engineering, both contact and non-contact measurement probes are used to gather measured surface points. However, due to the nature of these instruments, both the direction of the probe during measurement and the conversion of the gathered data to the appropriate computer aided models are currently very difficult.

This thesis addresses some of these problems. A stereo vision system employing neural network based image segmentation is implemented to automatically generate probe paths for either a touch trigger probe or an optical laser scanner. A fuzzy logic based iterative geometry fitting algorithm is used to fit geometric primitives to measured surface data. As modern computer aided drafting programs utilise parametric modelling methods and topology information, regarding the association of neighbouring surface patches is determined from the fitted geometric entities. Finally, utilising the extracted geometric and topology information, specific surface features, such as corners, slots and steps are detected using a feed-forward neural network.

The computational tools in this thesis provide methods that reduce the time and effort required to geometrically reverse engineer an existing physical object.

Examiners:

Dr. G.W. Vickers, Co-supervisor (Mechanical Engineering)

Dr. C. Bradley, Co-supervisor (Mechanical Engineering)

Dr. J. Wegner, Member (Mechanical Engineering)

Dr. Z. Dong, Member (Mechanical Engineering)

Dr. P. Driessen, Member (Electrical and Computer Engineering)

Dr. Y.F. Zhang, External Examiner (Mechanical and Production Engineering,
National University of Singapore)

Table of Contents

List of Tables

List of Figures

Nomenclature Table

A	parameter – learning rate of excitatory connectors
a	x-parameter for a plane
B	parameter – rate of increase for areas of similar grey level intensity
b	y-parameter for a plane
C	parameter – rate of increase of the momentum of the excitatory connectors
c	z-parameter for a plane
D	parameter – learning rate for inhibitory connectors
d	normal parameter for a plane
$E_{output,i}$	error at the output layer, neuron location i
$E_{hidden,i}$	error at the hidden layer, neuron location i
$G_{mean,k}$	mean grey level intensity for a patch
$k_{(p,q)}$	curvature between points p and q
n	number of data points in patch
$\overline{n}_p$	normal at point p
$\overline{n}_q$	normal at point q
$P_{sum,k}$	number of pixels in patch
p_i	point p at location i
q_i	point q at location i
R	radius
R_i	reflected vector at the output layer

x_o	x-centre
x_l	displacement in the left image
x_r	displacement in the right image
$x_{(x,y)(i,j)}$	strength of connector between neuron (x.y) and (i,j)
y_o	y-centre
z	height between the object point and the lens centre
z_o	z-centre
γ	strength of inhibitory connector
$\eta_{i,j,k}$	neuron value (0 or 1) at location i,j, level k
$\hat{\eta}_p$	unit vector of normal at point p
θ_{mom}	current momentum value
σ_{grey}	difference of pixel grey level intensity
v_i	distance between point I and the closest pixel

Chapter 1 – Introduction

The Need for Geometric Reverse Engineering

Reverse engineering of geometric models is the process of creating a geometric model such as a computer aided design (CAD) model, from an existing physical part. Reverse engineering is a rapidly evolving discipline[1] that is concerned with more than capturing the shape of the object but also involves interpreting spatial features on the object's surface. There are several applications of reverse engineering. For example, it may be necessary to produce a part where drawings or other documentation are not available. Clay or wood models are still used by designers and stylist to help evaluate real 3-D objects. Reverse engineering is used to generate measured surface points from these models in a CAD environment so that subsequent manufacturing processes may be implemented. For other parts, such as turbine blades or air foils, where extensive wind tunnel analysis and modifications have been made to its shape, reverse engineering is used to capture the changes. Finally, in another important area of research, reverse engineering has been used to create custom fits for human and animal prostheses. Applications of geometric reverse engineering are outlined in Table 1.

Table 1: Common applications of reverse engineering

Application	Description	Examples
1. Machine Parts	To re-create old machine parts where drawings or documentation do not exist	Replacement parts for ocean vessels.
2. Prototype or Stylist's Clay/Wood Models	Generation of measured data points from a scaled model.	Commonly used in the auto industry to create stamping dies of new car designs.
3. Modified Surfaces	To track changes to parts altered during analysis and testing.	Turbine blades or air foils that have been changed during wind tunnel testing.
4. Prosthesis	Custom fit prosthesis for better wear and comfort.	Knee and hip replacements, helmets, Formula 1 drivers seats.

The ultimate goal is to realise an intelligent reverse engineering system that can automatically capture an object's shape and translate the measured data points into a CAD model. Although several researchers (see Section 1.3) have made encouraging advancements, reverse engineering is a complex and difficult problem, to which an encompassing single solution has not been found. To this end, several solutions to different parts of the reverse engineering problem are presented.

1.1 Conventional Geometric Reverse Engineering

The reverse engineering process has traditionally employed a touch trigger probe mounted on a coordinate measuring machine (CMM)[2,3,4,5,6,7]. Advances in machine vision technology have enabled non-contact sensors (e.g. an active laser-based range finder) to be utilised for the collection of 3-D data from the surface of the object. The spatial data that

defines the object's form and features is then processed to create a CAD model suitable for any subsequent computer-based design, analysis or manufacturing tasks. The last decade has witnessed the adoption of machine vision-based reverse engineering to design studios and manufacturing enterprises such as custom injection molding firms[8]. Several 3-D machine vision systems and associated data processing software are commercially available as outlined in Section 1.3.2. The reverse engineering procedures can be characterised by four basic phases as outlined in Figure 1.

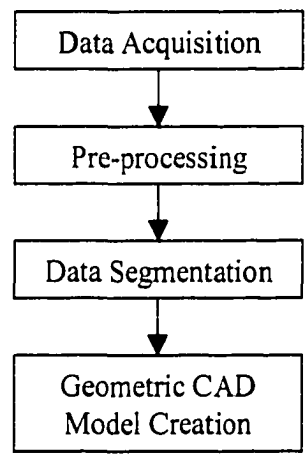

Figure 1: Conventional Reverse Engineering Process

Data acquisition uses a probe to measure a surface point. Whether a contact or non-contact method is used, the appropriate analysis must be applied to determine the position of the points on the object's surface from the physical measured data. Further analysis is required on the measured data in the pre-processing stage to filter the data for noise and as well, any alignment or any calibration that may be necessary. Depending on the data

acquisition method, the data set may need to be reduced to the proper density. Once the measured data points have been acquired, the surface is divided along its natural boundaries. These may exist along sharp corners of the object or along smooth transitions. For example, Figure 2 shows a surface where the six visible patches that constitute the surface have been segmented and labelled.

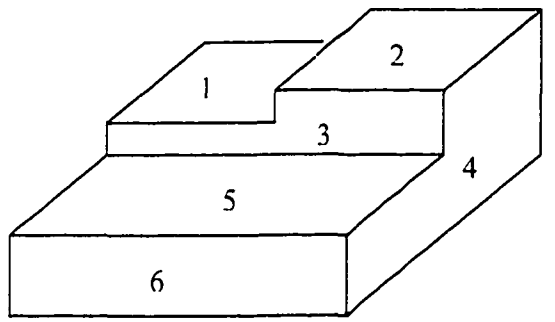

Figure 2: Three-level Segmented Surface

Finally, with the surface segmented into its constituent surfaces, geometric surfaces, such as planes and cylinders are fitted to the segmented data and a CAD model is generated. Further processing of the CAD model generates a boundary representation (B-rep) model of the object. B-rep model representation is a method used by high level CAD packages to depict solid models. This representation method requires knowledge (ie. topology) of both the geometry of the surface patches and the relationships of the surface patches to each other.

Problems in reverse engineering stem from either a sparse data set insufficient to

properly define the surface being reverse engineered or an extremely large data set of surface points that prove difficult to manipulate. Laser range sensors, similar to the one employed in this research, typically generate 3-D data files that have the following characteristics: i) size, several Mbyte range, ii) unstructured, or cloud data, that is not arranged in a spatially structured manner. Although the measured surface data is accurate (+/- 0.025mm over a 40mm depth of field, for example), the size and format of the cloud data renders the use of the data extremely difficult in engineering applications. Commercial software packages exist for the reverse engineering of this type of data but rely on the user to manually identify patch boundaries (ie. segment) between distinct features on the object. Further difficulties with laser scanning devices result from the limited range of the scanner, as well as the separation of the scanner from the surface by a pre-determined distance. Thus, during the scanning process, the operator must carefully control the scanner to maintain the proper stand-off distance within the boundary of the operating window.

Tactile sensors provide inadequate data density to define free-form surfaces. These sensors require that the operator manually direct the probe, a tedious and time consuming process especially for free-form surfaces that require a dense data set in order to be accurately defined. Software packages have been developed to automatically move touch probes in a grid like pattern, however, this method requires much effort to set and monitor.

1.2 Multi-Sensor Geometric Reverse Engineering

This research attempts to provide solutions to some of the problems encountered in current reverse engineering processes. The impediments to widespread acceptance of reverse engineering in industry are: 1) manual control of the digitising sensor 2) manual segmentation of data into distinct features and 3) inadequate methodology for guiding the construction of the CAD model from the segmented data.

1.2.1 Reverse Engineering System Design

In this work, automation of the scanning and construction of 3-D B-rep model is accomplished through three separate sensors, a black & white charged couple device (CCD) camera, a CMM touch probe and a Hymarc laser scanner, which is used to gather range data of varying degrees of accuracy and data density. A CCD camera is used, through the use of stereo vision, to determine the location of the object as well as segment the surface of the object into discrete patches. The laser scanner is used to gather accurate data from the surfaces. Special features, such as holes, detected by the stereo vision system are digitised with the touch probe as optical sensors are inappropriate in such situations where occlusions block the reflectance of laser light back to the laser scanner sensor. The collected measured data points are fitted with an appropriate primitive geometric shape. The combination of surface patches, which make up features, are then identified and an appropriate tolerance is estimated. An outline of the new engineering process is given in Figure 3:

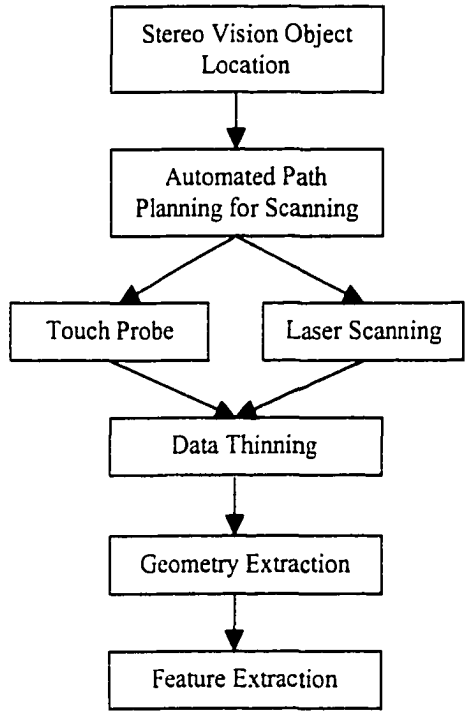

Figure 3: Proposed reverse engineering process

It is proposed that the application of the above methodology would forward the automation of the reverse engineering process. Solutions to the three important problems in reverse engineering outlined in Section 1.2 are:

1) A multiple sensor approach to digitisation is employed combining the relative strengths of three sensors; touch trigger probe, 2-D video camera and a 3-D laser-based range finder. The sensors also capitalise on the precise and repeatable positioning capabilities of the computer controlled CMM. The video camera, CMM and associated software are employed to build an approximate spatial model of the part using stereo vision imagery.

The touch trigger probe employs spatial information, defining the location of important features, derived from the first-cut computer representation. The laser-based range finder (a laser scanning sensor) employs the rough model but is utilised for the rapid digitisation of surface patches present on the object. Combination of the three sensing techniques, integrated on the CMM, offers greater flexibility for digitising a wider spectrum of parts and greater accuracy for defining functional engineering features such as bearing through holes, part datum, etc.

2) Images from the stereo process are used to determine how the object's surface should be segmented into individual surface patches, reducing the need for user intervention in outlining individual surface points. (i.e. The stereo pairs are used for automated patch boundary segmentation, where each patch is a recognisable geometric or engineering feature). After the measured data points have been collected, either with the touch trigger probe or the laser scanner, each surface patch of data points are automatically fitted with the appropriate geometric surfaces. A fuzzy logic algorithm is applied to reduce the number of iterations required to fit the geometric surfaces. The orientation and relative locations of the geometric surfaces to each other (i.e. the topology of the surface) are recorded in a "topology database".

3) A feed forward neural network is used to test the topology database for recognisable features. Engineering features, such as corners, slots and holes are identified. These features provide a more intuitive means for engineers to develop object definitions.

1.2.2 Potential Benefits of the Proposed Reverse Engineering Process

It is apparent that the multi-sensor feature based reverse engineering process offers a number of potential benefits relative to conventional reverse engineering practice. The digitisation of specific object features using a multi-sensor approach, combining a laser scanner with a touch probe, allows for the best qualities of each digitiser; the accuracy of the touch probe and the speed and consistency of a laser triangulation system. As well, the path of the probe, either the touch trigger probe and the Hymarc laser scanner can be optimised for scanning efficiency, an option not often realised in manual data point collection. The preliminary 3-D scan (i.e. stereo vision on the CDD camera) allows for the appropriate type of sensor, either the laser scanner or the touch trigger probe can be selected depending on the surface type.

Another potential benefit of the proposed system is the replacement of blanket scanning by employing more "intelligent" scanning of the object. This procedure reduces "over-scan", spurious data points that add to the complexity of the cloud data set but do not add to the definition of the surface patch being scanned. Also, as discussed in Section 1.2.1, by applying the segmentation process on the 2-D stereo images, the complex and difficult problem of segmenting the 3-D cloud data is avoided.

The creation of a B-rep model is further enhanced by the examination of the generated data for significant engineering features. Features are essential to automate the link between CAD and CAM, or in the case of reverse engineering, to automate the reproduction of engineered objects. Features provide a convenient, high level language for

specifying mechanical parts and for facilitating automated manufacturing.

A summary of the potential benefits of the multi-sensor feature based reverse engineering process is given in Table 2:

Table 2: Potential benefits of multi-sensor reverse engineering

	Conventional Reverse Engineering	Multi-Sensor Reverse Engineering
Measured data point collection	Manual, slow and laborious process.	Automated through the application of stereo vision.
Data pre-processing	Large set of cloud data points	Reduced set of cloud data points as each patch individually scanned.
Cloud data segmentation	Difficult and complex problem, often must be carried out manually through an interactive user interface	Problem of 3-D segmentation avoided through pre-segmentation of the stereo images.
Geometric surface fitting	User must select appropriate surface type to fit.	The "best-fit" surface is selected by applying a quick surface fitting algorithm.
Feature recognition	N/A	Allows for the automatic recognition of features to facilitate CAM processes.

1.3 Survey of Relevant Literature

An extensive literature search has revealed recent progress on the automation of reverse engineering. This review is divided between current research efforts and the "state of the art" for commercial reverse engineering packages.

1.3.1 Review of Current and Recent Research

Reverse engineering is a relatively new area of research that has borrowed many ideas and concepts from the areas of machine vision and image analysis. Previous research in reverse engineering has been focused on three main areas:

- *Surface fitting of curves and surfaces to cloud data.* Milroy et al.[9] examined techniques for fitting non-uniform rational B-splines (NURBS) curves to patches of data generated by a laser scanner using a least squares error minimisation approach. A smooth parametric surface approximation is obtained by Sarkar and Menq[10] through their B-spline surface fitting algorithm. Again, the parameters for the B-spline surface is determined through a least squares fitting algorithm. Gu and Yan[11] use an interesting method of applying a feed forward neural network to estimate parameters for a non-uniform B-spline surface. In their iterative procedure, initial parameters are used to construct a B-spline surface, which in turn is compared to the measured data points. The error between the parametric surface and the measured surface is fed back into the neural network for another parameter estimate. An alternate method developed by Liao and Medioni[12] used an initial simple surface, such as a cylinder, and deformed that surface to fit data points by minimising an energy function. Bradley et al.[13] utilised a quadric surfaces fitting method to 3-D cloud data employing a statistical parameter estimation technique. The method was found to be insensitive to outlier data points. Chivate and Jablokow[3] applied a least-squares approach to fit quadric surfaces to measured point data, resulting in an algebraic representation for each surface patch.

- *Segmentation of the data along its natural boundaries.* Bradley[14] used a manual computer workstation-based method for identifying and delineating surface patches present on an object. Milroy et al.[15] used an active contour algorithm to locate the boundaries on an object from an initial seed region identified manually. Sakar and Menq[10] applied a Laplacian of Gaussian edge detection operator to a range image. The operator identified potential candidates for edge points and a second pass by the algorithm is required to link the points to create boundary contours. An alternative method investigated by Jain et al.[16,17] sought to segment the surface by classifying each surface point by its local curvature.
- *Automation of the digitising process.* Milroy[18] implemented an in process method for calculating the position and viewing angle for a laser scanner head mounted on a CMM. The algorithm built an approximate model of each patch of data, acquired in any given viewing location, using the next best viewing location and orientation to determine the scanner orientation for subsequent passes. The process was repeated until the entire object had been digitised. Soucy et al.[19] used a voxel bin approach to compute sensor trajectories to achieve complete surface coverage. By placing scan data points into voxels, neighbouring voxels are examined for surface continuity. In their work, if the continuity did not exist, the scanner is directed to move in the vicinity of the discontinuity. Other previous research on automated digitisation has tended to focus on part digitisation or inspection employing a CAD model of the part. For example, Sobh et al.[7] used a CAD model to pre-plan the optimum (with respect to time) inspection path of a touch trigger probe mounted on a computer controlled CMM.

It should be noted that most of the above research[10,17] has been based on a limiting assumption: the 3-D data set is arranged in a well ordered matrix format. That is, the 2-1/2 D data is of the form:

$$\begin{bmatrix} x_{1,1} & x_{1,2} & & & x_{1,n} \\ x_{2,1} & x_{2,2} & & & \\ & & \ddots & & \vdots \\ & & & \ddots & \\ x_{m,1} & & \cdots & & x_{m,n} \end{bmatrix}$$

Figure 4: Ordered 2-1/2 D data format

This simplifying, assumption permits the application of common image processing operators and simple surface fitting techniques to the data set. Although convenient, true 3-D data are usually a result of tactile or optical sensors gathering massive amounts of unstructured data that completely blanket the object being scanned.

1.3.2 Comparison of Commercial Systems and Software

Commercial 3-D scanner systems have been available for a number of years. Several of the more advanced 3-D sensor systems available on the market today, suitable for performing reverse engineering, are discussed below.

Hymarc 3D Vision Systems [20]

Hymarc's Hyscan 45C is based on the unique patented synchronized scanning mechanism. The Hyscan 45C maps surface information in a continuous high speed non-contact manner with +/- 0.025 mm precision accuracy. The digitiser is designed to fit any CMM, custom translation device or CNC machine tool. The scanning system consists of a 45C camera head, real time controller, and host workstation. A photograph of the 45C, mounted on the end of a CMM arm, is shown in Figure 5. The head is translated during the scanning process and surface points are captured "on the fly". The relatively small field of view and the 3-D nature of the object can often necessitate multiple scanning passes to fully digitise the object. Each separate scanning pass is then combined into one global data file referenced to a single point. Typical working specifications for the Hyscan 45C are: the scanner is set to acquire 512 points in the 80 mm span of the scan line. The scanner is traversed at a uniform speed of 1.5 mm/second, yielding scan lines that are spaced 0.5 mm apart.

3D Scanners Ltd.[21]

The company's Web site contains detailed information on their two major products: REVERSA and ModelMaker. The REVERSA sensor is a non-contact scanning sensor for digitising models quickly and accurately when they are fixtured on a machine tool or CMM. The complete system includes a laser stripe triangulation range sensor head, data acquisition hardware, and software that allows immediate display of the data when it has been captured from the part. Data can be gathered at up to several thousand points per second permitting typical objects to be digitised in just a few minutes, independent of how complex their

surface geometry. The manufacturers specifications indicate REVERSA is capable of scanning points as close as 50μm apart (20 points per mm). System accuracy is dependent on the specifications of the machine tool or translation system on which the sensor is mounted.

In addition, 3D Scanners Ltd. produces the ModelMaker reverse engineering system that also employs the principle of laser triangulation. This sensor head is mounted on a compact arm and is manually positioned around the object during the digitisation phase. The arm has position encoders in each of its links that monitor spatial position. ModelMaker uses a dedicated 3-D image processing board to capture and process the gathered range data information in real time. The company supplies software to aid the engineer in processing and transferring the surface data to CAD/CAM packages.

Laser Design Inc.[22]

Laser Design Inc. makes the Surveyor 3D Laser Digitising System in several configurations depending on the size of the objects the user wishes to digitise. The company's product line can accommodate large automotive parts down to small electronic components. The company product specifications quote accuracy's of up to +/- 0.012 mm per axis and the 1200 and 6000 models are similar to conventional CMMs in that they also have a granite base that provides stability and accuracy for the range sensor head. The sensor head probes offer a 40-45 mm scan line, which increases scanning speed without sacrificing accuracy. The company also provides its DataSculpt scan data editing software that permits

supervision of the scanning process and preliminary data editing (such as filtering).

1.4 Scope of the Dissertation

The objective of this research is to develop an automated approach to the process of reverse engineering. This thesis is arranged in the chronological order of the steps required to carry out reverse engineering.

Part I - Automation Multi-Sensor Digitisation – Chapters 2 and 3

These chapters present the manner in which the three sensors are used to specify the objects location and orientation on a CMM. Furthermore, the method of creating the approximate 3-D model, using stereo vision and the subsequent digitisation of significant features and surface patches, using the touch probe and laser scanner is described. The spatial information generated by the stereo vision system is used to direct either the touch probe or laser scanner head. In Chapter 2, a neural network based algorithm is described that accomplishes the segmentation of surface patches in the images. Stereo correspondence and depth calculations are then made on the stereo image pair. Chapter 3 discusses the algorithm that plans the path of either the touch probe or laser scanning digitising head.

Part II – Creation of the Geometrical and Topology Database – Chapters 4 and 5

The emphasis in these chapters is the surface fitting of 3-D cloud data and structuring of the fitted surfaces, which comprise the CAD model, in a topologically ordered fashion.

An iterative least squares error minimisation routine is presented in Chapter 4 to fit the quadric surfaces to the data. A novel method to reduce the number of iterations to fit the surfaces, (i.e. planes, cylinders and spheres) to the measured data points is investigated. Using a voxel bin based method, the relative locations of the separate patches that comprise the surface is determined. The topology (i.e. the relative position of these surfaces and how they relate to one another) extraction algorithm is described in Chapter 5. Thus, with the topology of the object re-created, the B-rep model of the reversed engineered object is completed.

Part III - Feature Extraction – Chapter 6

The recent trend in commercial CAD software development has been toward parametric feature-based CAD. Features are the generic shapes of an object with which engineers can associate attributes useful in design and manufacturing. A feature encapsulates the engineering significance of the shape such as details related to form, tolerance and assembly. Feature-based CAD software provides a more intuitive approach for designers and engineers when developing new components. To extend the current status of reverse engineer research, this work extends the modelling past geometric descriptions and incorporates a few object features. The geometric and topology databases are examined for relevant features. In Chapter 6, a feed forward neural network is used to recognise relevant features in the database

Chapter 2 - Object Location and Segmentation

Automation of the digitisation process first requires that the location and orientation of the part be determined. This is accomplished through the application of stereo vision, a lower accuracy digitisation system that can view the entire workspace of the CMM. Stereo vision is a method to determine three-dimensional (3-D) information from two or more two-dimensional images taken from different locations. Using a single CCD camera attached to the end effector of the CMM as shown in Figure 5, two separate images are taken from different locations by translating the CMM a known distance. To construct a full picture of the objects surface, images of the object are taken from above, and from the four sides. As each stereo view requires two images, a total of ten images are needed for complete coverage of the object. Although many of the simple objects used in this thesis appear to be more easily reversed engineered with standard measuring devices (such as callipers), their non-orthogonal surfaces dictate the use of an sophisticated measuring device such as the CMM.

Figure 5: Picture of CCD camera mounted on CMM

Stereo vision allows for the extrapolation of depth due to the simple fact that two images taken from slightly different locations will show the subject of the images slightly shifted due to parallax. Figure 6 shows the effect of parallax between two stereo images.

Figure 6: Parallax between two stereo images

To extrapolate depth from a pair of two-dimensional images, corresponding points on each of the images must be identified and the shift of the corresponding points between the two images must be determined. The corresponding points in each image are determined through the application of a segmentation algorithm.

2.1 Review of Automated Digitisation Strategies

Previous research on methods of automating the process of digitisation can be divided into two groups:

- ***In process*** methods are techniques where a new scanning path is planned using the information gathered by the previous scanning pass. Milroy et al.[18] calculated the local surface slope based on the range data from the last data acquisition pass to plan the next best scanning pass. Mauer and Bajcsky[23] used occlusions found in the previous scan to plan the next view which would resolve as many occlusions as possible. Soucy et al.[19] placed digitised data into voxel bins so that neighbouring voxels could be examined for continuity. If a discontinuity was found, the scanner was directed to move to the vicinity.
- ***A priori*** based methods are where the features of an object are known beforehand. Lim and Meng[24] generate possible CMM probe orientations from CAD models prior to generating an inspection path. Sobh et al.[7] also used existing CAD data to direct a CMM probe to inspect a part.

In reverse engineering *a priori* knowledge is not known or available. However, *in*

process methods are often complex and computationally expensive. Milroy[18] reported taking four to ten minutes to calculate the orientation of the scanner head and scanning directions for typical scans of 10 to 20 scan passes. *A priori* methods do have the advantage of allowing the digitisation process to be optimised off-line, thus allowing for the most efficient utilisation of the digitiser. In this work, to take advantage of *a priori* efficiency, a process of generating dimensional knowledge of the object before digitisation was chosen.

A stereo vision system was developed to determine object location and orientation of the part that is to be reversed engineered. However, to apply stereo vision, a common point of interest must be located in both images of the stereo pair. To isolate this common point, a segmentation routine must first be applied to the images. For example, an edge based method of segmentation was devised by Canny[25] to delineate objects in an image based on the recognition of their edges. Although not specifically applied to stereo vision, Canny's work proved pivotal in the fields of edge based segmentation.

The segmentation of reverse engineered objects presents a number of problems due to the objects irregular shape, size and location. This lack of uniformity may defeat most rule-based methods of segmentation. As well, objects may be presented with an unknown number of surface patches. An alternative method, based on neural networks, has proven more robust at segmenting images, especially in the medical fields. Koh et al.[26] used a multi-layer self organising feature map for range image segmentation. To help in medical diagnosis Worth and Kennedy[27] employed a four layer neural network to segment grey matter from a brain scan image. The first layer represents the original image and the three

additional layers are used to separate the grey and white brain matter from the background.

After the images are segmented, identical patches from one image must be matched with the corresponding image in the second image. Marapane et al.[28] reconstructs surface range data through the correspondence of parameterised surface patches. Parameters are calculated for each patch, which describe certain attributes of each patch, such as the height, width and the number of pixels in each patch.

2.2 Development of the CMM Based Stereo Vision System

Depth information can be extracted from stereo images through the correspondence within two or more images of the same target. The distance the target has moved within the images taken from different viewpoints is used to calculate the depth of the target object from the lens. These images, acquired through a CCD camera attached to the end effector of a CMM, are acquired from a known position in space and at a set distance apart. Fortunately, the CMM provides an excellent platform to gather stereo images. It features an accurate and repeatable platform from which the CCD camera can be translated a known distance and to a known position in space. Figure 7 shows the possible movements of the CMM. Once the image pair is gathered, common points in the image must be found. The distance of the common points relative to the image frame is compared (parallax error) and the depth of the point from the front of the camera can be calculated. Thus for each view, a pair of images must be taken, common points found and compared to determine the three dimensional form of the object.

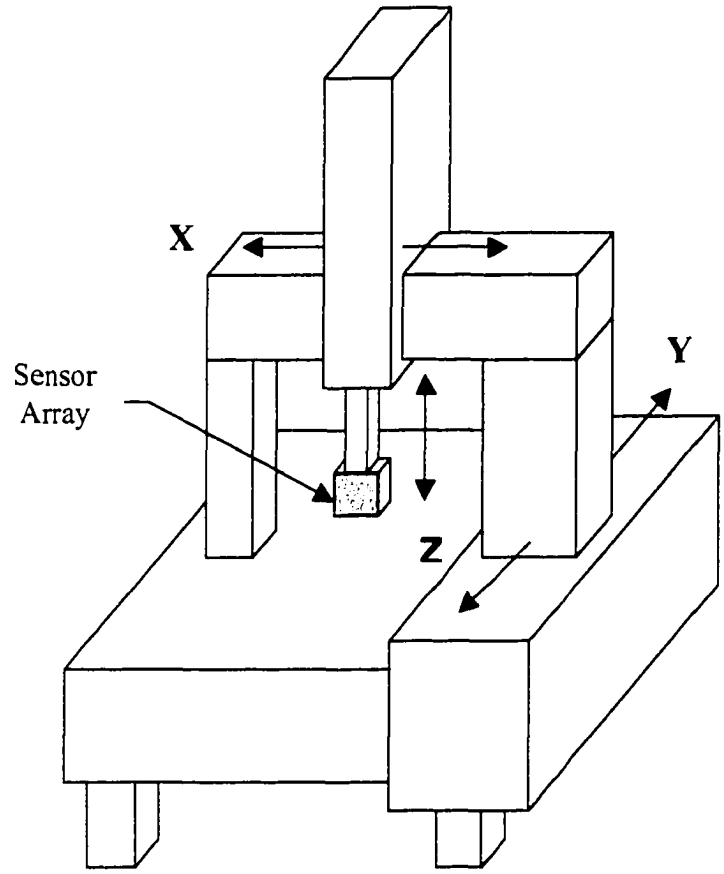

Figure 7: Co-ordinate measuring machine with axis movement shown

One popular method of determining common points between the stereo pair is to compare a specific pattern or combination of edges in the first image and with a similar pattern of edges in the second image. This method requires that an edge pixel enhancement and linking algorithm be applied to both images before the pattern matching algorithm. An alternative method is based on matching similar regions. Based on grouping pixels of similar

intensity, region correlation methods try to match similar regions on each image. Many benefits can be realised by applying region correlation instead of edge correlation, the most important is that of simplicity. For any image, there exist fewer regions than there are edges.

To match corresponding regions in each image, each region is labelled with parameters that describe the regions width, height, centre and mean pixel intensity. A separate algorithm then considers the best fit between parameters describing patch regions in image one, with parameters describing patch regions in image two. Finally, the parallax shift of the subject between the two images is determined and the distance of the subject from the camera is calculated. A flowchart of the stereo vision process is shown Figure 8.

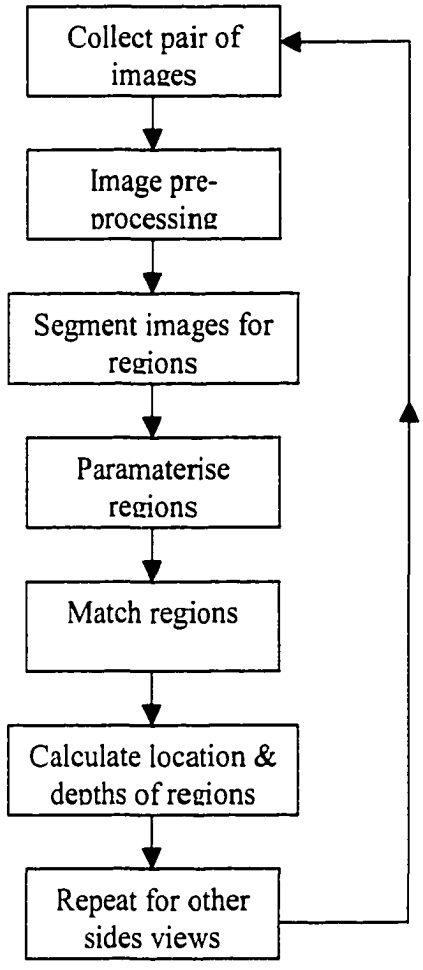

Figure 8: Stereo vision depth extraction process

A robust segmentation algorithm must be used to define the different patches in each of the images. As there is no *a priori* knowledge about the object being examined, the number of patches for each view of the surface is also not known. For this reason, a robust segmentation algorithm, such as those based on neural networks, is used in this work. However, before the segmentation program is applied to the CCD images, a pre-processing algorithm is first applied to reduce image noise and re-enforce image properties.

2.3 Image Pre-processing of the Stereo Pairs

The neural network segmentation program is a region growing algorithm dependent on a grey image gradient to define the boundaries of a surface patch. On sculpted surfaces, where curves are often blended into planar surfaces, resulting in a non-visible edge, it was found that the patches grown by the neural network segmentation algorithm tended to bleed into neighbouring surfaces if an edge is not clearly visible in the stereo image. For this reason, image enhancement routines are applied to the stereo images to better define partially exposed edges.

Therefore, the goals of the image enhancement routines are to reduce the amount of noise in the images and to enhance the natural edges in the image. This is achieved through a number of image processing routines that enhance the stereo images. A 3x3 linear smoothing mask is first used to reduce the effects of noise in the image. To find the edges which need to be enhanced, a 5x5 Laplace of Gaussian mask is used. However, the lines resulting from the mask are two to three pixels wide, a non-maxima suppression algorithm similar to the one described by Canny[25] is used to thin the resulting thick edge lines. An edge following routine is then applied to link pixels into lines, those lines consisting of more than five continuous pixels are extended to the image boundaries. The edge lines are then subtracted from the original stereo images, resulting in an image with enhanced edges.

To test the effectiveness of the edge enhancement routine, an object with a blended surface was selected. The object used is made of a flat L-shaped portion with a spherical portion blended into the elbow of the L. A photo of the object is shown Figure 9. The effect

of these first three image processing algorithms can be seen in Figure 10a, b and c. The edge following routine used to link pixels into lines is shown in Figure 10d. The enhanced edges are then subtracted from the original stereo images, resulting in the image shown in Figure 10e. These enhanced edges provide enough of a barrier to prevent the neural network patches from shifting over onto neighbouring surfaces.

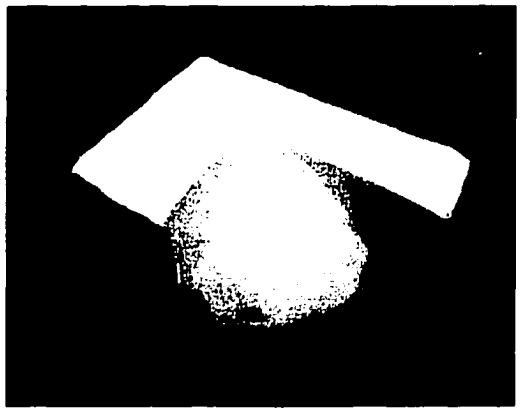

Figure 9: L-shaped blended surface test object

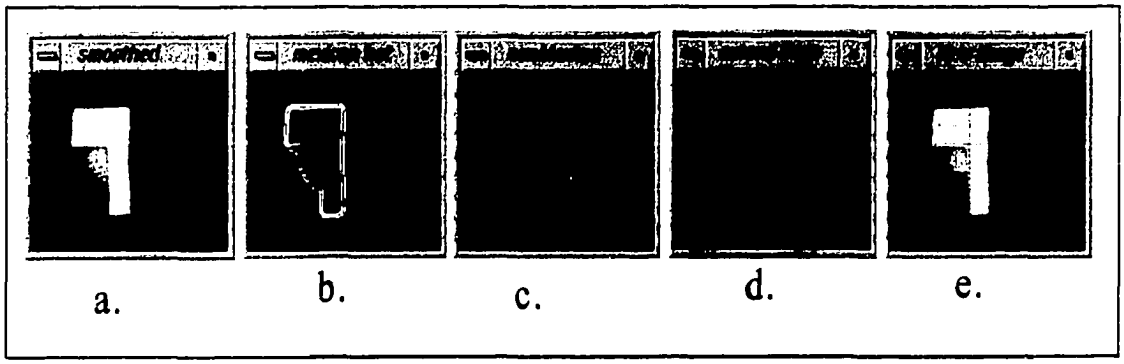

Figure 10: Enhancements routines applied to the left stereo image

2.4 Segmentation of the Stereo Pairs Using Neural Networks

Variation in object shape, size and location dictate the use of a robust segmentation method for reverse engineering applications. Recent research in image segmentation that has employed neural networks, has found that they have proved robust in segmenting regions with irregular and poorly defined boundaries. For these reasons, a neural network based segmentation algorithm has been chosen to define the patch surface boundaries on the objects surface. The network used in this research is based on the Kohonen Self Organising Map (SOM) network. The SOM, a competitive learning network, is based on the effects of a neighbourhood around each output node.

The SOM network consists of n layers of two-dimensional arrays of neurons, with each neuron connected to its immediate neighbours on its own layer and to $(n\text{-}1)$ neurons on all the layers below and above it. (see Figure 11) Each layer with winning neurons, after iteration, will represent a separate surface patch on the object. Therefore, with only nine layers above the input layer, a maximum of nine surface patches can be found. Every input neuron (x,y,0), on the bottom layer, is directly linked to $(n\text{-}1)$ neurons directly above it (i.e. one neuron per layer). These $(n\text{-}1)$ neurons are locked in a competition to be the winning output neuron for the input neuron (x,y,0). The winning neuron excites (strengthen) connectors in a neighbourhood on its own layer but inhibits the neurons on other layers from being declared winners for that specific location. Ten layers are used (n=10) in this work, one layer for the original input image and the remaining nine layers for the output. The shape of the neighbourhood is a square 5x5 neuron patch as shown in Figure 12. Once the network

is initialised, the learning of the network is self organising, with the segmentation routine complete when the output converges, i.e. no new or different winning neurons are declared.

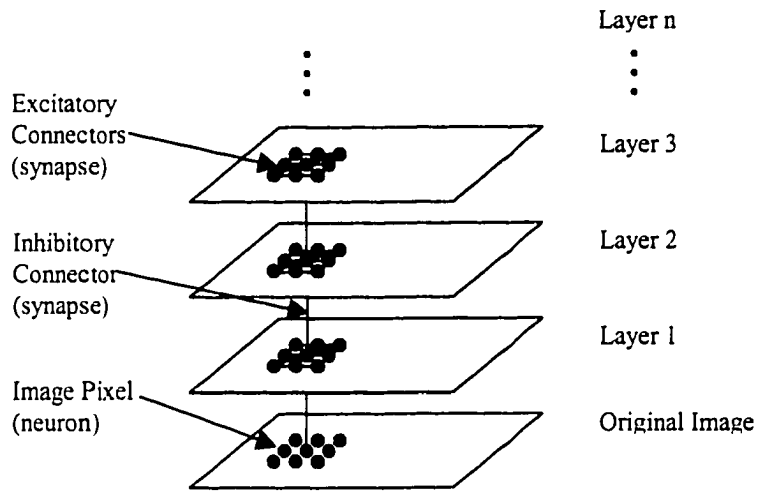

Figure 11: Neural network architecture

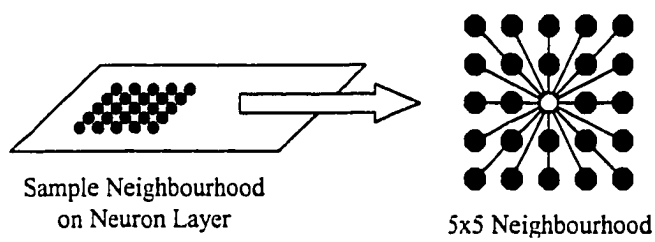

Figure 12: Detail of local neuron neighbourhood

Iteration of the network is accomplished by re-evaluating the excitatory connectors associated with the winning neuron and the inhibitory connectors during each cycle. A neuron is declared a winner for a given pixel location if it is the strongest neuron (determined

by the strength of it's connectors) for that pixel location (x,y,k), where k represents the patch layer. The maximum number of patches that can be found is limited to the number of layers minus 1 (n-1) upon which the neural network is built.

Before the iteration cycles can begin, the neural network must be initialised. A neuron is labelled either true (1) or false (0) to signify whether it belongs to a certain patch (layer k) for each pixel location (x,y,k). Only one layer can be true for each pixel location, (winner take all strategy) which is decided by picking the neuron (x,y,k) that receives the greatest excitation from it's connectors. The network is initialised by assigning one neuron from each layer to being a winner. Therefore, to facilitate the segmentation routine, the initial points for the neuron layer should be selected in areas where the grey levels are constant, allowing the seed points to strengthen their connectors faster, resulting in quicker growth of patches.

To initialise the network, the algorithm starts by scanning through the image, row by row, assigning a new label (1 or 0) every time a new pixel above a background threshold is encountered. This is similar to a raster scanning method described by Wahl[29]. Before a new label is assigned, the algorithm checks to see if surrounding pixels have a similar grey level (i.e. within a certain set range). If they are similar, then the same label is assigned. As the labels are being assigned, a histogram of the labels is kept. The top nine labels, which were assigned the greatest number of pixels, are assigned seed points on the neural network. A flowchart that outlines the method by which the initial seed points are selected is shown in Figure 13. The position of the initial point is determined by finding the centroid of its

labelled patch. An example of the initialisation sites can be seen in Figure 14.

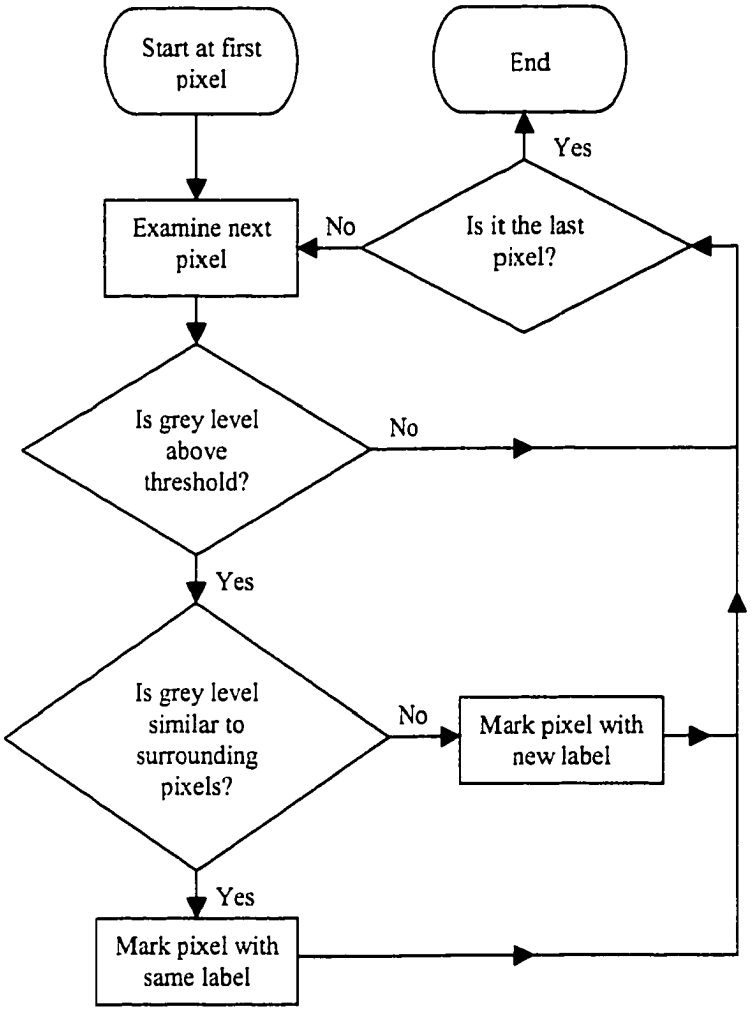

Figure 13: Algorithm to Determine Neural Network Initialisation Points

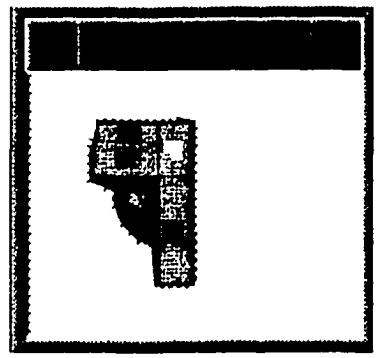

Figure 14: Sample initialisation points

Image segmentation into separate surface patches is the result of the competition between the neurons for each image pixel. The iteration cycle consist of three steps:

1) The excitatory connectors, consisting of the links between a pixel and its closest 24 neighbours, are updated.
2) The inhibitory connectors, linking each pixel at location (x,y) to its counterpart in the layers above and below it, are recalculated.
3) Finally, using the new values for the excitatory and inhibitory connectors, the strength of each neuron is calculated and a winner is declared.

The 24 excitatory connectors in the neighbourhood around every neuron only gain strength if they are attached to winning neurons. The rate at which the excitatory connectors are strengthened is given below (Equation 1):

$$\frac{dx_{(x,y)(i,j)}}{dt} = (A\, x_{(x,y)(i,j)} + B\, \sigma_{grey} + C\, \theta_{mom})\ \eta_{i,j} \qquad (1)$$

For each iteration *dt*, the strength of the connector (x) between central pixel (x,y) and (i,j) is increased by *dx*. The first term of Equation 1 is the growth strength; where constant A represents the learning rate of the term, the second term, B increases connector strength in areas where input neurons (image pixels) are of similar intensity, and the third term, C provides additional momentum for the growth of large patches. Through experimentation on over 25 different sets of stereo images, of different objects it was found that a value of A=1 allowed for a controlled growth rate of neuron patches without a single patch dominating all other patches. A value of B=10, promoted early growth and the eventual domination of patches that were seeded in areas of consistent grey level. Finally, a reduction in the total number of iterations was achieved by using a value of C=10 for the momentum term.

The inhibitory connectors (for a neural network of k layers, there are k-1 connections) are all set equal to 1. The rate at which the inhibitory connectors change is (Equation 2):

$$\frac{d\gamma_{(x,y,k)}}{dt} = + D\gamma_{(x,y,k)} \quad for \ \eta_{(x,y,k)} = 1 \tag{2a}$$

$$\frac{d\gamma_{(x,y,k)}}{dt} = - D\gamma_{(x,y,k)} \quad for \ \eta_{(x,y,k)} = 0 \tag{2b}$$

Referring to Equation 2, for each iteration dt, the inhibitory connectors change by a value of dγ. The constant D represents the learning rate and γ is the strength of the inhibitory connector at (t-1). A value of D=0.1 was found to produce satisfactory results, in preventing weaker patches from claiming pixels belonging to stronger patches.

The "strength" of which a pixel belongs to one patch, and not another is determined by the magnitude of its neuron, η, value. The relative value of the neurons determine which neuron (x,y,k) will be the winner for that particular pixel (x,y) location. The calculated value of the neuron is shown below in Equation 3:

$$\eta_{value(x,y,k)} = \sum_{m=1}^{5}\sum_{n=1}^{5}\left(x_{(x,y)(x-2+m,\,y-2+n)} \bullet \eta_{(x,y,k)}\right) - \gamma_{(x,y,k)} \bullet \eta_{(x,y,k)} \tag{3}$$

Employing the winner take all strategy, the neuron $\eta_{(x,y,k)}$ with the maximum value that is significantly larger than other values for that location, is declared the winner (1) while all other neurons on the other layers at pixel location (x,y) are losers (0). The connectors are iterated until the values converge. (i.e. no more winning neurons are declared).

2.4.1 Location of Concavities on the Patch Surface

One important aspect for which a CMM touch trigger probe is ideally suited to measure is the location and size of concavities, such as holes on an object being reverse engineered. Concavities, such as reamed holes, are used as bearing surfaces or for locating pins. In this work, concavities are found by searching for voids inside the patch boundaries previously defined through the neural network segmentation. To find potential concavities inside of each patch, a two step algorithm is applied. The first step uses a raster scan to identify and label the boundaries of a neuron patch. It then checks for non-neuron designated pixels inside of the neuron patch. If non-neuron pixels are found inside, they are labelled with either a new label, or the same label if neighbouring pixels have already been labelled. The raster scan is applied in both the X and Y directions. An example of a X-direction raster scan for one line of pixels is shown in Figure 15. The large X's denote boundary pixels of the neuron patch, whereas the small x's denote interior non-neuron pixels.

The second step is used to insure that the labelled pixels inside of the neuron boundaries are in fact potential candidate sites for a hole concavity. Pixels that have been tagged with the same label are use for boundary identification, as well, the centroid of the tagged pixels is calculated. If the labelled pixels are in fact a circular hole, their boundary pixels should be a constant distance, R, from the centroid. Therefore, from each boundary point, the distance was calculated to the centroid. A sample of some of the images with test hole concavities that were used to test the algorithm as shown in Figure 16. It was found

through testing, that if the standard deviation of the radius was less than 15% of the radius, R, that the labelled pixels were most likely a hole concavity.

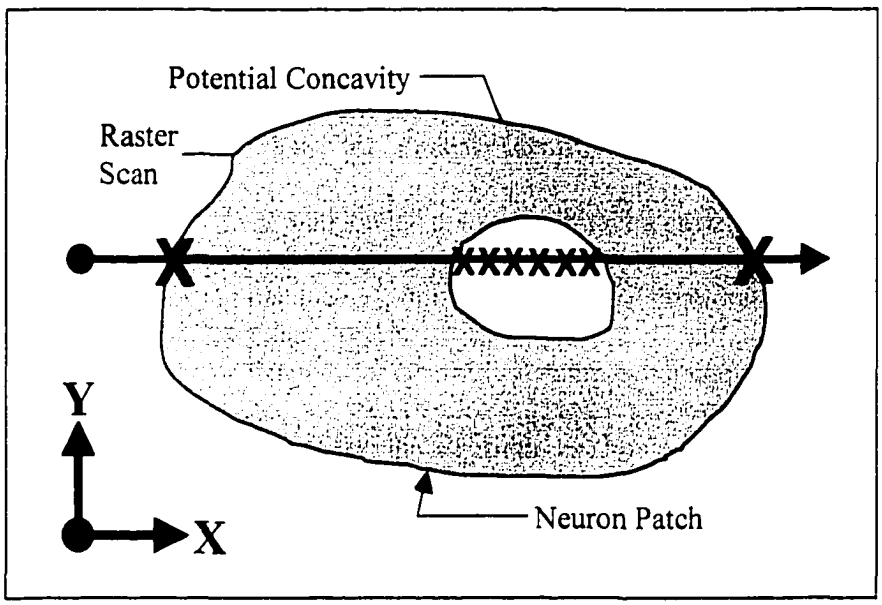

Figure 15: Raster scan example to find concavity

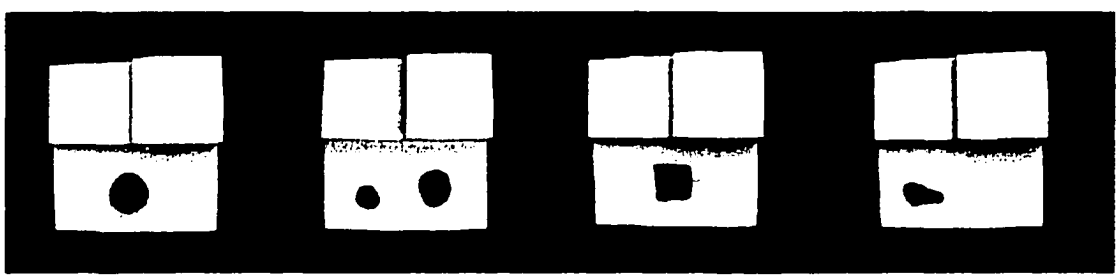

Figure 16: Sample of images used to test hole concavity algorithm

The exact location and radius of the holes found is derived from the co-ordinate information calculated for each patch. It is assumed that the top of the hole begins at the same height as the patch surface. A specific-hole measuring routine is then incorporated into the CMM probe tool path.

2.5 Stereo Image Pair Correspondence

The segmented stereo images are used to extract physical patch location and size by matching parameterised regions. Each patch is described by four parameters; i) Mean grey level for each patch (average of the image pixels intensity) is described in Equation 4. ii) Number of pixels labelled by the neuron patch as outlined in Equation 5. iii) Maximum width of the neuron patch, and similarly, maximum length Equation 6 iv) i and j co-ordinates of the centroid of each patch is calculated using Equation 7.

$$G_{mean,k} = [\sum_{x=0}^{m}\sum_{y=0}^{n} (\eta(x,y,k) \bullet \eta(x,y,0))] \ / \ P_{sum,k} \tag{4}$$

$$P_{sum,k} = \sum_{x=0}^{m}\sum_{y=0}^{n}\eta(x.y.k) \tag{5}$$

$$width = \max(x) - \min(x) \quad of \quad \eta(x,y,k) \tag{6a}$$

$$length = \max(y) - \min(y) \quad of \quad \eta(x,y,k) \tag{6b}$$

$$centroid \quad i = \sum_{x=0}^{m}\sum_{y=0}^{n} (\eta(x,y,k) \bullet x) / P_{sum,k} \tag{7a}$$

$$centroid \quad j = \sum_{x=0}^{m}\sum_{y=0}^{n} (\eta(x,y,k) \bullet y) / P_{sum,k} \tag{7b}$$

A patch from the first image of the stereo pair is compared with patches from the second image of the stereo pair. The first image patch which best fits a second image patch with less than 10% difference are considered a pair. It was found through experimentation with objects of different number of surface patches, that using a 10% difference gave consistently correct results. This process is continued until all the patches from the first image are matched with patches from the second image.

2.6 Depth Calculations and Transformations of Surface Patches

The extrapolation of 3-D depth information for a pair of images is accomplished through the matching of two images of the same area. The parallax shift caused be the different position from which the images were taken is proportional to the distance the patch is in front of the lens. The CMM controller provides two essential pieces of information; the exact position of the camera in space and the distance the camera was shifted.

The distance of the patch from the lens can be calculated using simple geometric principles. Figure 17 shows a diagram for calculating depth from a pair of stereo images. The centroid of each patch is used at the point of comparison between image 1 and image 2. Equation 8 is used to determine the depth of a patch from the camera lens.

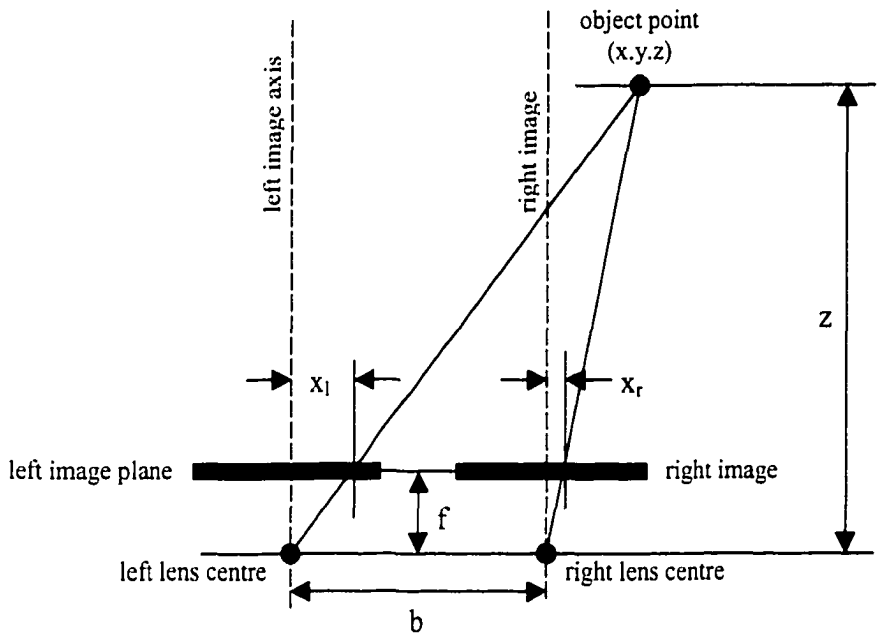

Figure 17: Diagram of stereo vision geometry

$$z = \frac{b \bullet f}{(x_{l'} - x_{r'})} \tag{8}$$

Where distance of separation between the two images is b, f the focal length of the lens and x_l and x_r are the location within the image. Therefore, for each matched pair, the depth of that patch can be calculated.

2.7 Examples of Neural Network Segmentation Performance

The algorithm was implemented with C++ code on a SGI Indy workstation. Although pre-written neural network algorithms do exist, such as the NN toolbox 3.0 for MATLAB[30], it were not used in this work, as it was not readily available. A stereo pair of images was taken from a view directly above the object. The photograph in Figure 18 shows the test object, comprised of a L-shaped planar surface and a blended spherical surface. Initial seed points are shown as small squares in Figure 14. For the purposes of clarity, only the right image is shown. A scale factor of five was selected to reduce the image size from the original 640x480 pixels so that the required computation could be reduced. It was found through experimentation that the factor of five proved to be a good compromise between image resolution and processing speed. Figure 10 shows the application of the edge enhancement algorithm before using the neural network segmentation.

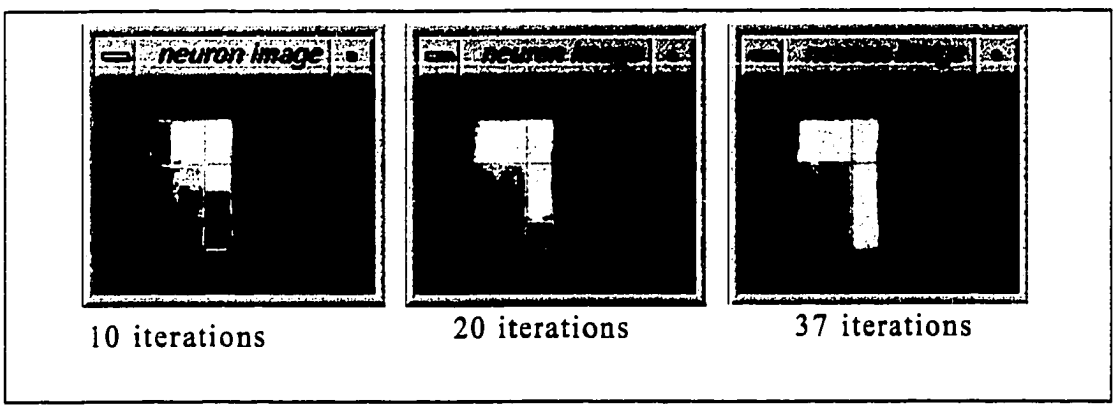

Figure 18: Neural network iterations of stereo image

From the initial nine seed locations, the algorithm correctly reduced the number of

patches visible on the top surface to two. The segmentation process for different iteration intervals is shown in Figure 18. Computation time of all 37 iterations was 12 minutes for the algorithm running on a SGI Indy with a R4600 processor running at 132 Mhz. It is interesting to note that the seed locations, along the centre in Figure 18 did not grow. As those patches were started in a location of changing grey values and did not grow as fast as patches started in even grey values and were soon overtaken.

2.8 Accuracy of Stereo Vision Positioning

The resolution of the stereo vision positioning system can be determined using a camera resolution test target. A test target as pictured in Figure 19 is used to test the physical limit of the resolution per pixel of the CCD image. Positioning the CCD camera at its furthest location from a surface (worst case scenario, with the CCM end effector at its maximum height) the resolution of the CCD system can be calculated. Examining the resulting CCD image, Figure 20, the minimum width possibly determined by the CCD camera, positioned 600mm away from the target is 3.8mm.

Figure 19: USAF camera resolution test chart

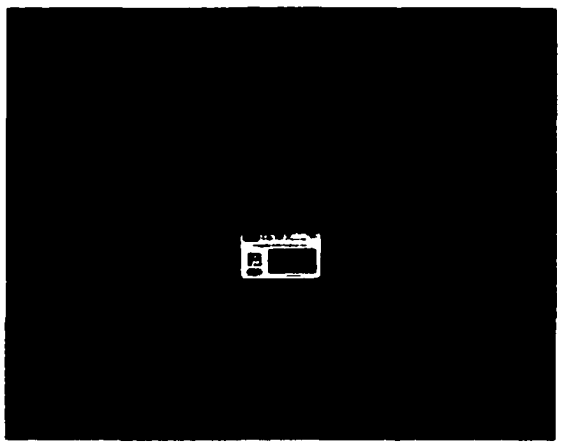

Figure 20: View of test chart from CCD camera

Calculating the minimum width discernible from a perfect lens and a CCD resolution of 480 by 640 pixels, using a 8mm focal length lens with a field of view of 41.2°, at a lens to surface height of 600mm, 2.8mm should be visible on the target. This assumes the lens projection covers the full width of the long dimension of the CCD array. This is in close agreement to the test results if allowances are made for lens distortions, noise from the CCD and poor image contrast due to insufficient lighting.

Therefore, from this analysis, it can be concluded that the stereo vision system has a positioning resolution of ±4mm. However, as the camera must be positioned closer to the object for side views, due to space limitations on the CMM, the resolution of the stereo vision positioning would improve to less than 4mm.

Chapter 3 - Automated Part Digitisation Employing a Multiple Sensor Approach

The current limitations of reverse engineering automation, outlined in Chapter 1, are addressed in this chapter. In particular, a multiple sensor approach to the digitisation automation issue is presented. This work demonstrates the considerable advantages inherent in concurrently employing a CMM touch probe and laser range finder to the problem. Using this technique, digitisation time is reduced, the spectrum of part types that can be digitised in one set-up can be increased and a greater accuracy in defining specific features is achieved.

Whether using a mechanical touch probe or a non-contact laser scanner, a reasonable amount of accuracy must be used to position the sensor above the object. The touch trigger probe requires that the tip of the probe contact an object's surface without disturbing the objects positioning and without damage to the probes delicate sensor. Whereas, the laser scanner has a limited range that the object's surface must lie within. For example, the

Hymarc laser scanner has a field of view limited to a 80mm depth and a 80mm width. Therefore the digitisation process is often based on operator judgement and experience, a situation where the most efficient path is not usually achieved. As a solution to these difficulties, automation of the digitisation operation is investigated in this work.

3.1 Previous Research on Multiple Sensor Digitisation

Limited research has previously been attempted on the problems faced by developers of reverse engineering software. Mason and Grun[31] created a software tool to automatically determine the locations for a multi-station vision system. The system consisted of standard off the shelf CCD sensors and frame grabbers, that applied "sensor placement constraints" to calculate sensor locations. For example, the feasibility of the object must be such that all features are visible from the selected viewpoints. The working environment places constraints on the sensor itself, as it should not conflict with any obstructions in the working space. The viewing angle of the sensor must also be selected such that the object features can be reliably measured. These constraints can then be combined to produce an acceptable viewpoint. However, their work did not address the direction and movement of any physical measuring devices.

An alternative approach was attempted by Tarbox and Gottschlich[32] to use a CAD model to find a set of sensing operations that completely cover an object to be measured. This sensing plan is generated using a hypothesise-and-verify approach, where the plan is verified using a simulation of the object. The hypothesise-and-verify cycle is repeated until

an acceptable plan is found. Finally, the sensor information collected is tested for completeness to determine if additional sensing must be performed.

In earlier work, Tarbox used a constraint based approach to determine if the sequence of sensing operations is capable of measuring every point on the surface of an object. To achieve a sensing plan with the fewest number of sensing operations, their algorithm selects hard to see regions and large surface areas as a priority. To achieve these two vastly different goals, a "difficulty-to-measure" weight is associated with each surface point. The algorithm then uses the first sensing operation where the sum of the weights is maximised (i.e. most hard-to-measure points).

In method applied by Sobh, et al.[33] at the University of Utah, a CCD camera is used to guide a CMM touch probe towards the object being measured. Although their research is still in the early stages, they hope to use the CCD camera to observe the touch sensor as it manoeuvres to contact the part so that breakage of the touch probe is avoided. As their reverse engineering system is stereo vision based, the touch probe is required to accurately measure features, which stereo vision cannot adequately resolve. Their research has been limited to the location of the touch probe in the CCD camera's vision by the use of simple thresholding of the CCD images.

A more comprehensive approach to the problem was formulated by M. Milroy[18], an automatic scanning process is used where the next scan is determined by the previous scan. First an initial scan is taken where the range of possible orientations of the outer edges of the scan is estimated to select subsequent gazes for the laser scanner. In order to select the

best gaze, "air vectors" are calculated at the scan edges. The air vectors are then used to calculate the next best gaze. This process works well except that the process is reliant on the user to supply the first initial scan and for the user to decide where to end the process.

A similar approach is used by Soucy et al.[19] to determine the next scanning pass from the previous scanning pass. Using a Hymarc laser digitiser, Soucy's algorithm actively analyses the previous scan data, finds the edges where data is incomplete and computes a new sensor trajectory to collect the missing data, taking into account the limitations of the sensor and manipulator. To find areas of the object where the scanner has yet to digitise, a voxel approximation of the surface is made of the model from the data that has been collected. Assuming C_0 continuity, any part of the voxel surface, where it is not continuous, subsequent scanning is required. Again, this method suffers from the same problems as those encountered by Milroy, the requirement that the first scan is made manually, and that each additional scan is made after calculations have be done on previous scans.

3.2 CMM Part Path Generation

Previous methods of automating the digitisation procedure have concentrated on a scan, calculate and re-scan iterative process to ensure the complete coverage of the object's surface. Although this method does eventually accomplish complete coverage of an object's surface, it suffers from two main problems. First, the number of required scanning passes is not known before hand. This is a problem, however, with objects which have occluded surfaces where a laser digitiser would be unable to gather data, the algorithm may repeatedly

try to gather data in those areas. Second, time is required between digitisation scans to calculate the next scanning pass. This is contradictory to the purpose of automating the scanning process, which is to minimise the time required for the collection of surface points by the digitiser.

These problems are addressed by the application of a preliminary coarse scan, such as the stereo vision algorithm discussed in Chapter 2. By pre-planning the scanning sequence beforehand, the digitisation process can be optimised. As well, the best digitisation sensor for the type of surfaces being reverse engineered can be selected.

At this stage in the reverse engineering process, the neural network based stereo vision algorithm has generated the following spatial information:

1) Number of patches comprising the object's top and sides.
2) The centroid (X_c, Y_c, Z_c) of each patch. (Note that the data generated by the stereo vision system, touch probe and laser scanner are both referenced to the same global co-ordinate system of the CMM.)
3) The length and width of each patch.

The object and its surface patches have been coarsely described by a rough range map created by the stereo vision system. This range map is the basis from which CMM path is planned. In this work, two types of digitisers are used to collect the measured surface data points, a CMM touch trigger probe and a 3-D laser scanner. Specifications of the CMM, CCD camera, Renishaw touch probe and the Hymarc laser scanner are given below in Table 3.

Table 3: Equipment Specifications

Equipment	Specifications
Mitutoyo BHN710 CMM	working vol.: 700x1000x600mm encoder resolution: 0.5μm controller: CMMC 35
Renishaw Touch Trigger Probe	PH8 probe head, TP2 touch probe max repeatability at stylus 0.35μm
B&W CCD Camera	NEC model TI-324A, 380,000 pixels, Computer 8.5mm f/1.3 close focus lens
PC	486 based PC, linked to CMM controller
Workstation	Silicon Graphics Indy, VINO video capture board

3.3 Touch Probe Part Path Generation

The application of the hole concavity location algorithm, as explained in Chapter 2.4.1, will result in the rough estimation of possible holes in the surface. The reverse engineering algorithm must determine the appropriate path to efficiently collect touch-trigger point data of the holes.

A Renishaw touch probe system used in this research is shown mounted on the Z-axis arm of a gantry style CMM end effector in Figure 21. The touch trigger probe is a contact digitising device, requiring the user to lightly contact the surface with the stylus of the touch probe. A highly sensitive, omni-directional electronic switch located in the probe head, signals the CMM when the probe stylus comes into contact with the object surface.

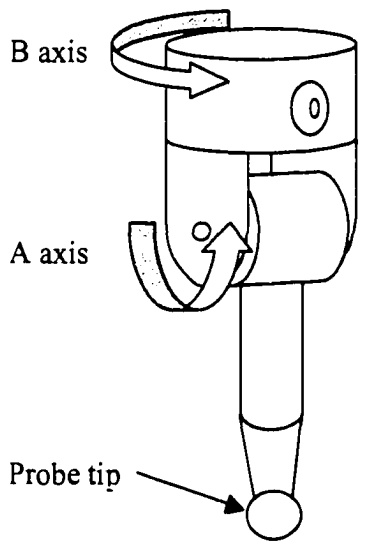

Figure 22: Touch trigger probe diagram

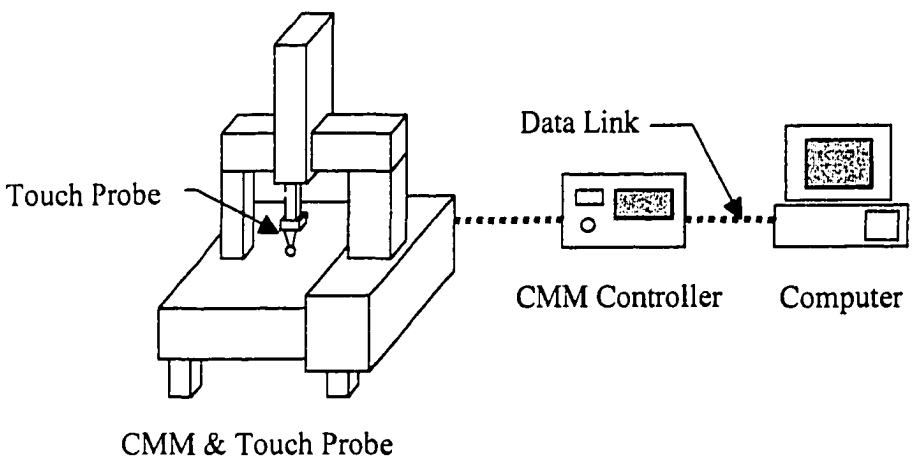

Figure 23: CMM touch trigger probe control schematic

To optimise the efficiency of the data collection process, one of the classical solutions of the "travelling salesman algorithm" is applied. According to the rules of this solution, the

path of the probe is constructed by selecting the next closest measurement point to the position where the probe is currently. Starting at the home position, the closest measurement point to the home position is chosen as the first point in the tool path. The second point in the tool path is the next closest point not chosen. The selection process is repeated until all measurement points are selected. No points are allowed to be selected twice.

The algorithm generates a program to direct the CMM movements of the touch trigger probe after the path of the touch probe is determined. The following details are accounted for when generating the part program:

1) One part path program is created for every orientation of the touch probe.
2) The CMM directs the touch probe to move at the required stand off distance above the surface and then measure the hole concavity at the required measurement speed.
3) Collision avoidance is achieved by withdrawing the probe to a distance of 100mm above the "imaginary plane" before the probe is moved to the next hole concavity. The imaginary plane is deduced from the probe orientation and the boundary box of the object as derived from the stereo vision algorithm.

3.4 Laser Scanning Path Generation

The Hymarc sensor head is directed to scan each patch individually from co-ordinate information derived from the stereo vision algorithm. A separate part program is created for each orientation of the Hymarc sensor head. A two degree of freedom trunnion allows the sensor head to be turned to a variety of orientations quickly and repeatedly. A diagram of

the possible sensor head orientations is shown in Figure 24 and a photo of the sensor mounted on the end of the CMM can be seen in Figure 5.

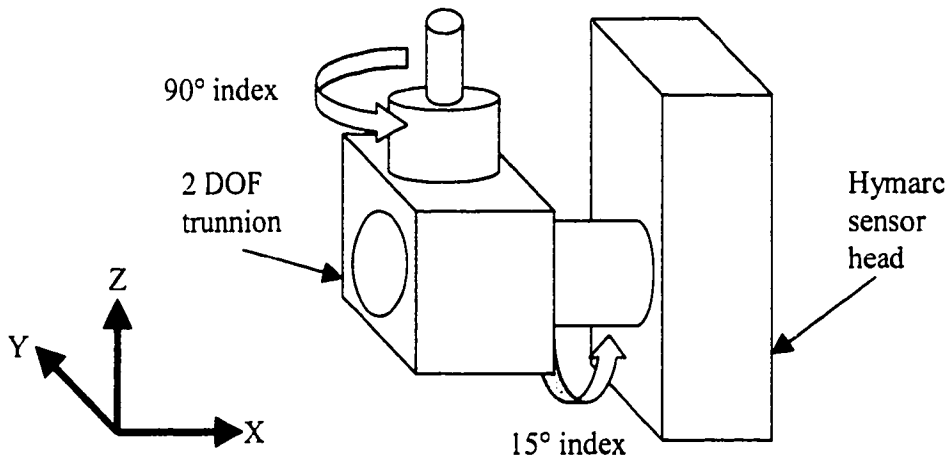

Figure 24: Hymarc sensor head trunnion mounting

The Hymarc laser system must have a minimum standoff distance of 80mm from the objects surface while it is being digitised. The scanning window is 80mm wide and 80mm deep. Digitisation of each surface patch is achieved by traversing the sensor head across the length of each patch. Should the width of a patch be greater than 80mm, two or more overlapping passes are required to digitise the surface. An example of a typical sensor head path plan can be seen in Figure 25. For the three surface patches shown, three separate passes, labelled 1, 2 and 3 are required. CMM movements, like the touch probe, are controlled through a PC link to the CMM controller. Digitised data from the Hymarc sensor head is sent via an Ethernet connection from the Hymarc controller to a SGI Indy workstation. A schematic of the control and data collection is shown in Figure 26.

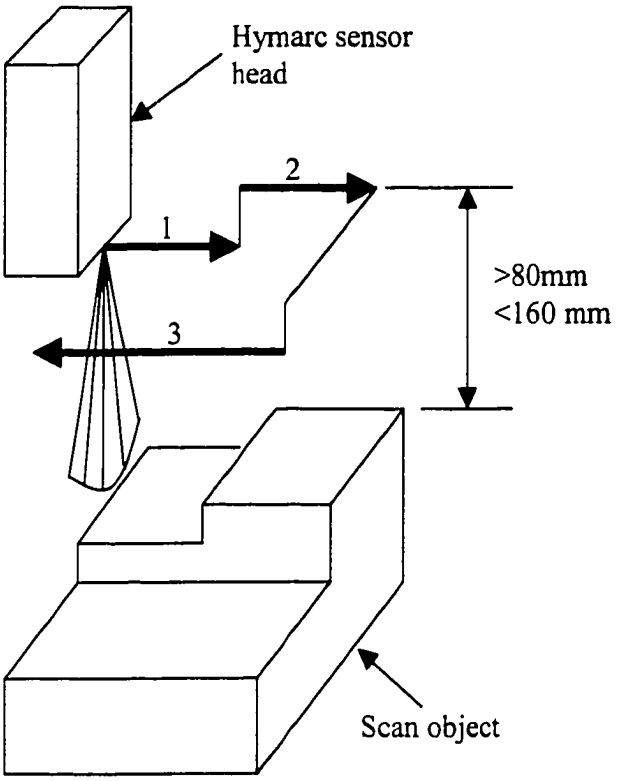

Figure 25: Sensor head traversing path

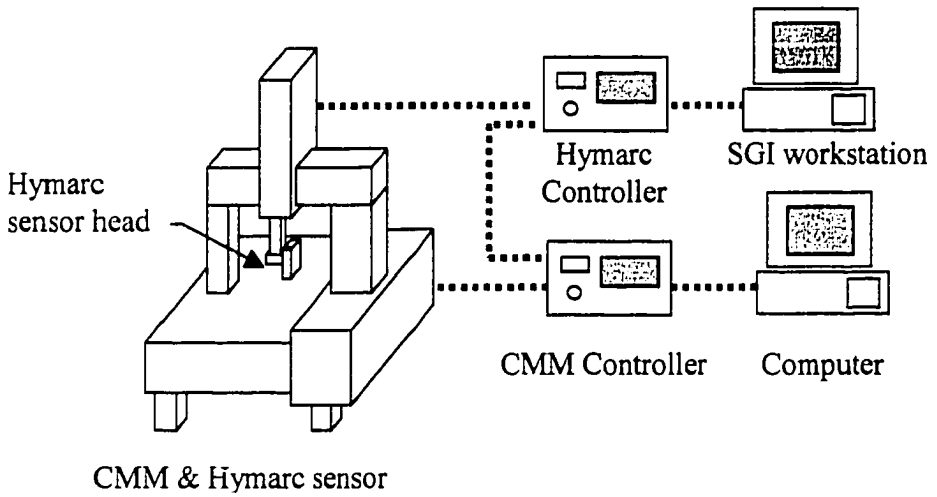

Figure 26: CMM/Hymarc control diagram

Path planning of the Hymarc sensor head is optimised for efficiency by employing the same "travelling salesman" solution as the used for the touch probe. For each patch, two points are calculated for each scanning pass: an initial position at one far end of the patch and the last position at the opposite far end of the patch. Should the patch require more than one scan pass, additional first and second positions are added for that patch. With the scanner starting a position at the centre of the CMM bed, 600mm above the scanning bed top, the distances to the remaining first and second points are calculated. The patch with the closest first or second point determines the first patch to scan. The next closest first or second point to the end of the last scanning pass determines the next patch to scan. This process is repeated until all of the patches have been selected. A 10mm overshoot is added to the start and end of each scan pass to allow the CMM time to achieve a constant scanning speed before data is taken. Figure 25 demonstrates the optimum path for the three-stepped object discussed earlier.

3.5 Examples of Scanning Path Generation from Stereo Images

To test the path generation algorithm, a few sample objects have been selected. The CCD images used to plan the scanning path for the three-stepped object can be seen in Figure 27. In Figure 28 the CCD images used by the segmentation algorithm to create a course description of the object are shown. For clarity, only the first image of each stereo image pair is presented. The neural network segmentation algorithm was able to quickly converge

to a solution for each of the images. The results of the segmentation can be seen in Table 4. Total time for acquiring and processing the images was less than one hour, of which much of the time was spent physically positioning and rotating the CCD camera to the proper orientation. A sample of the CMM path plan for the top surface is shown in Figure 29. The algorithm was able to generate five separate path plans, one for each of the five sides viewed by the CCD camera. The sixth side, bottom side is not visible by the CCD stereo vision system and thus a path plan was not generated. The object was scanned by the Hymarc sensor head in less than one hour using the generated path plan. The results of the scanning are shown in Figure 30. It should be noted that two surfaces were both not detected by the stereo vision algorithm and have thus not been scanned by the Hymarc scanner. These areas have been identified in Figure 30. Possible reasons for the exclusion of the surfaces is the lack of shadow delineation on the CCD images for these surfaces. This can be readily seen in Figure 28, images "b" and "d" where the excluded surfaces are not shown clearly separated.

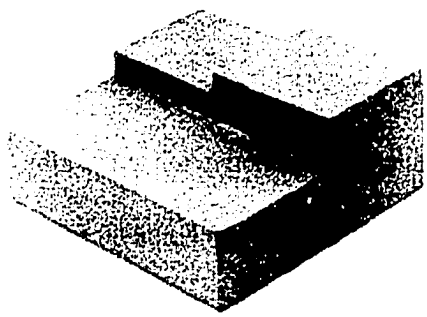

Figure 27: Three-stepped Planar Test Object

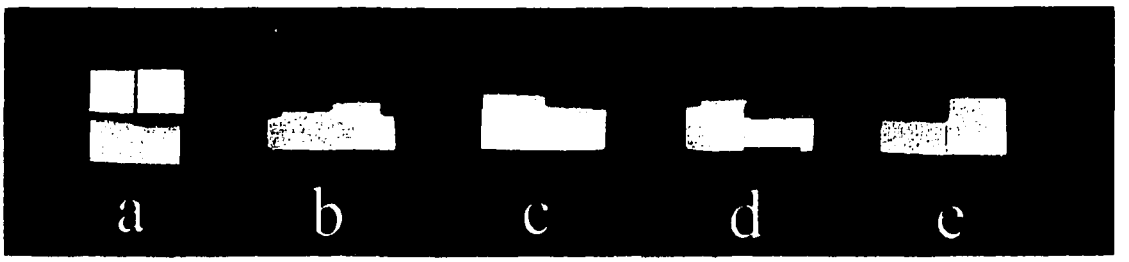

Figure 28: Three step object - CCD images
a – top, b – front, c – back, d – left side, e – right side

Table 4: Results for the segmentation of the 3-step object

	no. of patches	no. of seeds	no. iterations left image	no. iterations right image
View a	3	9	8	21
View b	1	9	11	14
View c	1	9	12	13
View d	1	9	14	16
View e	1	9	12	11

```
111, Vince-O-matic CMM geopak generated code
111, filename g_test.asc
111, number of matched patches: 3
79, 3
53, 7, 1
95
74, Press [F2] for step mode
99, 25.000, 8.000, 3.000, 3.000, 100.000
89, 33.4507, 485.517, 298.892
99, 25.000, 8.000, 3.000, 3.000, 100.000
89, -1.74727, 485.517, 218.134
99, 8.000, 8.000, 3.000, 3.000, 100.000
74, Prepare for Hyscan data gathering
89, 68.6488, 485.517, 218.134
99, 25.000, 8.000, 3.000, 3.000, 100.000
89, 51.5791, 482.885, 211.472
99, 8.000, 8.000, 3.000, 3.000, 100.000
74, Prepare for Hyscan data gathering
89, 123.615, 482.885, 211.472
99, 25.000, 8.000, 3.000, 3.000, 100.000
89, 124.354, 542.134, 204.381
99, 8.000, 8.000, 3.000, 3.000, 100.000
74, Prepare for Hyscan data gathering
89, 0.926384, 542.134, 204.381
89, 62.6402, 542.134, 298.892
99, 100.000, 8.000, 3.000, 3.000, 100.000
112
```

Figure 29: Sample of path code for CMM

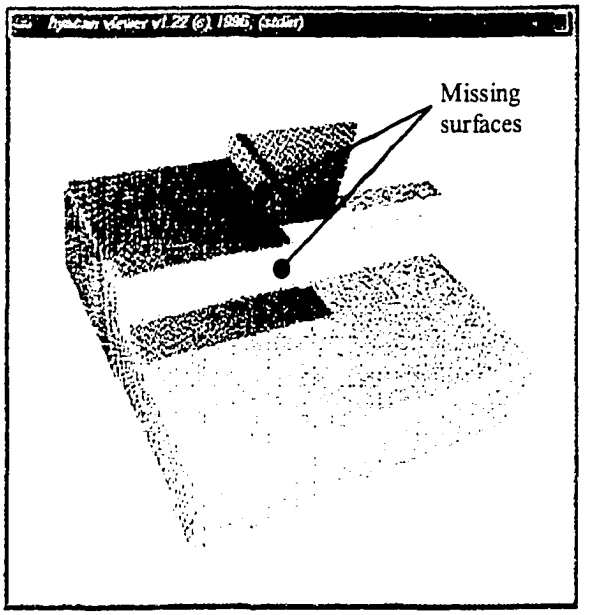

Figure 30: Scanning results of 3-step object

The object in Figure 31 was chosen to test the stereo vision system for both its ability to find surface patches but to also to test its ability to find hole concavities. Again the neural network based segmentation algorithm quickly converged to the correct number of patches for each of the stereo views. The CCD images used by the segmentation algorithm are shown in Figure 32. Less than one hour was required to acquire and segment the stereo image pairs. Results of the segmentation and matching algorithm can be seen in Table 5. A total of six part paths were created; five paths for the Hymarc sensor head, one part path for the touch trigger probe. A sample of one of the part paths, for the touch trigger probe, is shown in Figure 33. The digitised surface resulting from the automatic guidance of the Hymarc sensor head is displayed in Figure 34. Once again, the automated digitisation with the Hymarc sensor required less than one hour.

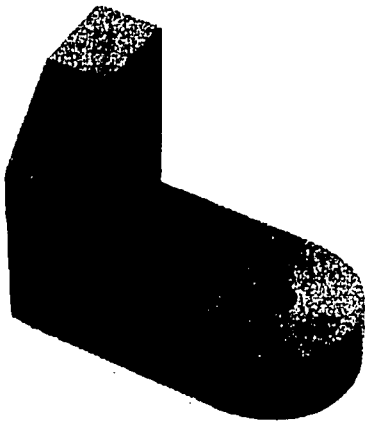

Figure 31: L-bracket test object

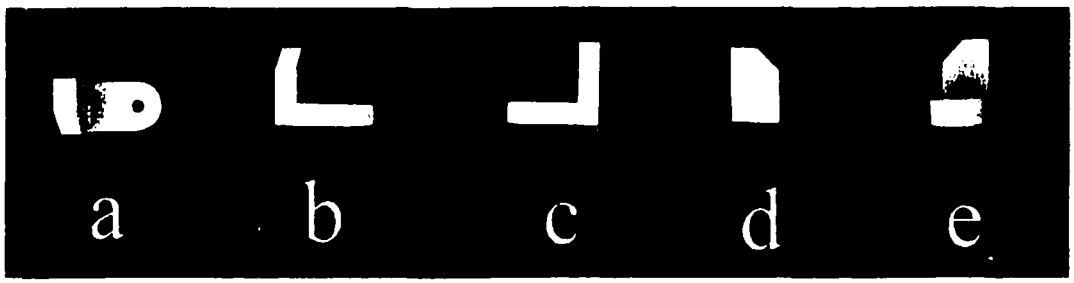

Figure 32: L-bracket with hole concavity

Table 5: Results of the segmentation of the bracket

	no. of patches	no. of holes	no. of seeds	no. iterations left image	no. iterations right image
View a	3	1	9	22	32
View b	2	0	9	24	30
View c	1	0	9	16	19
View d	1	0	9	23	14
View e	2	0	9	21	21

```
111, Vince-O-matic CMM geopak generated code
111, filename c_top.asc
111, number of matched patches: 4
111, probe position: A pos=0  B pos=90
111, probe direction: +ve X
34, 4
73, 1, c_top.dat , 1, 1
74, Is probe calibrated A=0  B=90?
79, 3
74, Is new probe calibration stored?
83, 2
53, 7, 1
95
99, 25.000, 8.000, 3.000, 3.000, 100.000
89, 60.1717, 271.391, 298.892
89, 40.7708, 271.391, 271.709
8, 0, 1, , 0
85, 40.7708, 271.391, 266.709, 0, 90, 0
85, 40.7708, 271.391, 266.709, 0, -90, 0
85, 40.7708, 271.391, 266.709, 90, 0, 0
85, 40.7708, 271.391, 266.709, -90, 0, 0
85, 40.7708, 271.391, 261.709, 0, 90, 0
85, 40.7708, 271.391, 261.709, 0, -90, 0
85, 40.7708, 271.391, 261.709, 90, 0, 0
85, 40.7708, 271.391, 261.709, -90, 0, 0
102
89, 40.7708, 271.391, 271.709
89, 60.1717, 271.391, 298.892
99, 100.000, 8.000, 3.000, 3.000, 100.000
112
```

Figure 33: Sample of CMM path code for touch trigger probe

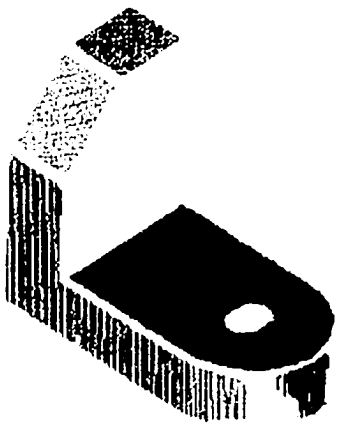

Figure 34: L-bracket 3-D digitised data

Chapter 4 - Surface Geometry Extraction

Geometry extraction is the process of finding standard geometric surfaces, such as planes, spheres and cylinders, in a cloud of surface points. These surfaces are common engineering primitives, often used in manufactured parts. Hakala[34] reports that 85% of manufactured parts can be modelled by simple quadric surfaces. Given that most manufactured parts are still made on either a vertical milling machine or turned on a lathe, this is not especially surprising. The automatic extraction of geometric entities without a high level of operator interaction should greatly increase the ease to which engineers can apply reverse engineering technologies.

One focus of this research is to replace the cloud of digitised data with standard CAD surfaces, thus it is important that the fitted surface accurately model the data points. However, as more parts are being made from plastics, free-form surfaces are beginning to play a larger role in manufacturing. In the event that a primitive quadric surface does not

accurately model the collected data points, the surface will be assumed by the program to be a free-form surface and a 3-D mesh will be fitted to the data points. It will be left to the user as to whether a more complicated surface, such as a B-spline surface, is fitted to the mesh, using built in surfacing routines in CAD packages, such as AutoCAD Surfacer.

4.1 Review of Reverse Engineering data fitting methods

Several different methods of fitting quadric surfaces to digitised data have been researched. Much of this work has focused on the classification of surface shape to help determine the quadric surface parameters. Hoffman and Jain[35] investigated a linear least squares method to calculate surface normals for a plane over a m x m neighbourhood in order to classify the surface type. Surfaces are classified as planar, convex or concave based on a statistical test for the trend in curvature values. After the surface segmentation and classification, the parameters can be used to fit the appropriate surface model, although this is not discussed in the paper.

In work by Bolle and Cooper[36] to estimate the position of an object, the range data is first partitioned into cubic windows (voxels). For each of these partitions, the likelihood of the data representing either a plane, cylinder or sphere is calculated and the appropriate surface is chosen using a Bayesian decision algorithm. Bradley[14], extends this work to reverse engineering by using a similar principle to refine the estimate of the parameters for quadric surfaces.

Flynn and Jain[37] used a hypothesis testing method to test if a surface is either planar

or non-planar. If the surface is non-planar, curvature features at each point in the data set are evaluated to classify the set as either spherical, cylindrical or conical. An optimisation technique is then used to refine the parameters of the resulting surface type. Locally estimated surface curvatures are used to tentatively give the surface its initial geometric parameters. A non-linear Levenberg-Marquardt technique is then used to refine the parameters of the quadric surface.

Bolle and Sabbah[38] use a similar approach to calculate surface properties. First a least squares error approximation of the surface is calculated. The surface is then smoothed with a linear regression technique. Secondly, differential geometry is applied to determine surface curvature, from which the maximum and minimum curvature and their principle vector directions can be used to obtain second order parameters (such as radii and axis of radii) of quadric surfaces.

A region growing technique developed by Faugeras et al.[39] to both segment range data and fit a quadric surface to the resulting patch. Segmentation is accomplished by first fitting localised quadrics through each point in the data set. Using the parameters from the quadrics, points in a neighbourhood where the surface orientation is rapidly changing are flagged as a surface edge. Finally, points with similar surface orientation inside of the flagged edge points are merged to form larger surface patches.

4.2 Reduction of Spurious Cloud Data

The collection of data points with a laser based digitiser often results in the retrieval of spurious data points. Although the neural network scanning algorithm directs the laser head to scan only the patch required, spurious data points result from the following conditions:

1) Scanning of patches narrower than the 80mm scanning window of the Hymarc scanner will pick up data points on either side of this swath.
2) The 10mm over-scan directed from the automatic scanning algorithm will sometimes acquire data from neighbouring patches.
3) If a non-rectangular shape is being scanned.

The above scanning situations and the spurious data that results is shown in Figure 35.

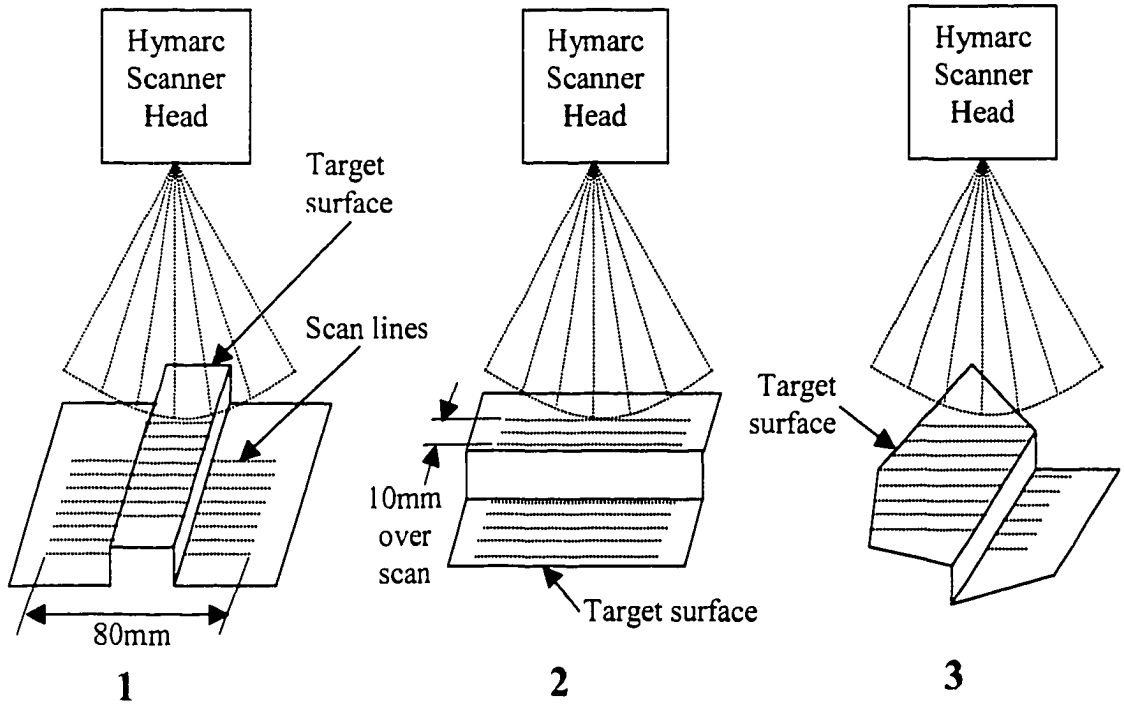

Figure 35: Spurious Data Resulting from Laser Scanning

A filtering algorithm was developed to separate spurious data from the main data set. Based on chordal deviation, the algorithm will flag data points when there is a change in curvature of the laser scanner line. Using a common starting point, the first vector is drawn between the first and second data points, and second vector is drawn between the first and third data points. The angle is measured between these two vectors, if the angle is above a predetermined value, the data point is flagged as a potential corner point. This method works well for finding the edges of planar surfaces. However, when this algorithm is applied to a spherical or free-form surface, a new minimum allowable deviation must be chosen or the algorithm must be abandoned altogether. As well, this algorithm is especially sensitive to noisy data. Should the data points vary widely, due to noise or due to a poor reflection

angle, the algorithm may often flag non-corner points.

To address these problems, the spurious data filtering algorithm developed for this work makes a few improvements on the chordal deviation method. To calculate a new vector a sampling of five data points is used as shown in Figure 36:

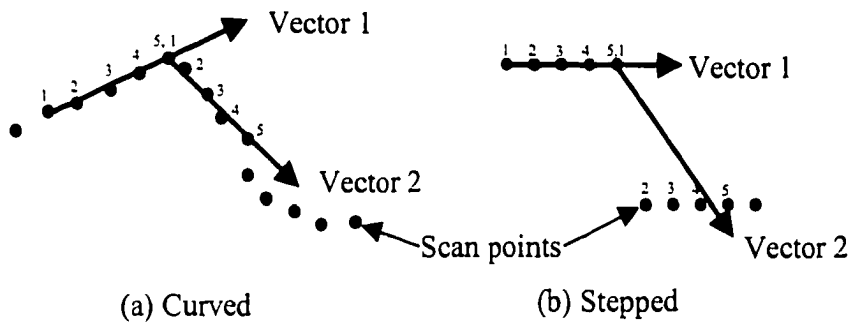

Figure 36: Example of the chordal deviation vector

The inclusion of five data points to determine the vector allows for the reduction in the sensitivity of the algorithm to noise in the data. At the end of each laser line, the vectors are determined by a minimum of two data points. As the point being examined is further away from the end points, three data points, then four and finally five data points are used to determine the first and second vectors.

Implementation of this algorithm has allowed the filtering of spurious data points, which belong to neighbouring patches. Figure 37 shows a typical planar patch, with spurious data points identified and separated to a new patch. A scan of the base of a computer mouse is shown in Figure 38 with the planar portions correctly separated from the filleted corner. The resulting model is now made of three separate surfaces, two planar surfaces and a

cylindrical portion.

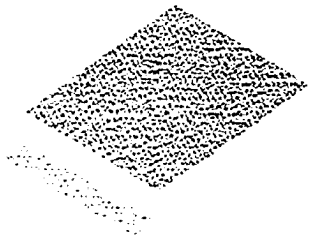

Figure 37: Scanning results – over scan of target surface identified

Figure 38: Scanning results - curved portions correctly segmented

Notice that the algorithm separated the filleted portion from the planar portion in the data. The spurious data filtering algorithm has separated the cylindrical portion of data, a patch not initially detected by the neural network segmentation algorithm.

4.3 Data Thinning and Boundary Identification through Voxel Binning

The large data sets created by the Hymarc laser scanning system often prove to be computationally difficult to process. One method, called voxel binning, can be used to

reduce the size of the cloud data set. Voxel binning can best be described as the process of subdividing the entire volume that a cloud data set occupies into smaller cubic volumes, called voxels. A single data point within each voxel is selected to represent all other data points in the cubic volume. In previous work, voxel binning typically involved calculating the average (x,y,z) value of all data points within a voxel bin and selecting the closest data point to be the representative value for that voxel.

In this work, the voxel binning algorithm has been expanded in flexibility to let the user to select between three different modes of voxel binning. The first mode is identical to previous work, in that the average data point for that particular voxel is calculated and the closest measured data point to the average is chosen to represent the voxel. The second mode selects the measured data point that is closest to the centre of the voxel. This allows users who want a more evenly spaced voxel bin data set than would otherwise result from selecting the average data point. The third mode of voxel binning allows the user to find the data point, which is closest to the centre with the exception of one of the axis (i.e. x, y or z) The third axis is allowed to float, where a point in that axis is selected closest to the calculated average data point for that voxel. This floating voxel bin point allows the users to create a more regularised set of data points without the representative inaccuracies of using the data point closest to the centre. The three modes of voxel binning are shown in Figure 39.

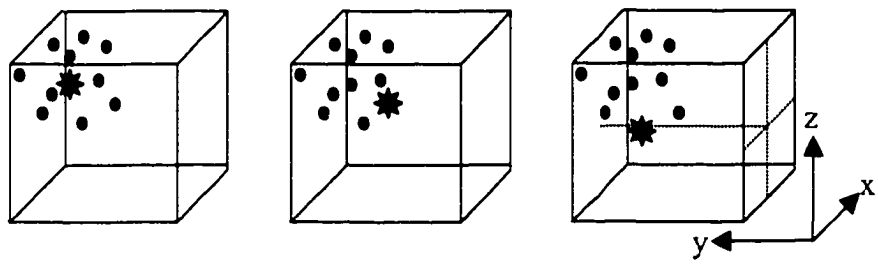

Figure 39: Three modes of voxel binning
(selected data point indicated by a star)

At this point in the algorithm, the user can select to output the data as a set of points for other types of surface fitting algorithms, such as polyhedral triangulation. These representative voxel bin points, referred to as fit points, are used as the points to which surface primitives or free-form surfaces are fitted. The second use of the voxel binning process is to decrease the time required to select the boundary points of the patch of data being fitted. As a consequence of the voxel binning process, all of the collected data points have been labelled with reference to a specific voxel bin.

A second consequence of the voxel binning algorithm is its use for identifying the boundary. The boundary labelling algorithm has two steps; first potential boundary points are labelled, second, the labelled points are joined to form an open ended polyline. A point is labelled a boundary point if it obeys the following criteria: The point being examined is not surrounded by other points on more than 3 of its 4 directions (east, west, north and south) This criterion is satisfied by examining all data points in the collected data. A reduction in the computation time is realised if the number of points being examined for each point tested

is limited to the surrounding 26 bins. (ie. a cubic volume of 3x3x3 bins, not including the centre bin). Depending on the size of the collected data set and the voxel bin size, this is a significant reduction in the time required to formulate the boundary. For example, the time required to determine the boundary of a turbine blade that occupied over 9,000 voxel bins, was reduced from three days to less than 20 minutes.

The next stage of the boundary classification process is to link the identified data points to form a continuous polyline while excluding spuriously flagged boundary points. Starting at the first flagged data point, the algorithm proceeds to link the next closest flagged data point. This process is repeated, until the next closest flagged data point is further than 10X the average distance between previous neighbouring connected data points. A flowchart of the algorithm is presented in Figure 40.

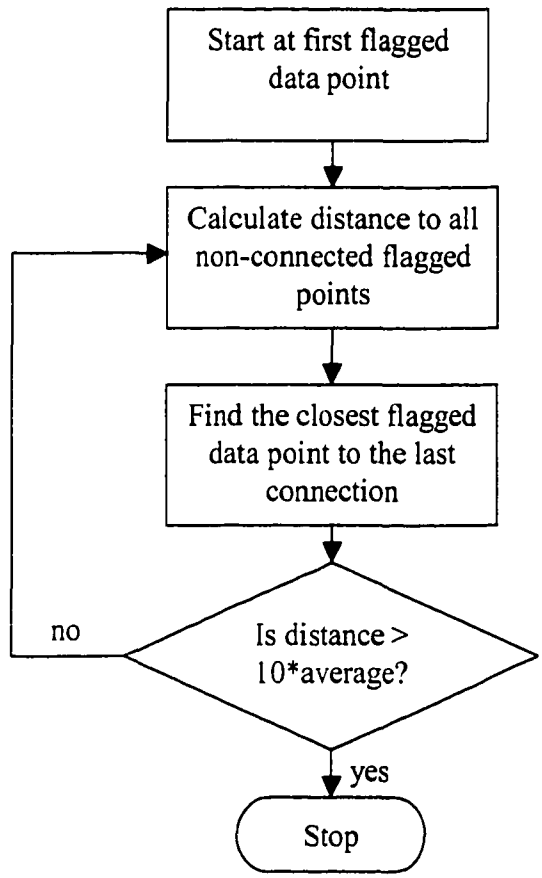

Figure 40: Boundary polyline linking algorithm

The value of (10 x average distance between data points) was found to be a reasonable compromise between arresting the algorithm before the boundary was sufficiently identified and allowing the algorithm to connect spurious data points. The connection of spurious boundary points can easily be corrected by dividing the polyline and erasing the spurious portion. However, if an insufficient number of boundary points were linked, the remaining data points may have to be manually linked to fully describe the boundary. This is a tedious endeavour. Figure 41 shows a patch with the boundary points linked together

with a single polyline. Notice that the excess boundary polyline can be easily cut from the main portion of the boundary line at the point noted in Figure 41.

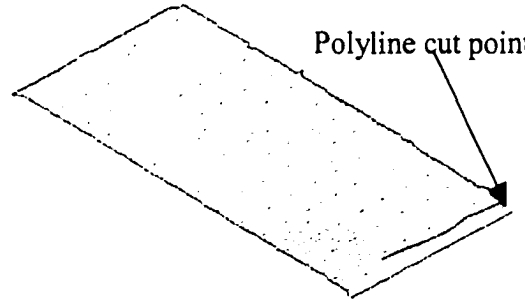

Figure 41: Results from the boundary identification algorithm

The final stage of the boundary identification algorithm is to sub-divide the boundary line into four individual segments. The sub-division is accomplished by applying a similar chordal deviation algorithm as that used in the spurious data filtering detailed earlier. Starting at the first point, a chord is fitted using boundary points 1 through 5. A second chord is fitted to boundary points 5 through to 10. The angle between these two chords is calculated, and if larger then a pre-determined value, point 5 is flagged as a potential corner boundary point. This process is repeated until all boundary pixels have been tested for being potential corner points.

The model of a rectangular patch is used in the determination of corner points. In the model, the corners of the rectangular patch are the four points that are furthest from the

centroid yet also being 90 degrees apart from one another. To determine which of the flagged points is a corner point, the program calculates the distance between the flagged points and the centroid of the patch. The flagged point with the furthest distance is chosen as the first corner point. The next corner point to be chosen is the next furthest flagged point that is also 90 degrees rotated from the first point. This process is repeated for the third and fourth corner points using the second and third corner points respectively to determine the rotation angle. Figure 42 shows a typical patch of which the potential corner points have been flagged (smaller circles) and the four corner points identified (four larger circles).

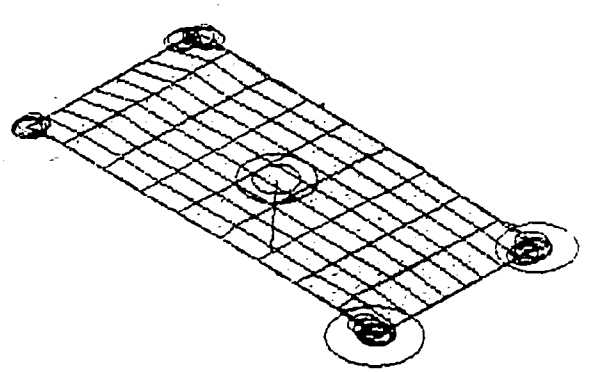

Figure 42: Plane surface with boundary corners identified

4.3 Fitting Planes to 3-D Scattered Data

Planar surfaces are often found in many manufactured items. This is not surprising as planner surfaces are often the easiest to manufacture. Given that the equation for a plane:

$$ax + by + cz + d = 0 \tag{9}$$

where the normal is given by vector (a, b, c) and d is given by:

$$d = a\bar{x} + b\bar{y} + c\bar{z} \tag{10}$$

The orientation and location of the plane is represented by the unit vector normal of the plane, η_p at point p. The unit normal is estimated by the average cross product between the first vector between the first point in the plane and the centroid of all the points, to the second vector point in the plane, made up between the second point and the centroid. This is described by Equation 11 and Equation 12 below:

$$\hat{\eta}_p = unit\_vector\_of\left(\sum_{i=0}^{n-1}(p_i - p) \times (p_{i+1} - p)\right) \tag{11}$$

$$p = \frac{\sum_{i=0}^{n} p_i(x, y, z)}{n} \tag{12}$$

Basically, two types of error can exist from the estimate of the plane. First, the direction of the unit vector normal. Second, the location of the base of the unit vector normal. Thus, in subsequent iterations, two different variables can be adjusted.

To correct the location of the centroid, first the error distance between the points used to fit with and the estimated plane is calculated. Taking the direction of the normal out of the plane to be positive, errors in the direction of the normal are considered positive, whereas errors in the opposite direction are considered negative. Using this convention, the cumulative error is calculated. If the cumulative error is zero, the centroid of the plane is correctly placed. If the cumulative error is positive, then the centroid is moved in the negative-normal direction and if the cumulative error is negative, the centroid is moved in the positive-normal direction.

The direction of the normal is corrected by first calculating the angle made between the estimated plane and the line between the fit-point and the centroid. Arbitrarily selecting the first fit-point as a basis, a plane is constructed from the line joining the first point and the centroid and the line consisting of the normal. All the error angles are projected on to this plane, and the cumulative error angles is calculated, using the right hand rule, between the normal and first fit-point line as the positive direction. Next, a second plane, at right angles to the first, but still parallel to the normal, is constructed, and new cumulative error angles are calculated. Using these two cumulative error angles, the direction of the normal is then adjusted.

A scan of a planar surface, approximately 50mm X 50mm was digitised to test the algorithm. A total of 3 iterations are required to best fit the plane to the 266 fit points. The mean error of the fitted plane to the fit points is 0.00352mm, with a maximum error of 0.148mm. A picture of a bounded mesh surface generated to fit the plane can be seen in Figure 43. The line drawn from the centre of the mesh represents the normal vector of the planar patch.

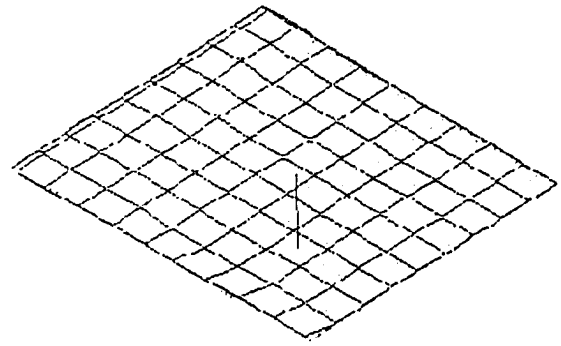

Figure 43: Example of a Plane Fitted to Measured Data

The robustness of the fitting algorithm is tested by subjecting the method to synthetically generated data with a known Gaussian noise. The effect of increasing Gaussian noise, from a zero mean noise variance of 0.05 to 1.0, on the ability of the algorithm to estimate the parameters of a plane from a set of 500 synthetic data points is illustrated in Table 6. For comparison, the effects of noise on the probability based algorithm used by Bradley[14] are also shown versus the iterative method used in this work. The percent error from the true parameters of the plane: a = 0.707, b = 0.00, c = 0.707, and d = 1.414 are

shown in Table 6. In comparing the results with those achieved by Bradley[14], it is interesting to note that the iterative solution is less effected by increasing Gaussian noise.

Table 6: Effects of Gaussian Noise on the Estimation of Planar Parameters

% Error		**a**	**b**	**c**	**d**
$\sigma^2 = 0.05$	Iterative	0.06	-0.57	0.09	0.32
	Probability	0.7	0.3	0.7	0.1
$\sigma^2 = 0.1$	Iterative	0.11	0.67	0.14	0.00
	Probability	1.5	0.6	1.5	0.2
$\sigma^2 = 0.5$	Iterative	0.05	-0.66	0.02	2.78
	Probability	5	1.7	5.5	0.1
$\sigma^2 = 1.0$	Iterative	0.00	-0.75	0.02	1.22
	Probability	13	4.8	16	2

Although the probability based results of Bradley should provide the optimal results, assumptions made about the completeness of the data set may lead to some error. Information about statistical parameter estimation can be found in works by Sorenson[40] and Bretthorst[41].

The effect of spurious data on the algorithm was tested by replacing the synthetic data by points, which were ten standard deviations from its proper position. Table 7 shows the results of varying amounts of outlier points and their effect on the accuracy of parameter estimation. Again, the algorithm performed well, with errors remaining small.

Table 7: Effect of outliers on planar surface fitting

% Error, $\sigma^2 = 0.1$	**a**	**b**	**c**	**d**
1 outliers out of 500 points	0.18	-0.69	0.16	0.30
20 outliers out of 500 points	0.30	0.59	0.33	0.43
125 outliers out of 500 points	0.48	-0.31	0.46	1.78

4.3 Curvature Determination

Curvature is a method often used to classify range data. The amount of curvature, and the directions of the minimum and maximum curvature provide a criterion for classifying the surface as a plane, sphere, cylinder, or a free-form surface. The local curvature of a surface is determine by Equation 13 below:

$$k(p,q) = \frac{\left\|\bar{n}_p - \bar{n}_q\right\|}{\left\|p - q\right\|} \tag{13}$$

Curvature is the measure of the rate of change in the surface normal from point p to point q. For the purpose of this work, the curvatures of interest are the minimum curvature and the maximum curvature. The direction of these two curvatures is assumed to be 90° to each other. Therefore, each fit point, the local maximum and minimum curvature is determined, as well, the local surface normal.

The local normal is calculated by first finding the closest neighbouring fit point. The next closest fit point whose vector, formed to the central point, is greater than fifteen degrees

but less than thirty degrees from the vector formed from the closest fit point is selected as the second fit point. This process is repeated until a total of eight fit points are found, evenly distributed around the evaluation point for which the normal is to be calculated. Eight vectors are calculated from the eight closest fit points and the central point as shown in Equation 14. The local normal for the fit point being examined is determined by Equation 15, which is the summation of the cross products of the first and second vectors, second and third vectors, etc.

$$v_i = p_{closest\_i} - p_{evaluation} \qquad for \quad i = 1 \quad to \quad 8 \qquad (14)$$

$$\hat{n}_p = unit\_vector\_of \quad \sum_{i=1}^{7} (v_i \times v_{i+1}) + (v_8 \times v_1) \qquad (15)$$

After the normals for all fit points have been determined, the maximum and minimum curvature at a point is calculated. The same eight closest points used to calculate the normal, are the same eight normals used to determine local maximum and minimum curvature. The curvature in each of the eight directions are calculated using Equation 13 between the evaluation point and its eight closest neighbours.

4.4 Fitting Spheres to 3-D Scattered Data

Spherical surfaces are another common surface type found on manufacture parts. Given the equation of the sphere as:

$$R^2 = (x - x_o)^2 + (y - y_o)^2 + (z - z_o)^2 \tag{16}$$

where the centre of the sphere is given by (x_o, y_o, z_o) and the radius by R.

A good estimate of the radius of a sphere can be arrived at by calculating the average of the inverse curvature (over all fit-points). Theoretically, there are no maximum or minimum curvatures on a sphere, both curvatures for each point can be used to calculate the mean curvature. Projecting each surface point towards the centre along the local normal at that point will produce a "mass" of points close to the centre, as shown in Figure 44. The centroid of the swarm of points is then used to estimate the centre of the sphere.

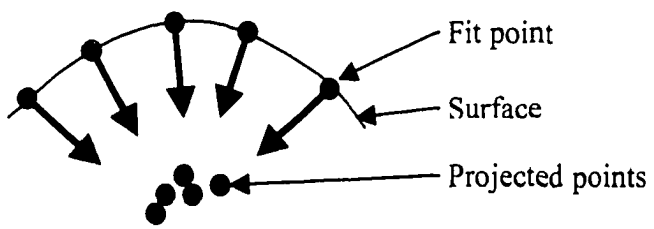

Figure 44: Projected points to determine centre

4.4.1 Improving the Fit of the Sphere

After the initial estimate has been made for fitting the sphere to the scan data, two variables can be changed. First, the position of the centre of the sphere can be re-positioned, and secondly, the radius of the sphere can be adjusted to better fit the data.

An iterative approach is used to minimise the error between the fit points and the sphere. The error between the fit point and the estimated sphere is calculated as illustrated in Figure 45.

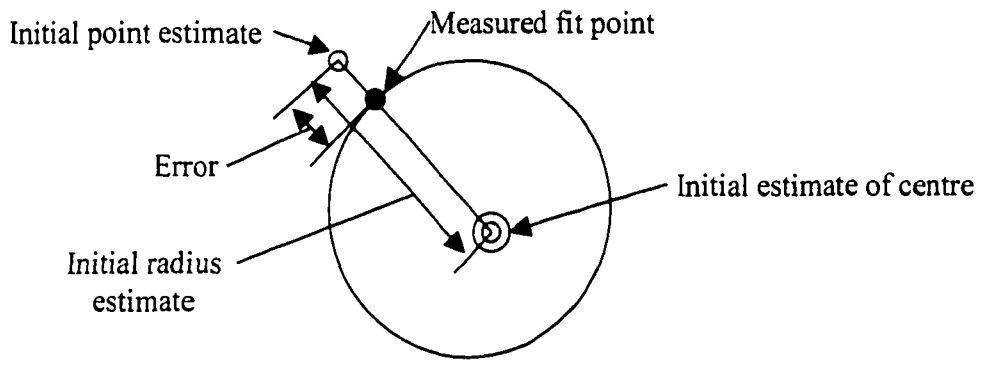

Figure 45: Initial sphere estimate error measurement

A rule-based algorithm is used to decide how the parameters of the sphere should be changed. Attention is paid to the sign of the error between the fitted surface and each fit point, with a cumulative positive and a cumulative negative error being calculated. If the total positive error is greater than twice the total negative error, or the total negative error is greater than twice the total positive error, the sphere centre is moved by amount and direction dictated by the error vector. The error vector is the summation of the vectors formed between the fit points and the perpendicular distance to the fitted surface. If the positive or negative error is less than two times the opposite error, the radius of the sphere is changed by the mean error. Refinement of the sphere to the fit points was continued until either 10,000 iterations have been completed or the change in the error between two consecutive

iterations is less than 0.01%.

An increase in the speed of which the final sphere dimensions was determined can be achieved through the application of fuzzy logic. Fuzzy logic deals with propositions that can be true to a certain degree. In the case of sphere fitting, two choices can be made to better fit the sphere to the measured data either the position of the centre, or the length of the radius. Both values may be changed to a certain degree to achieve the final result. In reference to the rule based fitting algorithm above, the closer in value of the positive and negative error, the greater the probability that the radius needs to be changed. Thus, the potential that the radius is required to be changed is calculated as shown in Equation 17.

$$Potential = \frac{|positive\_error| - |negative\_error|}{positive\_error + |negative\_error|} \qquad (17)$$

Note that Equation 17 will result in a value between zero and one. A random number function is used to generate a value between zero and one. Should the value of the random number fall below the potential, the position of the sphere centre is changed. Otherwise, the radius of the sphere is changed as outlined for the rule based fitting algorithm.

A spherical surface, 28.4mm in diameter was digitised to test the algorithm. A total of 117 iterations were required to best fit the sphere to the 373 fit points, with an initial guess of 14mm for the radius. The mean error of the fitted sphere to the fit points is 0.0382mm, with a maximum error of 0.177mm. A picture of the fitted sphere is shown in **Figure 46**. It

was found that the fuzzy fitting algorithm was able to reduce the number of iterations by about 10% of the rule-based algorithm.

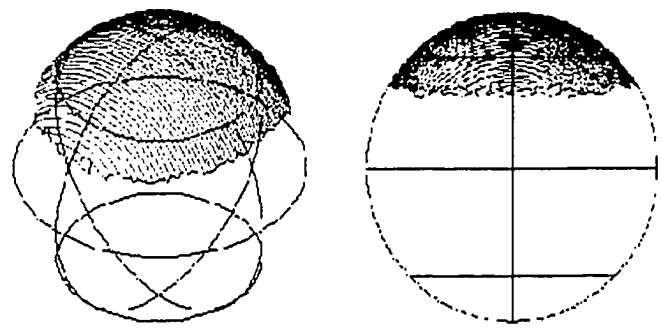

Figure 46: Sphere fitted to measured data

Synthetically generated data with a known Gaussian noise is used to test the robustness of the fitting algorithm. Again the zero mean noise variance is increased from 0.05 to 1.0, on a set of 500 synthetic data points. The results of the test, expressed in percent error can be seen in Table 8. For comparison, the effects of noise on the probability based algorithm used by Bradley[14] are also shown. The true parameters of the sphere, centred at X_o=0.5, Y_o=0.5, X_o=1.0 and a radius of R = 10.

Table 8: Effects of Gaussian noise on estimating spherical parameters

% Error		R	X_o	Y_o	Z_o
$\sigma^2 = 0.05$	Iterative	0.23	2.03	0.86	3.64
	Probability	N/A			
$\sigma^2 = 0.1$	Iterative	0.25	1.23	2.26	5.59
	Probability	0.1	0	0	0
$\sigma^2 = 0.5$	Iterative	0.11	8.30	3.42	7.21
	Probability	0.2	0.2	0.1	0
$\sigma^2 = 1.0$	Iterative	0.69	8.82	4.12	4.47
	Probability	N/A			

Differences between this work with those obtained by Bradley, may be a result on the converage of the synthetic data. To better simulate actual digitisation results, the synthetic data was generated to cover only half the sphere. A full sphere of generated synthetic data would have increased the accuracy of the parameter estimation.

Table 9 shows the results of varying amounts of outlier points and their effect on the accuracy of parameter estimation.

Table 9: Effect of outliers on spherical surface fitting

% Error, $\sigma^2 = 0.1$	R	X_o	Y_o	Z_o
1 outliers out of 500 points	0.07	0.01	2.83	5.76
20 outliers out of 500 points	0.16	1.95	4.42	1.11
125 outliers out of 500 points	0.22	63.90	37.63	4.13

As can be seen in Table 9, the algorithm produces acceptable results when up to 1% of the fit points are outliers. However, as the number of outliers increases to 20% of the total fit points, the results are definitely unacceptable. Again, as the synthetic data was only generated to cover half a sphere, a large number of outliers will skew fitting results.

4.5 Fitting Cylinders to 3-D Scattered Data

Cylindrical surfaces are often found on manufactured surfaces, such as fillets, and internal corners. A cylinder can be described by an axis and a radius. On a cylinder, the maximum curvature corresponds to the radius of the cylinder whereas the minimum curvature, corresponding to the cylinder's axis is zero. The main axis of the cylinder is specified by a start point, P_s and an end, P_e point. The radius of the cylinder is given by R. Assuming that the digitised data lies on the cylinder's surface, the cylinder's axis is first estimated by projecting the surface points by a distance of their inverse average curvature towards the centre of the cylinder along the normal calculated for that local point. This results in a cloud of points that roughly lie on the cylinder's axis. A least squares line fitting algorithm is used to fit the main axis to the cloud points.

4.5.1 Best Fit for the Cylinder

Once the initial estimate for the cylinder is made, a number of variables can be adjusted to best fit the cylinder to the scan data. The variables that are associated with positioning of the cylinder are the orientation and location of the cylinder axis. The radius

R, is associated with the cylindrical surface itself.

A rule base method was initially used to determine which of these variables to alter. First the cylinder axis is divided in its middle. The error is calculated between the fit-points and the estimated surface for the first half and the second half. The two errors are compared. If the front and back errors are similar, the radius is corrected, if they are different, the positions of the axis is adjusted.

Again, a fuzzy logic based algorithm was used to replace the hard rule algorithm to decrease the number of iterations. Two choices can be made to better fit the cylinder to the measured data, either the position of the main axis, or the length of the radius. Both values may be changed to a certain degree to achieve the final result. Using a similar strategy as the rule base method mentioned above, the closer in value of the front and back error, the greater the probability that the radius needs to be changed. The potential that the radius is required to be changed is calculated as shown in .

$$Potential = \frac{\left| front\_error - \left| back\_error \right| \right|}{front\_error + \left| back\_error \right|} \tag{18}$$

Fitting a cylinder to actual scan data tests the fitting algorithm. A cylindrical surface, with a radius 24.9mm and a length of 100mm was used in the test. A total of 167 iterations were required to best fit the sphere to the 286 fit. The mean error of the fitted cylinder to the fit points is 0.0818mm, with a maximum error of 0.103mm. A picture of the fitted sphere is

shown in Figure 47. Again, it was found that the fuzzy fitting algorithm was able to reduce the number of iterations by about 10% as compared to the application of the rule-based algorithm.

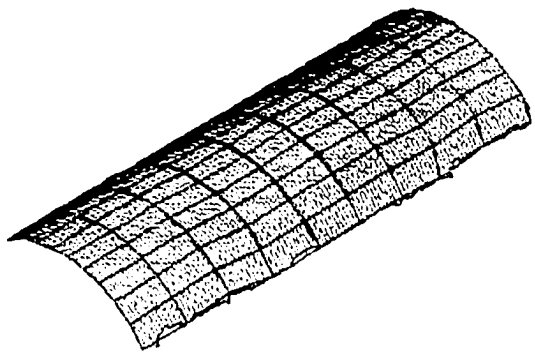

Figure 47: Cylinder fitted to measured data - mesh shown

To test the robustness, synthetic data is generated where the zero mean noise variance is increased from 0.05 to 1.0, on a set of 500 synthetic data points. The results of the test, expressed in percent error can be seen in Table 8 with comparison to the fitting algorithm used by Bradley[14]. The synthetic data is generated for a cylinder on the X-Y plane, with an axis through $(x_o, y_o) = (20.5, 5.0)$, rotated at an angle $\phi = 0.245$ radians around the Z-axis as measured from the X-axis.

Table 10: Effects of Gaussian noise on estimating cylinder parameters

% Error		R	X_o	Y_o	phi
$\sigma^2 = 0.05$	Iterative	0.05	0.48	0.48	0.26
	Probability	0.6	3.3	7.4	2.0
$\sigma^2 = 0.1$	Iterative	0.10	0.79	0.81	0.59
	Probability	1.3	6.8	13.6	26.0
$\sigma^2 = 0.5$	Iterative	0.11	1.98	2.08	0.93
	Probability	N/A			
$\sigma^2 = 1.0$	Iterative	0.79	5.90	6.94	3.22
	Probability	N/A			

As can be seen in Table 10, the algorithm performed quite well, with little noticeable effect on the parameters until $\sigma = 1.0$. To test the effects of outliers on the fitting algorithm synthetic data was generated with outlier points. The results of this test can be seen in Table 11.

Table 11: Effect of outliers on cylindrical surface fitting

% Error, $\sigma^2 = 0.1$	R	X_o	Y_o	phi
1 outliers out of 500 points	0.18	0.44	0.45	0.39
20 outliers out of 500 points	0.92	1.85	1.96	0.78
125 outliers out of 500 points	0.00	0.80	0.82	0.50

Again, little effect is seen from the increasing percentage of outliers, proving the robustness of the algorithm.

4.6 Fitting Free-Form Surfaces to Scattered Data

A free-form surface will be fitted to the measured surface points if neither of the previous primitives fit the data. A rule-based algorithm is used to decide if any of the primitives properly fit the measured data. A proper fit is determined when the error of one primitive surface is less than eight times the calculated error of any other primitive surface. If none of the primitives pass this test, the surface is then fitted with a free form mesh.

The first step to constructing a surface mesh is to define the four boundary lines as detailed in Section 4.3. Secondly, these four boundary polylines are further segmented to match the MxN mesh density as indicate by the user. A planner grid is then constructed using the segmented boundary lines as shown in Figure 48. The intersections between the mesh grid is then projected towards the patch surface, which as of yet is defined only by the fit points. The three closest fit points to the planer grid intersection are used to form a polygon, to which a point is calculated perpendicular to the planer grid line intersection. Using this first estimate of the mesh surface, individual mesh points are then adjusted to reduce the error between the mesh and the fit-points.

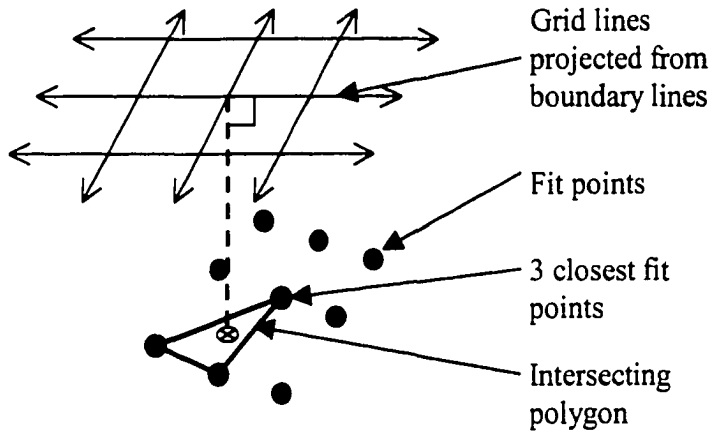

Figure 48: Determination of mesh point location

A sample of mesh fitting to a free form surface is shown in Figure 49 and Figure 50. The object, a water turbine blade roughly 30cm by 30cm square, was digitised using the Hyscan laser scanner. A total of five passes for each side were required to ensure complete converge of the turbine blade, requiring one hour on the scanner. Applying the meshing algorithm, a 15x15 mesh surface was generated with a mean error of 0.143mm error, with a maximum error of 1.45mm. Although the errors are large, better fitting can be achieved by choosing a finer mesh size.

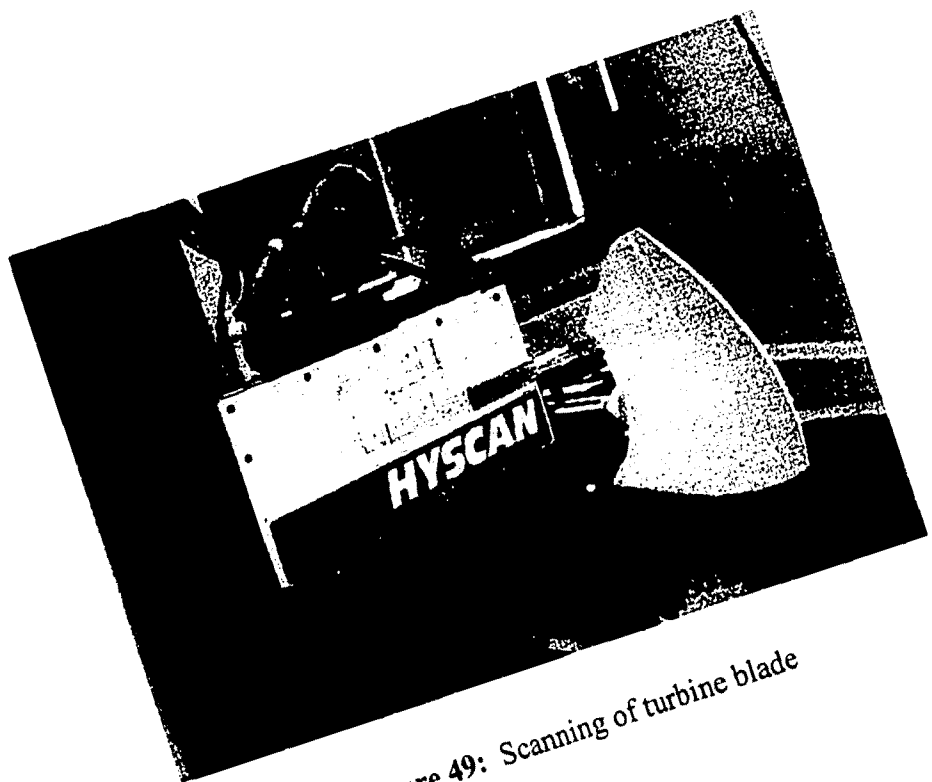

Figure 49: Scanning of turbine blade

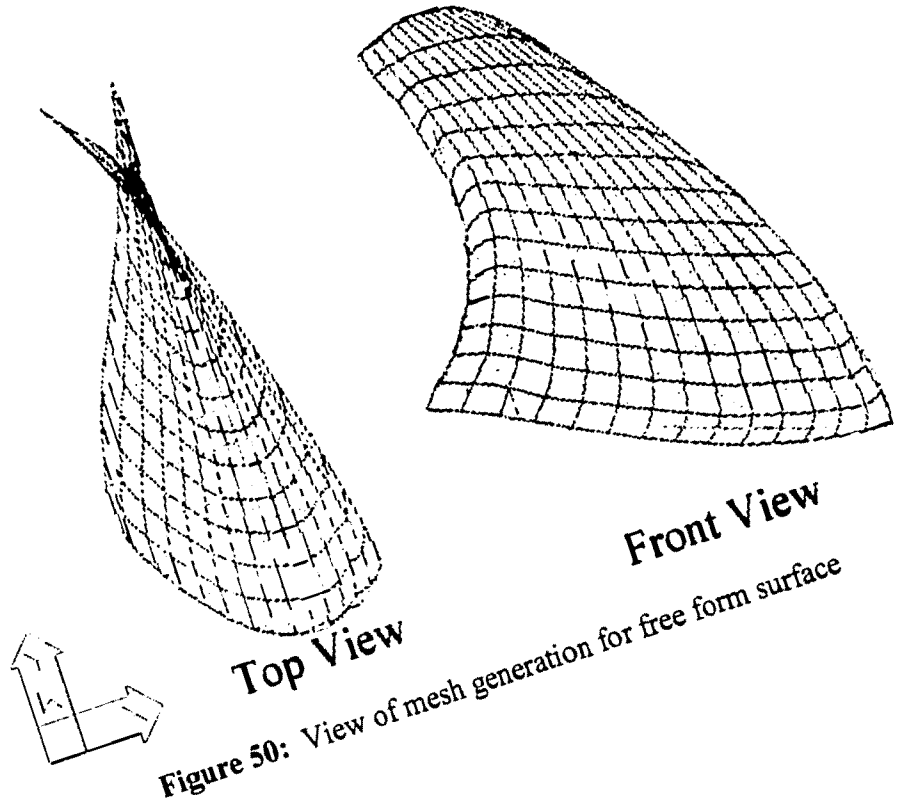

Figure 50: View of mesh generation for free form surface

Chapter 5 - Reconstruction of Model Topology

Topology reconstruction is the process of establishing the relationships between the different surface patches of a single object. This adjacency information is referred to as the topology of the B-rep model, whereas the actual descriptions of the surfaces, boundaries and the vertices of the boundaries are referred to as the geometry of the B-rep model. The details of the geometry of each surface, and how each surface patch fits together with adjacent surface patches must be determined in order to create a B-rep solid model. The topology can be interpreted as the "glue" holding all the geometric information together. In summary, a B-rep model contains at least two types of data, geometry and topology.

Research into topology has mainly focused on the generation of this information during the B-rep model creation stage of a CAD modeller. The creation of topology information from reversed engineered data has not been developed. Therefore, a review of present methods of organising topology information for a CAD modeller is presented.

5.1 Review of Topology Representation

In the book, 'Engineering Databases, Connecting Islands of Automation Through Databases' by Encarnacao and Lockemann[42], three main types of data models are presented, the hierarchical, the network and the relational data models. The hierarchical model is similar in appearance to a family tree, with each parent having several children. The network data model is a less restrictive model than the hierarchical database. The network data model consists of an arbitrary number of records and an arbitrary number of interrelationships between them. However, this data model still maintains a hierarchical data format. Finally, the third data model is the relational data model. This model consists of a list of relations and a list of attributes. No data structure is imposed on the data and the user is free to create relations between the data sets as needed. A graphical representation of each data model structure is shown in Figure 51.

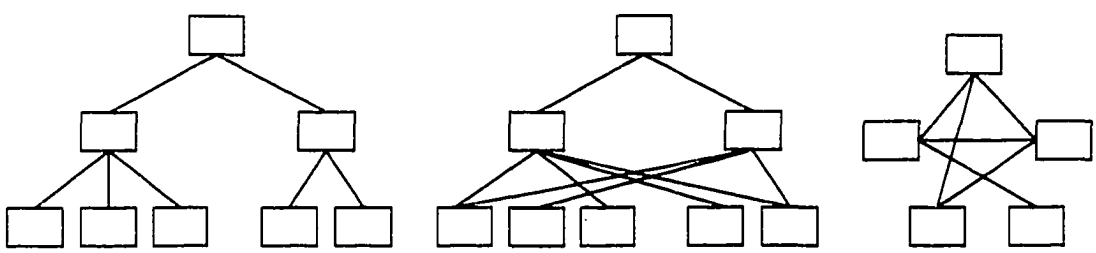

Figure 51: Three different model structures for storing topological data

Weiler[43] has developed a more efficient data structure for storing topological relationships. As B-rep models are composed of edges, vertices and faces upon which can be used as the "framework around which the rest of the solid modelling implementation can be built." Using only three primitive topological elements, nine adjacency relationships are possible. Weiler presents the winged-edge data structure, which is composed of the given edge, and it's adjacency with other edges, vertices and faces. The graphical appearance of the adjacent edges in relation to the reference edge gives this data structure the name "winged-edge" as can be seen in Figure 52.

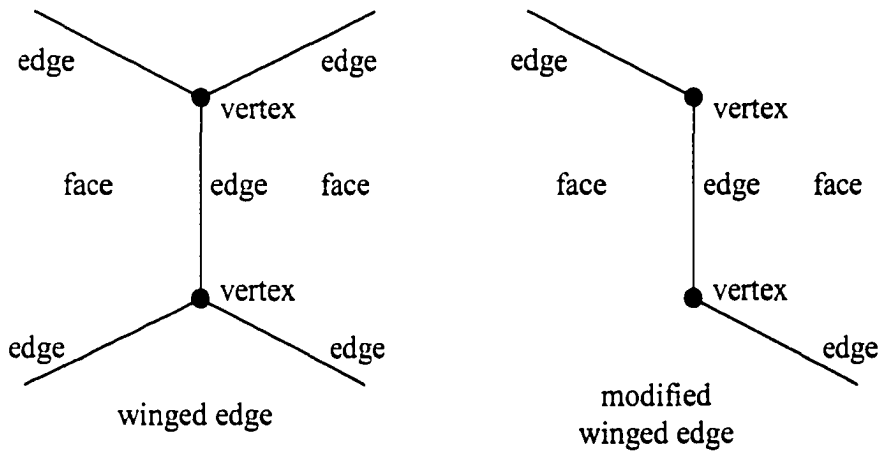

Figure 52: Winged edge data structures

The main advantage with the winged edge data structure is that the data length is the same for every edge. This makes the winged-edge structure efficient when processing topological entities in a computer. Toriya and Chiyokura[44] use a modified winged-edge data structure

in their DESIGNBASE software. Instead of having all four of the winged-edges, DESIGNBASE only stores two of the edges; the bottom-right edge and the top-left edge, as can be seen in Figure 52.

5.2 Topology Architecture

The syntax of the topology database must first be developed. As with any database design, it is a compromise between data storage space and data retrieval speed. A database with a high retrieval speed will generally have many redundant entries, thus using much data storage space. A database with a low retrieval speed will have very few, or no redundant entries, causing the program to spend more time searching for entries, but requires less space for data storage. Other factors to consider are the order in which the data is stored, the method by which the data is sorted and by which means the data can be edited. The database models discussed previously are analysed for all of these factors in Table 12 below:

Table 12: Benefits of different topology database formats

Type	Retrieval Speed	Redundancies (Memory)	Easily Edited?	Topology Maintained?
Hierarchical	fast	no	yes	no
Networked	fast	no	yes	yes
Relational	fast	no	yes	no
Winged Edge	slow	yes	no	yes
Modified Wing Edge	slow	yes	no	yes

Since the reverse engineering program will not need to access the database extensively (compared with a CAD system), access speed to the database is not of prime

importance. Also, as the reverse engineering package will only handle one scanned part at a time, the number of entities required to be stored will be minimal. Therefore, storage space is not a concern.

In examining the three classical data structures, the networked data model most closely resembles the topology relationships of a B-rep model, yet also provides easy access to relationships between different surface patches. To represent the geometrical data, a sequential data format is used. Although this may not be the most efficient method of data storage, there is no need to reference this data multiple times and it does provide for the most simplistic data structure.

The link between two entities in the network data model will be recorded as an element in a two dimensional array. For example, the links between Face 1 and 2, edges 1, 2, 3 and 5 is stored in a two-dimensional array as shown in Figure 53.

		Edges				
		1	2	3	4	5
Faces	1	1	1	1	0	0
	2	0	1	0	1	1

Figure 53: Face/Loop data storage format

A "1" represents the presence of a link between two entities while a "0" would represent no-links.

5.3 Establishing Patch Adjacency for the Topology Database

The adjacency of one patch with another is achieved through the examination of the edges for each patch. Edge information for each patch is compared with edge information of other patches to determine the existence of neighbouring patches. The algorithm assumes that neighbouring patches should have common edges. Therefore, the volume around each individual patch edge is inspected to determine if the volume is shared with another path edge.

The algorithm divides the entire workspace, which all surfaces occupy, into individual voxel bins. Voxel bins where edges occupy are identified. If it is found, as the algorithm continues with other edges, that another edge boundary occupies a previously labelled voxel bin, the edges are marked as well as the number of voxel bins they occupy. For example, in Figure 54, two surfaces are shown with the space sub-divided into voxel bins. One edge from the top surface and one edge from the front surface are shown sharing common voxel bins.

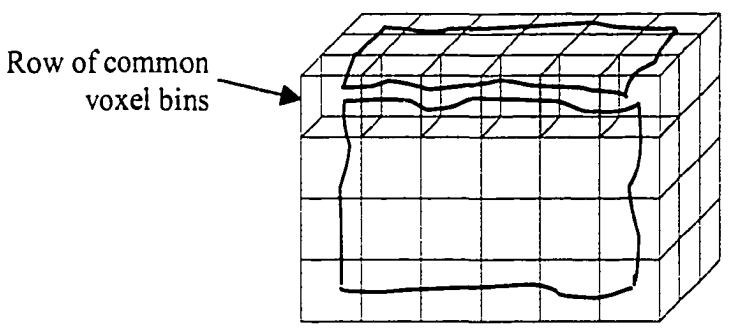

Figure 54: Voxel bin adjacency – common edge shown

The algorithm begins by subdividing the workspace into voxel bins, starting at the furthest data point of any surface. The edges of the first surface are examined, and the voxel bins that the edges traverse through are marked. The next surface is then examined, and the bins that its edges traverse through are marked belonging to that surface. A record is made if edges from one surface traverse through the same bins as edges from another surface. A flowchart of the voxel bin adjacency algorithm is shown in Figure 55.

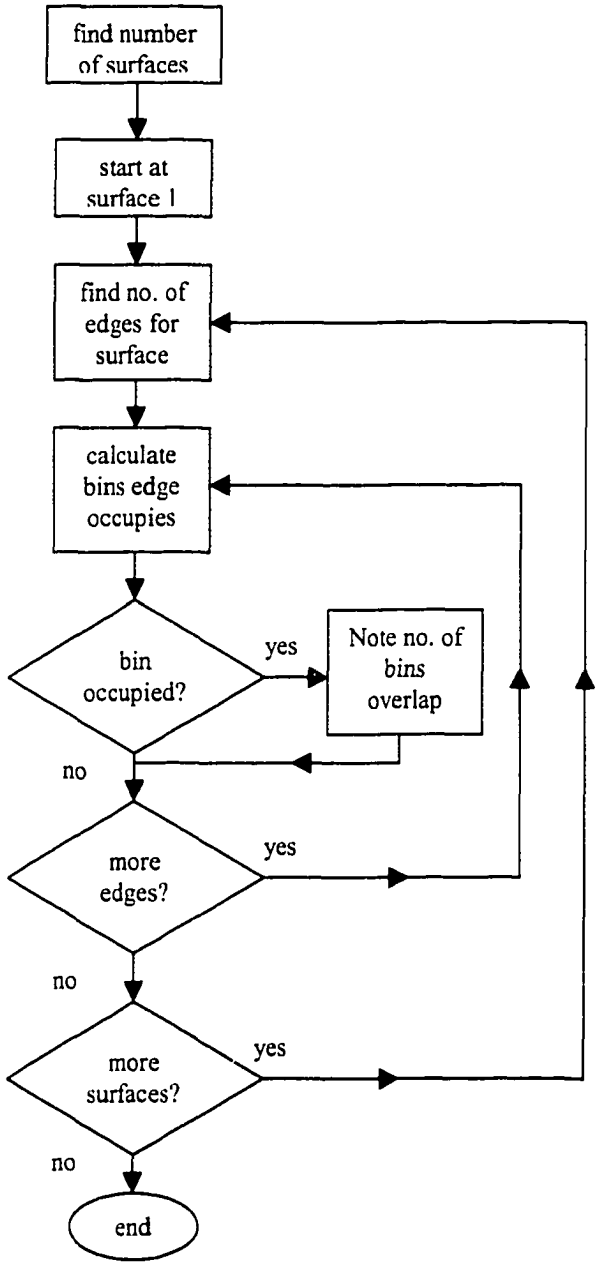

Figure 55: Algorithm to determine voxel bin adjacency

Edges that share a maximum percentage of common voxel bins (as determined during testing) are considered sharing an edge. This information is recorded in the face/loop

database for testing of possible surface features.

5.4 Testing the Topology Generation Algorithm

The effectiveness of the voxel bin adjacency algorithm in finding and organising topology information is tested with surfaces comprising of several patches. Two types of surfaces are tested, a surface where there is no ambiguity about which patches are neighbouring, and a surface where only parts of their edges are neighbouring.

The filleted corner surface first shown in Figure 38 is processed through the surface fitting algorithm, as explained in Chapter 4, with the results as shown in Figure 56. The surface consists of a planar top surface neighbouring a cylindrical corner surface (fillet) and a planar front surface. This surface was chosen to abate any chance of ambiguity about which surfaces are neighbouring.

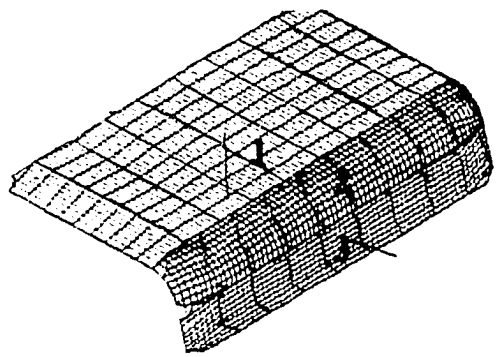

Figure 56: Filleted corner with surfaces fitted.

Selection of the voxel bin size has a direct bearing on the number of positive matches of surfaces sharing common edges. The objects shown in Figure 56 and Figure 57 were tested for different sizes of voxel bins. The length of the bin is non-dimensionalise by calculating the ratio of the longest orthogonal side of the object (along either the X, Y or Z axis) to the bin size. The results presented in Table 13 point out that a small bin size will result in the algorithm missing some of the common edges on the surfaces. However, if too coarse of a bin size is used, many of the edges may be missed. An object length/bin ratio of 15 was used for this work.

Table 13: Effect of bin size on finding common edges

Object length/Bin ratio	**All common edges identified?**
25	No
20	No
15	Yes
10	Yes
5	No

Results of the test confirmed that the algorithm could correctly identify neighbouring patches and the common edges that they share. Referring to Figure 56, the front side of the top patch (1) was identified as sharing the same voxel bins as the top edge of the fillet patch (2). Also the bottom edge of the fillet (2) was correctly identified as sharing the same voxel bins as the top edge of the front surface (3). Computation time for the algorithm was less than three seconds, including the writing of face/loop data to hard disk.

To test the algorithm further, the three-step object was digitised using the automated scanning algorithm presented in Chapter 3, Figure 30. The surface fitting algorithm of

Chapter 4 was applied to the seven patches of the object, with the results of the fitting shown in Figure 57. This object was chosen to test the algorithm's ability to cope with patches that share edges, but not completely.

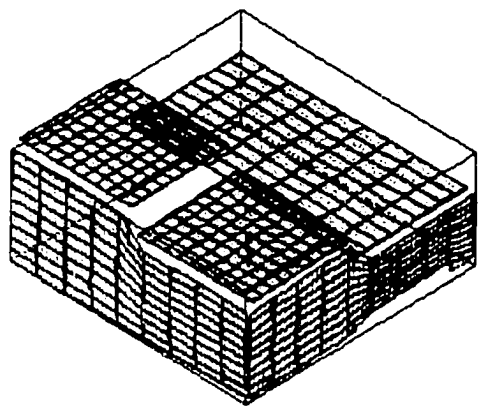

Figure 57: Three-step object with surfaces fitted with planes
(rear surfaces are not shown for clarity)

Applying the voxel bin adjacency algorithm, 11 shared edge pairs were correctly identified. As can be seen in Figure 57, a number of the top patches do not share 100% of their edges with the corresponding side patches. The rule algorithm is robust enough to account for this discrepancy. As long as one of the share edge pair has 100% of its voxel bins in common with another edge, it is considered a neighbouring pair. Computation time of the algorithm was less than ten seconds.

Chapter 6 - Feature Extraction

Engineers, designers and machinist do not naturally conceptualise in basic geometric entities, such as lines, points, curves or B-spline surfaces. Unfortunately, designers have been restricted in the past to thinking in terms of how a computer defines a solid object, and these restrictions have often limited the ingenuity and imagination of the designer. Features provide an essential link between CAD and CAM, or in the case of reverse engineering, to automate the reproduction of engineered objects. Features provide a convenient, high level language for specifying mechanical parts and for facilitating automated manufacturing.

Many software packages have been developed to create cutter paths from surface data points. Most use surface-following methods dependent on offset computations. These are not practical methods for producing a cutter path, as it does not necessarily generate the smoothest or most efficient motion. This "action" is very similar to the use of copy lathes or mills. However, if the object's features have been defined, the best cutter path and machining strategy can be employed to re-create that feature type.

6.1 Review of Features Literature in CAM

There has been an explosion of commercial "features" driven CAD solid modelling packages in the past few years in the market place. The CAM industry has "...comparatively lagged the almost exponential growth in CAD systems."[45] In the same way that most modern CAD packages take full advantage of features to help engineers design, CAM packages must extract sets of features in order to machine the solid object more efficiently with its available set of tools and machining conditions. However, if a part description has to be prepared manually, the quality of the resulting process plan depends on the user's experience and knowledge. It is this link between design and manufacturing that represents a vast field of research that is crucial to the success of the CAD/CAM industry. However, it should be noted that the features relevant for design are not the same as those relevant for machining.[46] Certainly designers should consider the manufacturing process as part of their designs, but it is unreasonable to force designers to specify features in terms of manufacturing processes when their main concern should be on the functional or form characteristics of the design. Therefore, whether the original design was created on CAD or the data was reverse engineered, a form of features extraction is required for the smooth link to CAM.

What exactly constitutes a feature is yet another area of ambiguity. Vandenbrande[46] informally defines a feature as regions of an object that are meaningful for a specific activity or application. Subrahmanyan[45] points out that the term "feature" is often a very general term used to indicate a non-unique shape that a desired part should possess as a consequence of applying a manufacturing process. Schulte et al.[47] puts forward a similar definition of a

feature "as geometry associated with a specific machining operation." Another definition of a feature is put forward by Shah[48]. A feature is described as a physical constituent of a part, mappable to a generic shape and "represents the engineering meaning of the geometry of a part of assembly." The word "features" also represents different things to different people, depending on the context in which it is used. For example, to a manufacturing engineer a feature might refer to a slot, or a hole or a pocket. To a designer, a feature might represent a web or an ergonomic handle, etc. For quality control and inspection, a feature is usually a datum or reference plane on a part.

Although feature recognition has been researched, it has mostly been focused on the recognition of features in CAD models. Most of this research in feature recognition has ceased since the introduction of feature based ("parametric") CAD programs. However, for the purposes of feature recognition in range data, it would be helpful to review a few of the methods.

One of the most popular methods of geometric feature recognition is by parameter matching. These features are first defined in terms of their geometric and topological characteristics. The feature matching algorithm searches the solid model database to determine if any of these characteristics exits. Since the topology of many solid models are stored as a graph, many matching algorithms which use only topological matching are often called graph matching algorithms. Pure graph matching done on solid models amounts to topological matching, the characteristics are based on the number of entities, topologic type, connectivity and adjacency. Another matching technique is syntactic pattern recognition.

In this method, the geometry is described by a series of straight, circular or other curved line segments. Languages have been developed for describing these sequences and manipulating them with operators that form a grammar. Features can be recognised by matching the feature against the object's description in the representation.

In "entity growing" feature extraction, features that are recognised are subtracted from the solid model by adding the geometric shape to the solid to "fill in" the feature. As the features recognised are not always an enclosed volume, new faces and topological entities are often created. Thus the term, "entity growing" is used. The advantage with entity growing is that with the resulting simpler solid model, features often not apparent with the complex model become obvious in the simpler solid model. Another method, volume decomposition, is a process where the material to be removed from the base stock is identified and is then broken down into distinct machining operations. The total volume of material to be machined is found by a Boolean difference between the stock and the finished part. This volume is then decomposed into smaller volumes or "features" which correspond to machining operations. By using a library of generic removal volumes, the features can be easily extracted from the removal volume.

Vandenbrande and Requicha[46], use a "generate and test" method to extract features for rapid prototyping on a 3-axis vertical milling centre. First, the complete data is processed for characteristic combinations of part faces to generate hints for the presence of machining features. These hints are then passed through a hint classifier that categorises them into three groups: promising, unpromising and rejected. The promising hints are further processed by

the algorithm, which searches for all the relevant data about the feature and tries to fit the largest possible feature volume that is consistent with the data. Unpromising hints are temporarily stored to see if they become part of a feature that was started by a promising hint. Otherwise, they are later discarded. Rejected hints are discarded right away. All the processed features are again stored where another algorithm attempts to combine them with other features. For instance, two coaxial and adjacent holes are combined to form a counter-bored hole, which is a specific type of composite hole. Their final step is the verification of the extracted features. Vandenbrande and Requicha generate their hints from characteristic patterns, combinations of faces that satisfy certain topological and geometric relationships. For example, a "hole-hint" is triggered by the existence of either a cylindrical or conical face. A linear slot and grooves are found by searching for two opposing and parallel planar faces, while a pocket hint is generated by a planar face surrounded by zero or more adjacent faces (walls). A good technique is the "generate and test" strategy, which is used to avoid unnecessary computations as hints are evaluated before further processing.

Henderson and Chang[49] use a feature recognition algorithm in order to perform automated process planning. The feature recognition process is split into a number of parts. First, the recognition algorithm searches for all depressions. It uses these depressions to extract recognisable features. Second, the extracted features list is checked for redundant representations. For instance, a through hole that may be extracted as two holes (one in each direction) would have one of the holes discarded to remove the redundancy. Then, the parameters that are related to the extracted feature are calculated. The algorithm developed

by Henderson only works on slots and holes. Holes are found by employing a multi-step pattern matching process. In the first step, the algorithm tries to match the topological and geometrical data with a predefined description of a convex circular edge. The second step is to match the possible hole components with the definition of a hole. To find an open simple slot, their algorithm searches for a component edge, which exist in the same depth with respect to its entrance face. No example of the slot definition was given, but there is an explanation of how to decompose a slot made of many smaller slots.

Sakurai and Gossard[50] divide their feature extraction procedure into two distinct segments. The first is to define the features to be recognised. The second step is to recognise the feature from a 3-D solid model. In their definition routine, Sakurai uses an interactive method in which the user selects faces from an example feature in a 3-D solid model shown on a graphic display. From the selected faces, the "facts" are defined and recorded. In the second step, feature recognition, the 3-D solid model is searched for each type of feature. Once a feature is found, the feature is checked to see if it is part of a more complex feature. Finally, before the algorithm sets off to find the next feature, the volume of the feature is computed and is combined with the original solid model to result in a new solid model without the feature. Sakurai explains that this last step is needed as the simpler solid model often reveals features that were previously hidden by the original solid model. Features are recognised by matching the topological information between the defined feature and a feature on the solid model.

A feature extraction method proposed by L De Floriani and E. Bruzzone[51] is entirely

based on the topological information contained in a solid model. As a first step the model is broken down into a "symmetric boundary graph" by extracting all the loops and connected faces from the solid model. Predefined features are matched with this boundary graph and are removed from the symmetric boundary graph. The algorithm then goes on matching with the rest of the data set until no more features can be found.

Joshi and Chang present a concept called the attributed adjacency graph (AAG)[52]. It is a graphical representation of the faces and edges of a part. Faces are represented by nodes, while the edges are represented by arcs. All convex edges are represented by 1's and concave edges by 0's. A matrix represents the graphs for the part and feature recognition is achieved by recognising the sub-graphs of features in a part. However, one of the drawbacks to this method is that it is restricted to 2-1/2 D parts and that other types of shapes, such as cones, spheres, taurus are also problems for this algorithm. Nezis and Vosniakos[53] combine the use of an AAG and a neural network. The neural network used is a three layer network, with 20 elements in the input layer, 10 elements in the single hidden layer and 8 elements in the output layer (one element each to represent the different types of features which can be found). Input to the network was through a 20-element vector based on an Attributed Adjacency Graph, which is reduced into an Adjacency Matrix. A set of heuristics is then used to break down the graph into a set of sub-graphs that are coded into an Adjacency Matrix. Face adjacency is represented in the Adjacency Matrix, where element (i, j), gives the relationships of face j to face i, depending on the binary value of the element. The Adjacency Matrix is then finally used as the input to the neural network recogniser.

Although the automation of reverse engineering is dependent on an algorithm to extract features, none of the reverse engineering packages use any kind of automatic feature detection. The main problem in applying the previously reviewed feature extraction methods to scanned data is that the data is often unstructured and incomplete. Therefore, reverse engineering packages with a provision to identify features rely on interactive user interface to extract the features from the data. The REFAB system by Thompson[54] uses an interactive graphics workstation to segment the scanned data into features. First the user picks the feature type and approximate location. Then the REFAB system, using an iterative refinement process, fits a feature to the scanned data. Work by Kwok and Eagle[5], allow the user to visually pick the features and then guide a CMM touch probe to the location. Their software then automatically stores the feature in an IGES format, ready to be imported to a CAD system.

6.2 Feature Extraction in Reverse Engineering

There have been various techniques developed to extract features from a geometric modelling database. It is, however, difficult to classify feature recognition methods into clean, uncluttered groups, as there is considerable overlap between the various techniques. The majority of methods use a matching algorithm, which compare data with predefined generic features. Feature recognition algorithms may include the following specific tasks:

1) Generic definition of a features topology.

2) Searching the database to match topologic patterns.

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

4) Determining feature parameters (hole diameter, pocket depth).

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

In this work a "feature" will be defined as a recognisable topological pattern of a set of faces and edges. The features will be further limited in scope to specific machining operations for the creation of slots, corners and steps. 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 part, not on the design of parts. The reverse engineering package is to model the part with some rudimentary editing features. It is not meant as a replacement for a CAD package.

6.4 Neural Network Feature Extraction Method

The automation of feature extraction has not been addressed in other reverse engineering packages. Due to the nature of reverse engineered data, to implement a feature extraction algorithm, a robust method is required.

Rule based algorithms rely on concise and accurate data to which the rules are tested.

Because reverse engineered data is often incomplete and subjected to measurement noise, a neural network based algorithm is thought to be more robust. Artificial neural networks are based on their biological counterparts, where a large number of simple processing elements are interconnected. Typically the processing units are either a binary element, and perform a simple arithmetic operation on its input. The weighted inter-connectors transmit values between elements, adjusting the values depending on their weighting. During the training stage, the weighting of the connectors are adjusted to give the appropriate results. The main difference between neural networks and rule base algorithms are that the neural networks are dependant on simple arithmetic between it's elements, whereas rule based algorithms are dependant on logic statements.

Neural networks have seen successful applications in machine vision pattern recognition and in signal analysis problems. However, little work has been done on applying neural networks to feature extraction, especially in extraction from range data. The network used in this work is also a feed-forward based network with one hidden layer. The number of layers in a neural network cannot be pre-determined by any rules, only through experimentation. 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. Again, the number of elements in the hidden layer cannot be determined except through experimentation. Nezis and Vosniakos[53] found that by increasing the number of hidden layers did not alter the results but did increase the training times considerably. 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.

The neural network used in this research, as shown in Figure 58, is trained by using back-propagation methods. Size of the network is dictated by the size of the input and output vector. In a feed-forward back-propagation network, errors between the output and the expected output are back-propagated during the training period, to adjust the connector weights between the input, hidden and output layers. The amount of adjustment of the connector weight is determined by multiplying the error by a learning rate. Adjusting the weights between these sets of neural elements is an iterative process, with pairs of input and output vectors chosen to train the network. The training that is carried on until the error falls below a threshold. Once training is completed, the weights are set and the network can be used for recognition.

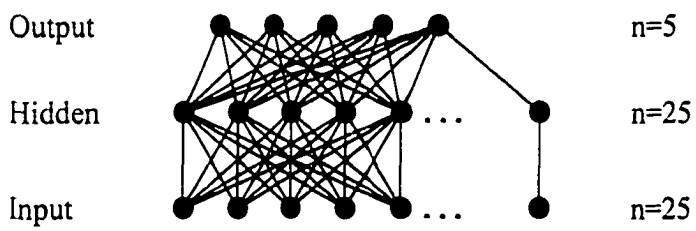

Figure 58: Neural network model used for feature recognition

Let the input neuron be represented by $\eta_{1,i}$ the hidden layer neuron $\eta_{2,i}$ and the output layer neuron $\eta_{3,i}$ (where i = 1 to n, depending on the respective layer as shown in Figure 58).

The weight of a connector between a neuron on the input layer and the hidden layer is represented by $x_{(1,i),(2,i)}$ and the weight of a connector between the hidden layer and the output layer is represented by $x_{(2,i),(3,i)}$. Therefore, the value of a neuron on the hidden layer is given by Equation 19.

$$\eta_{(2,i)} = \sum_{m=1}^{n} x_{(1,m),(2,i)} \bullet \frac{\eta_{(1,m)}}{90} \tag{19}$$

Similarly, the values of the output neurons are calculated as the product of the connector weights and the corresponding neuron values on the hidden layer. However, to bias the output neurons towards unity, a sigmoid function is used in Equation 20 below:

$$\eta_{(3,i)} = \frac{1}{(1 + \exp[\ -(\sum_{m=1}^{n} x_{(2,m),(3,i)} \bullet \eta_{(2,m)})]} \tag{20}$$

Determination of the weights of the connectors is completed during the training stage. Errors between the calculated output and the desired output are used as a basis for the adjustments of the connection weights between the output and hidden layers, and the hidden and input layers. The output error vector is calculated as:

$$E_{output,i} = desired\_output_{i} - \eta_{(3,i)} \tag{21}$$

The reflected vector is the dot product of the error vector, E_{output} and the calculated output vector, η_3 scaled by the complement of the output vector ($1-\eta_{(3,i)}$). Effectively, this vector

results in higher values for outputs with large error and outputs that are neither close to 1 or 0 in value. The product is expressed as:

$$R_i = E_{output,i} \bullet \eta_{(3,i)} \bullet (1 - \eta_{(3,i)}) \tag{22}$$

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

$$\frac{dx_{(2,j),(3,j)}}{dt} = A \bullet R_j \bullet \eta_{(2,i)} \tag{23}$$

where the constant A is the learning rate.

The error of the neurons on the hidden layer is calculated by taking the dot product between the reflected vector and the vector consisting of the connector weights between the hidden neuron $\eta_{(2,i)}$ and the output. (effectively $x_{(2,i),(3,m)}$ for m=1 to n) Again, this product is scaled by the value of neuron $\eta_{(2,i)}$ multiplied by its complement, $(1-\eta_{(2,i)})$. Therefore, the hidden layer error can be written as:

$$E_{hidden,i} = \eta_{(2,i)} \bullet (1 - \eta_{(2,i)}) \bullet \sum_{m=1}^{n} R_m \bullet x_{(2,i),(3,m)} \tag{24}$$

Finally, the adjustment of the connector between the i^{th} input neuron, $\eta_{(1,i)}$ and the j^{th} neuron

of hidden layer, $\eta_{(2,j)}$ is given by:

$$\frac{dx_{(1,i),(2,j)}}{dt} = B \bullet E_{hidden,j} \bullet \frac{\eta_{(1,i)}}{90} \tag{25}$$

where B represent the learning rate.

Adjusting the two sets of weights between the layers and recalculating the outputs is an iterative process that is repeated until the errors fall below a pre-determined tolerance level. The allowable error tolerance was determined through experimentation. Different values for the allowable error, ranging from a high of 1.0 to 0.05 were tested. It was found that a large error resulted in a poorly performing neural network, giving many incorrect answers. However, a very small allowable error resulted in excessively long training times. An error tolerance of 0.10 was selected as a compromise between training speed and algorithm accuracy in determining the correct feature.

Similarly, the effect of the learning rate A and B on the training speed and accuracy was determined through experimentation. A value of 0.2 was selected from both learning rates A and B.

6.5 Input Representation

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. In work by Nezis and Vosniakos[53] and by Prabhakar and Henderson[55] an Adjacency Matrix was created from the topology data provided by the CAD software. However, with reverse engineered data, the geometry may also prove to be an important indicator of possible features, as the faces that make up a feature may not be fully defined. Therefore the method used in this work will be a Modified Adjacency Matrix, where instead of simply representing whether two faces are adjacent, an added factor, the angle formed by their surface normal vectors are added. Figure 59 shows a sample of an Adjacency Matrix for a face surrounded by four other faces, resulting in a 5x5 matrix.

	1	2	3	4	5
1					
2	87				
3	90	85			
4	93		90		
5	91	91		93	

Figure 59: Adjacency matrix for feature recognition

In the Modified Adjacency Matrix, the upper triangle of the matrix represents convex angles, while the lower triangle of the matrix represents concave angles. For example, in Figure 59 the surfaces numbered 2 and 5 make a 91° concave angle.

Data for the Modified Adjacency Matrix is drawn from both the topology and geometry databases. Each face in the topology database and its surrounding faces as outlined by the face/loops (edges) is coded to its adjacency and the angle between its surface normals.

Because of the 5x5 size of the adjacency matrix, a 25 element input vector to the neural network feature recognition algorithm is required.

6.6 Feature Definitions

Features will be defined by a combination of geometric shapes associated with manufacturing operations. Each element of the output layer is used to represent a single feature. A simple description of the features is presented below:

- SLOT is a depression with a bottom face and two side faces.

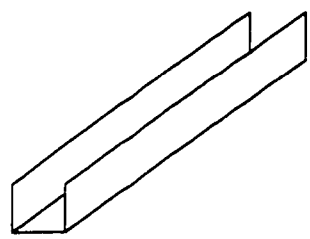

Figure 60: Diagram of slot feature

- STEP is a depression with a bottom face and one side face.

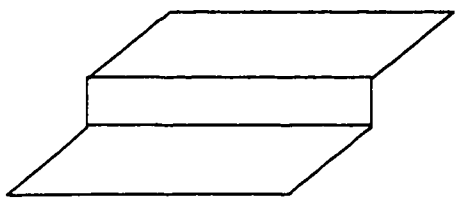

Figure 61: Diagram of step feature

- CORNER is a face forming a convex with two side faces.

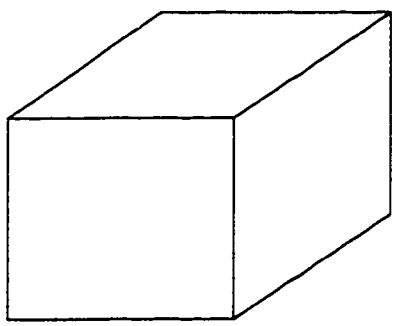

Figure 62: Diagram of corner feature

The neural network recognition algorithm must first be trained to recognise the above three features. Synthetic data was manually created to submit to the neural network for evaluation. During the training stage, the solution to the input vector is known beforehand, thus the neurons can be corrected through back-propagation. Sample training vectors are shown in Figure 63. The first line in Figure 63 represents the input vector, the second line

represents the correct output vector. The next two lines represent the next input and output pair of vectors. This format continues until the end-of-file is reached.

```
25 0 87 90 0 0 0 0 89 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
6 1 0 0 0 0 0
25 0 0 90 0 0 0 0 89 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
6 0 0 0 0 0 1
25 0 0 0 0 0 90 0 0 0 0 90 0 0 0 0 0 0 0 0 0 0 0 0 0 0
6 0 0 1 0 0 0
25 0 0 90 0 0 0 0 89 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
6 0 0 0 0 0 1
```

Figure 63: Sample training vectors - input vector followed by output vector

Through testing it was found that the algorithm was sensitive to three items. First, the order in which the training vectors were presented. For example, if the last 10 vector pairs presented were to represent slots, the neural network would be biased towards indicating slot features. Therefore, a random mix of training vector types is important to successful training. Second, the number of training vectors played a significant role in the accuracy of the neural network in deciding the type of feature that was being presented. It was found that a minimum of 150 vectors were required to train the neural network to recognise three different types of features. Third, the error allowed before the back-propagation algorithm stopped correcting the neuron connectors. Again, through testing, an error value was found (0.01) that gave reliable results.

6.7 Testing of the Neural Network Algorithm

The neural network feature extraction method described in this work is based on both topologic and geometric information. Although the neural network has been trained with topological data, it has only been with synthetic data. To fully test the robustness of the algorithm, real reverse engineering data was utilised.

The neural network was presented with the three-step object (see Figure 2) introduced earlier in this thesis. Face/loop and topology data associated with each patch was derived from the algorithm presented in Chapter 5. Using the extended data feature in AutoCAD to store the topology data, neighbouring information was examined, and the appropriate 25-element input vector for the neural network algorithm was derived. In situations where more than two neighbouring patches (three patches), the maximum number of combinations of input vectors are created. For example, if there are three patches that are neighbouring, there are four possible unique vectors.

In analysing the three-step object, a total of 31 input vectors were created from the seven surface patches. The processing of the 31 vectors required less that 1 second and correctly identified the four outer corners on the object as shown in Figure 64. A typical output suspecting a possible corner feature is shown in Figure 65.

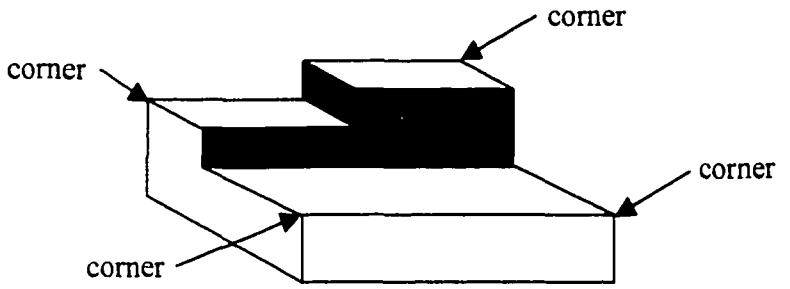

Figure 64: Corners identified on the three-step test object

```
Reading in vector 3 has: 0, 82, 96, 0, 0, 0, 0,
82, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0,

output layer elements= 0.923111, 0.0161301,
0.000258668, 0.00470541, 0.00217018, 0.0341589,

Max element is:  0    suspect corner
```

Figure 65: Possible corner suspected flag from feature recognition algorithm

However, in analysing the same 31 input vectors, none of the possible step features were identified by the algorithm. One possible explanation of this failing, is the lack of a side face as described by the definition in the previous section. These missing side faces and the possible steps that they form are shown as the shaded planes in Figure 66. The surfaces were initially left missed by the automated scanning algorithm, and thus were not on the part path derived to direct the Hyscan laser scanner. The neural network feature recognition algorithm did not prove to be robust enough to distinguish partial step features.

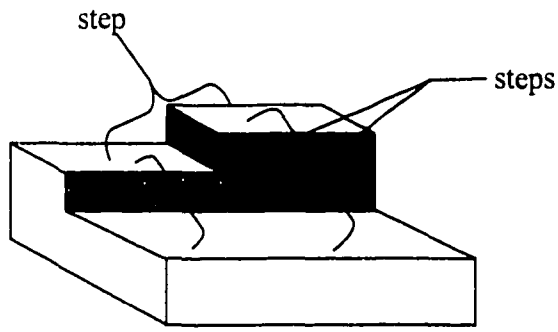

Figure 66: Possible step features missed by the algorithm

Chapter 7 - Summary and Conclusions

In this research, progress in the field of geometric reverse engineering has been made. Through the application of neural network based machine vision, the directing of both a physical touch trigger probe and a non-contact laser scanner has been automated. This has reduced the requirement of user intervention to collect surface data points on an object. A chordal deviation algorithm has been developed to separate "over-scan" of neighbouring surface patches when digitising with a laser scanner. In order to reduce the size of data sets, a new flexible voxel binning program has been developed. This program calculates the best representative pixel for a group of pixels occupying a defined cubic volume. An algorithm that finds the boundaries of surface patches without user intervention was realised. Using fuzzy logic, a new geometric primitive fitting program was developed which has decreased the number of iterations required to fit quadric primitives, such as planes, spheres, and cylinders, to cloud data. Finally, a neural network base feature recognition was implemented to find important geometric features on objects being reversed engineered. This is a new application of feature recognition research that has not been applied to reverse engineering.

A summary of the significant contributes is listed in Table 14.

Table 14: Contributions to geometric reverse engineering

Area of Research	Significant Contribution
Automated Data Collection	Developed a new neural network based machine vision to guide the data collection process
3-D Cloud Data Purging	Refined procedures of cleaning cloud data through the application of chordal deviation, voxel binning and boundary following.
Primitive Fitting	Applied fuzzy logic methods to decrease iteration time for fitting of geometric primitives to cloud data.
Feature Recognition	To fully reverse engineer an object, developed an algorithm to recognise certain geometric features on the object's surface.

Although significant work has been made towards applying feature recognition towards reverse engineering, future research should include the application of feature recognition earlier in the reverse engineering process. This may include the use of a lower resolution 3-D laser scanner device to gather preliminary surface data from which features on the surface of the object can be identified. By applying feature recognition earlier in the reverse engineering stage, an even more directed collection of exact measured data points can be achieved than that accomplished in this work.

Appendix A

Glossary of Terms

Term	Page First Used	Definition
2-1/2 D	13	A model space where typically the Z-axis is constrained.
A Priori	20	Information that was known beforehand.
AAG	110	Attribute Adjacency Graph - Method of encoding the adjacency of different attributes in a matrix.
B-rep	4	Boundary Representation – A method used by CAD programs to model a solid with its boundaries.
B-splines	11	Parametric method of representing a curve
Back-Propagation	114	Method to update connector weights in neural networks based on the error.
CAD	1	Computer Aided Design – a computer program that allows for design on a computer
CAM	9	Computer Aided Manufacturing – a computer program to aid in the planning of a manufacturing process.
CCD	6	Charged Coupled Device – a light sensitive microchip used to capture images in video cameras.
Chordal Deviation	67	Measure of the amount of deviation of a chord from a reference chord.
Cloud Data	4	Term used to describe the cloud like structure of data points collected by a range finder sensor.
CMM	2	Coordinate Measuring Machine – a precise machine used in industry to measure surface points.
CNC	14	Computer Numeric Control – a control system used to program the movements of a machine tool.
Competitive learning	28	Type of neural network where in training, there can only be one correct neuron.

Connectors	28	Either excitatory or inhibitory in nature, the are the weighted links between neurons.
Depth of Field	5	The range where measurements are considered accurate.
DOF	52	Degree of freedom – the number of axis that can be manipulated.
Feature	17	A combination of geometric entities that together have a meaningful purpose.
Frame grabber	44	Plug-in card for a computer that can accept a video signal.
Free-form surface	62	A surface not made of any geometric primitives.
Fuzzy logic	8	A method for decision not based on absolute rules. Answers may vary with each decision.
Gaussian noise	77	Noisy data which is distributed normally about a datum.
Grey level	30	The intensity of a pixel in an image.
Hymarc laser scanner	9	A brand of laser scanner based on triangulation range finding.
In process	20	Information is gathered as the process is running.
Kohonen SOM	28	Self Organising Map – A neural network based on competitive learning among neighbouring neurons.
Laplacian of Gaussian	12	A common filtering algorithm for smoothing and edge finding in machine vision images.
Lathe	62	A metal cutting machine where the workpiece is rotated against a stationary cutter.
Least squares	11	An error minimising technique based on the square of the error.

Milling machine	62	A metal cutting machine where a rotating cutter is used to machine a workpiece that is held stationary.
NN	8	Neural network – a computer algorithm based on the architecture of a biological brain.
Neuron	28	A node in a neural network.
NURBS	11	Non-uniform rational B-splines – a parametric representation of curved lines.
Outlier	11	A data point that is outside of the normal group. Usually attributed to noise.
Parallax	19	The shift in two similar images resulting from two different viewpoints.
Parametric CAD	106	A CAD program that allows designers to specify models with parameters, such as features.
Part Datum	8	A surface on an object from which other dimensions are referenced.
Pixel	33	The smallest light gathering element in a computer image.
Quadric surface	62	Surfaces such as planes, cylinders, spheres and cones which follow the general quadric: $F(x,y,z)=Ax^2+By^2+Cz^2+2Dxy+2Eyz+2Fxz+2Gx+2Hy+2Jz+K=0$
Range finder	2	A device used to measure in 3-D space.
Range map	47	A map of data points in 3-D space.
SGI	40	Silicon Graphics Inc. – a maker of computer workstations.
Standoff	52	A distance that a sensor must be positioned above the object being measured.
Topology	17	The spatial relationship of different surfaces to each other.

Touch trigger probe	2	A sensor used for making measurements, see Fig.21.
Travelling salesman problem	50	A classical optimisation problem for determining the shortest path for a "travelling salesman".
Trunnion	51	A device which allows for the rotation in one axis.
Voxel bin	12	A cubic volume derived from a larger volume.

Appendix B

Header files for program code

Object location and Segmentation – Automated Part Digitisation

Chapters 2 and 3.

```
//stereo_main.h
//by Vincent Chan
//=====================================================================
//header file for
//main program to read in SGI rgb file and segment the image into
//different patches using fuzzy NN
//
//will read in either a colour rgb file or b&w rgb file
//
//complie with:
//                read_rgb.o
//                display_image.o
//=====================================================================


#define compress_ratio 4            //ratio to compress original image
by
#define width_factor 0.0161         //factor to convert pixels to mm
#include <math.h>
#include <fstream.h>
#include <stdio.h>
#include <stdlib.h>


extern "C" int read_image(short picture[][640], int *pxsize, int
*pysize);
extern "C" int display_image(short picture[][640], int xsize, int ysize,
       int refresh, int neurons, char win_name[20]);

struct patch      {     //structured variable to carry patch parameters
    int grey;           //mean grey level
    int num_pixels;     //number of pixels in patch
    int hieght;         //height of patch in pixels
    int hieght_max;     //max j pixel of patch
    int hieght_min;     //min j pixel of patch
    int width;          //width of patch in pixels
    int width_max;      //max i pixel of patch
    int width_min;      //min i pixel of patch
    int cent_i;         //i co-ord of centroid
    int cent_j;         //j co-ord of centroid
    int original_layer; //original layer the patch was found on
};

int segmentate(int hole_cent[4][1000], short picture[][640], int xsize,
int ysize, struct patch *,
                int *pnum_patches);

int display_neurons(short pneurons[][640/compress_ratio][10], int
xcompress,
```

```
      int ycompress, int refresh, int neurons, char win_name[20],int
individual);


int match_patch(int right_holes[4][1000],
            struct patch *right,  int num_patches_right,
            struct patch *left,   int num_patches_left);

void calculate_z (int angle,  float centroid_dist, float z1,
            int cent_i,  int width, int width_max, int width_min,
            float *cent_z, float *width_z, float *max_z, float *min_z);


void calculate_2 (int angle, int vector, float centroid_dist, float x1,
                int cent_i, int width_i, int width_max, int width_min,
                float *cent, float *width, float *max, float *min);


void calculate_3 (int vector, float centroid_dist, float x1, int cent_j,
                int hieght_j, int hieght_max, int hieght_min,
                float *cent, float *width, float *max, float *min);


int gcode_laser (int num_matches,  float cent_x[9],  float cent_y[9],
                  float cent_z[9], float width_x[9],
                  float width_y[9], float width_z[9]);

int check_width (int width);

float swaths(int passes, int k);

void gcode_touch (int num_matches,  float cent_x[9],  float cent_y[9],
                  float cent_z[9], float min_x[9], float max_x[9],
                  float min_y[9], float max_y[9], float min_z[9],
                  float max_z[9], int slot,  int angle);


void salesman(int num_matches, int path_order[9], float cent_x[9],
              float cent_y[9], float cent_z[9]);

void probe_point (float cent_x, float cent_y, float cent_z,
                int vector[3], int approach[3], int memory_no,
                char file_name[25]);


void anti_bleed (short picture[][640], int xsize, int ysize, int
backgound);
```

```
/* entlayer.cc ads program
 * vincent
 */
#include  <stdio.h>
#include  "adslib.h"
#include  <math.h>

//define the maximum spacing between lines
#define MAX_SPACE 2.0
#define MAX_ANGLE 0.1745
#define MAX_NUM_GROUP 100

class clean_patch   {
      ads_name line_name[10000];
      ads_point start_pt[10000];
      ads_point mid_pt[10000];
      ads_point end_pt[10000];
      int group1[10000];
      int group2[10000];
      char patch_name[25];
      char axis;
      int group_count;      //number of groups grouping

    public:
      int entlayer();
      void find_centroid(ads_point centroid);
      char find_axis();
      float calculate_angle(ads_point old_point,
          ads_point new_point);
      float parallel_dist(ads_point old_point,
          ads_point new_point);
      float perpen_dist(ads_point old_point,
          ads_point new_point);
      float perpen_dist_calc(float old1, float new1,
                     float old2, float new2);
      float calculate_dist(ads_point new_point,
                     ads_point old_point);
      void change_colour(int array_name, int colour);
      void group_line(int line_count);
      int between(ads_point test, ads_point start, ads_point end);
};

struct resbuf *assoc_rb(struct resbuf *rb, int group);
int adsfunc();
int entlayer();
void set_name(int num, char *layer_name);
int convert_int (int mesh_num, char mesh_name[25]);
```

```
//geo_fit.h
//header file for geometry fitting
//and topology generation
//V. Chan

#include <stdio.h>
#include "adslib.h"
#include <math.h>
#include <iostream.h>
#include <fstream.h>
#include <string.h>
#include "dxf_codes.h"
#include <stdlib.h>

#define STEP 0.5
#define MAX_DATA 10000
#define PI 3.14159

struct resbuf *assoc_rb(struct resbuf *rb, int group);
int twirly_thing(int count);
int cross_product(ads_point first,  ads_point sec,  ads_point normal);
int unit_vector(ads_point unit);
float distance(ads_point pt1, ads_point pt2);
float angle_formula(ads_point first, ads_point sec);
float projection(ads_point normal,  ads_point centroid,
            ads_point old_point, ads_point new_point);
int rotate_z(ads_point point, float phi, ads_point new_point);
int rotate_x(ads_point point, float theta, ads_point new_point);
int Get_Int_Var(const char* varname);
void Register_App (char app_name[25]);
int Table_Object_Exists(const char* Table, const char* Object);
int convert_int (int mesh_num, char mesh_name[25]);
int factorial(int n);


struct scan_data{
      ads_point point[1000000];
      int link[1000000];
};

struct feature_data{
      char name[25];
      char type[25];
      ads_point centre;
      float radius;
      ads_point first;
      ads_point second;
};
```

```
class fit_ent   {
    ads_name line_name[MAX_DATA];
    ads_point start_pt[MAX_DATA];
    ads_point end_pt[MAX_DATA];
    ads_point first_polyline[MAX_DATA];
    ads_point last_polyline[MAX_DATA];
    int first_poly_num,  last_poly_num;
    int line_count;
    int axis;
    char axis_direction;
    ads_point fitpts[MAX_DATA];
    long num_fitpts;
    ads_point centroid;
    ads_point curve_cent;
    float avg_curvature;
    ads_point projections[MAX_DATA];
    ads_point avg_min_vector;
    ads_real voxel_size;
    int bin_size[3];
    float line_spacing;
    struct scan_data data;
    int num_points;
    int bin_ptr[50][50][50];
    int num_bpoints;
    float boundary[3][2];
    ads_point bound_data[8*MAX_DATA];
    struct feature_data cylinder;
    struct feature_data sphere;
    struct feature_data plane;
    struct feature_data feat_data[50];


    public:

    int geo_fit();
    int seleciton(char layer_name[25]);
    int find_ends(char patch_name[25]);
    int move_entities(char first_oatch_name[25],  char
sec_patch_name[25]);
    int plot_transfer(char patch_name[25],
                ads_point patch_data[8*MAX_DATA],  long *num_points);
    int find_border(char patch_name[25]);
    int follow_around(ads_point first_patch[8*MAX_DATA],
                  long num_points);
    int erase_poly(char patch_name[25]);
    int cross_section(char layer_name[25]);
    int voxel_bin(char layer_name[25]);
    int voxel_out(int x_bins, int y_bins, int z_bins, int bin_flags);
    float fit_plain();
    int fit_curvature(char layer_name[25]);
    ads_point std_normal;
    float fit_sphere();
    float fit_cylinder();
    int check_quad(ads_point centre, ads_point points[10*MAX_DATA],
                int num_points);
    int read_fitpts(char layer_name[25]);
```

```
    int make_mesh(char layer_name[25], int surface_type);
    int selection(char layer_name[25]);
    int read_boundary(char layer_name[25]);
    int find_bounds();
    int select_corner();
    int chordal_deviation(int tagged_index[MAX_DATA]);
    int cross_points( ads_point point1a, ads_point point1b,
                      ads_point point2a, ads_point point2b,
                      ads_point cross_pt1, ads_point cross_pt2,
                      ads_point mother_normal);
    int closest_meshpt(int style, ads_point cross_pt1, ads_point
cross_pt2,
                       ads_point meshpoint, ads_point emergency,
                       ads_point mother_normal);
    int Read_Xdata_Str(const ads_name E_Name, char app_name[25], int
ent_num);
    int Attach_Xdata_Str (const char* Val, const ads_name E_Name,
                          char app_name[25]);
    int Attach_Xdata_Name (const ads_name E_Name);
    int Attach_Xdata_Type (const char* Val, const ads_name E_Name);

    int Attach_Xdata_Centre (ads_point centre, const ads_name E_Name);
    int Attach_Xdata_Radius (ads_real radius, const ads_name E_Name);

    int Attach_Xdata_Point1 (ads_point centre, const ads_name E_Name);
    int Attach_Xdata_Point2 (ads_point centre, const ads_name E_Name);

    int read_2_array();
    int create_3vectors(int num_entities);
    int create_topology();
    int scale_patch(char patch_name[25]);

};
```

```
// features.cc  program
// written by Vincent Chan


//*****************************************************************
// program to train a feed forward neural network to recognize
// features, using pseudo feature vectors
// requred files: features.dat      training vectors
//
//*****************************************************************


#include  <stdio.h>
#include  <stdlib.h>
#include  <fstream.h>
#include  <iostream.h>
#include  <string.h>
#include  <math.h>

#define THRESHOLD 254.0
#define LEARN 0.2


class neurons  {
      float input_layer[25];       //elements in the input layer
      float hidden_layer[25];      //elements in the hidden layer
      float output_layer[6];       //elements in the output layer
      float in_hidden[25][25];     //connectors between input and hidden
elements
      float hidden_out[25][6];     //connectors between hidden and output
elements

    public:
      void training();
      void find_features();
};
```

Bibliography

[1] Varady T, Martin R and Cox J, "Reverse engineering of geometric models – an introduction", Computer Aided Design, v29, n4, 1997, pp255-268.

[2] Carter B, "Software and Digitizers Push Reverse Engineering Forward", Machine Design, Aug 8, 1996.

[3] Chivate PN, Jablokow AG, "Solid-model Generations from Measured Point Data", Computer-Aided Design v25 n9 Sept 93 pp587-600.

[4] Hansen F, Pavlakos E, Hoffman E, Kanade T, Reddy R, Wright P, "PARES: A Prototyping and Reverse Engineering System for Mechanical Parts-on-Demand on the National Network", Journal of Manufacturing Systems v12 n4 1993 pp269-281

[5] Kwok WL, Eagle PJ, "Reverse Engineering: Extracting CAD Data From Existing Parts", Mechanical Engineering v113 n3 Mar'91 pp52-55.

[6] Raab S, "Coordinate measurements accelerate reverse engineering", Machine Design v66 n22 Nov.21/94 pp50-53.

[7] Sobh T, Owen J, Jaynes C, Dekhil M, Henderson TC, "Industrial Inspection and Reverse Engineering", 2nd CAD-Based Vision Workshop, Proceedings IEEE 1994.

[8] Milroy M, Weir DJ, Bradely C, Vickers GW., "Reverse Engineering Employing a 3-D Laser Scanner: A Case Study", International Journal of Advanced Manufacturing Technology, v12, 1996, pp111-121.

[9] Milroy M, Bradley C, Vickers GW, "G1 Continuity of B-spline surface patches in reverse engineering", Computer Aided Design, v27, n6, 1995 pp471-478.

[10] Sakar B, Menq CH, "Smooth-Surface approximation and reverse engineering", Computer Aided Design, v23, n9, 1991, pp623-628.

[11] Gu P, Yan X, "Neural Network Approach to the Reconstruction of Free Form Surfaces for Reverse Engineering", Computer Aided Design, v27, n1, pp59-64.

[12] Liao CW, Medioni G, "NOTE: Surface Approximation of a Cloud of 3D Points", Graphical Models and Image Processing, v57, n1, Jan. 1995 pp67-74.

[13] Bradley C, Milroy M, Vickers GW, "Reverse engineering of quadric surfaces employing 3-D laser scanning", Proceedings of the Institution of Mechanical Engineers – Part B. v208, pp21-28.

[14] Bradely C, Numerically Controlled Machining from Three Dimensional Machine Vision Data, Ph.D. Dissertation, University of Victoria, 1992.

[15] Milroy M, Bradley C, Vickers GW, "Segmentation of a Wrap-Around Model using an Active Contour", Computer Aided Design, v29, n4, 1996, pp299-320.

[16] Flynn PJ, Jain AK, "Surface Classification: Hypothesis Testing and Parameter Estimation", Proceedings of the 1988 IEEE Conference on Computer Vision and Patttern Recognition, pp261-267.

[17] Hoffman R, Jain K, "Segmentation & Classification of Range Images", IEEE Transactions on Pattern Analysis & Machine Intelligence, v9 n5, Sept 1987, pp608-620.

[18] Milroy M, Automation of Laser-Scanner-Based Reverse Engineering, Ph.D. Dissertation, University of Victoria, 1995

[19] Soucy G, Callari FG, Ferrie FP, "Uniform and Complete Surface coverage with a Robot-Mounted Laser Rangefinder", Proceedings of the 1998 IEEE/RSJ International conference on Intelligent robots and Systems, Victoria, B.C., Canada, Oct. 1998, pp1682-1688.

[20] Hymarc 3D Vision Systems, 35 Antares Drive, Nepean, Ontario, Canada, K2E 8B1, URL: http://www.hymarc.com

[21] 3D Scanners Ltd., South Bank Technopark, 90 London Road, London SE1 6LN, United Kingdom, URL: http://www.3dscanners.com

[22] Laser Design Inc., 9401 James Ave. South, Suite 162, Minneapolis, Minnesota, 55431, USA, URL: http://www.laserdesign.com

[23] Mauer J, Bajesy R, "Occlusions as a Guide for Planning the Next View", IEEE Transactions on Pattern Analysis & Machine Intelligence, v15, n5, May 1993, p417-433.

[24] Lim CP, Meng CH, "CMM feature accessibility and Path Generation", International Journal of Production Research, v32 n3 Mar 1994, p597-618.

[25] Canny J, "A Computational Approach to Edge Detection", IEEE Transactions on Pattern Analysis and Machine Intelligence, v PAMI-8, n6, Nov 1986, pp679-698.

[26] Koh J, Suk M, Bhandarkar SM, "A Multilayer Self-Organizing Feature Map for Range Image Segmentation", Neural Networks, v8, n1, 1995, pp67-86.

[27] Worth AJ, Kennedy DN, "Segmentation of magnetic resonance brain images using analogue constraint satisfaction neural networks", Image and Vision Computing, v12, n6, July/Aug 1994, pp345-354.

[28] Marapane SB, Trivedi MM, "Region-Based Stereo Analysis for Robotic Applications", IEEE Transactions, 1991.

[29] Wahl FM, *Digital Image Signal Processing*, Artech House, Boston, 1987.

[30] MATLAB, neural network toolbox 3.0, The Mathworks Inc, 3 Apple Hill Drive, Natick, MA, URL:http://www.mathworks.com

[31] Mason SO, Grun A, Automatic Sensor Placement for Accurate Dimensional Inspection, Computer Vision and Image Understanding v61 n3, May 1995 pp454-467

[32] Tarbox GH, Gottschlich SN, Planning for Complete Sensor Coverage in Inspection, Computer Vision and Image Understanding v61 n1, Jan 1995 pp84-111.

[33] Sobh TM, Owen J, Jaynes C, Dekhil M, Henderson TC, Industrial Inspection and Reverse Engineering, Computer Vision and Image Understanding v61 n3, May 1995 pp468-474

[34] Hakala, DG, Hillyard RC, Malraison PF, Nource GF, Naturals quadrics in mechanical design, SIGGRAPH/81, Seminar: Solid Modeling, Dallas, TX, 1981.

[35] Hoffman R, Jain K, Segmentation &Classification of Range Images, IEEE Transactions on Pattern Analysis & Machine Intelligence, v9 n5, Sept 1987.

[36] Bolle RM, Cooper DB, On Optimally Combining Pieces of Information, with Application to Estimating 3-D Complex-Object Position from Range Data, IEEE Transactions of Pattern Analysis and Machine Intelligence, v PAMI-8, n5, Sept. 1986.

[37] Flynn P J, Jain A K, Surface Classification: Hypothesis Testing and Parameter Estimation, IEEE Computer Vision and Pattern Recognition 1988, pp261-267.

[38] Bolle R M, Sabbah D, Differential geometry applied to least-square error surface approximations, SPIE – Optical and Digital Pattern Recognition, 13-15 January 1987, LA, California, pp117-127.

[39] Faugeras GD, Herbert M, Pauchon E, Segmentation of Range Data into Planar and Quadratic Patches, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1983, pp8-13.

[40] Sorenson H W, *Parameter Estimator, Principles and Problems*, Marcel Dekker Inc., NewYork, 1980.

[41] Bretthorst G L, *Bayesian Spectrum Analysis and Parameter Estimation*, Springer-Verlag, New York, 1988.

[42] Encarnacao J L, Lockermann P C, *Engineering Databases, Connecting Islands of Automation*, Springer-Verlag, New York, 1993.

[43] Weiler K, Edge-Based Data Structures for Solid Modeling in Curved-Surface Environment, IEEE Computer Graphics & Application v5 n1 pp21-40 Jan'85.

[44] Toriya H., Chiyokura H., *3D CAD Principles and Applications*, Springer-Verlag, New York, 1990.

[45] Subrahmanyan S, Wozny M, An overview of automatic feature recognition techniques for computer-aided process planning, Computers in Industry v26 n1 Apr'95 pp1-21.

[46] Vandenbrande JH, Requicha AAG, Spatial Reasoning for theAutomatic Recognition of Machinable Features in Solid Models, IEEE Transactions on Pattern Analysis and Machine Intelligence v15 n12 Dec'93 pp1269-1285.

[47] Schulte M, Weber C, Rainer S, Functional features for design in mechanical engineering, Computers in Industry v23 n1 Nov'93 pp15-24.

[48] Shah JJ, Assessment of Features Technology, Computer-Aided Design v23 n5 Jun'91 pp331-343.

[49] Henderson MR, Chang GJ, Frapp: Automated Feature Recognition and Process Planning From Solid Model Data, ASME Computers in Engineering Conf. San Franciso 88.

[50] Sakurai H, Gossard DC, Shape Feature Recognition From 3D Solid Models, ASME Computers in Engineering Conf. San Franciso 88.

[51] De Floriani L, Bruzzone E, Building a feature-based object description from a boundary model, Computer Aided Design v21 n10 Dec 89 pp602-610.

[52] Joshi S, Chang TC, Graph-based heuristics for recognition of machined features from a 3D solid model, Computer Aided Design v20 n2 Mar'88 pp58-66.

[53] Nezis K, Vosniakos G, "Recognizing 2 1/2D shape features using a neural network and heuristics," Computer-Aided Design, v29, n7, 1997, pp523-539.

[54] Thompson WB, Reverse Engineering of Mechanical Parts, University of Utah web site http://www.cs.utah.edu/projects/robot/sam/sam.html.

[55] Prabhakar S, Henderson MR, Automatic form-feature recognition using neural-network-based techniques on boundary representations of solid models, Computer-Aided Design, v24, n7, July 1992, pp381-393.