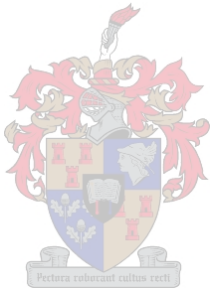
EDGE SCANNING AND SWEPT SURFACE APPROXIMATION IN REVERSE ENGINEERING

Kristiaan Schreve



Dissertation presented for the Degree of Doctor of Philosophy (Mechanical Engineering) at the University of Stellenbosch.

Thesis Supervisor: Prof. A.H. Basson

Department of Mechanical Engineering

University of Stellenbosch

September 2001

Declaration

I the undersigned hereby declare that the work contained in this dissertation is my own original work and has not previously, in its entirety or in part, been submitted at any university for a degree.

Abstract

Broadly speaking Reverse Engineering is the process of digitising a physical object and creating a computer model of the object. If sharp edges formed by two surfaces can be extracted from a point cloud (which is the set of measured points) it can speed up the segmentation of the point cloud and the edges may also be used to construct swept surfaces (or various other types of surface that best captures the design intent).

A strategy is presented to "scan" edges. The strategy simulates a CMM (Coordinate Measurement Machine) as it would scan a sequence of short lines straddling the edge. Rather than measuring on a physical object, the algorithm developed in this dissertation "scans" on the points in the point cloud. Each line is divided in two parts, or line sections, belonging to the surfaces forming the edge. The points of the line sections are then approximated with polynomials. Each edge point is the intersection of two such polynomials. In many engineering components sharp edges are replaced with fillet radii or the edges become worn or damaged. This algorithm is capable of reconstructing the original sharp edge without prior segmentation.

A simple analytical model was developed to determine the theoretically achievable accuracy. This Analytical accuracy was compared with the accuracy of edges extracted from point clouds. A series of experiments were done on point clouds. The input parameters of the experiments were chosen using the technique of Design of Experiments. Using the experimental results the parameters that most significantly influences the accuracy of the algorithm was determined. From the Analytical and experimental analysis guidelines were developed which will help a designer to specify sensible input parameters for the algorithm. With these guidelines it is possible to find an edge with an accuracy comparably with an edge found with the traditional method of finding the edges with NURBS surface intersections.

Finally the algorithm was combined with a swept surface fitting algorithm. The scanned edges are used as rails and profile curves for the swept surfaces. The algorithms were demonstrated by reverse engineering part of another core box for an inlet manifold.

If the edge detection parameters are specified according to the guidelines developed here, this algorithm can successfully detect edges. The maximum gap size in the point cloud is an important limiting factor, but its effect has also been quantified.

Opsomming

In Truwaartse Ingenieurswese word 'n fisiese voorwerp opgemeet en 'n rekenaar model word daarvan geskep. Die segmentering van die puntewolk (dit is die versameling gemete punte) sal aansienlik vergemaklik word indien dit moontlik is om skerp rante in die puntewolk te identifiseer. Die rante sal dan gebruik kan word om veegvlakke (*swept surfaces*), of enige ander tipe oppervalk wat die ontwerp die beste beskryf, te konstrueer.

Hierdie proefskrif beskryf 'n strategie wat die rante kan opmeet. Dit simuleer die manier waarvolgens 'n Koördinaatmeetmasjien 'n reeks lyne, wat oor die rant lê, sou meet. In plaas van op 'n fisiese voorwerp op te meet, "meet" die algoritme op 'n puntwolk. Elke lyn word dan in twee dele verdeel (elke deel word 'n meetlynseksie genoem). Elke meetlynseksie behoort aan een van die twee oppervlaktes wat die rant vorm. Die rant punte word bereken as die interseksie van twee polinome wat deur die punte van die meetlynseksie gepas is. Dit is dikwels die geval met meganiese onderdele dat skerp rante vervang word met 'n vulstraal of dit kan ook gebeur dat die rant verweer het of beskadig is. Die algoritme, wat hier beskryf word, kan selfs die oorspronklike skerp rant in sulke gevalle herkonstrueer.

'n Eenvoudige analitiese model is ontwikkel om die teoretiese akkuraatheid van die algoritme te bepaal. Die teoretiese akkuraatheid is vergelyk met die akkuraatheid van rante wat uit puntewolke bepaal is. 'n Reeks eksperimente is op puntwolke gedoen. Die parameters vir die eksperimente is gekies deur van Eksperimentele Ontwerp gebruik te maak. Met behulp van hierdie tegniek kon bepaal word watter meet-parameters die grootste invloed op die akkuraatheid van die gemete punte het. Die teoretiese en eksperimentele resultate is gebruik om riglyne daar te stel waarmee die intreeparameters van die algoritme gekies kan word. Met hierdie riglyne is dit moontlik om 'n rant te vind met 'n akkuraatheid vergelykbaar met die tradisionele metode om die rante te vind met behulp van NURBS oppervlakte interseksies.

Laastens is die algoritme gekombineer met 'n algoritme wat veegvlakke deur punte kan pas. Die gemete rante word gebruik as spore en profiele vir die veegvlakke. Die

tegnieke is gebruik om 'n CAD model van 'n sandkernvorm (vir die giet van 'n inlaatspruitstuk) te maak.

Deur die riglyne te gebruik om die intreeparameters vir die algoritme te spesifiseer, kan rante suksesvol uit puntewolke bepaal word. Die maksimum afstand tussen naburige punte in die puntewolk beperk die gebruik van die algoritme, maar die effek hiervan is ook vasgevat in die riglyne wat ontwikkel is vir die algoritme.

Dedication

To my parents and family without whom this thesis would not have been possible.

Acknowledgements

The author wishes to express his gratitude towards the following individuals and institutions for contributing towards this study:

- My Heavenly Farther for allowing me to do this project, without whom nothing is possible and to whom all credit is due.
- The National Research Foundation, GCC, University of Stellenbosch Research Committee, Harry Crossley Foundation, Prof. A.H. Basson and my parents for financial assistance.
- Prof. A.H. Basson, my dissertation supervisor, for his valuable advice and criticism during the research for this dissertation.
- My parents for assistance and support from the beginning to the end of this project.
- My fellow students, Andreas, Corné, Liu and Charl for their support and advice, especially Charl for his unselfish help in debugging the computer code.
- Dr. D. Page for introducing me to Design of Experiments.
- Finally my friend, Lourens, for constantly reminding me about the deadlines for this project!

Table of Contents

List of Figures

Chapter 5

List of Tables

Nomenclature

A	[mm]	Scanning amplitude
$A_{optimum}$	[mm]	Recommended scanning amplitude
$\boldsymbol{a}$		Unit direction vector (a_x, a_y, a_z) of a line
a	[mm]	Torus major radius
b	[mm]	Position of torus centre on z-axis
$\boldsymbol{b}_j$		The j'th plane origin (see equation 4.5.4.2.)
$\boldsymbol{c}$		Position (c_x, c_y, c_z) of probe ball centre
$\boldsymbol{c}(u)$		Point at parameter u on the offset curve (equation 5.2.2.9.)
$\boldsymbol{c}_j^k(u)$		Point on the k'th offset curve of the j'th scan line. (See definition of $\boldsymbol{n}_j^k(u)$ for an explanation of the superscript k.)
d	[mm]	Distance
df	[mm]	Length of the section of the scan line scanned on the fillet radius
$\boldsymbol{d}_s$		Scanning direction (Chapter 4, Figure 4)
$\boldsymbol{n}$		Plane normal vector
$\boldsymbol{n}_j^k(u)$		The unit normal vector on the j'th curve at parameter u. The superscript k indicate the line section of the scan line. Each scan line is divided into two scan line section belonging to the two neighbouring surfaces. Therefore its value is either 1 or 2.
$\boldsymbol{n}_s$		Scanning plane normal vector
$\bar{\boldsymbol{p}}$		Analytical edge point (equation 5.2.1.3.)
$\boldsymbol{p}_i$		The i'th scanned point.
$\boldsymbol{p}_{i,j}$		The i'th scanned point on the j'th scan line
$\bar{\boldsymbol{p}}_j$		Edge point of the j'th scan line
$\boldsymbol{p}_j^k(u)$		Point at parameter u on the k'th scan line section of the j'th scan line. (See definition of $\boldsymbol{n}_j^k(u)$ for an explanation of the superscript k.)

$(\boldsymbol{p}_j^k(u))^o$		Point at parameter u on the k'th scan line section of the j'th scan line compensated with the probe radius. (See definition of $\boldsymbol{n}_j^k(u)$ for an explanation of the superscript k.)
R	[mm]	Torus section radius
R_f	[mm]	Fillet radius
R_p	[mm]	Probe ball radius
r	[mm]	Torus section radius
r	[mm]	Radius of curvature (see paragraph 5.2.2.)
$\boldsymbol{S}(u,v)$		Point on a surface at parameters u and v
s	[mm]	Arc length of circle segment (see paragraph 5.2.2.)
$\boldsymbol{s}_j^k(u,v)$		Point on the k'th ruled surface of the j'th edge point. (See definition of $\boldsymbol{n}_j^k(u)$ for an explanation of the superscript k.)
$\boldsymbol{s}_l$		Origin (s_{lx},s_{ly},s_{lz}) of a line
$\boldsymbol{t}$		Edge curve's tangent vector
$\boldsymbol{t}_j$		Edge curve's tangent vector at the j'th edge point
$\boldsymbol{t}(u,v)$		Point on torus surface at parameters u,v
u		Curve or surface parameter
v		Surface parameter
α	[rad]	Angle (Chapter 4, Figure 1)
ε	[mm]	Scanning tolerance
ε_{max}	[mm]	Maximum error
ε_{min}	[mm]	Minimum recommended scanning tolerance (equation 5.3.3.5.1.1.)
$\varepsilon_{analytical}$	[mm]	Analytical error
ϕ	[rad]	Angle between scanning plane normal vector and edge tangent (Chapter 3, Figure 6)
ϕ	[rad]	Half the arc angle of the fillet (Chapter 4, Figure 1)

γ	[rad]	Arc angle (Chapter 5, Figure 1)
κ_e	[mm^{-1}]	Edge curvature
κ_s	[mm^{-1}]	Surface curvature in the direction perpendicular to the edge
λ		Independent line parameter
η	[mm]	Point cloud noise level
θ	[rad]	Intersection angle between two surfaces (Chapter 4, Figure 1)
ρ_c	[mm]	Point cloud pitch
$(\rho_c)_{ideal}$	[mm]	Recommended point cloud pitch
ρ_e	[mm]	Edge pitch
ρ_L	[mm]	Line pitch

Chapter 1.

Introduction

1.1. What is Reverse Engineering?

Broadly speaking Reverse Engineering is the process of digitising a physical object and making an electronic copy of it. Normally the electronic copy is used to make a physical copy of the object. Often, the manufacturing step is not included. In these instances the electronic copy is only used for record keeping, visualisation or geometric input for analysis software such as Finite Element Methods or Computational Fluid Dynamics packages. This thesis is mostly concerned with the making of the electronic copies. Varady et al., 1997, give a good review of the Reverse Engineering process and the related technologies.

There is nothing new about making copies of existing objects. However, Reverse Engineering became a buzzword during the early 1960's (Bidanda and Hosni, 1994). The increasing availability of Computer-Aided Design (CAD) technology, computer hardware, Coordinate Measuring Machines (CMM) and Computer Numerically Controlled (CNC) machining made it cost effective to make copies of objects with really complex geometry. In 1972 David McMurtry developed and patented the first touch trigger probe (Anonymous, 2001b). This made highly accurate computer controlled scanning on standard CNC machines possible.

The Reverse Engineering process starts with digitising the object. Digitising is the generic term for measuring any number of coordinates on the surface of the object. Thus, digitising gives a discrete representation of the object. In rare cases this might be sufficient. The set of coordinates is called a *point cloud* in Reverse Engineering jargon. A typical point cloud can contain millions of coordinates. There is a very wide variety of digitising techniques as will be described in the literature review.

Usually some kind of surface representation of the points must be created so that the object can be visualised on a computer screen and re-manufactured. This can be a very difficult step. The sheer size of a point cloud can create numerous problems. The topology of the points is also seldom known. Some heuristic must be used to determine this. Measuring noise, combining point clouds scanned in different coordinate systems and identifying surface patches in the point cloud are further challenges addressed in recent and ongoing research projects. It is with the latter that this thesis is concerned.

The two most important surface representations in Reverse Engineering are Non-uniform Rational B-splines (NURBS) and STL (Stereolithography). NURBS curves and surfaces are used by most of the surface modelling CAD packages on the market today. Thus, if this format is used, the model can be modified using all surface modelling capabilities of the CAD package. It can, however, be a rather time consuming process. Creating a NURBS model normally involves a surface fitting step. Surface fitting is the process of finding the surface parameters that best represent the point cloud. STL models are a triangulation of the point cloud. Automatic triangulation software is commercially available. Unfortunately it is hard to edit these models and since the triangulation interpolates all the points, noise in the point cloud must be kept to a minimum. The algorithms developed in this thesis is not aimed at STL modelling.

Once a surface representation is available various Computer Aided Manufacturing (CAM) or Rapid Prototyping (RP) processes can be used to manufacture a copy of the original object.

The rest of this chapter looks at some interesting applications of Reverse Engineering. Then a pilot study conducted at the beginning of the project is described. The findings of this study led to the identification of the research needs addressed in this thesis.

1.2. Interesting Applications of Reverse Engineering

A quick glance at the literature reveals many interesting applications of Reverse Engineering. The die and mould industry is probably the most frequent user of Reverse Engineering technology in South Africa, as the author's own experience at

the Global Competitiveness Centre seem to indicate. The pilot study described in the following paragraph is an example of this category.

One misconception that must be removed right away is that Reverse Engineering is only about making copies of someone else's design, similar to making paper copies on a photocopier. A good example to counter this misconception is the design of a rear view mirror for an automobile as reported by Puttré (1994). In order to eliminate vibrations caused by wind on the rear view mirror, the designer experimented with alternative clay models and tested them in a wind tunnel. The final model was Reverse Engineered in order to create the CAD documentation and manufacture the production tooling. Similar examples can be found in Chant et al. (1998) and Chapdelaine (1998).

Reverse Engineering is used for such diverse applications as developing replacement tiles for NASA's space shuttles (Hosni and Ferreira, 1994; Perreault and Ward, 1999), the interior design of the F16 fighter aircraft (Perreault and Ward, 1999) and shoe lasts (Bao et al., 1994; Danckaerts and Yudhira, 1997; Schneider, 2001). Reverse engineering the classic teardrop-shaped fuel tank of a Harley-Davidson is a recent project (Anonymous, 2000). Reverse Engineering is used to manufacture replacement parts if it cannot be obtained from the original supplier for whatever reason (Metwalli, et al., 1999; Hegazi and Metwalli, 1999). The human form is also often reverse engineered with applications in the garment industry (Au and Yuen, 1999), the medical profession (e.g. Liu and Ma, 1999), and artistic sculpturing (e.g. Ip and Loftus, 1996).

As can be seen from the above, Reverse Engineering technology is certainly not limited to the engineering profession. Some of the more imaginative applications are found in the fields of archaeology and anthropology. Archaeologists use the technology to scan broken artefacts and then they try to use intelligent software to connect all the pieces and rebuild the original object, almost like building a puzzle (Üçoluk and Toroslu, 1999). Anthropologists use Reverse Engineering technology to digitise human remains since they are under increasing pressure to return the remains to the communities they belong to (Puttré, 1994).

Perhaps the example that will capture the imagination of most people, especially mothers and fathers to be, is creating a 3D image of an unborn baby from ultrasound scans of the baby in the mother's womb (Anonymous, 2001a).

1.3. Pilot Study

The die and mould industry mostly need complete NURBS surface models. Creating such a model from a point cloud can be a very time consuming process. A pilot study was undertaken to understand the difficulties inherent in the process and to identify areas where time savings can be made by implementing intelligent techniques to speed up the process.

1.3.1. Selection of a Product

A core box was chosen as a typical example of a product in the automotive industry. It is used to make sand cores for an IC inlet manifold.

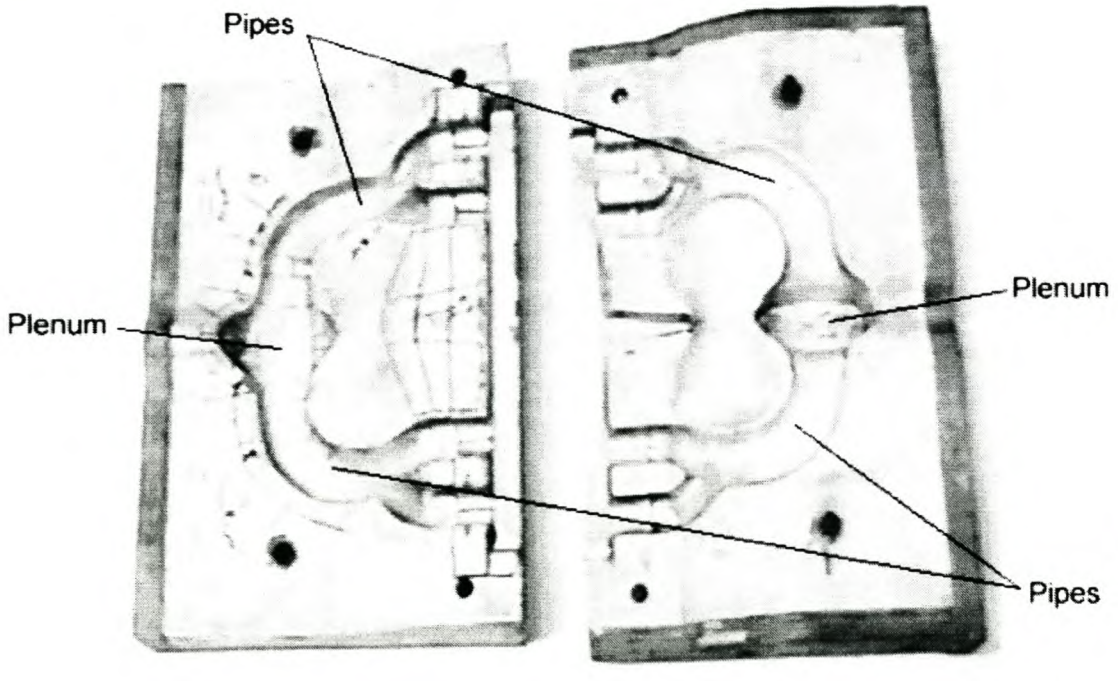


Figure 1 Core Box (Bottom Core Box on the Right).

1.3.2. Reverse Engineering Facilities

The Global Competitiveness Centre (GCC), at the University of Stellenbosch, uses a Mitutoyo Bright 710 CMM with a Renishaw PH10M tactile probe. Scanning and manipulation of the surface data are done with the METRIS software. Briefly, reverse engineering is done as follows with the METRIS software. A point cloud is measured with SURFEYOR. The cloud is manipulated in SHAPID. Then the cloud is approximated with a base surface. Finally a least squares approximation of the surface is done to obtain a NURBS surface representing the data.

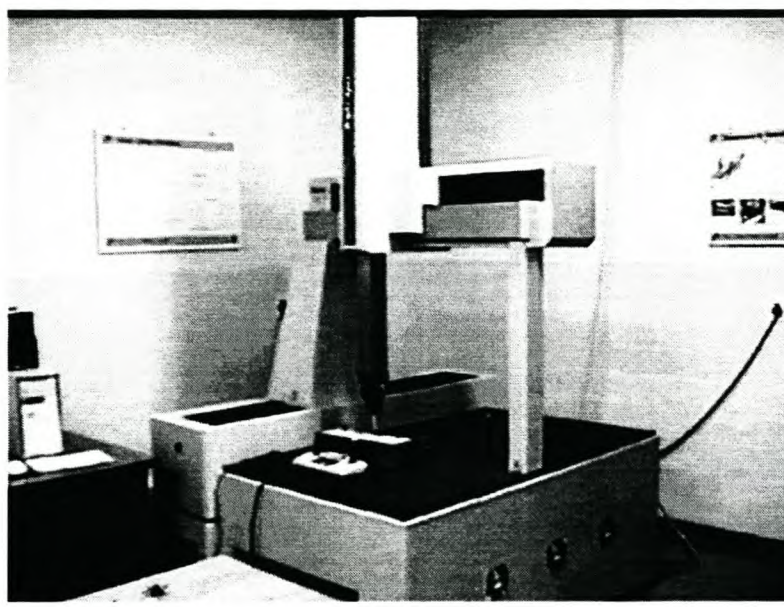


Figure 2 The Mitutoyo CMM at the GCC.

1.3.3. Time Study Results

A complete summary of the time study can be found in Appendix B. It took 44.43 hours to reverse engineer the bottom core box and 32.02 for the top core box. This is the operator hours. Some of the measuring and modelling operations were done simultaneously and therefore the total time is not simply the sum of the measuring and modelling time. A summary of the measuring and modelling time of the two core boxes is given in Table 1.

Table 1 Summary of Reverse Engineering the Core Box.

	Bottom Core Box	Top Core Box
Measuring	26.9 h	28.4 h
CAD Modelling	30.8 h	9.1 h
Actual Total	44.43 h	32.02 h

The difference in the modelling time is largely due to the fact that the modelling of the split plane is included in the time for the bottom core box. The modelling of the split plane took 12 hours 51 minutes. The learning curve also contributes to the difference. The reason the split plane took so long is that it was necessary to accurately define the edges of the split plane. It is best that this process be described in some detail.

Surfaces were fitted to different point cloud patches, for example the pipe sections, the plenum chamber or the split plane. There was a gap between these point cloud patches due to the manual subdivision process. This gap can be fairly large, in this case a few millimetres, due to such factors as the roughness of the scan or irregularities near the edge. The edges were then found by extending the surfaces and finding their intersections. Extending NURBS surfaces can cause some very erratic behaviour of the surface as shown in the figure below. Considerable time goes into trying to reduce the gap to improve the result of the surface extension. This figure shows a surface on the right and its extension on the left. It was extended using AutoCAD (2000). The extension is done over a longer than normal distance to highlight the problem of surface extensions. Note the waviness that appears, and increases, towards the end of the extension.

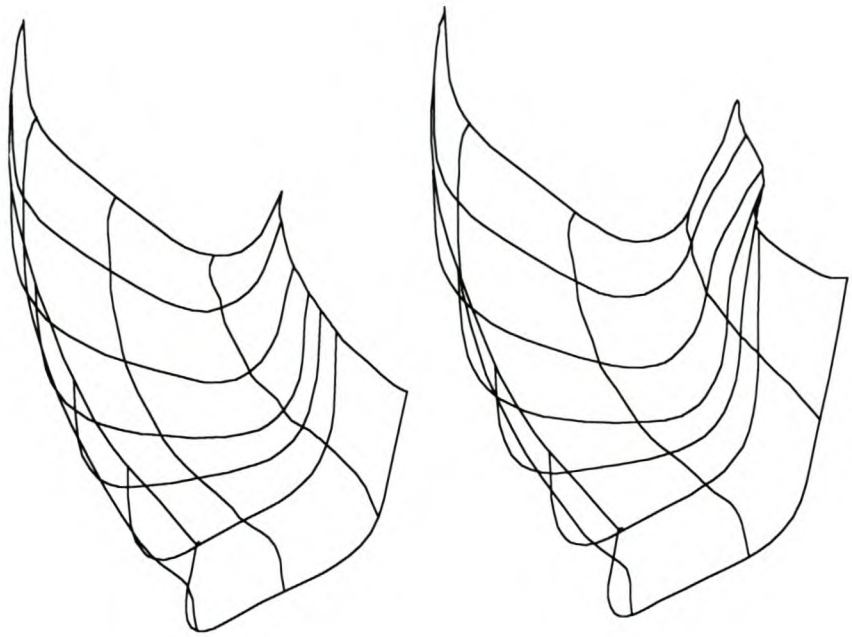


Figure 3 Surface Extension.

Since the two parts are largely similar, many of the modelling difficulties were sorted out while modelling the bottom core box. It was more difficult to model the plenum chamber on the bottom core box due to the deep hole and the flat surface at the bottom of the chamber. It took 5 hours 54 minutes to model the plenum on the bottom core box and only 43 minutes for the plenum part on the top core box. It must be stated that a complete, machine ready, CAD model was not generated.

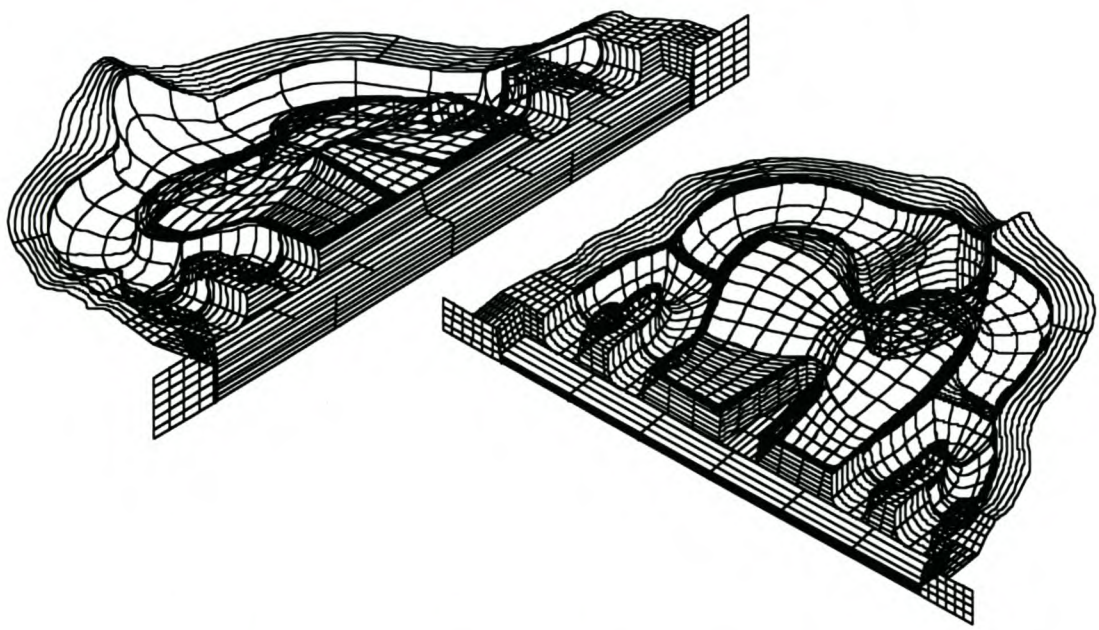


Figure 4 CAD Drawing of the Reverse Engineered Core Box.

1.3.4. Conclusions

The first question to ask is whether the time study results are representative of industry practice. The core box was also reverse engineered by a South African toolmaker. They also have a CMM with a touch trigger probe and reverse engineered the core box to about the same level of detail as was done at the GCC. The toolmaker reported a total time 52 hours (measuring and modelling) for the bottom core box and 34.5 hours for the top core box. This reflected the same trend as the result of the GCC time study.

The times are also of the same order of magnitude. Two time studies are also reported in the literature, but it is hard to compare the results from literature with this time study. So many factors influence the result, such as the level of detail of the modelling, experience of the designers and CMM operators, the software used, number of free-form surfaces, etc. Milroy et al. (1996) report the reverse engineering of a water timer housing consisting of 12 surface patches. The total modelling time reported is 12 hours, i.e. 1 hour per patch. They did a complete CSG (Constructive Solid Geometry) model from the point cloud. Rolls et al. (1999) report the reverse engineering of an automotive bracket consisting of 77 surfaces patches. The total modelling time is 38 hours, i.e. almost 30 minutes per patch. The core box reverse engineered by the GCC has 176 surface patches and took 39.9 hours to model, i.e. almost 15 minutes per patch. Clearly the GCC times are by far the fastest, but too much should not be read into this. As already stated there are so many factors that play a role in the total modelling time, it is just not fair to blindly compare the time study results. It does, however, seem that the times reported in the pilot study is comparable with the findings of other researchers.

Obviously big improvements can be made in the scanning time. The CMM with a touch trigger tactile probe is possibly the slowest method of scanning an object. However, the scanning time is not addressed in this thesis since there are many devices, such as laser scanners that can do the job much faster.

The difficulties with modelling the split plane are clearly illustrated in the time study results. As it is shown, the problem is related to finding the intersection between the

split plane and the neighbouring surfaces. An edge detection function will be very useful in these cases.

Another important point to note is that a lot of the design intent is lost in the Reverse Engineering process. Surfaces that were originally modelled as swept surfaces, extrusions, etc. are now all NURBS surfaces. It is therefore rather difficult to make changes to the design if that is required.

1.4. Research Objectives and Motivation

The main objective of this project is to address the need for an accurate tool that can be used to detect edges in an unstructured point cloud. It cannot be expected that the tool must conjure an edge that is more accurate than the accuracy of the points in the point cloud. Therefore, the goal is to develop a tool that can detect an edge with an accuracy of the same order of magnitude as the accuracy of the points in the point cloud. Of course, the tool must also be easy to use and it must give results quickly.

The detected edges can be used in the modelling of swept surfaces. Integration of the edge scanning algorithm with a swept surface approximation algorithm is presented and demonstrated in a case study.

1.5. Thesis Outline

Since improving the Reverse Engineering time is one of the main themes of this work, the literature review will look at the different scanning methods. Naturally, it will also look at what other authors did about the edge detection problem.

Chapters 3, 4 and 5 are the main body of this thesis. They describe the edge detection algorithm. Chapter 3 describes a virtual CMM that is used to "scan" the edge. Chapter 4 describes the various aspects of the algorithm and Chapter 5 describes the analysis of the method.

Various aspects of NURBS surface construction related to Reverse Engineering are discussed in Chapter 6. The chapter discusses NURBS surface fitting, surface extensions, the calculation of NURBS surface intersections and finally the sweep

surface fitting algorithm that is combined with the edge detection algorithm is also described.

All the tools described in chapters 3 to 6 are combined in a simple computer program. This is discussed in Chapter 7. Another case study is used to illustrate the practical use of the tools.

The thesis ends with the usual conclusions.

Chapter 2.

Literature Review

2.1. Introduction

The purpose of this chapter is not to give an in depth account of the various edge detection methods, but rather to introduce the alternative approaches. It is to raise the different issues involved in edge detection. In the process a survey of the published methods is done. In order to appreciate the intricacies of the proposed method, it is necessary to understand the nature of the data used by edge detection algorithms. Therefore a brief description of some of the most important digitising techniques is given.

2.2. Digitising Techniques

Various techniques have been developed to sample points on the surface of physical objects. On one end of the spectrum there are techniques that sample one view of an object almost instantaneously. The other end of the spectrum is characterised by techniques that laboriously sample one point at a time. The latter must be combined with a scanning strategy that will guide the digitiser/probe over the entire unknown surface. The scanning methods are normally classified as either contact scanners or non-contact scanners. (Várady et al. (1997) and Rolls et al. (1999) present brief, but comprehensive, reviews of digitising techniques.)

2.2.1. Point Scanners

Point scanners are probably the most important digitising devices for Reverse Engineering because of their versatility and accuracy. Unfortunately, the advantages often come at the expense of speed and cost. Tactile and laser probes belong to this family. All these probes are held in a CMM (Coordinate Measuring Machine). A CMM can take the form of either the traditional gantry system or an articulated arm.

2.2.1.1. Laser Point Probe

Laser point scanners, such as the Hyscan 45C, measure the range by observing the reflection of a laser spot on the object surface. This is schematically illustrated in the figure below. The laser beam is focussed on the object surface and the range is determined by the angular position of the rotating, two-sided mirror and the location of the imaged spot on the photosensitive array (Milroy et al. 1996). Rüther and Craigie (1998) developed an alternative laser based system at the University of Cape Town. Their system uses three CCD cameras to determine the position of the laser spot on the object's surface. Their system's accuracy is 0.2mm compared to the reported accuracy of 0.1mm by Milroy et al. (1996). Yau et al. (2000) use two CCD cameras to locate the laser spot.

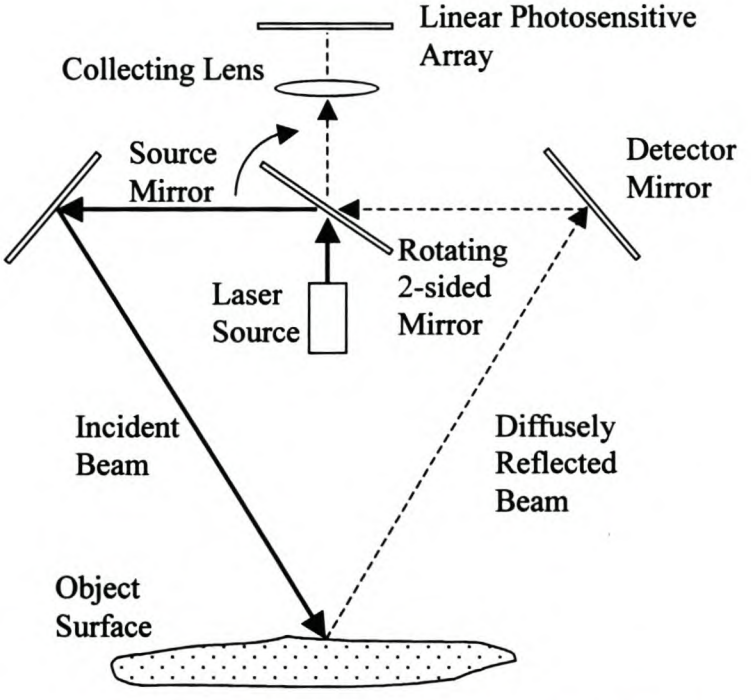


Figure 1 3D Range Sensing (Milroy et al. 1996).

The accuracy of systems such as Milroy's, depend on the focal length, i.e. the distance from the probe to the object's surface. They report an accuracy of 0.1mm at 100mm. Reported accuracies range from 0.05mm to 0.254mm for focal lengths from 40mm to 120mm (Hosni and Ferreira, 1994; Milroy et al., 1996 and Ebenstein at al., 1999). The focal length restricts the depth of holes that can be measured.

Ebenstein et al. (1999) argue that even though points scanned with a laser scanner sometimes have high noise values, high accuracy results can still be obtained because the noisy points can be filtered out. They suggest that after an initial entity approximation, points having residuals of more than 2.4 standard deviations must be discarded. With their procedure they were able to measure spheres to within 0.02mm. Of course, this can only be done if the noise is Guassian. They also show that points scanned on a sphere with a laser scanner have larger errors near the North Pole (due to specular reflection) and the equator (due to too little diffuse reflection). Smith and Zheng (1998) analysed the nature of these errors in detail. While the assertion of Ebenstein et al. (1999) is valid if primitive analytic surfaces are scanned, such a method can produce incorrect results if a NURBS surface is fitted to the data. NURBS surfaces adapts itself to the local nature of the point cloud during the fitting process. Therefore NURBS surfaces will approximate any structured error.

It is often necessary to reorientate parts during scanning in order to capture the entire object. Most laser point scanners are not truly 3D, but in fact only 2.5D. There is a time and risk penalty with every reorienation. It can also be very difficult to integrate the point clouds if accurate reference points cannot be measured, a problem that led Yau et al. (2000) to investigate the registration of multiple point clouds. If a point cloud has any structure, it is lost through such reorientations. Milroy et al. (1996) discuss the scanning of a water timer housing requiring scanning from five views. They scanned 220000 points at a rate of 2000 points per second, yet it took 90 minutes to complete the scanning. It clearly shows that more time on a laser point scanner can be spent at set up and reorienation than at the actual scanning.

As the angle between the surface normal and the laser beam, the incidence angle, becomes smaller, the diffuse reflection of the beam becomes weaker. Ebenstein et al. (1999) show that the errors increase under these conditions. This means that even if the entire object can be viewed in one view, it may be necessary to reorientate the part to scan surface areas where the angle of incidence is high. Laser scanners cannot measure vertical walls. It is necessary to reorientate the object to measure these surfaces. Scanning regions close to vertical walls can also be difficult, because the wall obscures the reflection of the laser beam.

2.2.1.2. Tactile Probes

Due to the high accuracy achievable with tactile probes, and their versatility, tactile probes are used very often in coordinate metrology and Reverse Engineering. Accuracies of 0.001mm and lower are reported (Weckenmann and Knauer, 1998). With a rotateable probe, occluded regions can be measured and it is not sensitive to the inclination angle, such as is the case with laser probes. Deeper holes can also be measured than with laser probes (Puttré, 1994). They are also insensitive to surface effects such as colour and texture. Of course, the object must be able to withstand the contact force exerted by the tactile probe. This eliminates soft clay models from being measured with tactile probes.

Puttré (1994) reports that tactile probes can be significantly cheaper than laser probes, although it is very hard to compare the cost of the probes, since it depends on so many aspects.

The smallest feature that can be measured with a tactile probe is primarily limited by the size of the probe ball. Another consideration is the method of placing the probe. With manual placing, as is the case with articulated arms, the smallest feature that can effectively be measured is 0.8mm (Raab, 1994).

Most tactile probes work like an electrical switch. It rests on three, or more, contact points. As soon as the probe ball makes contact with the object, the probe, which is held down with a spring, deflects and at least one contact is broken. As soon as the probe deflects the CMM will stop. This leads to a unique source of error, the probe lobing error (Shen and Springer, 1998). The force needed to lift the probe off a contact depends on the direction in which the probe is moving. This causes a directional error. The deflection of the probe stem may also be a source of error. However, compared to laser scanners, the errors of tactile probes are well understood and very predictable. It depends mostly on the probe and CMM itself and not on the characteristics of the object, such is the case with laser scanners.

Tactile probes of the touch trigger variety are very slow. It seems to be seldom used for complete surface scans. If used for scanning, the point cloud is as sparse as possible for economic reasons. Analogue probes are much faster, though not as fast as laser scanners, but this comes with an accuracy penalty. (The Renishaw Cyclone used

here is accurate within 0.05mm.) Much denser sampling is possible at a fraction of the cost of scanning with a touch trigger probe.

The measured points must be compensated with the probe radius. This can be a source of error and difficulty, especially if the points are randomly distributed.

2.2.1.3. Digitising Techniques

Bradley and Chan (2001) argue that digitising time is one of the most important limitations in Reverse Engineering. Surface scanning strategies have automated the process, but often sampling still happens at about one second per point. This is not satisfactory.

Regular grid strategies, either Cartesian or polar, are most often used to digitise surfaces with point scanners. Danckaerts and Yudhira (1997) developed such a strategy for a laser scanner. They stated that at the time of their publication, surface scanning strategies were very rare in commercial software. The biggest challenge to developing such a strategy is to safely steer the probe over the unknown topology of the surface using only the already measured points to prevent a collision (unintentional contact) between the probe and the object.

It is important to select a strategy that covers all the features of the surface with sufficient points in the minimum time. This consideration led Janssens (1998) to develop a curvature based scanning strategy. After an initial grid scan the points are triangulated and curvatures are approximated with height deviations. In regions of high curvature, more points are measured until the density is sufficient.

Song and Kim (1997) developed a similar strategy. They also measure an initial grid and then refine it based on curvature until sufficient density is obtained. They calculate the centroid of each triangle and then try to measure it. If the deviation from the expected point is more than a specified tolerance, the refinement will continue. Janssens (1998) does not measure the centroid, but rather measures a point in the middle of each side of the triangle and he then calculates the deviation from the expected points to check if the refinement must continue. Janssens (1998) also checks the length of the sides of the triangles. An additional tolerance is specified for the length of the triangles' sides. It seems that the two strategies were developed

independently at about the same time. No comparative study was found in the literature. It might seem that by measuring three points on each triangle instead of just one in the middle that Janssens's method is less efficient, but this thorough measurement seems to make it more robust under local surface conditions, see Kruth, et al. (1997). It may also converge faster. Song and Kim (1997) do not provide a similarly thorough analysis of their method's robustness.

Chen and Lin (1997) also developed a curvature based method such as the two described above. However, they start directly to refine the triangular mesh after measuring a boundary. No initial grid scan is done. They use a rough bi-cubic B-spline surface approximation to determine the scanning path and estimates of the next points. As Song and Kim (1997) they also measure the geometric centre of the triangular patches. Yau (1997) also use a B-spline surface approximation to determine the scanning path. He starts with a grid and refines the surface approximation with new scanned points until the approximation is within the desired limits.

Both these methods provide a very thorough scan of a free-form surface. It can also be a very economical scanning method. Timesavings are made by measuring fewer points in reasonably flat areas. The CMM is then free to spend more time refining intricate parts of the surface. It must be noted that point clouds scanned in this manner are no longer structured in regular grids.

In some instances the CMM may spend too much time refining insignificant detail. These surface patches will be cut from the point cloud after measuring and NURBS surface approximation. For example, fillet radii can be added in the CAD software, without doing another time-consuming surface fit. In such cases it is unnecessary that so much time is spent on refining a fillet radius. The same is true for sharp edges.

2.2.2. Line Scanners

For surface scans, Bradley and Chan (2001) feel that laser line scanners are the best compromise between speed and accuracy.

Rather than sampling one point at a time such as laser point scanners discussed earlier, laser line scanners samples a line of points at once. A laser line is projected onto the surface. This line moves a few times up and down the surface, rather like a

paintbrush, until the entire surface is covered with points. This makes laser line scanners a few orders of magnitude faster than point scanners.

Accuracy is not sacrificed in the process. Bradley and Chan (2001) report accuracies similar to laser point scanners (±0.025mm over a 40mm depth of field).

Line scanners work on the same principles as a laser point scanner. Therefore the same limitations and advantages apply as already discussed for laser point scanners.

2.2.3. Machine Vision Techniques

There are a number of digitising techniques that can be described under the generic heading of Machine Vision. These techniques all take one, or more, image of an object and use some method of assigning depth information to each pixel in the image. These techniques are fast and often do not require expensive sensors.

2.2.3.1. Automatic Analysis of Silhouette Images

Huang and Motavalli (1994) mounted an object on a light table on a CNC machine bed. A CDD (Charge Coupled Device) camera is secured in the tool holder of the CNC machine. As the object traverses below the camera it takes small pictures (50×50mm) of the object. The back lighting of the light table provides a good contrast between the object and the environment. This is used to automatically find the profile of the object. The reported accuracy is 0.254mm.

Armstrong and Adonis (2000) extended the idea to three dimensions. They mounted the CCD camera on a gantry system so that it can accurately rotate around the object. Any number of images of the object can then be taken. Each image provides a boundary profile of the object. Using the principle of occlusion, the boundaries are combined to obtain a 3D image of the object. Obviously it is not possible to measure doubly concave surfaces in this way. They report an accuracy of 0.05mm.

2.2.3.2. Shape from Shading

Shape from Shading receives frequent attention in the literature. With standard photographic equipment and a PC, very good reproductions of physical objects can be

made. Shape form Shading is basically the reverse of the rendering algorithms used in CAD software.

Ip and Loftus (1996) tried the lighting models of Lambert and Phong to determine the local surface normal at every point in the image. The surface normal vectors are then integrated to determine the depth of every pixel. They mention that the main problem with the method is that accurate illumination parameters are needed for every type of material and light source. There are quite a few parameters that must be determined. They used a trial and error method to find good values for these parameters. Their investigation shows that there is a visible discrepancy between the original object and the reverse engineered object. Peng and Loftus (1998a) report accuracy results when using the Phong model. The average error varied from 2.13% to 6.31% for the case studies they report. They aimed for a good machining dimensional tolerance of less than 0.3%.

Ip and Hou (1999) describe the reverse engineering of a computer mouse using Phong's illumination model. The average error is 2.15mm, which they grant is a bit high. They ascribe the error to ambient light effects.

Peng and Loftus (1998b and 2001) found better results with a modified Torrance-Sparrow illumination model. Their main contribution is that they use a neural network to obtain the illumination parameters. The neural network uses an object of known geometry, e.g. a sphere, to determine the illumination parameters of the material and light sources.

2.2.3.3. CT Scanning

Menon et al. (1997) and Liu and Ma (1999) use ultrasonic scanning to digitise an object. This process is often referred to as CT scanning, i.e. Computer Tomography. Tomography refers to the cross sectional slices that are obtained during ultrasonic or X-ray scanning. It is possible to scan internal geometry in this way. However, the result is not very accurate. It suffers from speckle noise generated by the ultrasonic scans. Nonetheless, the method has important bio-medical applications. Liu and Ma (1999) state that medical CT scanners have an accuracy in the order of a few tenths of a millimetre. They further claim that there are industrial CT scanners with accuracies that can compete with CMMs and laser scanners.

2.2.3.4. Moiré Contours

Moiré methods are very old, first described in 1948 according to Del Taglia et al. (1995). Simply put, a fringe pattern is projected on an object and the depth information is obtained by counting the light and dark fringes. Del Taglia et al. (1995) used a laser beam to generate the fringe pattern and a CCD camera to capture the image. All the equipment is standard, off the shelf equipment. This is maybe why the accuracy is only 0.1mm. Problems with automatic fringe analysis and poor fringe resolution also contribute to the error. It is further important to note that the Moiré method works best if the surface has a white matt finish. Wykes and Morshedizadeh (1995) improved the accuracy to ±0.01mm. They decreased the noise in the fringe projection system by using better equipment.

The Moiré method, as in fact all the Machine Vision methods, suffer from the problem of integrating multiple scans. Jun et al. (2001) addresses the problem by minimising the square of the distance between point pairs. This is done at the loss of the grid structure of the point cloud.

Bradley and Chan (2001) point out that there are often height discontinuities in the scanned image, because one or more fringes may be hidden making the fringe analysis difficult.

Another problem of this, and some other Machine Vision methods, is that of uniform density. If measurements from only one view are taken, the density of the point cloud varies with the angle between the local surface normal and the line of sight. These techniques are analogues to projecting a regular grid of points onto a surface. In areas where there is a large angle between the surface normal and the line of sight, or projection direction, the density will be very low.

2.2.3.5. Photogrammetry

The intersection of rays can be used to determine depth values for stereo images. Only two images are needed to measure one view of an object. However, the initial camera calibration will require several redundant images. After this is done, the calibration only needs to be updated periodically. The digitisation of the object can then continue with only two images from different camera angles. Initial user interaction is required

to define reference points in the images. Some tie points will also have to be measured with some other device (e.g. a CMM) to provide scale and referencing with respect to the coordinate system (Bonitz and Krzystek, 1997).

2.2.4. Hybrid Scanning Techniques

Speed, accuracy and cost are the important efficiency metrics used when comparing digitising techniques for reverse engineering. None of the above mentioned methods is a clear winner in all three aspects. The CMM with a touch trigger probe is by far the most accurate, but speed and cost are sacrificed. All the rest are attempts at faster scans, but this is achieved at the cost of accuracy. A number of researchers tried to avoid this seemingly inevitable compromise by using a hybrid scanning method. They normally do a rough scan with one of the faster techniques and then a CMM is used to take critical measurements.

Suzuki and Aoyama (1997) used a CCD camera to take three orthographic views of an object. This is used to create a rough model of the object. Once the geometry is known, the CMM scanning path can be optimised. There are quite a few studies on finding optimum scanning (or rather inspection) paths for an object of known geometry, for example those reported by Kim and Kim (1996), ElKott et al. (1999) and Lin and Chen (2001). The method of Suzuki and Aoyama (1997) will run into trouble with geometry that is hidden in all three orthographic views. Motavalli et al. (1998) try to improve the situation by taking five orthographic views. Still, doubly concave surface patches may create problems. Chan et al. (2001) solve this problem by taking stereo images with a CCD camera and thus creating real 3D images of the object. They also do not limit the number of views of the object. A very good 3D model of the object is then used for the CMM inspection path planning. Deep holes, causing shadow effects, are the only features that may not be detectable with the CCD camera. Any other part of the object that is still not detected with the CCD camera will in all likelihood not be measurable with a CMM.

Combining the scans from different devices may be a problem unless each device can measure accurate reference points. Often this is not possible because the accuracies of the two systems can differ by one or two orders of magnitude, e.g. a CMM and any of the Machine Vision techniques. Rolls et al. (1999) investigated the problem of

combining CMM data with laser scanned data. Although the two systems complement each other well in terms of accuracy and speed, there is a loss of accuracy as soon as the data sets are combined. In other words, the whole point of augmenting the accuracy with CMM data is lost. They tried using reference spheres to determine the transformation matrices, but found that the sphere fitting results from laser scanned data often is not accurate enough. A least squares approach such as presented by Jun et al. (2001) may address the problem.

Bradley and Chan (2001) use a laser line scanner to do a surface scan. A CMM, in manual operating mode, is used to digitise the boundaries of the surfaces patches. Their approach is aimed at fast and accurate segmentation of point clouds.

2.3. The Case for Edge Detection

There is a lively debate about edge detection in the research community. Researchers in the fields of Reverse Engineering and also Machine Vision are the principle participants. The debate revolves around whether a face-based or edge-based (explained in the following paragraph) technique should be used to segment a point cloud. There is a strong case for both methods. The fact that as yet there is no clear winner in this debate is probably due to the fact that nobody was able to present a segmentation technique that can solve all the problems the research community are faced with. In the following paragraphs the difference between edge-based and face-based methods is described and then the main arguments of the debate are presented.

2.3.1. Edge-based and Face-based Edge Detection

Várady et al. (1997) classified edge detection methods in two basic categories: edge-based methods and face-based methods. Many researchers use this distinction.

Researchers that follow an edge-based approach segment a point cloud by first finding the patch boundaries and then all the internal points that belong to the patch are selected.

The face-based approach is basically the inverse of the edge-based approach. All the points that belong to a patch are selected. Some surface definition is then assigned to

the patch, e.g. plane, cylinder or B-spline. The patch boundaries are then the intersections of these surfaces.

2.3.2. Amount of Information Used in Segmentation

Point clouds are only a discrete representation of the geometry of an object. It has no topology. It seems obvious under these circumstances that all the points must be used so that the best possible definition of the edge can be found. Yet, Várady et al. (1997) point out that with the edge-based approach the cloud is segmented by only using the edge points. The problem remains to find the points that belong to a patch once the edge is known. Since the point cloud does not have any topology, there is no clear cut way of knowing which points lie "inside" the boundary. Proponents of the edge-based approach, e.g. Milroy et al. (1997), circumvent the problem by constructing a wire frame model of the cloud, thus assigning topology to the points, before they do the segmentation. This is a good solution provided that a unique wire frame can be constructed. However, constructing a wire frame, or doing a triangulation of the point cloud, is a very complex problem as the recent literature on the matter indicates (Hoppe, et al., 1992; Edelsbrunner and Mücke, 1994; Choi, et al., 1998; Bernardini, et al., 1999 and Wang and Chen, 1999).

2.3.3. Face-based Methods Yield Edge and Surface Definition

Várady et al. (1997) and Besl and Jain (1988), both advocates of the face-based approach, argue that face-based methods do not only provide the edge information, but they simultaneously give the surface definition of the patch. The surface definition and edge definition are closely linked. This is a good approach if the correct surface model is used. For example, the way to find the intersection between a cylinder and a plane is first to fit the entities and then calculate their intersections. A smooth edge, consistent with the rest of the model is obtained. On the other hand, an edge-based method may yield a rather erratic edge that is not consistent with the definition of the plane or cylinder.

However, Várady et al. (1997) and Peng and Loftus (1998a) point out that this approach forces a specific surface model on the object that is not necessarily appropriate. Furthermore, most face-based methods work only with simple algebraic

surface models, e.g. cylinders, planes, quadratic polynomial surfaces, etc. It is simply impractical to allow any surface model for the segmentation. However, many engineering surfaces require a much more complex surface definition than one of the above mentioned. Várady et al. (1997) actually argue for the use of face-based methods if the surface definition is known to be one of the simple algebraic surfaces, but their opinion is that a combination of face-based and edge-based techniques are better if there are free-from surfaces in the point cloud. They point out that blindly applying a face-based method means that the surface representation, albeit accurate, may not be functional from an engineering point of view.

2.3.4. Face-based Methods do not Work for Free-form Surfaces

Quite a number of researchers argue that face-based methods do not work well for free-form surfaces. Members of this camp are Fan et al. (1987), Várady et al. (1997), Milroy et al. (1997), Fitzgibbon et al. (1997), Horváth and Vergeest (1998) and Yang and Lee (1999). Most face-based methods use a region growing strategy to find the surface that represents the object. Fitzgibbon et al. (1997) argue that region growing strategies often do not work well for free-form surfaces. They propose a hybrid method. First they do a rough edge-based segmentation based on curvature estimates for each point. Then the patches are approximated with surfaces and, where applicable, the surfaces are merged.

Fan et al. (1987) say that region growing methods require very complex control algorithms. Normally some statistical goodness-of-fit testing is done to decide whether or not to include a point in the patch. The algorithm must also know when to stop growing. Variable order growing strategies such as those of Besl and Jain (1988) and Taubin (1991) also have to determine the order of the surface patch. Not only are the control algorithms very complex, but it also is very difficult to determine the control parameters. The parameters normally are tolerances that are used in the statistical testing. Park and Yun (2001), doing a rough edge-based segmentation based on curvatures estimates, have the same difficulty. They say that it is mainly a problem of finding control parameters that work for the general case. Good parameters for one example may not be applicable in another case.

Besl and Jain (1988) disagree with this view. They say that their method works fine for free-form surfaces. It must be noted, however, that an algebraic polynomial surface, of no more than degree four, is used to represent free-form surfaces in their case.

2.3.5. Computation Time

Computation time is a highly contentious issue with representatives of both sides of the divide arguing that the other approach is computationally more time consuming. See for example Bradley and Chan (2001) who argue against edge-based methods and Peng and Loftus (1998a) who argue against face-based methods, both on the same ground. Actually this is a futile argument since computation times reported in the literature are first of all hard to compare, because it is so hardware dependent. It is also a function of how optimised the code is. Further, it depends on the details of the specific method.

2.3.6. Edge-based Specific Problems

The edge based methods reported in the literature (Chen and Liu (1997), Horváth and Vergeest (1998), Liu and Ma (1999) and Yang and Lee (1999)) often use gradient or curvature estimates to segment the point cloud. Some method is used to estimate a gradient, or curvature, for each point in the cloud and then the points of gradient, or curvature, extrema are labelled as edge points. Trucco and Fisher (1995), Várady et al. (1997), Liu and Ma (1999) and Yang and Lee (1999) all point out that curvature estimates calculated directly from the raw data are highly noise sensitive. Várady et al. (1997) say that near the edges the effect is even worse due to sensor effects, especially if the data is scanned with a laser scanner. Liu and Ma (1999) further say that the curvature estimations of discrete data depend on the specific formula used and they thus raise the question whether it is at all possible to have a unique solution. The problems of finding good curvature estimates also make it very difficult to find smooth edges. Furthermore, Fitzgibbon et al. (1997) point out that it often leads to very erratic edges.

A further problem of some edge-based techniques is connecting the edge points if the segmentation is done in the manner described above. Milroy et al. (1997) used an

active contour, which behaves in the manner of an expanding snake moving over the point cloud, to connect the edge points. Yang and Lee (1999) use a special edge-neighbourhood chain-coding algorithm to do the same.

Park and Jun (2000) argue against edge-based methods because of the amount of user interaction required in most reported methods. However, their attempt at an automatic segmentation method is only designed for primitive analytic surfaces.

2.3.7. Summary

What then must be learned from this debate?

The debate describes the nature of the problem of point cloud segmentation very well. The problems and pitfalls are well highlighted. It is, however, hard to determine a real winner of the contest.

Maybe Várady et al. (1997) have the most balanced view in this author's opinion. As already stated, they argue that face-based methods work really well if the surface model is known beforehand as is the case with primitive analytic surfaces. They state that for Reverse Engineering it might be necessary to capture the *design intent* of the object and in this case, blindly applying a face-based method is wrong. They argue that both methods must be available in a good Reverse Engineering system.

2.4. Edge Detection in Reverse Engineering

There is a vast number of segmentation methods coming from the Machine Vision community, some of which have been referred to in the previous section. However, the nature of point clouds in the field of Machine Vision is such that they are normally in a regular grid and that they are intersected only once by the line of vision. These are rather severe simplifying assumptions for Reverse Engineering where the point clouds are often unstructured (structured clouds are often combined resulting in an unstructured cloud) and the clouds often define a closed volume of complex geometry. Therefore Chen and Liu (1997) and Bradley and Chan (2001) advocate methods specially developed for the Reverse Engineering community. In this section some of the attempts at segmentation and edge detection in Reverse Engineering are presented.

2.4.1. Face-based Methods for Reverse Engineering

In the previous paragraph arguments are presented against the use of face-based segmentation methods for free-form surfaces. In practice, it seems that most successful systems do use a face-based method. Most practical systems follow the approach of the SURFEYOR system reported by Kruth et al. (1997) and Janssens (1998). In this system the user must cut the point cloud using the computer's mouse. This is done after the point cloud is triangulated. Other researchers using this approach are Sinha and Seneviratne (1996), Zhao et al. (1997) and Motavalli et al. (1998). After the cloud is segmented, B-spline (or NURBS) surfaces are normally fitted to the patches. The surfaces are then extended and trimmed if a sharp edge is required, or joined smoothly otherwise.

It is simply not possible to accurately cut a point cloud manually on a computer screen by picking points with the mouse. The practice is that the designer cuts the cloud on the inside of the patch boundary to ensure that no points of another patch are included. This leaves a gap between the patches, which must be bridged by extending the surfaces. As explained in the introduction, this can be a considerable source of error (and frustration for the designer). This method is also very time consuming and sometimes leads to a long trial and error cycle. However, it is computationally robust and it works!

In their discussion of the problem, Chiang and Chen (1999) suggest that a system should be developed that can update the surface locally in the boundary region after the manual segmentation. Local updating of B-spline surfaces is a topic addressed by Ma and He (1998). It is possible to update the surface definition of the boundary region with a refitting procedure without having to recalculate the interior region. Remeasuring the boundary region seems like a good idea, but it would be very difficult to decide whether to include points measured very close to the boundary. Wrong inclusion will distort the surface definition.

Fitzgibbon et al. (1997) first do a rough edge-based segmentation using mean and Gaussian curvatures and then they use a face-based region growing method. Provided that the algorithm operates robustly, this method will include more points belonging to the surface patch. This means that there will be no gaps between the surfaces. An

unspecified Machine Vision system is used in their case study. Bearing this in mind, the reported average accuracy of the surface fits of between 0.2mm and 0.32mm is perhaps not bad. However, only planar and quadric surfaces are used. Their system operates automatically; therefore the system decides whether to fit quadric or planer surfaces to the data. This algorithm is definitely much more difficult to control if higher order or more complex surfaces such as B-spline surfaces are used. It also has already been mentioned that such an approach does not necessarily lead to a segmentation that captures the design intent of the part, neither will the surfaces necessarily be functional from an engineering point of view.

A number of researchers have proposed a face-based region growing approach to fit primitive analytic surfaces (e.g. planes, cylinders and spheres) to the data (Thomson et al. 1999, Park and Jun, 2000 and 2001 and Goussard and Basson, 2001). Provided that these methods allow the designer to control the type of surface that is fitted, this technique is very useful. It leads to functional engineering surfaces and the designer can embody the surface definition in the design intent. However, it is necessary that developers of these methods must give the designers good guidelines to help them specify the control parameters necessary to fit the entities. The algorithm of Goussard and Basson (2001) requires three different tolerances. A user that does not know the intricacies of their method will be at a loss to understand the meaning and implications of these parameters, let alone trying to specify intelligent values. In his thesis, Goussard (2001) provides guidelines on choosing the tolerances.

Vergeest et al. (2000) extended these ideas in their feature based reverse engineering system to extrusions and swept surfaces. They argue that the designer should determine the feature type that must be extracted from the cloud.

2.4.2. Edge-based Methods for Reverse Engineering

Proponents of the edge-based approach in Reverse Engineering frequently stress the importance of having edge information in an environment where free-form surfaces are the order of the day. Horváth and Vergeest (1998) go so far as to propose a Natural Shape Representation (N-rep), as against a Boundary Representation (B-rep). Their philosophy is that the designer should start with modelling the edges and then continue to fill in the surfaces. They propose a method to segment a regular grid point

cloud essentially using the 2D gradients and curvatures to find C^1 and C^2 singularities. Of course, the regular grid is an important impediment. They report no results on the accuracy and robustness against noise. It must be noted that they only use their system to propose *hints* for the segmentation. It is to be assumed that the user will ultimately accept or reject the proposed edge points. Sarkar and Menq (1991a), Seiler et al. (1996), Chen and Liu (1997) and Liu and Ma (1999) use the same concept, also for regular grids, but their implementations differ considerably in the detail. Yang and Lee (1999) used 3D curvatures, but they still require points in a regular grid in order to calculate the curvatures. Motavalli and Bidanda (1994) used this idea to segment the profile curve of a rotational part into lines and arcs.

All these gradient and curvature based methods have problems when segmenting objects consisting of free-form surfaces. For example, the curvature of a cubic NURBS surface changes continuously. For such objects it is very difficult to specify good threshold values for the segmentation.

Milroy et al. (1997) use 3D curvature to segment an unorganised point cloud. They construct a wire frame model (note, not a triangulation) of the point cloud to establish the topology and then calculate the curvatures. They use an active contour to identify and join the true edge points. Flexural stiffness and an inflation force must be assigned to the active contour and a seed point must be specified. A gravity force proportional to the principle curvature is assigned to the initial edge points. The energy of the active contour is then minimised to expand it. From the discussion of the case studies they did it is clear that false edge points can impede the expansion of the active contour. High signal to noise ratios in the data is therefore a problem, but this is probably always a problem. The point is that spurious points can cause incorrect edge detections. They also describe the difficulty of assigning appropriate flexural stiffness and inflation force values. Furthermore, it seems that these values are geometry dependent. A proper guideline is needed.

From the same research group comes a very novel approach. Bradley and Chan (2001) propose a sensor based approach. Realising the difficulties of manual segmentation on the computer screen and the above mentioned methods, they manually measure the patch boundaries with a CMM. These boundary points are then used to construct a rough surface, e.g. a Coons patch. All points in the cloud that can

be projected unto this surface and those that lie within a user specified tolerance belongs to the patch. For complex surfaces it might be necessary to use some curves across the interior of the patch so that a Gordon surface can be constructed. They claim, without evidence, that the manual scanning of the boundaries is just as fast as doing a manual segmentation on the computer screen. Furthermore, they claim that the edge points will be more accurate because they are measured with a very accurate CMM. This is, however, subject to how accurate the CMM operator can place the probe ball on the edge. Raab (1994), for example, states that the smallest feature a user can measure with an articulated arm is 0.762mm (0.03inches). Granted, placing an articulated arm is not the same as guiding a CMM, but one wonders whether there will be a big difference in the result, since in both cases the user has only his eyes to determine the position of the probe. Chiang and Chen (1999) also agree that measuring an edge with a CMM is a very great difficulty. Bradley and Chan (2001) give no information on how to compensate the measured edge points with the probe radius. If the points lie in a plane this is not too difficult, but how do they treat an arbitrary edge in 3D? In the last case the compensation will definitely be a source of error, possibly significantly larger than the measuring error.

2.5. Recommendation

The pilot study done in the beginning of this project, and reported in Chapter 1, highlighted the need for a robust edge detection method. A number of useful methods are proposed in the literature, but they all have important limitations or are only applicable for specific purposes. There is clearly a need for a computationally robust method in the field of Reverse Engineering. Simplicity and efficiency are also important practical requirements. Any automatic method must not only detect edges based on the geometry, but somehow it must also capture the design intent. This is a topic of much research in the field of feature recognition. It does not seem that a generally applicable method will be found soon. This rules out completely automating the process.

Another shortcoming in the literature is that very few researchers report the accuracy of their proposed methods. A thorough investigation of the accuracy of any new

method must be done. Preferably guidelines must be given to help the designer know beforehand what accuracy can be expected of the results.

Methods that work with unorganised point clouds, describing a closed volume, are in short supply. Edge detection for regular grid point clouds is a well covered topic. New methods must address the need for edge detection methods that work for unorganised point clouds.

Judging from the comments made by the researchers cited in this chapter, it seems that using an edge-based method is better when working with objects having free-form surfaces.

Chapter 3.

Virtual CMM

3.1. Introduction

In order to implement the edge detection method proposed in this thesis, it is necessary to be able to select a number of points, according to a pattern, from a point cloud. The selection method must be independent from the point cloud structure, since no structure is assumed in this research work. The accuracy of the scanned edge depends on the maximum distance between points in the cloud. This has a rather important implication. It was already stated that not all scanning methods gave a point cloud with uniform density. Problems often occur near vertical walls. However, if accurate edge information is required, care must be taken that sufficient points are scanned in the region of the edge. It cannot really be expected that the detected edge will be accurate if there are relatively few points in the region of the edge.

3.1.1. Neighbour Finding Method

The literature suggests a number of ways that points can be selected in this fashion. One way is to select a start point for the pattern and then search amongst its neighbours for the next point that lies closest to the desired pattern. This method, as all the methods described here, require that the points be stored in an octree data structure (Meagher, 1982) to minimise the search time. A neighbour searching method, such as developed by Goussard and Basson (2001) or Vörös (2000), can then be used to find the points lying on the desired pattern. This is a very efficient and reliable method of finding neighbouring points. The neighbour finding algorithm developed by these researchers use a simple binary numbering scheme to number each node in the octree as it is built. From the binary number, it is possible to determine in which level of the octree the node is contained and in which octant it lies. By doing simple binary arithmetic on the number and stepping through the

octree, all the nodes neighbouring a specific point can be found. The algorithm uses the previously selected points, and based on the specific pattern, calculates a direction in which the next point should lie. If this direction is not a very reliable estimate of a vector in the tangent plane of the current manifold, there is always the risk that the neighbour search method can select a point that does not lie on the current surface. Think for example of a thin walled object. If the wall thickness is of the same order of magnitude as the cloud density, there is no guarantee that points belonging to the same surface will in fact be selected.

3.1.2. Ball Pivoting Method

Bernardini, et al. (1999) successfully used a ball pivoting algorithm to triangulate an unstructured point cloud. Their algorithm uses the analogy of a ball "rolling" on the point cloud and thus filling in the triangles that connect the points. This "rolling ball" can equally well be used to select an arbitrary pattern of points. The ball can "roll" along the required pattern and the points touching the ball that lie closest to the desired pattern are selected. This algorithm requires that a reasonable estimate of the surface normal must be available at each point. In the application considered by Bernardini, et al. (1999), this is not a problem, indeed for many reverse engineering tasks this is not difficult to obtain. If points are scanned in a regular grid and the grid contains only points from one viewpoint, it is easy to make good estimates of the normal vectors. When the grids are combined, the normal vectors can be transformed with the points. Unstructured point clouds present more difficulties to calculate the normal vectors. It might require a complete surface triangulation. Suzuki and Aoyama (1997) developed a scanning method that also returns the normal vectors at each scanned point. Essentially their method does a 2.5D scan and then uses a triangulation to find the unit normal vectors at each scanned point. The triangulation method of Miyake, et al. (1997) can be used.

3.1.3. Virtual CMM Method

Another analogy from the physical world is to simulate a CMM. Janssens (1998) developed such a method; he calls it a virtual CMM and uses it for cloud thinning. Rather than scanning on a physical object, the virtual CMM "scans" on a point cloud. Again, the points are stored in an octree data structure. The virtual CMM uses the

octree to search for the point that lies closest to the line representing the movement. It does this by first finding the cluster containing the closest points and then selecting the closest point. This method is very efficient in computation time and it provides a means of ensuring that the virtual CMM remains on the correct surface. (The details of the last mentioned fact are discussed later in this chapter.)

There is another advantage to using a virtual CMM in context of this research project. By replacing the virtual CMM with a real CMM, it should be possible to use the same algorithm to scan an edge on a physical object. In order to study the viability of using the method on a real CMM, a simple analytical model was developed on which a virtual CMM can scan. The possibility of using the method on a real CMM is not considered further than this investigation in this project.

For these reasons, it was decided to implement a virtual CMM to obtain the necessary points from the point cloud. The implementation of the virtual CMM on a point cloud is further discussed in this chapter. It follows the discussion of the analytical method.

3.2. Analytical Virtual CMM

The analytical method was used in the initial testing of the algorithm because it eliminates the effect of point cloud density. As the experimental analysis shows later in this thesis, density has an important influence on the success of the edge scanning. By using an analytic CMM this effect was eliminated in the initial effort to understand the intricacies of the algorithm.

Two intersecting torii are used as shown in Figure 2. This model makes it possible to investigate all the parameters influencing the method. It is also simple to change the parameters of the model so that a series of tests can be performed. A third torus can be added as a fillet radius.

The axis of revolution of the torus model is the z-axis and the centre point can lie anywhere on the z-axis. The points where the virtual CMM will touch the object are the intersections of the lines representing the movement with the torii. The effect of the probe ball radius can easily be incorporated by simply adding or subtracting the probe ball radius to the section radius of the torii. Then the same algorithms for

finding the intersection points can be used. The only tricky part is finding these intersection points.

Let a be the major radius of the torus and r the section radius. The position of the centre on the z-axis is b. Any point on the torus can then be defined in terms of parameters u and v as shown in the equation below. The parameters are illustrated in Figure 1. The torus is defined by the revolution about the z-axis of the circle shown in the figure. Parameter u, not shown, is the rotation angle about the z-axis.

$$\boldsymbol{t}(u,v)=((a+r\cos v)\cos u,(a+r\cos v)\sin u,r\sin v+b) \qquad (3.2.1.)$$

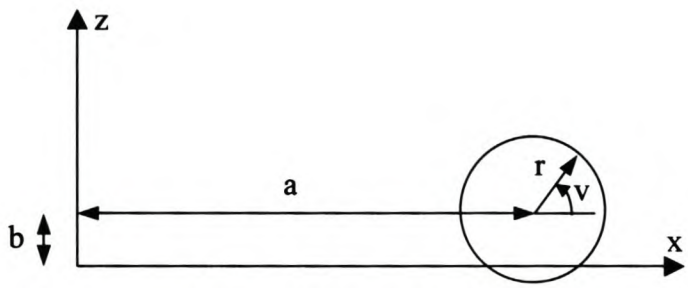


Figure 1 Torus Model.

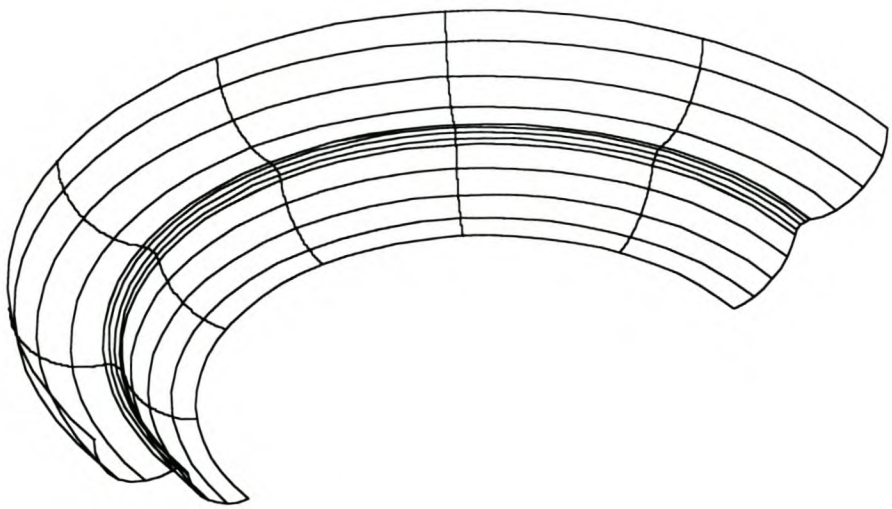


Figure 2 Torus Model in 3D.

If the effect of measuring noise must be investigated, a noise vector can be added to the point calculated with equation 3.2.1.

If the usual definition of a parametric line is used (see Appendix A), then the intersection point can be found by solving the following three equations.

$$s_{lx} + \lambda a_x = (a + r\cos u)\cos v \qquad (3.2.2.a)$$

$$s_{ly} + \lambda a_y = (a + r\cos u)\sin v \qquad (3.2.2.b)$$

$$s_{lz} + \lambda a_z = r\sin v + b \qquad (3.2.2.c)$$

The only three unknowns in these equations are the parameters u, v and λ. By substituting the above three equations into each other and doing some algebraic manipulations, a polynomial equation of degree four, given below, is obtained. The equation is derived in Appendix G.

$$0 = (K_1^2 - K_3) + (2K_1K_2 - K_4)\lambda + (2K_1 + K_2^2 - K_5)\lambda^2 + 2K_2\lambda^3 + \lambda^4 \qquad (3.2.3.)$$

with

$$K_1 = \|\boldsymbol{s}_l\|^2 - a^2 - r^2 + b^2 - 2b\,s_{lz}$$

$$K_2 = 2\boldsymbol{s}_l \cdot \boldsymbol{a} - 2b\,a_z$$

$$K_3 = 4a^2\left(r^2 - b^2 + 2b\,s_{:z} - s_{lz}^2\right)$$

$$K_4 = 8a^2\,a_z(b - s_{lz})$$

$$K_5 = -4a^2\,a_z^2$$

Once λ is known from equation 3.2.3., it is trivial to find the parameters u and v. Unfortunately, it is not so trivial to find the λ. One can easily think that values of λ at the start and end point of the CMM's movement can be used as search limits in a root finding method such as Newton's method. There are two problems with this approach.

Newton's method can find only one root. It is certainly possible that there can be more than one root between λ_{end} and λ_{start}. The virtual CMM must ensure that the λ closest to the start point of the movement is found. Newton's method cannot guarantee that.

A method such as Laguerre's method presented by Press, et al. (1997) finds all the real roots of any polynomial equation. They state that Laguerre's method guarantees convergence to all real roots from any start point. Although not proven, experience also suggests that non-convergence to complex roots is "extremely unusual," in the words of Press, et al. (1998). This algorithm was implemented with great success for this project.

3.3. Discrete Virtual CMM

Three variations of the virtual CMM developed by Janssens (1998) are considered here. The first variation is almost exactly the same as his virtual CMM, the only significant difference is that non-measuring moves are also considered. The second variation uses a virtual ball probe, with finite ball radius, to scan on the cloud. The last variation returns the intersection point of the search line and a triangle formed by three points close to the scan line. These three strategies are considered in this section. They all require an efficient search algorithm to find specific points in a point cloud. The search algorithm is considered first.

3.3.1. Search Algorithm

To make the edge detection algorithm as widely applicable as possible, no assumption is made of underlying structure in the point cloud and, following Janssens (1998), an octree data structure (Meagher, 1982) is used to handle the point cloud.

The octree algorithm first determines a box containing all the data. This is the root node. This node and successive nodes can be divided into eight child nodes, hence the term octree. Janssens (1998) suggested that the refinement of each node should continue until the length of the shortest side of the child node is less than three times the average pitch of the data or until the child node contains only one data point. Figure 3 gives an illustration of an octree. The selective refinement of the nodes is illustrated.

The octree has *internal* and *leaf* nodes. An internal node contains no data points, just pointers to its eight child nodes. A leaf node is found at the end of a branch and may contain data points, but has no children. This is illustrated in Figure 4 below.

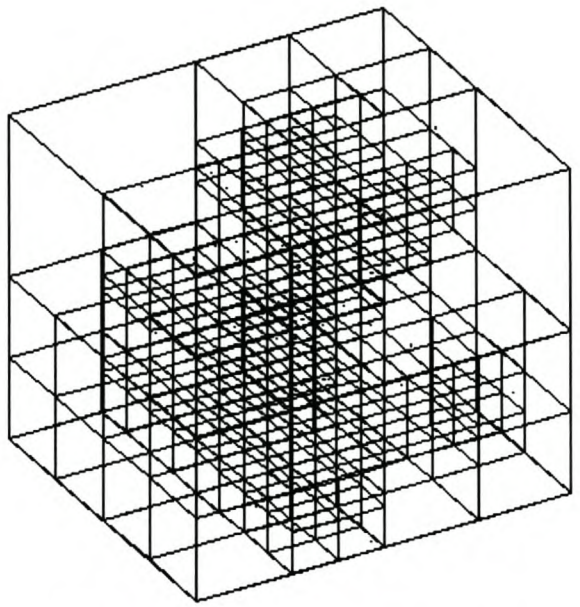


Figure 3 Example of an Octree.

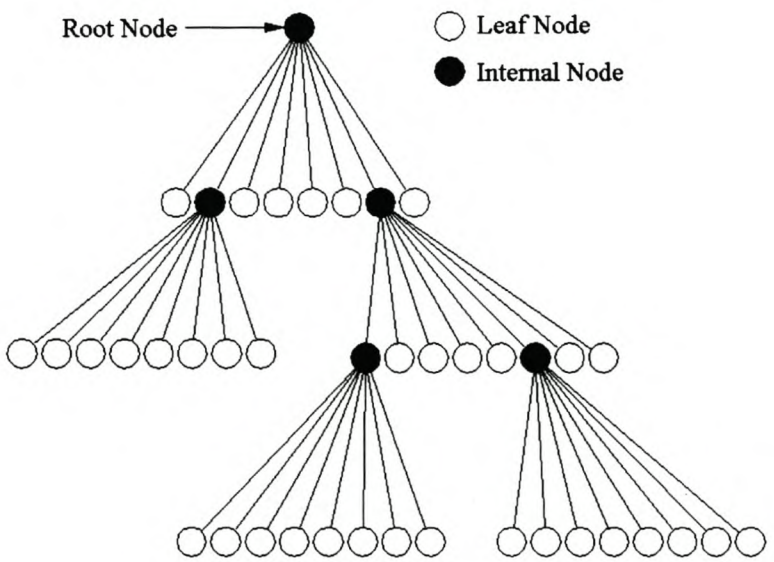


Figure 4 Octree Nodes.

The octree implemented here uses an object oriented programming approach adapted from Jones (1999). Octree-node and octree objects were developed. The octree-node object only contains data pertaining to the node: the position of one corner, the level

of the node in the octree data structure and the node type. If it is a leaf node, it also contains the number of points in the node and an integer pointing to the point in the data array corresponding to the first data point in the leaf node.

When the octree object is constructed, it reads the point cloud from a text file and calls the *BuildTree* procedure to create the octree data structure. Initially, all the data points belong to the cube-shaped root node. They are then sorted, using the *QuickSort* algorithm (Cooke, et al. 1985). The node is divided into eight equal cubes representing the child nodes, and the sorted data points are assigned to the respective child nodes. After a node has been divided into eight child nodes, the procedure is repeated on each child node until the stop criteria are satisfied.

Finding a specific point in an octree data structure means that the box containing the point must first be found. The child nodes of the root node are tested. All those that can contain the point are selected and the algorithm repeats the search step for all the selected nodes. This is best done in a recursive implementation. This process continues until a leaf node is reached. If the leaf node contains points, all the points in the node are tested. This minimises the search time, because only a fraction of the points in the cloud are actually tested.

3.3.2. Point Selection

The search line will seldom actually intersect a specific point in the cloud. Therefore the strategy suggested by Janssens (1998) selects the point that lies closest to the search line. This is a very simple calculation (see Appendix A).

As already mentioned in this chapter, a very important consideration in finding a specific pattern of points in a point cloud, is that the method must ensure that selection of points is consistent. In other words, it must be certain that neighbouring points in the pattern actually are neighbours in the manifold. The point selection method attempts to ensure this in a number of ways. First of all, the start point of the search line must lie outside the object represented by the points. Since the search line is a parametric line, this means that only points that have a positive parameter when projected onto the line have to be considered. (See Appendix A for projecting a point onto a line.) Points that correspond to a negative parameter obviously lie on a surface

in the opposite direction of the scanning movement. Thus, the point with the smallest positive parameter is selected. This is illustrated in two dimensions in the figure below. The line represents the search line. The arrow indicates the direction of the search line. The black point at the top of the line is the origin of the search line. The white points belong to the cloud.

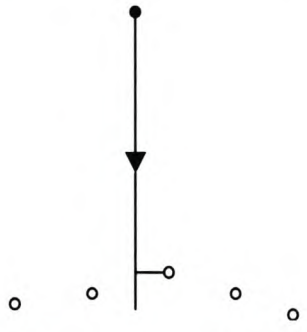


Figure 5 Finding the Closest Point to the Search Line.

When scanning in the bottom of a valley, the points at the top of the valley corresponds to a smaller, positive, parameter value than the ones at the bottom if the scanning movement started near the top of the valley. So, another test is necessary. The points at the top of the valley are removed from the list of candidates by calculating the distance to the search line (see Appendix A). If this distance is more than the cloud density, the point is removed from the list. This has an important implication for the point cloud. The density must be sufficiently uniform so that gaps in the point cloud that is more than the pitch apart really do represent a gap in the surface.

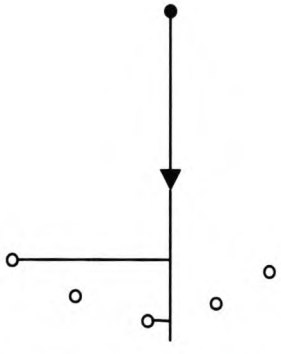


Figure 6 Selecting Points in a Valley.

Another problem arises when scanning close to a vertical wall. If the distance from the search line to the wall is less than the point cloud density, then it is unavoidable

that the point on the wall that corresponds with the smallest positive parameter value will be selected.

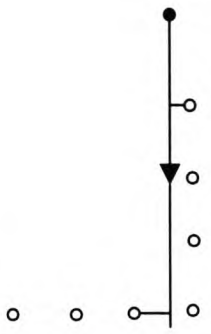


Figure 7 Selecting Points Near a Vertical Wall.

Janssens (1998) does not simulate non-measuring movements. This means that during a non-measuring movement the "probe" can move through the surface without the scanner knowing it! If there is another surface behind the one being scanned upon, there is a serious risk that the next scanning movement will be on the wrong surface. Thus, non-measuring movements are simulated in the virtual CMM developed here. This is another mechanism that helps to ensure that the scanner remains on the correct surface. Thus, the virtual CMM can do "collision" testing. It will be stopped from moving through the object.

This method has the advantage that it returns the best possible available data to the downstream algorithms. No assumption is made about the nature of the surface between the points. The only error is the noise in the point cloud. Another advantage is that this method requires the least amount of computation time of the methods presented in this chapter. The obvious disadvantage is that the selected points will not lie on the desired pattern. This may cause problems when doing manipulations with the pattern of points.

These reasons made this method the method of choice in this research project, but two other methods, discussed below, were also considered.

3.3.3. Scanning with a Virtual Probe

While simulating a CMM, why not simulate the probe as well? As with a real CMM, the virtual CMM then returns the centre of the probe ball. This can at least solve the problem of the points not lying on the search line.

If $\boldsymbol{c}$ is a point in the cloud, R_p is the probe radius and the search line is as given in Appendix A, then the position of the probe ball centre can be found from the following equation.

$$\|\boldsymbol{c} - (\boldsymbol{s}_l + \lambda \boldsymbol{a})\| = R_p \qquad (3.3.3.1.)$$

Expanding this equation and remembering that $\boldsymbol{a}$ is a unit vector, gives the expression for λ in equation 3.3.3.2.

$$(\boldsymbol{c} - (\boldsymbol{s}_l + \lambda \boldsymbol{a})) \cdot (\boldsymbol{c} - (\boldsymbol{s}_l + \lambda \boldsymbol{a})) = R_p^2$$

$$\|\boldsymbol{c}\|^2 - 2\boldsymbol{c} \cdot (\boldsymbol{s}_l + \lambda \boldsymbol{a}) + \|\boldsymbol{s}_l + \lambda \boldsymbol{a}\|^2 = R_p^2$$

$$\lambda = \frac{\boldsymbol{c} \cdot \boldsymbol{a} - \boldsymbol{s}_l \cdot \boldsymbol{a} \pm \sqrt{(\boldsymbol{s}_l \cdot \boldsymbol{a} - \boldsymbol{c} \cdot \boldsymbol{a})^2 - \|\boldsymbol{a}\|^2 \left(\|\boldsymbol{s}_l\|^2 + \|\boldsymbol{c}\|^2 - 2\boldsymbol{c} \cdot \boldsymbol{s}_l - R_p^2 \right)}}{\|\boldsymbol{a}\|^2}$$

$$\lambda = \boldsymbol{c} \cdot \boldsymbol{a} - \boldsymbol{s}_l \cdot \boldsymbol{a} \pm \sqrt{R_p^2 - \left[\|\boldsymbol{s}_l - \boldsymbol{c}\|^2 - (\boldsymbol{s}_l \cdot \boldsymbol{a} - \boldsymbol{c} \cdot \boldsymbol{a})^2 \right]} \qquad (3.3.3.2.)$$

The shortest distance from a point in the cloud to the line is given by the expression in the []-brackets in equation 3.3.3.2. If this distance is equal to or less than the probe radius, then the equation is used to calculate λ. Both λ-values (given by the ± sign) for each point must be checked. Negative λ-values can be ignored because they represent situations where the CMM have to move in the negative $\boldsymbol{a}$ direction. The smallest positive value of λ given by all the points in the cloud represents the distance that the CMM must move to make contact with the point cloud.

If the probe radius is larger than the point cloud density, this method helps to ensure that the scanner remains on the correct surface. It also has no problem scanning near vertical walls or down in valleys. There are two significant disadvantages though.

Since the virtual CMM returns the points at the probe ball centre, these points must be compensated with the probe radius. Under certain circumstances this can both be difficult and inaccurate.

The second disadvantage is that this method induces noise. Scanning on a discrete surface the probe can in fact already have passed through the actual surface when it eventually touches a point in the cloud. If it is assumed that the surface connecting three neighbouring points does not deviate significantly from a flat surface, the order of magnitude of the noise can be determined as follows. In the following figure let *A*, *B* and *C* be three neighbouring points. *D* is the centre of the probe ball. Let the distance between any neighbouring point *A*, *B* or *C* be *d*. If the surface *ABC* is flat, the contact point should be *E*, but the virtual probe will move beyond that point until in makes contact with a point *A*, *B* or *C*. In the worst case, *E* it is exactly in the middle of the equilateral triangle *ABC*. The noise is then the difference between the probe radius and the distance *DE*.

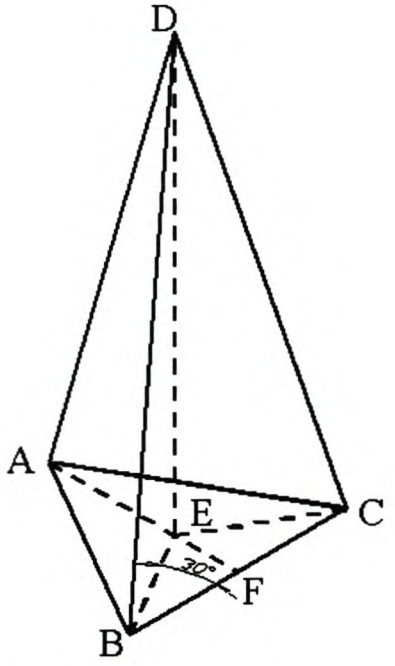


Figure 8 Noise Induced by Scanning with a Virtual Probe.

The noise is found as follows.

$BD=R_p$

$BC=d$

$BF=0.5d$

$$BE = \frac{BF}{\cos 30°} = \frac{d}{\sqrt{3}}$$

$$DE = \sqrt{BD^2 - BE^2} = \sqrt{R_p^2 - \frac{d^2}{3}}$$

$$\therefore \varepsilon = R_p - DE = R_p - \sqrt{R_p^2 - \frac{d^2}{3}} \qquad (3.3.3.3.)$$

Clearly, if the probe radius is significantly larger than the point cloud density, the noise is small. It decreases as the ratio of probe radius to point cloud density increases.

3.3.4. Triangular Approximation

In order to scan points that lie on the desired pattern and that do not need radius compensation the virtual CMM can scan on a triangulated surface. Triangulation of points scanned on physical objects is a very difficult and unresolved problem, judging by the number of papers that regularly appear on the topic (some of them are cited in this section). Problems with triangulations also often arise near sharp edges, exactly the region that is of interest for this project. A complete triangulation of the point cloud is therefore not within the scope of this project. A much more simplified approach is taken here.

A triangle is calculated for each scanning movement. This triangle is constructed by first finding the two closest points to the search line using the point selection method described above. These two points are A and B respectively in Figure 9. The algorithm then selects a third point, C in Figure 9, to form triangle ABC. Triangle ABC is formed so that the search line, represented by E, intersects it. C is the closest point to E that will form such a triangle. Now there can be no other points in the dashed circle, with centre point E and radius EC, which can be used to construct a triangle with points A and B that will be intersected by the search line, E.

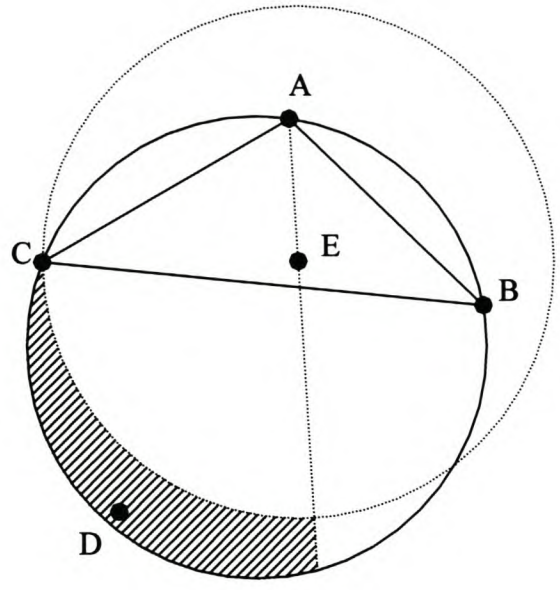


Figure 9 Triangle Construction with Virtual CMM.

Is triangle ABC a "good" triangle, in other words, does it really represent the surface on which the points A, B and C were scanned?

De Berg, et al. (1997) and Bernardini, et al. (1999) show that a 3D Delaunay triangulation guarantees that the topology of the points is correct and that the surface converges to the true surface as the point cloud density increases. Computing a 3D Delaunay triangulation is expensive in terms of computational time and memory required (Hoppe, et al., 1992; Edelsbrunner and Mücke, 1994; Choi, et al., 1998; Bernardini, et al., 1999 and Wang and Chen, 1999). For this project it is only necessary to compute a 2.5D Delaunay triangulation of the region being scanned, as described by De Berg, et al. (1997) since the virtual CMM is only concerned with a small part of the surface during each scanning movement. Briefly, the requirement for a 2.5D Delaunay triangulation as described by De Berg, et al. (1997) is as follows.

If triangle $P_1P_2P_3$ (Figure 10) defines a circumscribed circle and P_4 lies inside the circle, edge P_2P_3 is illegal and must be flipped. The Delaunay triangles are then $P_1P_2P_4$ and $P_1P_3P_4$.

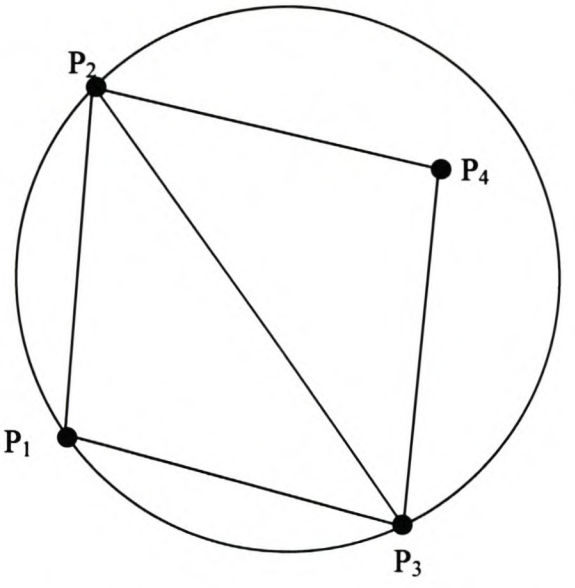


Figure 10 2.5D Delaunay Triangulation Rule.

From this it follows that triangle ABC in Figure 9 is not necessarily a Delaunay triangle. If there is a point, D, in the cross hatched region of that figure, it means that the edge BC is illegal. The Delaunay triangles will be ACD and ABD.

Now, the question might be asked why a fourth point, D, is not tested in order to ensure that Delaunay triangles are indeed formed. Well, the problem is that there can be more points in the region that can complicate matters. Consider only triangle ABC in Figure 9 as duplicated in Figure 11.

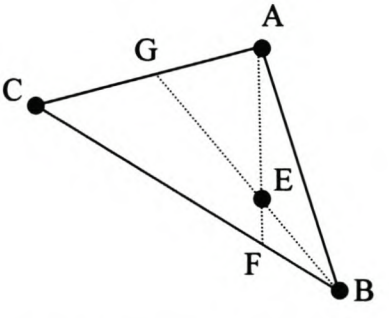


Figure 11 Regions that can Contain more Points.

There cannot be more points in the triangle AEB, because A and B are the closest points to E. If there was another point in the region described by the polygon CGEF, it would have been used to complete the triangle that is used to form the triangle that is

intersected by search line E. Therefore there cannot be more points in the polygon region CGEF. The triangles AEG and BEF remain. Any point in one of these triangles together with A and B cannot be used to form the initial triangle that is intersected by the search line. Thus, it is possible that there can be one or more points in this region. These points must also be considered in the Delaunay triangulation of the surface region.

On top of all this, the surface does not necessarily end beyond the polygon ABCD. Points outside this region must by triangulated until it is possible to say that all the edges of the triangle intersected by search line E are legal. It becomes a complicated triangulation process that is beyond the scope of this research project. So, the method is left as it is.

The triangle that is intersected by the search line is not necessarily a Delaunay triangle. In other words, it does not necessarily represent the real surface. Although some tests were done with this scanning method, this reason and the extra computation time are the most important reasons why this method is not further used for edge detection.

Chapter 4.

Edge Scanning Algorithm

4.1. Introduction to the Edge Scanning Method

4.1.1. Background and Description of the Method

In previous chapters it is shown that segmenting point clouds presents many difficulties, not only in terms of finding the boundaries of surface patches, but also with the subsequent surface modelling. It is often a very time consuming process to reconstruct a CAD model consisting of many surface patches from a single point cloud. The question is whether a tool that can segment the point cloud and simultaneously find the intersection curves between neighbouring surfaces would not make modelling much faster. In this chapter such a tool is described.

Here, a virtual CMM is used to scan the edge. Sometimes it might only be necessary to find the surface boundaries and use these curves as generators when recreating a surface model of the object.

Rather than defining some of the points in the point cloud as edge points, the method described here calculates new edge points. This is an important advantage over image segmentation methods such as described in the literature review. The errors in the case of the image segmentation methods can be as large as the point cloud density, which means that a very dense point cloud must be scanned to accurately define the edge. Although the accuracy of the edge scanning method also depends on the point cloud density, the error of the edge points is normally much less than the cloud density.

The idea of the edge scanning algorithm is to scan as few as possible points in a small region around the edge so that the scanning time is reduced as much as possible and then calculating the edge points from the scanned points. This means that the

algorithm must be able to anticipate direction changes of the edge. Another important characteristic of the algorithm is that it must be robust against measuring noise. The latter requirement proved to be the most difficult to achieve. An example of the points scanned by the algorithm is shown in Figure 1.

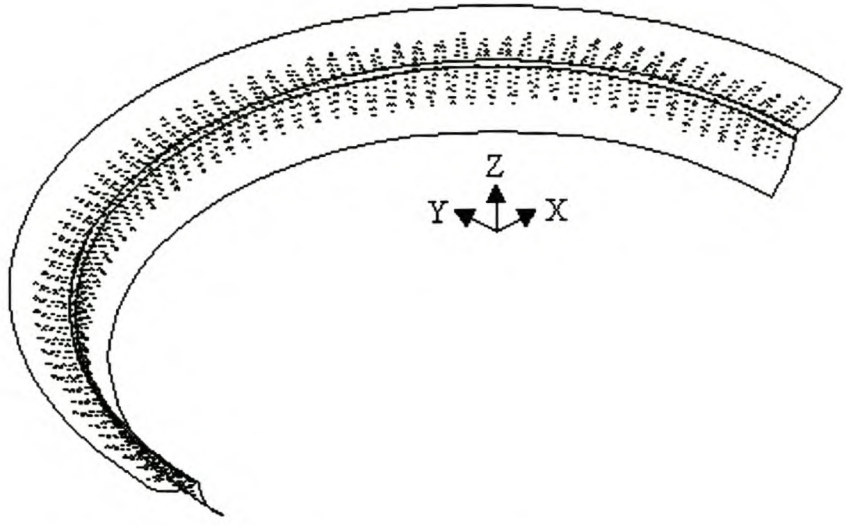


Figure 1 Example of a Zigzag Edge Scan.

A brief description of the method is as follows. Given a starting condition, a line is scanned and the point is calculated where the scan line intersected the edge. The edge point is the intersection of two polynomial functions that approximate the scan line sections (lines or quadratic polynomials). The position of the next edge point is then estimated. The scanner then tries to scan the next line so that it will intersect the edge near the estimated edge point. The process of estimating edge points, scanning lines and calculating edge points continues until the calculated edge passes through a "gate" defined by the user. Finally the polynomial curves are compensated with the probe radius, if necessary, and the edge points are recalculated.

4.1.2. Definition of Terminology

Before starting the detailed description of the scanning methods, it is necessary to define the terminology that is used in this chapter.

Two *patterns* that can be used to scan the edge are discussed. Since both patterns share many basic algorithms and differ only in a few details, they are discussed

simultaneously. Where there are differences, they will be pointed out. A *zigzag* pattern (Figure 1) and a *square* pattern (Figure 2) are described.

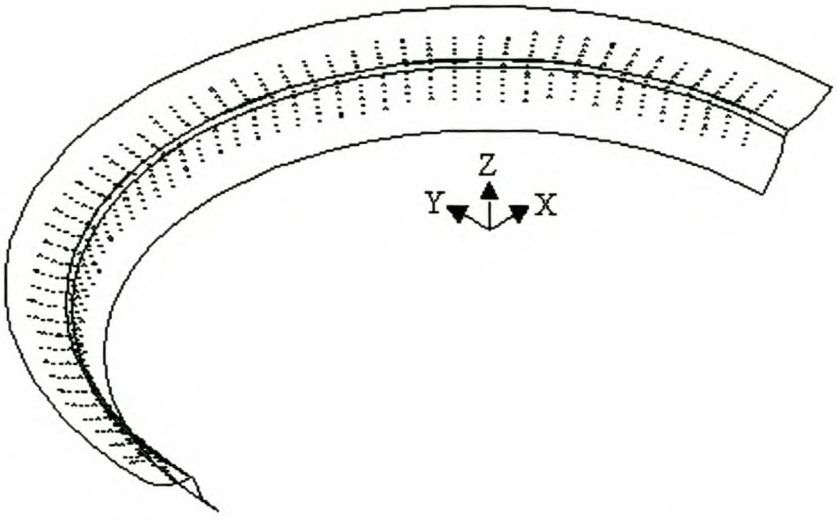


Figure 2 Example of a Square Edge Scan.

Each pattern consists of a sequence of *scan lines*. The scan line starts on one surface, crosses the edge and terminates on the second surface forming the edge. The points on the scan line that was scanned on the first surface are called the first *line section*, and the points on the second surface form the second line section. The scanned points are designated with $\boldsymbol{p}_{i,j}$, indicating the *i*'th scanned point on the *j*'th scan line. Where the meaning is clear, the subscript j is dropped so that the notation is not unnecessarily cluttered. *Edge points* are designated $\bar{\boldsymbol{p}}_j$, meaning the edge point of the *j'th* scan line. The *line pitch* is the distance that the scanned points must be apart. It differs from the *edge pitch*, which is the distance that the calculated *edge points* are apart. Edge points are calculated from the points on the scan line. The *amplitude* defines the region around the edge where the scanner can safely sample points to calculate the edge points. As shown in Figure 3 the amplitude is defined in the way it is normally done for waves. This figure also illustrates the schematic representation of the scanning patterns that will be used further on. Rather than drawing a 3D view, the pattern is drawn as though it is folded open. This is easier to view on paper.

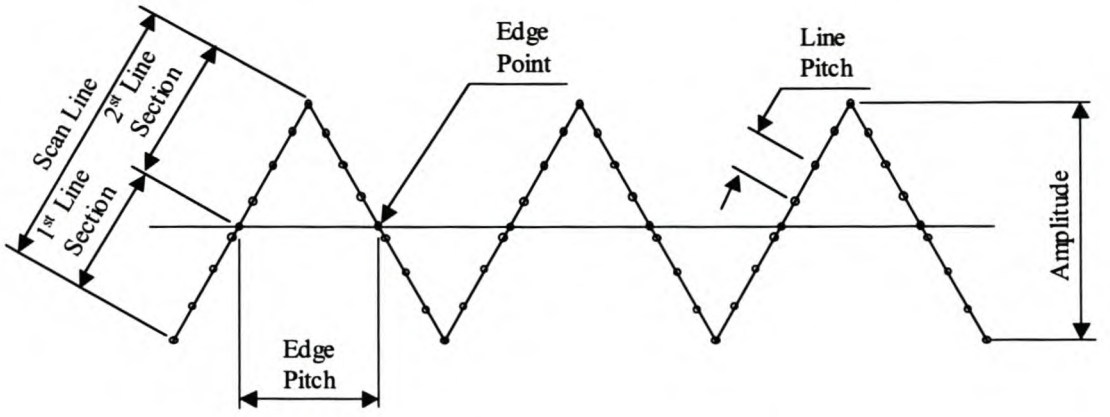


Figure 3 Scanning Terminology.

When using the virtual CMM to scan on a point cloud, the *point cloud pitch* is very important. Here it is defined as the average distance between neighbouring points in the point cloud.

The rest of this chapter describes the edge scanning method in detail. The chapter ends with a discussion of the limitations of the method. The evaluation of the method is left for the next chapter.

4.2. Line Scanning

4.2.1. Defining the Scanning Plane

The CMM must scan a number of lines that straddle the entire length of the edge. When scanning a surface using a standard parallel scanning pattern, the planes in which the scan lines lie, are determined by the parallel pattern parameters. This is not the case when scanning an edge. Each consecutive scanning plane depends on the local geometry of the edge. The scanning plane should be orientated in such a way that a safe scanning path can be obtained. Ideally this means that the scanning plane must be perpendicular to both surfaces around the edge.

It is not possible to define such a scanning plane, i.e. one that is perpendicular to both surfaces, when using the zigzag pattern. However, by choosing the amplitude much larger than the edge pitch, the scanning plane will be close to perpendicular to both surfaces. Finding a reliable definition of the scanning plane orientation in the case of the zigzag pattern proved difficult. Obviously, the vector from the last scanned point

to the estimate of the next edge point, i.e. the new scanning direction, must lie in the new scanning plane. Various other vectors were tested to completely define the scanning plane, such as the normal vector on the current surface. This means that the new scanning plane is at least perpendicular to the first surface, but not to the second surface. The best compromise was to use the tangent vector at the last edge point and the scanning direction. These two vectors are t and d_s respectively in Figure 4. This means that the new scanning plane orientation is given by the following equation.

$$n_s = d_s \otimes (t \otimes d_s) \qquad (4.2.1.1.)$$

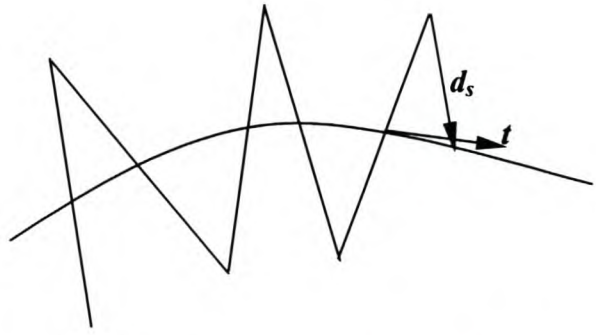


Figure 4 Determination of Scanning Plane Orientation.

The tangent vector, t in Figure 4, is the obvious definition for the scanning plane orientation for the square pattern since it is perpendicular to the normal vector of both surfaces at the edge. (This fact is demonstrated in paragraph 4.4.1.)

4.2.2. Ensuring a Constant Pitch

For reasons of economy and accuracy, it is necessary to keep the pattern in the smallest possible region around the edge and to scan as few points as possible. The first requirement is met by calculating the length of the scan line as it is being scanned. As soon as the length is equal to the amplitude, the scanning will stop.

It is possible that the gap between the last point scanned before the scanner crossed the edge and the first point scanned after it crossed, is much larger than the required line pitch. The sharper the angle is between the two neighbouring surfaces, the bigger is the chance that this will be the case. If corrective measures are not taken, it can happen that there will not be enough points on the second section of the scan line to accurately calculate the edge. When the algorithm detects that the gap between points

is too large, it will go back and measure enough points in the gap to ensure that consecutive points on the scan line are not more than the required line pitch apart.

4.2.3. Amplitude Adjustment

As the scanner moves along the edge, it can happen that the pattern slowly moves off the edge. As it moves along the edge, the new edge points will no longer lie in the middle of the scan lines. This phenomenon is schematically illustrated in Figure 5. It can easily happen that the pattern moves completely off the edge. A further danger related to this, is that there will be too few points on the one line section to calculate the line or polynomial needed to calculate the edge point.

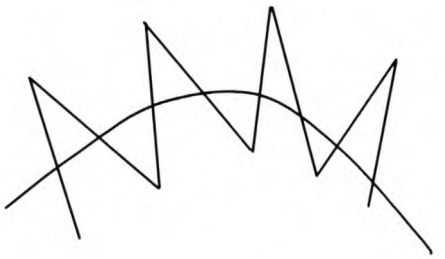


Figure 5 Schematic Illustration of a Zigzag Pattern Slipping Off an Edge.

The solution to this problem is to scan more points on the line if the second line section is too short. From Figure 5 it might seem that another solution is to decrease the pitch when scanning in a region of high edge curvature. Although this pitch adjustment was implemented, see paragraph 4.4.4., it was found that it is sufficient to extend the length of the scan lines.

Ideally, if the length of the second line section multiplied by the cosine of the angle between the scanning plane normal and the edge tangent, ϕ_t in Figure 6, is shorter than half the required amplitude, then more points must be scanned to make up the deficit.

However, this method produces bad results if the tangent is not calculated very accurately. With a bad tangent, the angle ϕ_t is much larger than it should be, which in turn means that a large deficit is calculated. The scanner will make up the deficit. This causes much longer scan lines than desired. The calculation is made independent

of the tangent by comparing the length of the scan line with half the amplitude, in the case of a square pattern, and with $0.5\sqrt{\rho_e^2 + A^2}$, in the case of a zigzag pattern.

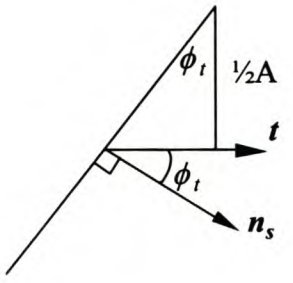


Figure 6 Angle between Scanning Plane Normal and Edge Tangent.

4.3. *Intermediate Edge Point Calculation*

An intermediate edge point is calculated for each line of the scanning pattern. The scanned line is first split in its two parts, the line sections. Then each part of the line is approximated with a polynomial function. The intermediate edge point is the intersection of these two functions.

4.3.1. Scan Point Projection

Measuring noise and the discrete nature of point clouds mean that the scanned points do not necessarily lie on the scanning plane. This can cause considerable problems when fitting curves to these points, because the degree of freedom of the fitting process is often very small. The maximum distance between the fitted curve and the scanning plane can be very significant. The fitted curves can then intersect the edge very far from the scanning plane. This is illustrated in the Figure 7.

Figure 7 illustrates how noisy points can distort the curve approximations. The fitted lines do not intersect at the edge. The distortion becomes much bigger as the gap at the edge increases, due to a fillet radius. Edge points calculated with these lines often have very large errors, sometimes so large that algorithm is unable to follow the edge.

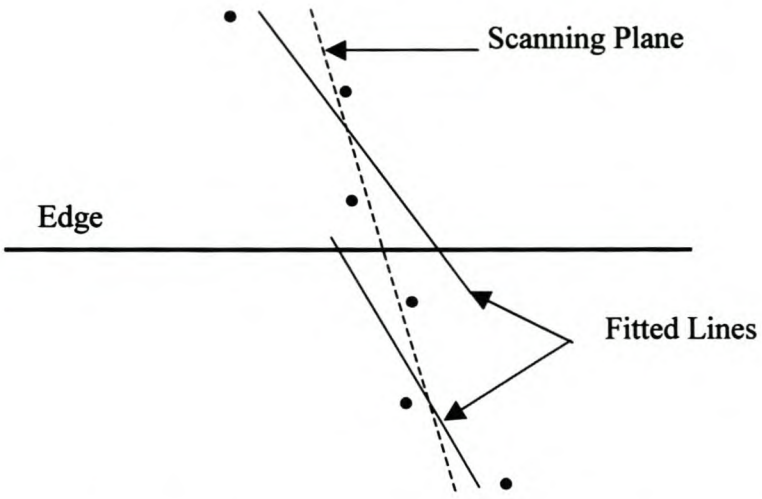


Figure 7 Problems Due to Curve Approximation of Noisy Points.

Projecting either the points or the polynomials to a common plane can solve the problem. Both methods were tried (using only first order polynomials) and no significant difference, in terms of accuracy, was found. Since it is easier to project the points to a plane than a polynomial function, it was decided to project the points before fitting a polynomial function to the points. The points are projected onto a plane that is fitted to all the points of the scan line using the equation below. (The equation is derived in Appendix A.)

$$\boldsymbol{p'}_i = \boldsymbol{p}_i - \{\boldsymbol{n} \cdot (\boldsymbol{p}_i - \boldsymbol{s}_p)\}\boldsymbol{n} \qquad (4.3.1.1.)$$

The projected points are then approximated with polynomial curves and these curves are used to calculate the edge points as described in what follows.

4.3.2. Splitting the Scan Lines

A scan line straddles the edge. It is divided in two scan line sections at the point that lies furthest from the line connecting the first and last point of the scan line. This point is deleted from the scan line and is not further used in the calculation of the edge point.

4.3.3. Cleaning the Line Sections

Often the edge between two intersecting surfaces is not sharp. A small fillet radius is often added, or the edge might simply be worn out and damaged. Points scanned on

this part of the object must first be removed before the edge can be calculated. These points are identified by checking the distance from the projected point to the polynomial. If there are points of which the distance to the fitted polynomial is more than a user specified tolerance, the point closest to the edge is deleted and a new polynomial fitted. This process is repeated until all the points are within the tolerance or until the number of remaining points equals the degree of the polynomial.

The distance to the curve can easily be calculated using the equation below if it is a line. (This equation is derived in Appendix A.)

$$d = \left\| \boldsymbol{a} \otimes \left(\boldsymbol{p}_i - \boldsymbol{s}_l \right) \right\| \qquad (4.3.3.1.)$$

However, when a polynomial of degree more than one is used, the calculation is a little more intricate. Since parameterised polynomials are used, the parameter of the point on the polynomial closest to the scanned point must be found before the shortest distance to the polynomial can be calculated. Brent's method (Press et al. 1997) is used to solve the non-linear problem. Since the polynomial represents only a short curve on the surface of the object, the problems of finding good initial values for Brent's method and of having more than one point on the polynomial that are the same distance to the scanned point, can be ignored.

It is important to note that the tolerance mentioned here is only to be used for deleting points that do not belong to the surface, e.g. points on a fillet radius. As far as possible, points on the actual surface must not be deleted. That will leave very few points to fit a curve to. Therefore, the tolerance must at least be more than the noise in the data. Since the algorithm works with as few points as possible, it cannot afford to throw away noisy points.

4.3.4. Fitting Polynomials

Polynomials are fitted to the projected points of each line section. Theoretically polynomials of any degree can be fitted as long as there are enough points on the line section. However, the polynomials will have to be extrapolated so that their intersection can be calculated. In order to ensure a well behaved extrapolation, cubic

polynomials is the practical limit. Typically there will be no more than ten points on each line section, another reason not to fit polynomials of degree higher than three.

The type of polynomial that is fitted is another source of error in the calculation of the edge points. Surfaces are seldom perfectly flat, so a straight line might be a very rough assumption. However, it provides the opportunity to scan the least number of points to calculate the edge and thus save scanning time. A quadratic polynomial is probably the best option provided that the surfaces are not doubly curved in the scanning region. Cubic polynomials will only be used in rare cases.

If a line is selected, the least squares fitting algorithm of Forbes (1991) is used. It is not necessary to parameterise the scanned points before the lines are fitted.

In order to fit quadratic or cubic polynomials the scanned points must first be parameterised using the chord length method. The usual least squares problem (Lawson and Hanson, 1974) is then formulated as

$$[\boldsymbol{U}]^{\mathrm{T}}[\boldsymbol{U}][\boldsymbol{C}]=[\boldsymbol{U}]^{\mathrm{T}}[\boldsymbol{P}] \qquad (4.3.4.1.)$$

where $[\boldsymbol{U}]=\begin{bmatrix} 1 & u_1 & u_1^2 \\ 1 & u_2 & u_2^2 \\ & \vdots & \\ 1 & u_n & u_n^2 \end{bmatrix}$ and $[\boldsymbol{C}]=\begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \end{bmatrix}$

In the above equation $\boldsymbol{U}$ and $\boldsymbol{C}$ are given for the quadratic case. The changes needed to fit a cubic polynomial are trivial. $\boldsymbol{C}$ is the matrix of coefficients of the quadratic polynomial in parametric form. $\boldsymbol{P}$ is the matrix of measured points. If $\boldsymbol{U}$ is known, $\boldsymbol{C}$ can be solved using Gauss elimination.

Finding the coefficients is a non-linear problem since the parameterisation is unknown. However, it was found that the initial parameterisation using the chord length method is sufficient. A non-linear optimisation method, such as Powell's method (Press et al. 1997) will put an unnecessary computational burden on the algorithm.

4.3.5. Calculating the Intersection Point

The calculation of the intersection of two lines is straight forward. The calculation thereof is separated from the calculation of polynomials of degree more than two.

4.3.5.1. Line Intersections

Finding the intersection of two lines in a plane is straight forward. Equations for finding the intersection are derived in Appendix A. If the one line is represented by $\boldsymbol{x}_2 = \boldsymbol{s}_{l2} + \lambda_2 \boldsymbol{a}_2$, then λ_2 can be solved using the equation below. If the denominator in this equation is zero, then one of the alternative equations given in Appendix A can be used, provided that the lines are not parallel. Of course, the two lines must also lie in the same plane.

$$\lambda_2 = \frac{a_{1x}\left(s_{l2y} - s_{l1y}\right) - a_{1y}\left(s_{l2x} - s_{l1x}\right)}{a_{2x}a_{1y} - a_{2y}a_{1x}} \text{ if } a_{2x}a_{1y} - a_{2y}a_{1x} \neq 0 \qquad (4.3.5.1.1.)$$

4.3.5.2. Polynomial Intersections

A polynomial of the fourth degree must be solved to find the intersection points of two quadratic polynomials. For two cubic polynomials, a polynomial of the ninth degree must be solved. For this reason, a numerical method is used to find the intersection point. The distance between the two polynomials is minimised using Powell's method (Press et al. 1997).

As shown in the figure below, the iteration algorithm uses the parameter values of the points that lie closest to the edge as starting values. These normally are good starting values. However, sometimes it happens that at least one of the polynomials looks like the 2nd polynomial in Figure 8. In these cases it can happen that the algorithm finds the wrong intersection. In the cases where this problem is known to occur, decreasing the step size solved the problem.

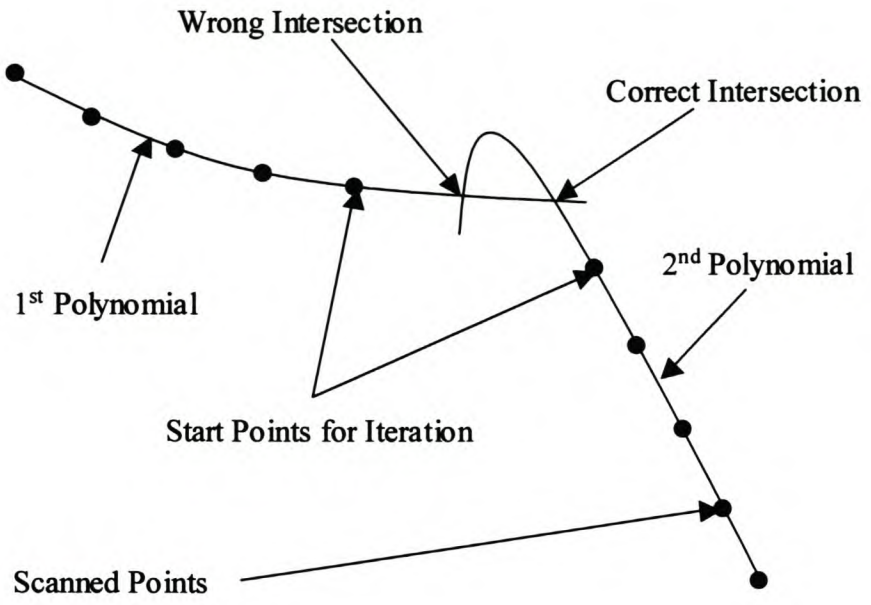


Figure 8 Finding Polynomial Intersections.

4.3.6. Discussion

Some edge detection techniques for reverse engineering do a search of the points in the cloud and then define selected points as edge points (Milroy, et al., 1997 and Yang and Lee, 1999). This means that the selected edge points can be as far from the real edge as half the point cloud pitch. Furthermore, these methods do not take into account that the edge might be round due to a deliberate fillet radius or due to wear and damage. The method proposed here in essence extrapolates the boundary region of two neighbouring surfaces and calculates the intersection points. Provided that the assumptions made for the extrapolation, i.e. linear or quadratic, are correct, the calculated edge point should be much closer to the real edge than the point cloud pitch. The measuring noise and the probe radius effect further influences the accuracy. A quantitative analysis of the accuracy of the method follows in a later chapter.

Robustness against noise in the measurements is achieved by first projecting each scan line onto a plane that was fitted through the points of the scan line. Polynomial approximation rather than interpolation further enhances the robustness against noise.

4.4. Finding the Scanning Direction

Two methods are investigated to determine the direction that the pattern must take after doing each line scan. The first method takes the estimated tangent of the edge as the scanning direction. The second method takes the curvature of the edge into account and determines the scanning direction accordingly. For both these methods, a robust estimate of the edge's tangent direction is vital.

4.4.1. Finding the Edge's Tangent Direction

Three methods of finding the edge's tangent direction were tested. The challenge is to find a robust estimate of the tangent direction since this determines the ability of the algorithm to follow the edge without error.

4.4.1.1. Edge Point Interpolation

The obvious, and easiest way, to find a tangent vector is to interpolate the last two edge points as

$$\boldsymbol{t}_j = \frac{\overline{\boldsymbol{p}}_j - \overline{\boldsymbol{p}}_{j-1}}{\left\| \overline{\boldsymbol{p}}_j - \overline{\boldsymbol{p}}_{j-1} \right\|} \qquad (4.4.1.1.1.)$$

Clearly, this method depends on the accuracy of the calculated edge points. Inaccurate edge points will lead to inaccurate tangent vectors that can in turn direct the algorithm away from the edge. If an occasional edge point is significantly erroneous, one would still want the algorithm to complete the scan. It is therefore desirable to have a method that does not depend on the calculated edge points.

4.4.1.2. Cross Product of the Surface Normals Using a Flat Surface Approximation

A better estimate of the tangent direction can be done as follows. From differential geometry it is known that the normal vector at a point on a surface is perpendicular to the tangent plane at the same point (Do Carmo, 1976). The tangent vector at a point on a curve in the surface lies in this tangent plane. Thus, when the normal vector of the surface is calculated at a point on the edge, the edge's tangent vector lies in the tangent plane. Since the edge is a curve mutual to the two neighbouring surfaces, two

surface normal vectors can be determined at the edge and their cross product gives the direction of the edge's tangent vector.

The normal vector on the surface at the edge point is approximated by fitting a plane to the points of the last two scan lines that also belong to the same surface. The cross product of the normal vectors of these planes is the tangent direction.

This method of finding the tangent direction resulted in a very significant improvement. The angle between the approximated tangent direction and the real direction is plotted in Figure 9. In this figure, the two techniques are compared. The tangent directions were calculated at about 50 points on the specific edge. The angles for the old technique are very scattered and the deviation from the real tangent is very high. The improvement with the new method is clearly visible.

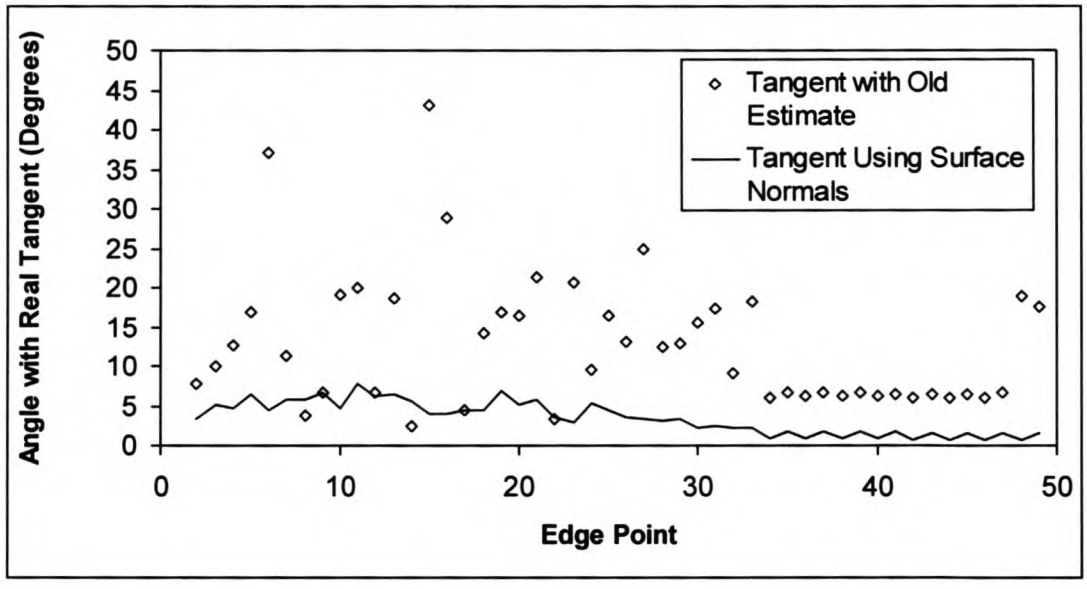


Figure 9 Comparison of Tangent Vector Estimates.

This method often fails when the scan lines are much longer than they are apart, i.e. the ratio A/ρ_e is very large. This means that a flat surface is fitted to points that are almost co-linear. The approximation of the surface as flat might also be over simplified.

4.4.1.3. Cross Product of the Surface Normals Using Polynomial Approximations

A very accurate approximation of the tangent vector is

$$\boldsymbol{t}_j = \frac{\boldsymbol{n}_j^1(u) \otimes \boldsymbol{n}_j^2(1-u)}{\left\|\boldsymbol{n}_j^1(u) \otimes \boldsymbol{n}_j^2(1-u)\right\|}, \quad u=1 \qquad (4.4.1.3.1.)$$

$\boldsymbol{n}_j^k(u)$ are the unit surface normal vectors at the parameter u of the polynomial curve approximating the scan line sections. The nomenclature used in this and the following equations is explained in the following figure. Two polynomials are shown. They can be of any degree. These are the curves as they are fitted to the line section, thus they do not meet at the edge point. The curve on the left was scanned first (the superscript 1 is used to indicate this). In the following equations the superscript k replaces the superscripts *1* and *2* in the above equation. The superscript k is used to indicate the specific scan line section of the j'th scan line. Thus $\boldsymbol{p}_j^k(u)$ is the polynomial fitted to the k'th scan line section of the j'th scan line. The parameterisation of the curves is such that the parameter at the first point is 0 and the last point is 1. The subscript j indicates that this curve belong to the *j'th* scan line. The unit normal vector to the surface at u is $\boldsymbol{n}_j^k(u)$. The subscript and superscript have the same meaning. This nomenclature is illustrated in Figure 10. In this figure the scanning was done from left to right.

The cross product of the two unit surface normal vectors at the edge has the same direction as the edge's tangent vector. Instead of a flat surface approximation, the curves that were fitted to the scan line sections are used. The surface normal is taken as the cross product of the tangent vector of the polynomial curve with the vector connecting the end points of the last two polynomial curves. This is done in the next equation.

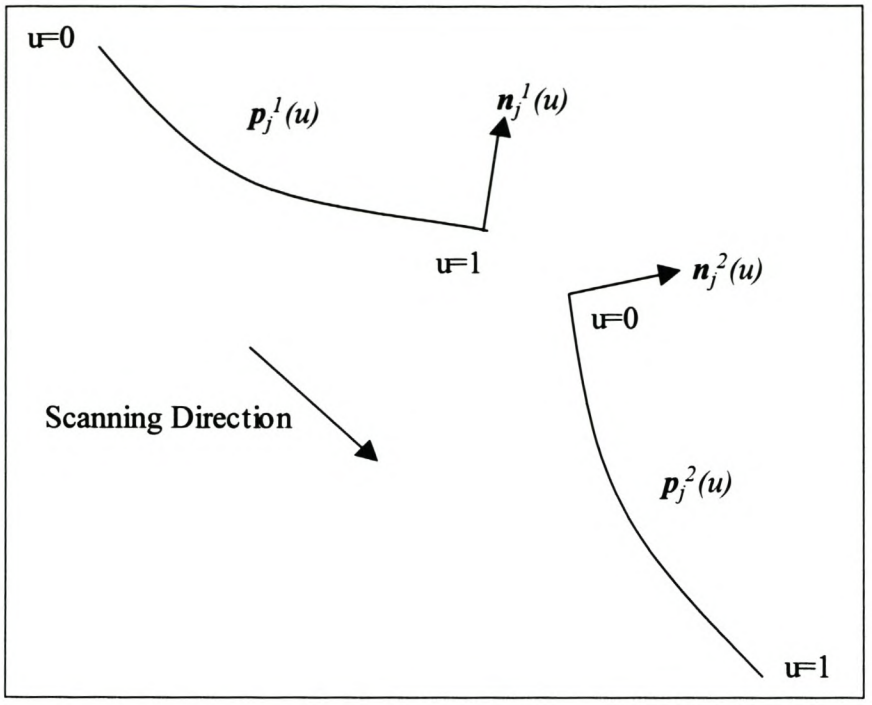


Figure 10 Polynomial Nomenclature.

$$\boldsymbol{n}_j^k(u) = \frac{\left(\frac{d}{du}\boldsymbol{p}_j^k(u)\right) \otimes \left(\boldsymbol{p}_j^k(u) - \boldsymbol{p}_{j-1}^{k'}(1-u)\right)}{\left\| \left(\frac{d}{du}\boldsymbol{p}_j^k(u)\right) \otimes \left(\boldsymbol{p}_j^k(u) - \boldsymbol{p}_{j-1}^{k'}(1-u)\right) \right\|}, \quad k=1,2 \qquad (4.4.1.3.2.)$$

with

$k'=1$, $u=0$ if $k=2$ and $k'=2$, $u=1$ if $k=1$

The following graph shows an instance when this method produced a significantly better result than the method of using flat surface approximations. This method provided the most robust results. For this reason it is the preferred method used in this project.

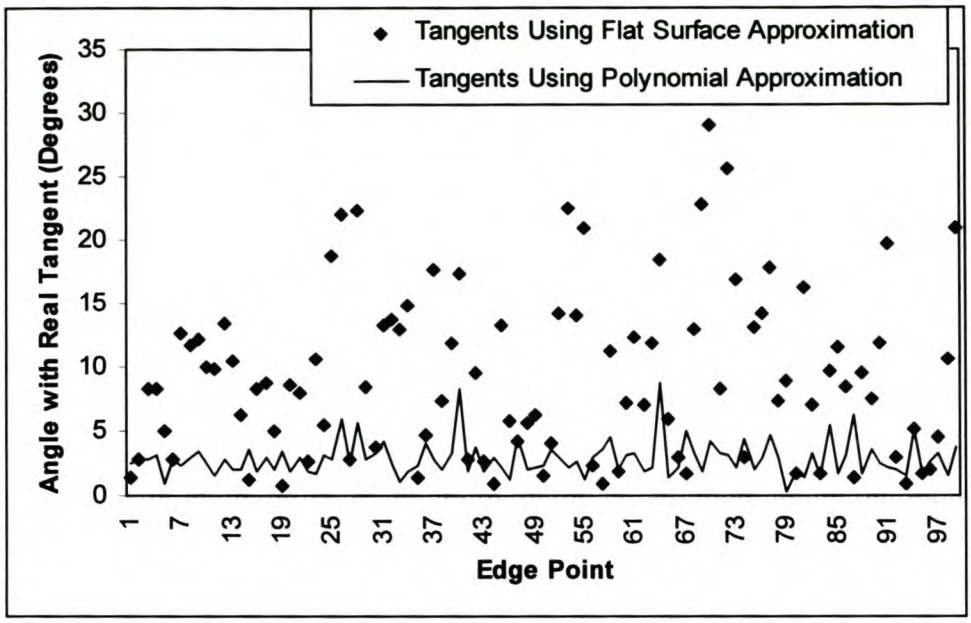


Figure 11 Comparison of Tangent Vector Estimates.

4.4.2. Tangent Extrapolation

The simplest method of steering the pattern is to follow the tangent direction. Using the tangent direction and the required edge pitch, the position of the next edge point is guessed and the virtual CMM will scan the next line of the pattern towards this point (equation 4.4.2.1.)

$$\bar{\boldsymbol{p}}'_{j+1} = \bar{\boldsymbol{p}}_j + \rho_e \boldsymbol{t}_j \qquad (4.4.2.1.)$$

Since the estimates of the tangent direction and the edge point are very robust, this method gives a robust pattern direction. However, where the curvature of the edge is high, the pattern does not follow the edge closely. Extending the scan lines (discussed in 4.2.3.) and changing the edge pitch (discussed in 4.4.4.) ensure that the pattern continues following the edge even in regions of high curvature.

4.4.3. Curvature Based Extrapolation

A curvature based method for steering the pattern was tested. A circle is interpolated through the last three scanned points and then the next edge point is estimated to be on the circle at a distance equal to the edge pitch away from the last edge point. This method proved to be highly sensitive to the accuracy of the calculated edge points. In

fact, it was not possible to obtain a successfully scanned edge using this method. It was thought that the method would be more robust if the edge points are projected onto the osculating plane of the edge curve. (See Do Carmo, 1976, pp 17 for a definition of the osculating plane of a curve.) However, a robust method of finding the osculating plane was not found. Therefore, the method of curvature based extrapolation was discarded due its sensitivity to errors in the edge point calculation.

4.4.4. Edge Pitch Adjustment

In order to optimise the scanning time the edge pitch can be adjusted according to the edge curvature. Fewer points can be scanned on sections of the edge where the edge curvature is small. Similarly, the pitch can be decreased when the curvature becomes very high in order to improve the definition of the edge. This will also improve the robustness of the algorithm as described in paragraph 4.4.2.

The chordal deviation is used as an indication of the local edge curvature. The chordal deviation can be defined as the shortest distance from a point B to the line connecting points A and C. The chordal deviation of the three edge points can be found as follows (referring to Figure 12.)

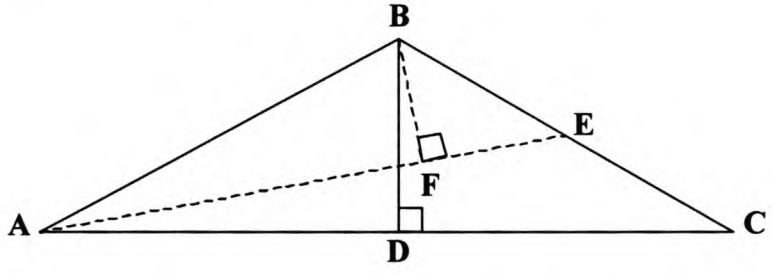


Figure 12 Determination of Chordal Deviation.

Let A, B and C be the last three edge points. $\|\boldsymbol{DB}\|$ is the chordal height for these points. $\|\boldsymbol{BF}\|$ is the prescribed chordal height. $\|\boldsymbol{DB}\|$ can then be found by using the cross product to find the sine of the angle *BAD*.

$$\|\boldsymbol{DB}\| = \frac{\|\boldsymbol{AB} \otimes \boldsymbol{AC}\|}{\|\boldsymbol{AC}\|} \qquad (4.4.4.1.)$$

The ratio of $\|\boldsymbol{DB}\|$ to $\|\boldsymbol{BF}\|$ is used to adjust the pitch. The pitch, $\|\boldsymbol{BE}\|$, that would have resulted in the correct chordal height can easily be derived from Figure 12. However, the pitch that is thus calculated cannot be used as the new pitch. The triangle ABC in Figure 12 is normally very slender, i.e. $\angle ABC$ is very close to 180°. This means that the value determined for $\|\boldsymbol{BE}\|$ is very sensitive to errors in the calculation of the points A B and C. For this reason, the ratio of $\|\boldsymbol{DB}\|$ to $\|\boldsymbol{BF}\|$ is rather used as an indication of the adjustment that must be made to the edge pitch.

The adjustment is further limited to ensure that the pattern does not suddenly make a very big or very small step compared to the previous step. Currently the adjustment ratio is limited between 0.5 and 3. These values seem to balance the requirements for a fast scan and a robust scan.

When scanning on a point cloud, it is also necessary to check that the edge pitch does not decrease below some multiple of the cloud pitch. When scanning diagonally across a regular grid, the distance between consecutive points would be $\sqrt{2}$ times the cloud pitch if it is assumed that the grid has the same pitch in the two grid directions. (See Figure 13.) It makes no sense to scan at a pitch less than the distance between these points. Therefore the pitch is further limited to be no less than 1.5 times the cloud pitch, which is on the safe side of $\sqrt{2}$ times the cloud pitch.

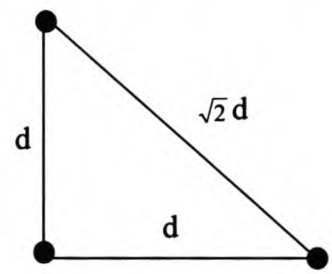


Figure 13 Maximum Distance between Points in a Regular Grid.

Sometimes it happens that the zigzag pattern degenerate due to a systematic deviation from the desired scan lines. This is illustrated in Figure 14. Ideally, the zigzag pattern should look like the isosceles triangle ABC. The virtual CMM cannot follow the desired scan direction exactly due to the discrete nature to the cloud. It sometimes happens that a systematic deviation to the same side of the desired scan line occurs at the end point of the scan line, as shown in the figure. The result is that the pattern

degenerates unless it recovers by itself due to a deviation to the other side of the desired scan line. This can happen rather quickly, after only a few scan lines, especially if the cloud pitch is large compared to the pattern's amplitude.

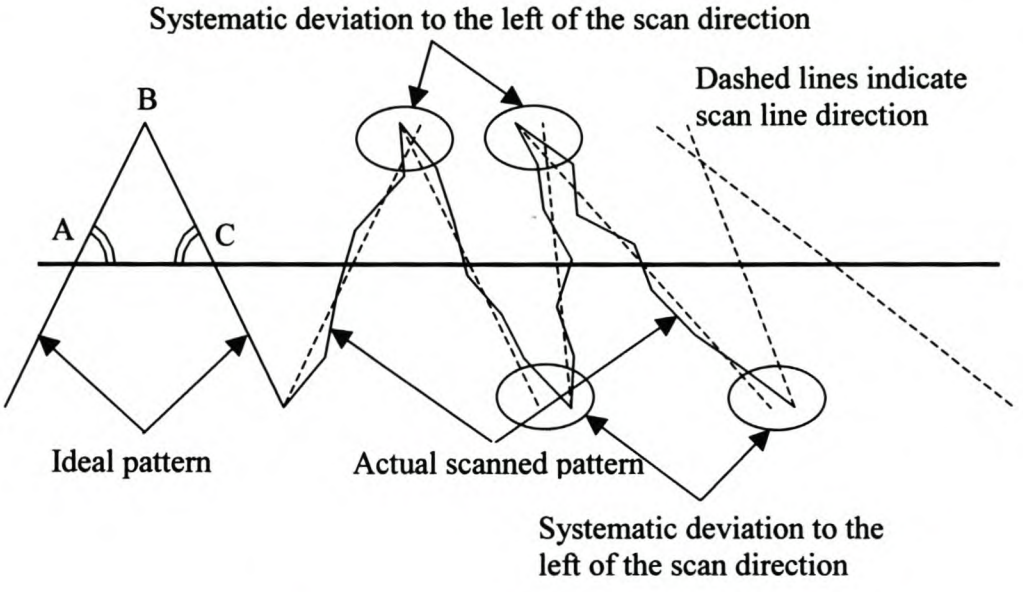


Figure 14 Degenerate Zigzag Pattern.

4.5. *Probe Radius Compensation*

When scanning with a probe of finite radius, the points must be compensated with the radius before the edge can be calculated. Experimentation showed that the accuracy of the edge is very dependent on the accuracy and robustness of the compensation method. Furthermore, the compensation must be done very carefully since points on two surfaces in a small region around a sharp edge are used to calculate the edge. Three alternative methods were tested. But, first it will be shown that the scan line sections do not necessarily intersect once they are compensated.

4.5.1. Why Line Sections do not Intersect

In order to accurately compensate the scanned points for the probe radius, the compensation on each line section is done separately. This, however, causes a problem for calculating the edge points. The compensated points do not lie in the same plane and neither do the polynomial curves that are fitted to the compensated

points. Therefore, the curves will not intersect each other. The problem is illustrated in the figure below.

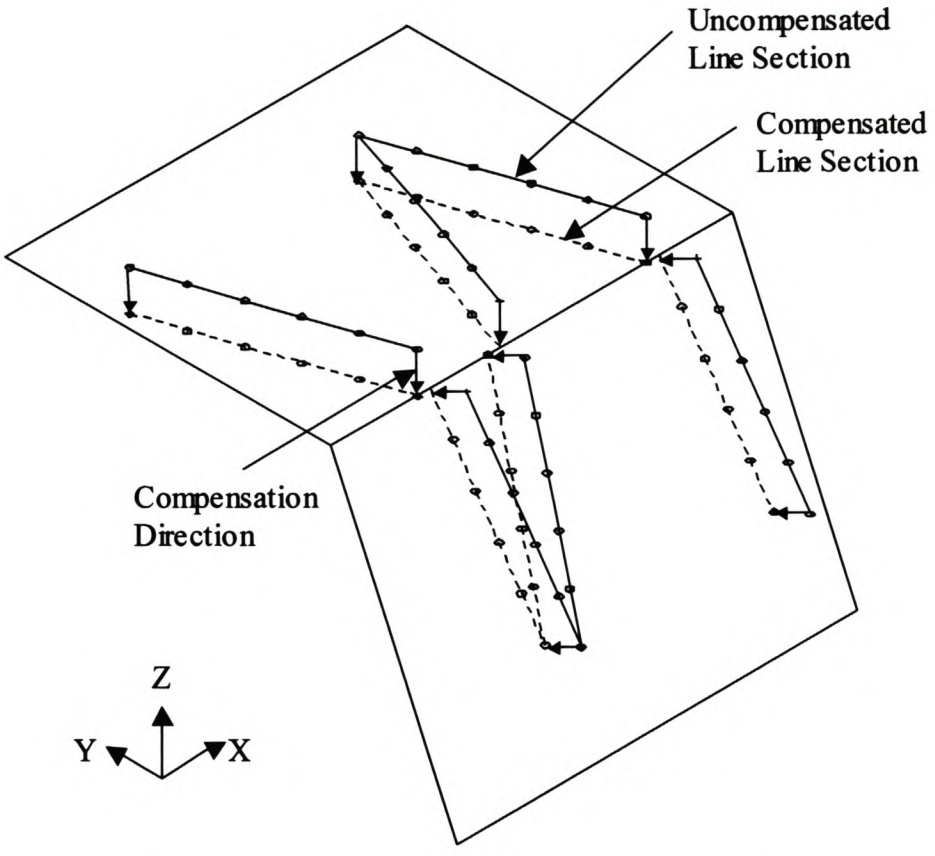


Figure 15 Errors Due to Probe Radius Compensation.

4.5.2. Point Compensation

Early testing of the edge scanning method was done with a point compensation method (Schreve and Basson, 2000). With this method, the local surface normal vector must be calculated for each point. The local surface unit normal of a parametric surface is found from equation 4.5.2.1.

$$\boldsymbol{n}(u,v)=\frac{\dfrac{\partial \boldsymbol{S}(u,v)}{\partial u}\otimes\dfrac{\partial \boldsymbol{S}(u,v)}{\partial v}}{\left\|\dfrac{\partial \boldsymbol{S}(u,v)}{\partial u}\otimes\dfrac{\partial \boldsymbol{S}(u,v)}{\partial v}\right\|} \qquad (4.5.2.1.)$$

The local surface unit normal for the *i'th* point on the *j'th* scan line can be calculated by replacing the partial derivatives in equation (4.5.2.1.) with central differences as follows.

$$\frac{\partial \boldsymbol{S}(u,v)}{\partial u} = \boldsymbol{p}_{i+1,j} - \boldsymbol{p}_{i-1,j} \qquad (4.5.2.2.a)$$

$$\frac{\partial \boldsymbol{S}(u,v)}{\partial v} = \boldsymbol{p}_{i,j+1} - \boldsymbol{p}_{i,j-1} \qquad (4.5.2.2.b)$$

In equation (4.5.2.2.) $\boldsymbol{p}_{i+1,j}$ and $\boldsymbol{p}_{i-1,j}$ are the immediate neighbours on the same scan line of the point, $\boldsymbol{p}_{i,j}$. $\boldsymbol{p}_{i,j+1}$ and $\boldsymbol{p}_{i,j-1}$ are the closest points to $\boldsymbol{p}_{i,j}$ on the *j+1'th* and *j-1'th* scan lines. The cross product of the vectors constructed with these points is a good approximation of the local surface normal vector provided that the distances from the point $\boldsymbol{p}_{i,j}$ to be compensated to the four other points ($\boldsymbol{p}_{i,j+1}$, $\boldsymbol{p}_{i,j-1}$, $\boldsymbol{p}_{i+1,j}$ and $\boldsymbol{p}_{i-1,j}$) are the same. The method is illustrated in Figure 16. The compensation vector is found by multiplying the unit normal with the probe radius.

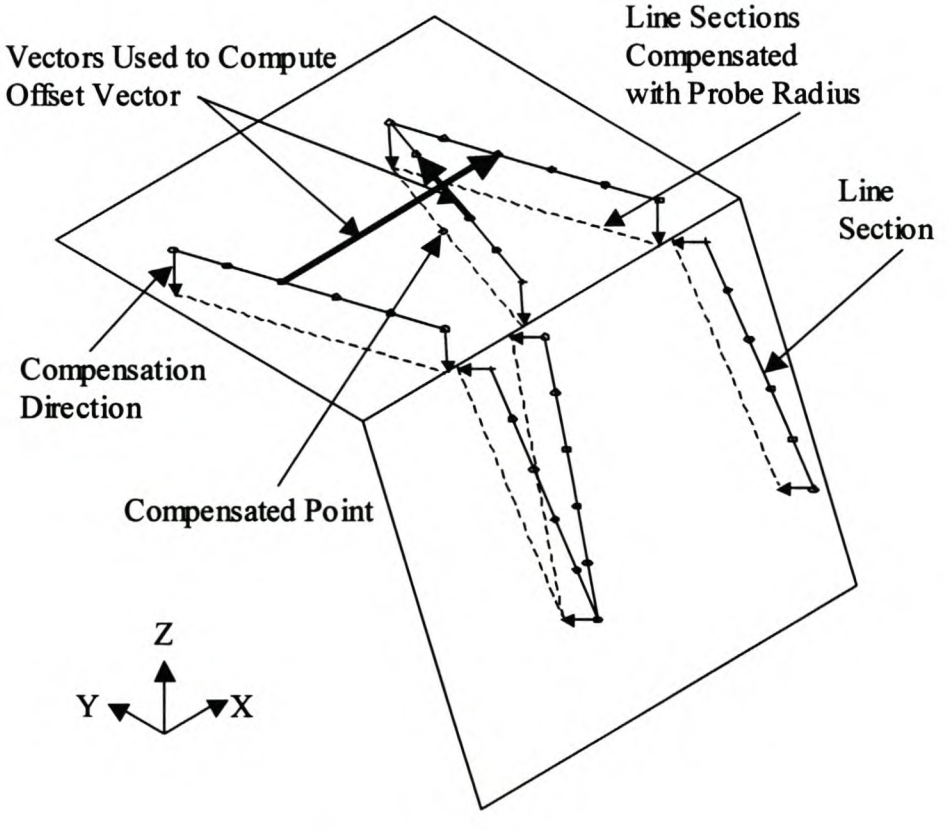


Figure 16 Point Compensation Method.

This method implies that the points on each line section are first compensated with the probe radius and then the curves are fitted which are used to calculate the edge points. Very good results are obtained with this method provided that the distance between consecutive points on the line sections are significantly more than the measuring error. Thus, the estimate of the local surface normal vector is very sensitive to noise in the data.

Another source of error is the fact that the two line sections do not meet at the same point on the edge. This problem is discussed in paragraph 4.5.1. It has the same result as the problem discussed in paragraph 4.3.1. and Figure 7, but the cause is completely different. In paragraph 4.3.1. the lines do not meet because the points do not lie in the scanning plane. Even if the points are projected onto the scanning plane, the two line sections still will not meet at the edge after radius compensation, because of the reason discussed in paragraph 4.5.1.

4.5.3. Curve Compensation

Due to the sensitivity of the point compensation method to noise in the data, a curve compensation method was also tested. This implies that the curves are fitted to the uncompensated points and then the curve is compensated with the probe radius.

4.5.3.1. Compensation Plane

In order to compensate the curves in R^3, a plane must be defined on which the compensation will be done. The curves of both line sections are compensated on the scanning plane that is fitted to the scan line. This means that a small error is made if the scan line is not perpendicular to the edge. This error is described in paragraph 4.5.1. Note that the point compensation method does not have this problem, but the error then occurs when projecting the points on the scan plane before calculating the curves. Thus, this error is unavoidable with these methods, but a careful selection of the scanning parameters can minimise this error.

4.5.3.2. Line Compensation

A line is easy to compensate, since only the line origin has to be moved, the line direction remains unchanged. If $\boldsymbol{x}=\boldsymbol{s}_l+\lambda\boldsymbol{a}$ is the definition of the line, $\boldsymbol{n}$ is a normal

vector to the scanning plane, then the compensated line, x', can be found from equation 4.5.3.2.1.

$$x'=s_l+\lambda a+\frac{a\otimes n}{\|a\otimes n\|}R_p \qquad (4.5.3.2.1.)$$

4.5.3.3. Polynomial Compensation

If $p_j^k(u)$ is a point on a polynomial curve, $t_j(u)$ is the tangent to the curve at the point, n is a normal vector to the scanning plane, then a corresponding point on the compensated curve can be found with equation 4.5.3.3.1.

$$p'^k_j(u)=\frac{t_j(u)\otimes n}{\|t_j(u)\otimes n\|}R_p+p_j^k(u) \quad k=1,2 \qquad (4.5.3.3.1.)$$

This equation looks similar to equation 4.5.3.2.1., but it should be noted that the curve obtained from compensating a polynomial curve in this way is not a polynomial anymore. However, the same numeric procedure used to calculate the intersection of the polynomials (see paragraph 4.3.5.2.) can be used to calculate the intersection of the compensated curves.

4.5.4. Intersection Curve Compensation

The above two methods both suffer from the fact that the two compensated line sections do not meet each other at the edge. The reason, as discussed earlier, is that the plane in which the points lie is not perpendicular to the edge. The intersection curve method calculates new curves that lie in a plane that is perpendicular to the edge. It does this by using the curves of the neighbouring scan lines as rails for a ruled surface. The curve that is formed by the intersection of the ruled surface and the perpendicular plane is then used to calculate the edge.

The direction of the edge's tangent vector is the desired orientation of the perpendicular plane. The tangent vector can be found using equation 4.4.1.3.1., but a better approximation of the surface normal is

$$\boldsymbol{n}_j^k(u) = \frac{\left(\frac{d}{du}\boldsymbol{p}_j^k(u)\right) \otimes \left(\boldsymbol{p}_{j+1}^{k'}(1-u) - \boldsymbol{p}_{j-1}^{k'}(1-u)\right)}{\left\| \left(\frac{d}{du}\boldsymbol{p}_j^k(u)\right) \otimes \left(\boldsymbol{p}_{j+1}^{k'}(1-u) - \boldsymbol{p}_{j-1}^{k'}(1-u)\right) \right\|}, \quad k=1,2 \qquad (4.5.4.1.)$$

with

$k'=1, u=0$ if $k=2$ and $k'=2, u=1$ if $k=1$

The above equation uses a central difference to calculate the derivative in the direction parallel to the edge. This is a good approximation as long as the distances between points $\boldsymbol{p}_{j+1}^k(1-u)$ and $\boldsymbol{p}_j^k(u)$ and points $\boldsymbol{p}_{j-1}^k(1-u)$ and $\boldsymbol{p}_j^k(u)$ are the same. This is the case close to the edge where this distance is equal to the edge pitch. In the case of the zigzag pattern this will be a very bad approximation at the start and end points of the scan lines. However, the calculation of the edge point is not done in this region, but rather in the region near the edge where the distances are equal (the curves are extrapolated to find their intersection), so the approximation is a good one.

The position of the perpendicular plane must be on the edge point, but this point is not known beforehand. It is sufficient to position the perpendicular plane halfway between the endpoints of the two line sections, as shown in the following equation.

$$\boldsymbol{b}_j = \frac{\boldsymbol{p}_j^1(1) + \boldsymbol{p}_j^2(0)}{2} \qquad (4.5.4.2.)$$

The position and orientation determines the perpendicular plane.

The nomenclature is illustrated in Figure 17. Here, only the compensated curves are shown for the sake of clarity. The ruled surface that is used to calculate the intersection curve is shown in thin solid lines. The intersection curve is shown as a dashed curve. The perpendicular plane that is used to calculate the intersection curve is not shown. The origin of this plane is at the point $\boldsymbol{b}_j$ and the orientation is the tangent vector to the edge, $\boldsymbol{t}_j$.

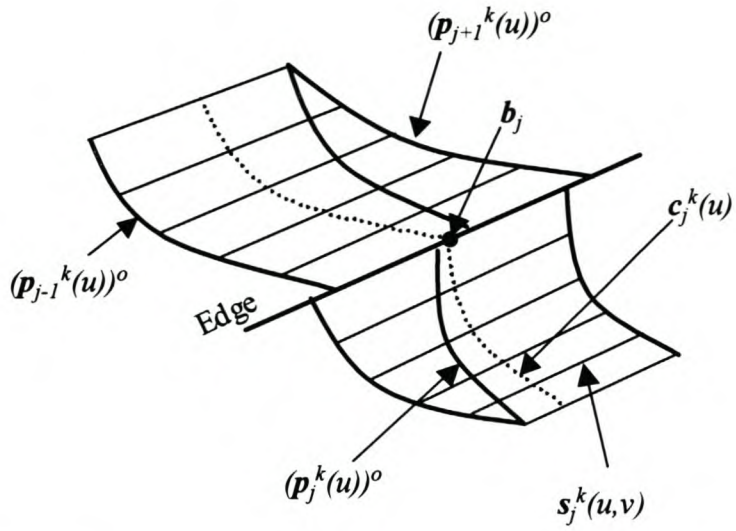


Figure 17 Illustration of Intersection Curve Compensation Method.

Any point on the compensated curves can now be found with the following equation. (The superscript ^{o} in this and following equations indicates the compensated curve.)

$$\left(\boldsymbol{p}_j^k(u)\right)^o = \boldsymbol{n}_j^k(u)\cdot R_p + \boldsymbol{p}_j^k(u) \qquad (4.5.4.3.)$$

The surface between the k'th line section of the j-1'th and j+1'th compensated scan lines is now approximated as a ruled surface. The edge is found by calculating the intersection of the curves that is formed by the intersection of the perpendicular plane and the ruled surface. Points on this curve are found by calculating the intersection of lines in the rule direction and the perpendicular plane. The ruled surface is given by the following equation.

$$\boldsymbol{s}_j^k(u,v)=\left(\boldsymbol{p}_{j-1}^{k'}(1-u)\right)^o+v\left[\left(\boldsymbol{p}_{j+1}^{k'}(1-u)\right)^o-\left(\boldsymbol{p}_{j-1}^{k'}(1-u)\right)^o\right]\ (u,v=0..1) \qquad (4.5.4.4.)$$

with

k'=1 if k=2 and k' =2 if k=1

Any line in the rule direction is found by keeping the parameter u constant. The parameter in the rule direction is v. Any point on the curve formed by the intersection of the ruled surface and the perpendicular plane is calculated by taking the intersection of the lines in the rule direction with the plane. The equation for

calculating the intersection of a line and a plane is given in Appendix A. The intersection curve is

$$\boldsymbol{c}_j^k(u) = \left(\boldsymbol{p}_{j-1}^{k'}(1-u)\right)^o + \frac{\boldsymbol{t}_j \cdot \left(\boldsymbol{p}_{j-1}^{k'}(1-u)\right)^o - \boldsymbol{t}_j \cdot \boldsymbol{b}_j}{\boldsymbol{t}_j \cdot \left(\left(\boldsymbol{p}_{j+1}^{k'}(1-u)\right)^o - \left(\boldsymbol{p}_{j-1}^{k'}(1-u)\right)^o\right)} \left(\left(\boldsymbol{p}_{j+1}^{k'}(1-u)\right)^o - \left(\boldsymbol{p}_{j-1}^{k'}(1-u)\right)^o\right)$$

(4.5.4.5.)

with

k'=1, u=0 if k=2 and k' =2, u=1 if k=1

Note that the above equation is derived in terms of the parameters of the polynomials representing the original, uncompensated points. The intersection point, or edge point, is the intersection of the curves $\boldsymbol{c}_j^1$(u) and $\boldsymbol{c}_j^2$(u). Powell's method (Press et al. 1997) is used to find the intersection points.

4.5.5. Discussion

In the figures below, the three probe radius compensation methods are compared. (The results are for the Experiment 1 with a square quadratic scan in Appendix E.) The three methods follow the same trend, except for tests 6, 15 and 16, where the point compensation method compares badly. This is due to the noise sensitivity of the method as explained earlier. If the data is good, the point compensation method can produce very good results, as indicated in experiment 12. However, since the method cannot produce consistent and reliable results, it is not useful for engineering purposes.

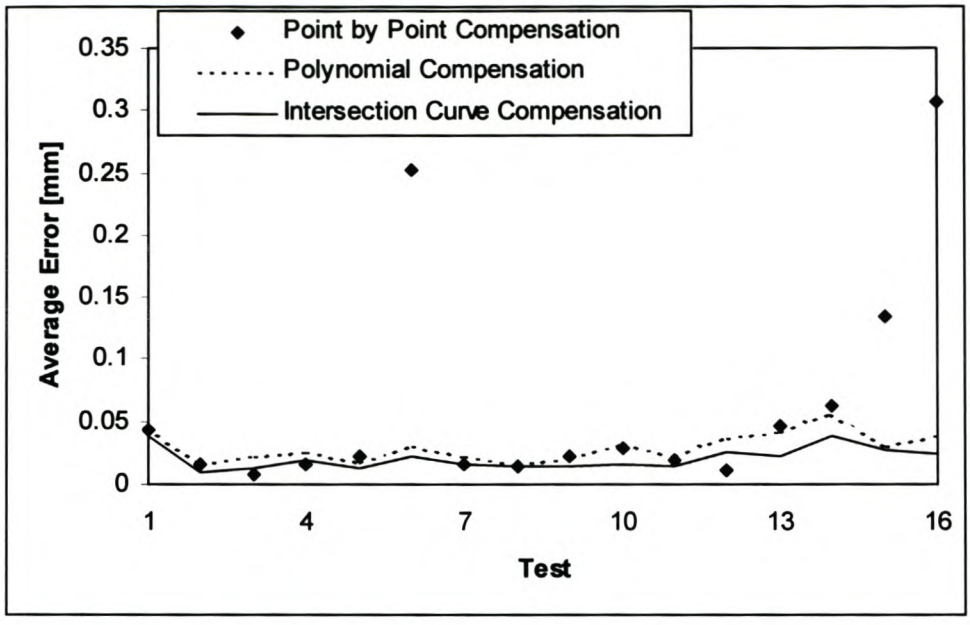


Figure 18 Comparison of Average Errors of the Compensation Methods.

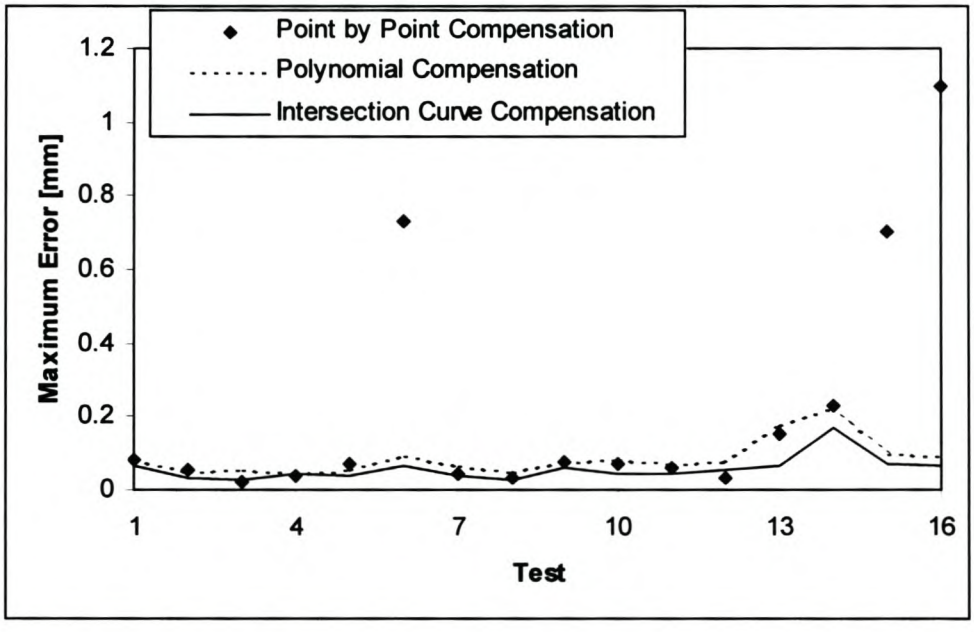


Figure 19 Comparison of Maximum Errors of the Compensation Methods.

The intersection curve compensation method is consistently better than the other methods, because the two compensated curves meet at the edge. The quadratic polynomials are only an approximation of the actual surface. Furthermore, the polynomials must be extended over the gap between the two surfaces caused by the fillet radius. It cannot be guaranteed that this quadratic extrapolation will follow the

actual surface. Finally, there was measuring noise of between 0.01mm and 0.02mm in all the tests. These factors all contribute to the errors observed in the experiments.

Due to the combination of robustness and accuracy, the intersection-curve method is the best way to compensate the points for the probe radius and calculating the edge points.

4.6. Start and End Conditions

In addition to specifying all the scanning parameters such as the edge pitch, amplitude, line pitch, etc., the user must specify a start point and scanning direction. The start point will be the first point on the first scan line. The scanner will scan the first line in the prescribed scanning direction. The orientation of the first scanning plane is determined by this initial scanning direction and the orientation of the probe. The scanner must also know if the edge lies in the direction of the scanning plane normal vector or in the opposite direction. These parameters can easily be determined by scanning three points on the object, the first being the start point, the second fixing the scanning direction and the third indicating the direction of the edge.

The scanner must also know where to stop. A gate is defined by an end point and an end direction. The width of the gate is equal to the amplitude. As soon as the calculated edge passes through the gate, the scanning will stop. The gate is defined by scanning two points on the object, the first point is the end point and the second determines the gate direction.

4.7. Error Handling

This paragraph does not describe computer or programming related error handling such as checking the type of input parameters, it rather describes the way the algorithm handles errors related to the scanning process itself. If any of these errors occur, it will not be possible the continue scanning the edge. Whether the virtual CMM should be stopped at once if such an error occurs, is debateable since it may be possible to find some way around the error and still continue scanning. However, the philosophy during the development of the algorithm was that it is better to stop and allow the user to make a decision about correcting the error.

4.7.1. Unacceptable Input Parameters

When using the virtual CMM to scan on a point cloud, it does not make sense to specify an edge pitch or line pitch that is less than the point cloud density. Experimentation has indicated that the line pitch should preferably be at least 1.5 times the point cloud pitch if it can be assumed that the point cloud has uniform density. If the point cloud's density is non-uniform, it would be best to specify a line and edge pitch that is more than the largest gap in the point cloud.

There must be enough points in each line section to do the curve approximation. This means that the following inequality must be satisfied, where q is the degree of the curve that will approximate the line section.

$$\frac{A}{2\rho_L} > q+1 \qquad (4.7.1.1.)$$

The ratio of the amplitude to the line pitch indicates the number of points there will be on the scan lines, so half that ratio must be the number of points on the line section. Normally, there must be more than A/ρ_L points per scan line, because some points will always be removed from the scan line during the line splitting and cleaning operations. The above ratio serves as minimum ratio. A more accurate ratio can be obtained by taking the fillet radius, probe radius and intersection angle into account.

4.7.2. CMM and Scanning Errors

As stated in Chapter 3, the virtual CMM used here simulates non-measuring moves as well and therefore the edge scanning algorithm must be able to handle "collisions" that occur during these movements. When scanning an unknown surface, collisions often happen, but they can be minimised by selecting good scanning parameters. This largely depends on the experience of the operator. When the virtual CMM collides, this algorithm will try to re-measure a point at the point of collision. In this way the scanner should be able to complete the scan line.

If the calculation of the tangent vectors becomes unstable, there is a big risk that the algorithm will try to steer the scanner far from the actual edge. The scanner will start

moving outside the region that the operator tried to specify. For this reason, the algorithm will stop immediately if a collision occurs while the scanner is trying to measure the first point of the scan line. This will stop the virtual CMM before it moves outside the unspecified, and thus unsafe, region.

It can happen that the virtual CMM selects the same point twice. The risk of this error increases if the point cloud pitch and line pitch are approximately the same. Since the algorithm uses the last two points on the scan line to estimate the next point on the scan line, it will fail under these circumstances. One way of solving the problem would be to increase the line pitch and try again to measure a new point. However, increasing the line pitch holds some risk of collisions since the operator probably selected the maximum pitch with which the scanning can safely be completed. Therefore, the algorithm will stop scanning and display an error message.

4.7.3. Errors Prohibiting the Calculation of the Edge Points

The calculation of the edge points requires that the curves representing the line sections are good approximations. Of course, enough points are needed to fit the curves, in the case of a line at least two points and at least three points in the case of a quadratic polynomial. The algorithm will try to ensure that there are always enough points on the scan lines to estimate the curves by scanning additional points if necessary and splitting the scan line so that there are enough points on both line sections.

If the algorithm is unable to scan additional points, because it reached the end of the search region, it can try to reduce the degree of the polynomial so that an edge point can still be calculated. If the user chose to use line approximations, this is not possible. The algorithm then stops and displays an error message. One can argue that it might be better to move outside the search region so that one or two more points can be scanned simply to keep the virtual CMM going. However, it is the author's opinion that it would be safer, and it would ultimately produce more accurate results, if the virtual CMM is stopped immediately with an appropriate error message. The user can then specify better scanning parameters and repeat the scan.

4.8. Limitations of the Edge Scanning Method

There is a number of instances when this algorithm will fail to detect an edge. These instances are described in the following figures. Where necessary, the results of quantitative analysis into these limitations are presented in a later chapter.

- When scanning in a section of the edge with high curvature and a sharp angle between the two intersecting surfaces, it is possible that the scanner will be unable to remain on the object (Figure 20). Increasing the *search distance* can partly solve the problem, but this is not always feasible. (The *search distance* is the distance that the scanner will move beyond the point where the scanner expects to make contact with the object before it will give up the search for the object.) The first time that the scanner reaches the end of the search distance, it will try to correct itself by trying to scan a point between the last point of contact and the position of the probe at the end of the search movement. If the virtual CMM again does not make contact with the object, the algorithm will stop the scanning and display an error message.

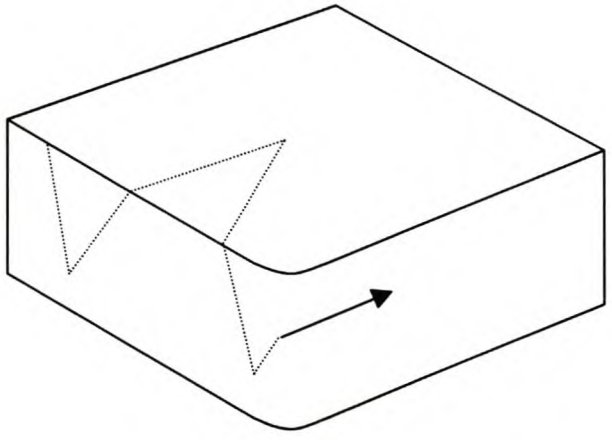


Figure 20 Scanning beyond Search Envelope.

- It must be possible to specify a scanning region that does not contain any other sharp edges. Should the scanner intersect another edge, the results will firstly be very inaccurate and secondly, it is possible that the algorithm will start following the wrong edge (Figure 21). This might be a problem when trying to find the boundaries of thin walled objects.

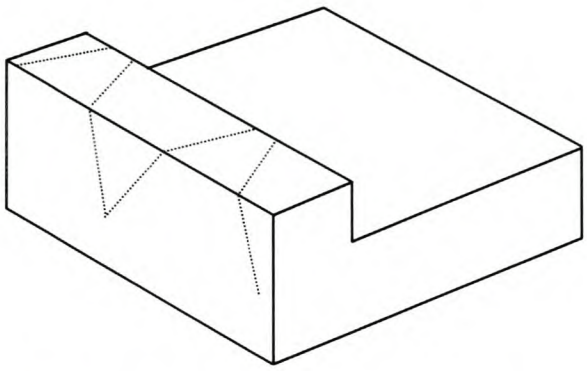


Figure 21 Error Due to Scanning on Multiple Edges.

- When a bad selection of scanning parameters is made, it is possible that a line will be scanned that never intersects the edge (Figure 22). In its current form the algorithm will detect that the scanner did not intersect the edge. An error message will be returned and the scanning will stop. No corrective measures are taken, because a similar problem will occur if the intersection angle between the surfaces becomes very blunt. In the latter case, corrective measures might steer the scanner away from the edge. The best way of avoiding this kind of problem is to follow the guidelines for specifying the scanning parameters. The selection of the *edge pitch* will be very important in this case.

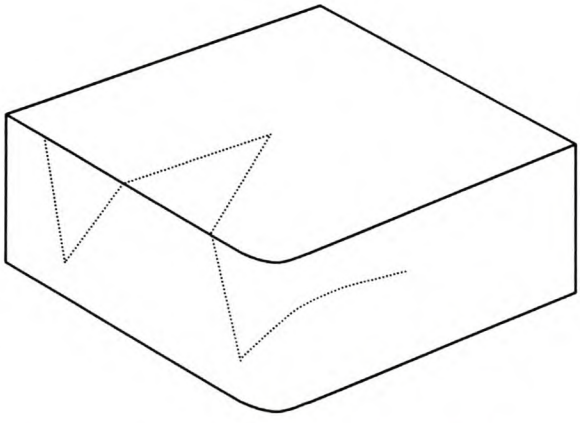


Figure 22 Error Due to Smooth Edge.

- The algorithm can only detect an edge curve that is itself at least C^1 continuous (Figure 23). If a vertex on the edge is reached it will not be possible to accurately calculate the edge point and there is the further risk that the scanner will start following the wrong edge. Since the algorithm is able to scan filleted edges, the

same problem might occur when there is a section of the edge with very high curvature.

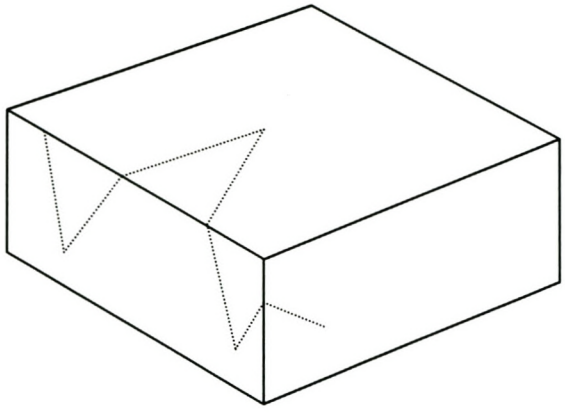


Figure 23 Scanning the Wrong Edge.

Chapter 5.

Analytical and Experimental Evaluation

5.1. Object Model Used for Testing

The torus object described in Chapter 3, and discussed in detail in Appendix G, is used in all the tests of the edge scanning algorithm. Point clouds are generated from this object so that the distance between the points in the parametric directions is always at the specified point cloud pitch. With this model it is easy to test all the parameters that govern the algorithm. The experiments can be automated with little difficulty. The surface curvature is constant in the direction perpendicular to the edge. The edge curvature is also constant. This is very convenient because it simplifies the interpretation of the results.

5.2. Analytical Accuracies

In this section a simple analytical model is derived to study the edge scanning algorithm's analytical performance. The influence of the main scanning parameters is explained and finally the analytical results are compared with actual edge scanning results.

5.2.1. Analytical Error Model

Some simplifying assumptions are necessary to derive the analytical error model. It is assumed that the surface on which the scanning is done is a perfect cylindrical section. This means that the principle curvatures are constant. The curve that best approximates the scan line section is found by approximating the continuous line section rather than a discrete line section. In other words, instead of approximating a number of measured points by minimising the square of the distance between the curve and the points, the square of the distance between the curve and the arc is minimised. This will give the polynomial curve that best represents a given arc.

Measuring noise is not included. It is also assumed that the scanning plane is perpendicular to the edge. This means that the scan line sections are perfect arcs. The analysis is done in two dimensions, since it is only necessary to calculate the intersection point of the two curves. Therefore, the subscripts used in the previous chapter are dropped.

Figure 1 illustrates the object and approximating arcs that is used to derive the analytical model. The object is hatched. The path of the probe ball centre is shown only for the upper cylindrical surface. The curve that approximates the path of the probe ball centre is shown as a dashed line. The unconventional orientation of the probe is chosen so that the angle γ starts at the 0 for the arc described by the probe ball centre. This simplifies the subsequent derivation of the analytical model. Since the object is symmetrical about the line AB, the calculated edge point is the intersection of the compensated curve and the line AB. The radius of the probe ball is R_p and the fillet radius between the two cylindrical surfaces is R_f.

The angle ϕ, which is half the angle of the fillet arc, is equal to

$$\phi = \arcsin\left(\frac{R\sin\left(\frac{\pi-\theta}{2}\right)}{R+R_f}\right) \qquad (5.2.1.1.)$$

The angle α is the arc angle that the circle segment must extend over the gap formed by the fillet radius to the intersection point.

$$\alpha = \arccos\left(\frac{R\sin\left(\frac{\pi-\theta}{2}\right)}{R+R_f}\right) - \frac{\theta}{2} \qquad (5.2.1.2.)$$

With α known, the actual intersection point is

$$\bar{p} = \begin{pmatrix} R\cos\alpha \\ -R\sin\alpha \end{pmatrix} \qquad (5.2.1.3.)$$

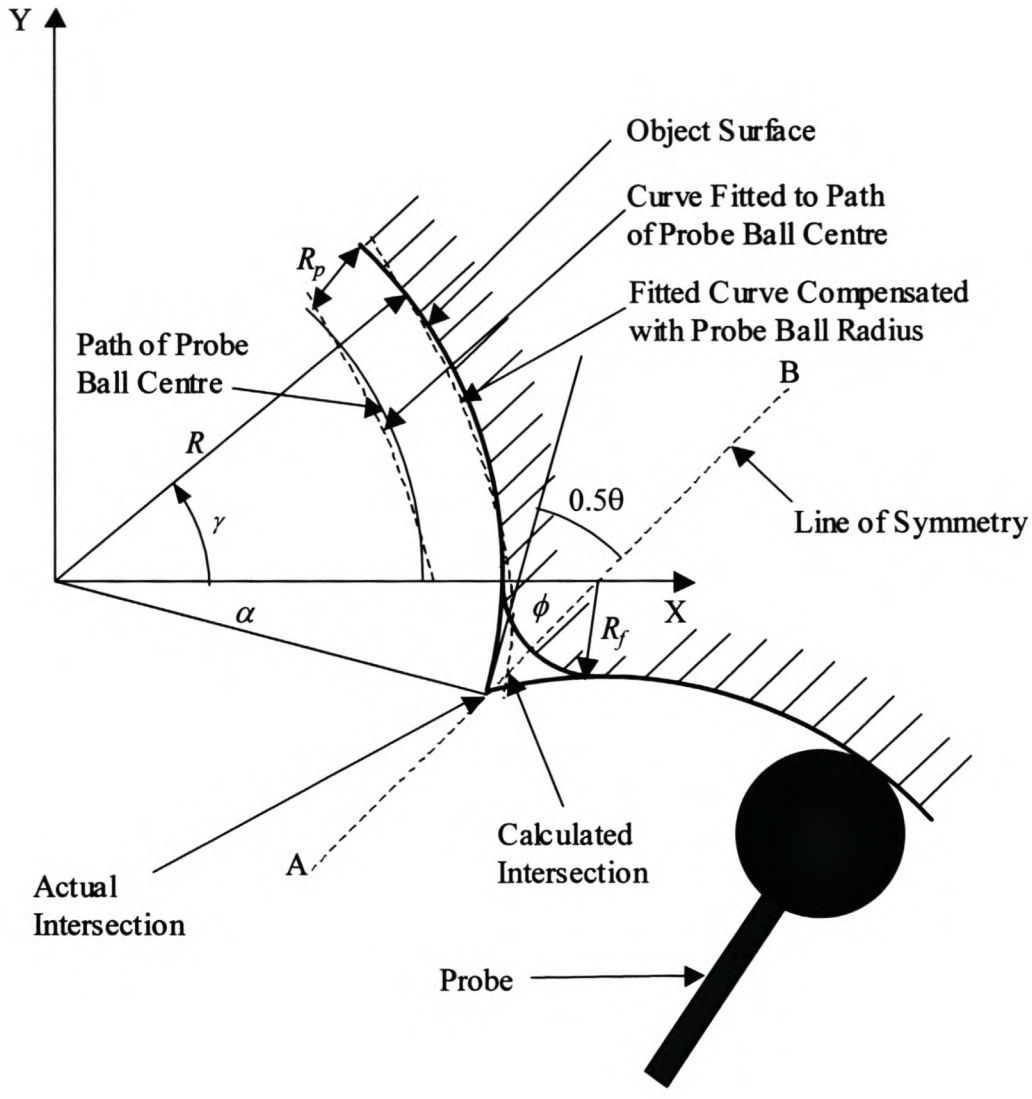


Figure 1 Model Used for Calculating the Analytical Accuracy.

5.2.2. Best Approximation of the Offset Curve

The curve that best approximates the circle segment is found by minimising equation (5.2.2.1.). Here, $f_c(u)$ is the arc and $f_p(u)$ is the curve approximating the arc. If the arc is approximated by a quadratic polynomial its six coefficients must be found. (Three coefficients for the X coordinate and three for the Y coordinate. The Z coordinate is not considered in this discussion.) If $f_p(u)$ is a line, then four coefficients are needed.

$$F = \int_0^s \left\| f_c(u) - f_p(u) \right\|^2 du \qquad (5.2.2.1.)$$

Equations for the six coefficients of the quadratic polynomial are derived in Appendix C and repeated below. The same is done for the lines in Appendix D. In these equations s is the arc length of the circle segment formed by the uncompensated points, i.e. the curve spanned by the angle γ in Figure 1, and r is the radius of curvature of the same segment, thus $r=R-R_p$. (The rest of this discussion focuses mainly on the quadratic polynomial curves. Similar results derived for lines are given in Appendix D.)

$$c_x = \frac{30r^2}{s^5}\left(s^2 \sin\left(\frac{s}{r}\right) - 12r^2 \sin\left(\frac{s}{r}\right) + 6rs\cos\left(\frac{s}{r}\right) + 6rs\right) \qquad (5.2.2.2.)$$

$$b_x = \frac{1}{s^3}\left(12r^3 \cos\left(\frac{s}{r}\right) + 6r^2 s \sin\left(\frac{s}{r}\right) - c_x s^4 - 12r^3\right) \qquad (5.2.2.3.)$$

$$a_x = \frac{1}{6s}\left(-3b_x s^2 - 2s^3 c_x + 6r^2 \sin\left(\frac{s}{r}\right)\right) \qquad (5.2.2.4.)$$

$$c_y = \frac{30r^2}{s^5}\left(s^2 - s^2 \cos\left(\frac{s}{r}\right) + 6rs \sin\left(\frac{s}{r}\right) - 12r^2 + 12r^2 \cos\left(\frac{s}{r}\right)\right) \qquad (5.2.2.5.)$$

$$b_y = \frac{1}{s^3}\left(-c_y s^4 - 6r^2 s - 6r^2 s \cos\left(\frac{s}{r}\right) + 12r^3 \sin\left(\frac{s}{r}\right)\right) \qquad (5.2.2.6.)$$

$$c_y = \frac{-1}{6s}\left(3b_y s^2 + 2s^3 c_y - 6r^2 + 6r^2 \cos\left(\frac{s}{r}\right)\right) \qquad (5.2.2.7.)$$

With these coefficients the maximum deviation from the circular curve to the curve that approximates it is derived in Appendices C and D. The result is represented in the figure below. The maximum deviation and the arc length are both made non-dimensional by dividing by the radius of curvature. Notice the power relationship between the arc length s and the maximum error. From this result it is clear that the arc length must be kept at a minimum to achieve the best results with the edge scanning algorithm.

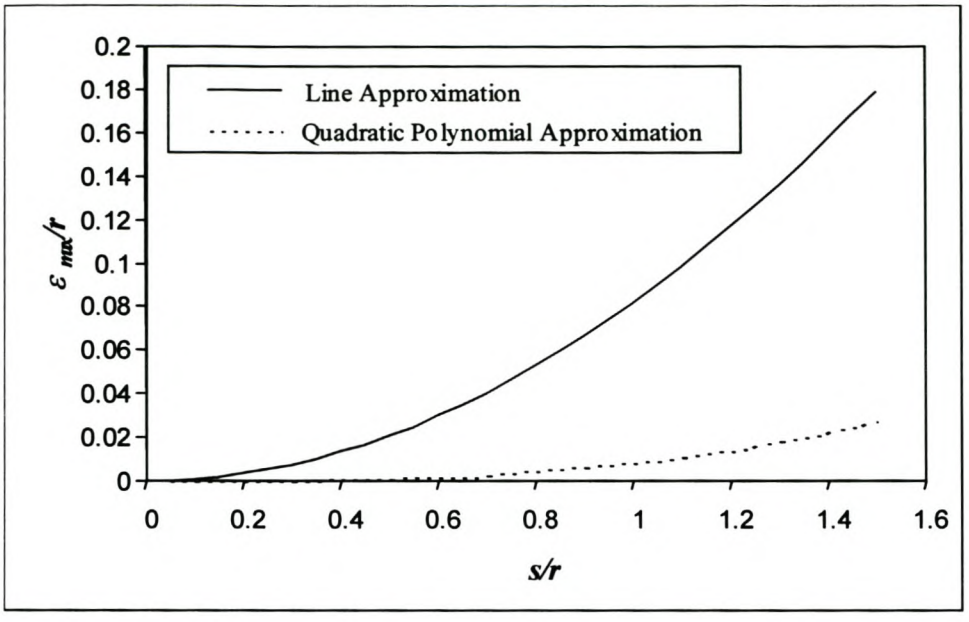


Figure 2 Maximum Deviation from a Circle Segment for a Best Fit Curve.

The next step is to find the offset curve. The offset vector is

$$\boldsymbol{d}_p(u) = \begin{pmatrix} \dfrac{R_p \dfrac{df_{py}(u)}{du}}{\sqrt{\left(\dfrac{df_{px}(u)}{du}\right)^2 + \left(\dfrac{df_{py}(u)}{du}\right)^2}} \\ \dfrac{-R_p \dfrac{df_{px}(u)}{du}}{\sqrt{\left(\dfrac{df_{px}(u)}{du}\right)^2 + \left(\dfrac{df_{py}(u)}{du}\right)^2}} \end{pmatrix} \qquad (5.2.2.8.)$$

Any point on the offset curve is given by

$$\boldsymbol{c}(u) = \boldsymbol{f}_p(u) + \boldsymbol{d}_p(u) \qquad (5.2.2.9.)$$

5.2.3. Anallytical Error

The calculated intersection point is shown in Figure 1. It is the intersection of the offset curve and the line AB. This is the point where the offset curve of the second surface intersects the one of the first surface, because the second curve is a mirror image of the first curve.

The line AB is given by

$$\boldsymbol{L}_{AB}(\lambda) = \begin{pmatrix} R + R_f \\ 0 \end{pmatrix} + \lambda \begin{pmatrix} \cos\phi \\ \sin\phi \end{pmatrix} \qquad (5.2.3.1.)$$

The intersection point is where $\boldsymbol{c}(u)=\boldsymbol{L}_{AB}(\lambda)$. If $\boldsymbol{c}(u)$ is a line, the intersection is found with the equation for line intersections derived in Appendix A. An explicit solution for u or λ for the quadratic polynomial is not possible, so any of the root finding algorithms can be used to find u_{int}, the parameter value at the intersection point.

It is clearly illustrated in Figure 1 that the calculated intersection point and the actual intersection point are not the same point. The distance between these points is the analytical minimum error of the edge points. Therefore, the analytical error is

$$\varepsilon_{analytical} = \sqrt{(c_x(u_{\text{int}}) - \bar{p}_x)^2 + (c_y(u_{\text{int}}) - \bar{p}_y)^2} \qquad (5.2.3.2.)$$

5.2.4. Influence of the Process Parameters on the Analytical Error

The influence of the scanning parameters is investigated with the analytical error model. The results are given in the graphs in this paragraph.

The first observation to note from all the graphs given below is that the analytical error shows a power relationship with the arc length of the surface. As shown in Figure 2 the maximum deviation of the approximated curve also shows a power relationship with the arc length. This means that the scanning amplitude must be kept as small as practically possible in order to achieve the best accuracy.

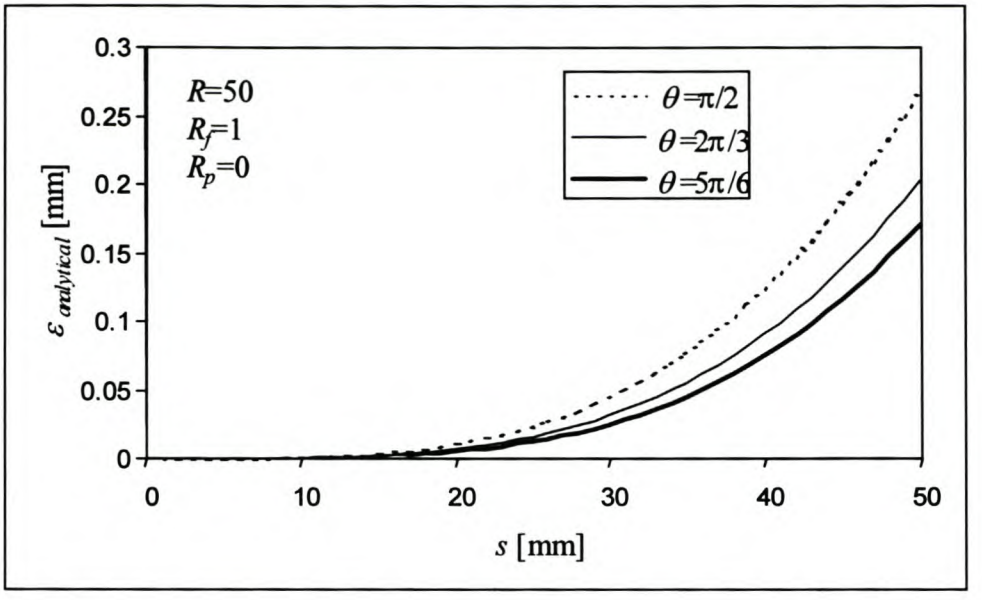


Figure 3 Influence of Intersection Angle on the Analytical Error.

Clearly the intersection angle of the two surfaces has an important influence on the analytical error, as shown in Figure 3. As the intersection angle between two surfaces becomes sharper, the distance that the offset curve must be extrapolated increases. The further the curve is extrapolated, the more it deviates from the circle segment. That is why the error increases as θ decreases. It also seems that the error does not increase linearly with a decrease of the intersection angle. The same analytical error can be achieved for a smaller intersection angle by decreasing the length of the line section, by implication the scanning amplitude.

Figure 4 shows that the error increases with the probe radius. The deviation of the polynomial curve from the object is amplified through the offset process; therefore the analytical error must increase with the probe radius. This increase is linear with the probe radius. Again the implication is that the amplitude must be decreased to achieve the same analytical error if the probe radius increases, only now it decreases linearly with the probe radius. Since the relationship between the error and the probe radius is linear, it is not expected that its influence will be as significant as the intersection angle's influence.

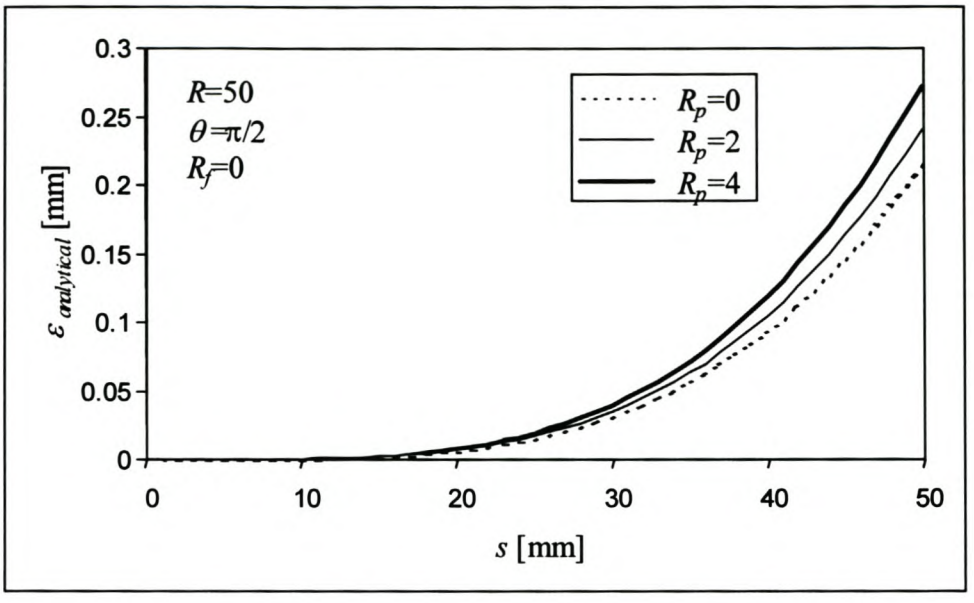


Figure 4 Influence of the Probe Radius on the Analytical Error.

The gap between the two surfaces increases as the fillet radius increases. Therefore the error must increase. This is clearly shown in Figure 5. It is further clear that the error increases linearly with the fillet radius. This implies that the amplitude must decrease in proportion to the increase of the fillet radius to obtain the same error.

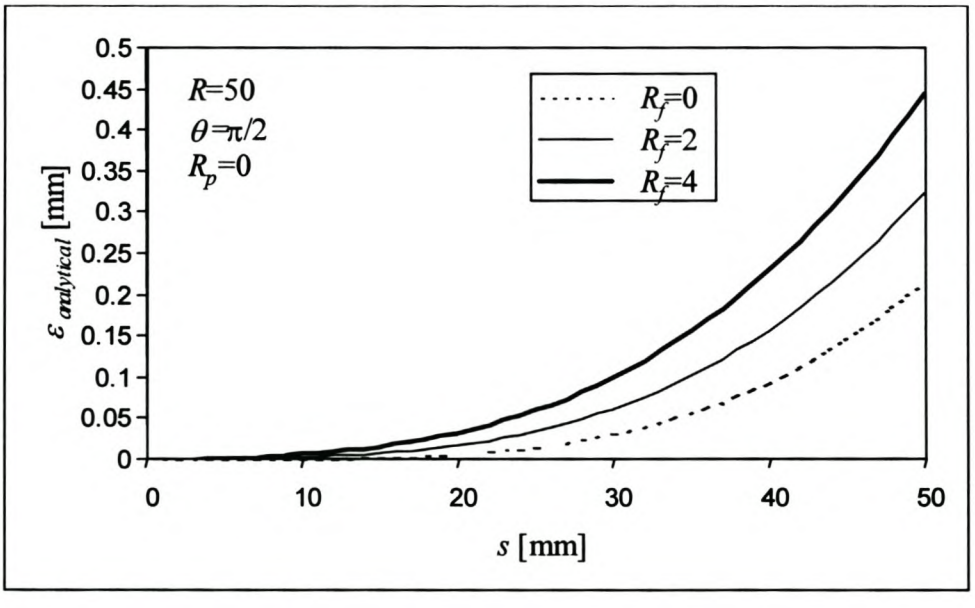
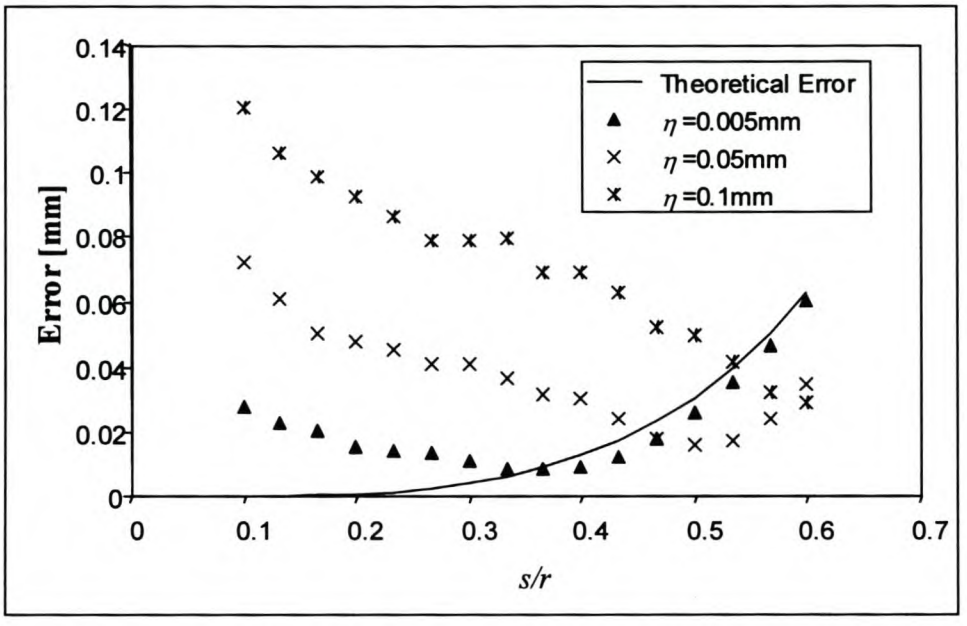


Figure 5 Influence of the Fillet Radius on the Analytical Error.

5.2.5. Difference between Analytical and Experimental Results

The comparison of the analytical model with results obtained with the edge scanning algorithm reveals some interesting insights into the influence of the noise in the point cloud. The results shown in the following two graphs were obtained with square scans. The average error of the edge points is compared with the analytical error.



$\rho_L = \rho_e = 2\text{mm}$ $R_p = 0.5\text{mm}$ $\varepsilon = 0.2\text{mm}$ for $\eta = 0.005\text{mm}$
$\kappa_e = 0.004\text{mm}^{-1}$ $R_f = 0\text{mm}$ $\varepsilon = 0.25\text{mm}$ for $\eta = 0.05\text{mm}$
$\theta = \pi/2$ $\rho_c = 1\text{mm}$ $\varepsilon = 0.3\text{mm}$ for $\eta = 0.1\text{mm}$
$r = 50\text{mm}$

Figure 6 Comparison of the Analytical Error with Results from the Edge Scanning Algorithm for Square Quadratic Scans.

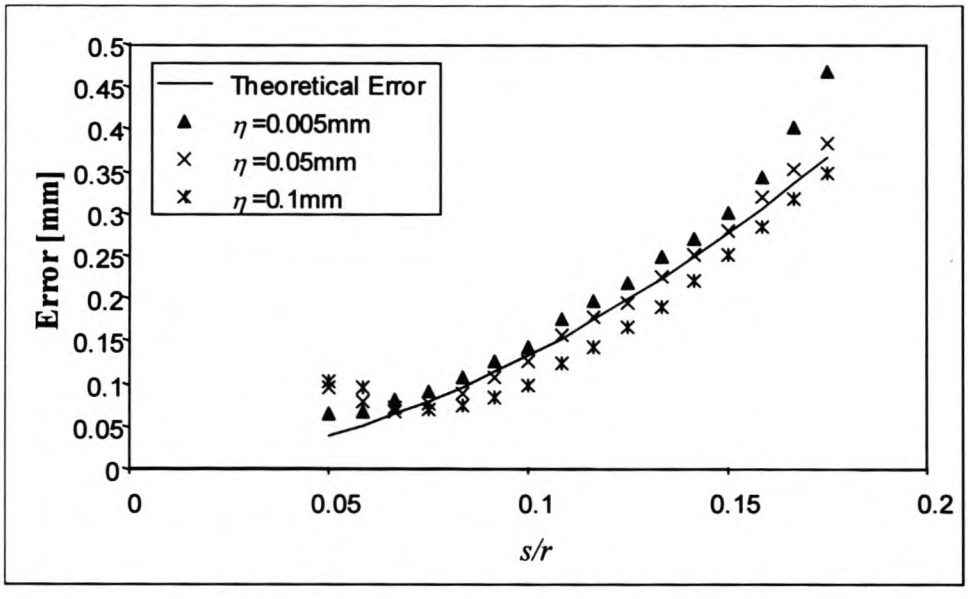
First of all Figure 6 shows the significant influence of noise in the point cloud on the results. The error decreases as s/r increases. A reason for this can be that as s/r increases, the number of points on the scan line section increases, and therefore the approximation of the curve improves. Clearly then, it is desirable to have a reasonable number of points on the scan line. At the point where $s/r=0.4$ there were about 20 points per scan line section.

This also indicates that the remark made earlier in this chapter that the amplitude must be as small as possible is also not correct for real scans. The remark was based purely on the analytical model. Figure 6 and Figure 7 indicate that there is an optimum

scanning amplitude. A guideline for finding the optimum amplitude is given at the end of this chapter.

Then, at a certain point, the error starts to increase with *s/r*. At this point, the deviation of the approximated curve from the arc becomes more significant than the noise. From this point, the analytical error and the real error grow together. This is better illustrated in Figure 7. This turning point happens at different values of *s/r* for the three curves. One possible explanation for this is that noise actually has a positive influence in the region of this point. Perhaps, due to the random distribution of the points the approximated curve remains for a short while closer to the desired curve. This can also be the explanation of why the experimental results are slightly better than the analytical results for this part of the curves.

Figure 6 also indicates that the ratio *s/r* should be kept below about 0.4 to 0.6 to avoid that the errors goes into the power region. Figure 7 shows that this ratio must be much lower when linear scans are used. For linear scans it must be as low as 0.1.



ρ_{LP_e}=2mm R_p=0.5mm ε=0.2mm for η=0.005mm
κ_e=0.004mm^{-1} R_f=0mm ε=0.25mm for η=0.05mm
θ=π/2 ρ_c=1mm ε=0.3mm for η=0.1mm
r=50mm

Figure 7 Comparison of the Analytical Error with Results from the Edge Scanning Algorithm for Square Linear Scans.

Figure 7 shows how quickly the errors reach the power region for the linear scans. Once this region is reached, the experimental errors and the analytical error grow closely together.

Should the quadratic scans then always be used? The results in Figure 6 and Figure 7 indicate that the error at low values of *s/r* is about the same order of magnitude. When scanning in this region, and if the point cloud pitch allows only a few points per scan line section, it is advisable to use a linear scan. It is computationally more robust since the linear approximation has one more degree of freedom than the quadratic approximation for the same number of points. When scanning with only 4 or 5 points per scan line section, this difference can be significant.

There is a number of reasons why the experimental results deviate from the analytical results.

- First of all the experimental results include the effect of measuring noise. Its effect has been discussed in detail above.

- The number of points used for curve fitting determines how well it approximates the ideal curve derived in paragraph 5.2.2. The more points there are per scan line, the better this approximation should be.

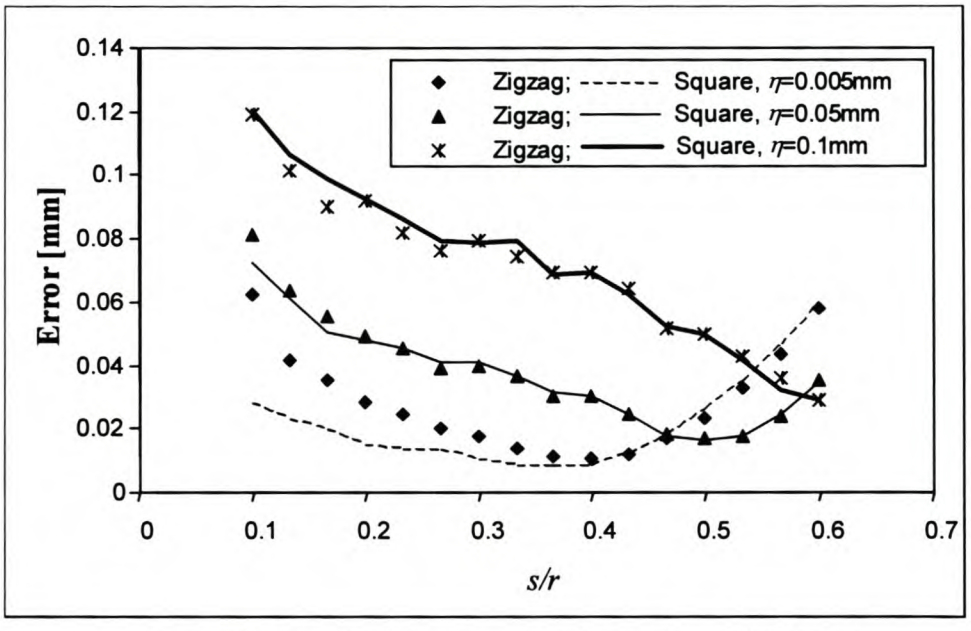
- In Chapter 4 it is shown that the offset vector is the cross product of the tangent vector to the curve and a vector that connects points on the two neighbouring curves. Even in the analytical model, the tangent of the curve is not tangent to the actual surface. The second vector can also not be tangent to the actual surface. This further increases the error of the offset vector.

- The analytical model assumes that the scanning plane is perpendicular to both surfaces, therefore, the scan line segments form perfect arcs. In practice the orientation of the scanning plane depends on the tangent vector calculated for the edge. It is shown in Chapter 4 that this vector can deviate considerably from the actual tangent vector. This means that the scanning plane is not necessarily perpendicular to the surfaces and the line segments therefore not necessarily perfect arcs.

- The scanning tolerance determines how many points are included in the curve approximation. If the tolerance is too small, many points are ignored that might have improved the polynomial approximation. If it is too large, points that lie on the fillet radius are included in the curve approximation. This has a very adverse effect on the error.

5.2.6. Comparison of Zigzag and Square Patterns

The results shown in the previous paragraph were all obtained with square scans. In the following two graphs, these results are compared to zigzag scans on the same objects and with the same scanning parameters.



ρ_L=ρ_e=2mm R_p=0.5mm ε=0.2mm for η=0.005mm
κ_e=0.004mm^{-1} R_f=0mm ε=0.25mm for η=0.05mm
θ=π/2 ρ_c=1mm ε=0.3mm for η=0.1mm
r=50mm

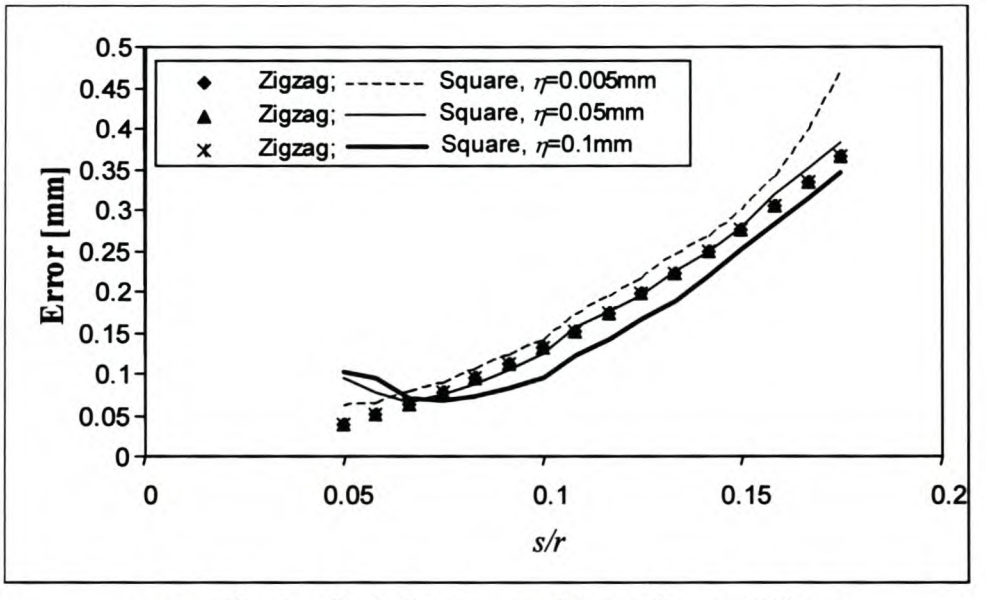
Figure 8 Comparison of Zigzag and Square Patterns for Quadratic Scans.

Figure 8 shows that the zigzag and square patterns very closely follow the same trend for quadratic scans except for the one scan with η=0.005mm. For this scan, the square pattern performs better for low values of s/r. In Chapter 4 it is mentioned that the points on the scan line sections are projected unto the scanning plane before the curve approximation is done. If the scanning plane orientation is not perpendicular to the local surface normal, a small error is induced by this projection. The scanning planes

of the zigzag cannot always be perpendicular to the local surface normal. However, the square pattern was developed so that the orientation of the scanning plane is parallel to the edge's tangent vector. As mentioned earlier, this means that the orientation of the scanning planes of the square pattern most closely represents a plane that is perpendicular to the local surface normal. This can explain the difference between the two curves. It can further be that the noise in the point cloud has a more significant effect than this induced error as the noise becomes larger. Therefore the curves do not deviate significantly for the higher noise values.

The difference diminishes as the value of s/r approaches the point where the optimum amplitude is reached.

All this means that when it is necessary to do a very accurate scan at less than the optimum amplitude it is better to use a square pattern. Of course, this means that the point cloud noise must be very small. A value of η=0.005mm corresponds to the noise that can be expected from a CMM with a touch trigger probe, as shown in Chapter 2.



ρ_{LP_e}=2mm R_p=0.5mm ε=0.2mm for η=0.005mm
κ_e=0.004mm^{-1} R_f=0mm ε=0.25mm for η=0.05mm
θ=π/2 ρ_c=1mm ε=0.3mm for η=0.1mm
r=50mm

Figure 9 Comparison of Zigzag and Square Patterns for Linear Scans.

Figure 9 shows that the zigzag and square patterns follow each other closely when doing linear scans. However, it is perhaps interesting to note that the zigzag scans

follow each other very closely, whereas there is some difference between the three square scans. The reason, if it is significant, is not understood at the time of writing.

5.3. Experimental Testing*

A series of experiments were done to test, first of all, the robustness of the edge scanning algorithm. The technique of Design of Experiments was used to ensure a good coverage of the range of parameters that influence the algorithm. A short background of this technique is given in the following paragraph. This section also contains a discussion of the experimental results. (The actual results are given in Appendix E.) The section closes with some general observations of the experimental results.

5.3.1. Design of Experiments

In this study the influence of 11 parameters are investigated. If the relationship between these parameters and the outcome of the experiments is linear, then at least 2^{11} experiments must be done to study all the combinations! After the testing the researcher must draw some conclusions from 2048 experiments. In this case the relationship between the parameters and the outcome of the experiments is not linear. This means that many more than 2^{11} experiments are needed.

Design of Experiments is a technique developed to analyse the sensitivity of a process to any number of parameters without testing all the possible combinations (O'Connor, 1991). Standard tests can be used or derived with the method of O'Connor (1991). This method reduces the number of experiments dramatically. For example, in this case only 16 experiments (using a standard test from the Statistica (2000) package) are enough to analyse the sensitivity of the parameters. (This specific experiment is given in Appendix E.)

---

* Although the experimental work is discussed after the analytical model, it was actually done the other way round. This is mentioned simply because some of the parameter selections for the experiments turned out to be very unfortunate once the results from the analytical model were known. The insight gained from the experimental work is nonetheless very useful as described in this chapter.

The interaction of the parameters can also be investigated, but then more experiments are needed to avoid the aliasing effect. Basically, aliasing means that the effect of two or more parameters or interaction of parameters cannot be separated in the results. However, in the experiments done in this project, the interactions are not investigated.

The experiments assume that the relationship between the experimental outcome and the parameters is linear. This shortcoming is avoided to some degree by not doing one experiment for the entire parameter range. The range is divided into smaller regions and the investigations are done for each region individually.

5.3.2. Choice of Parameters

Eleven parameters were identified that influence the accuracy of the results.

The surface curvature (κ_s) in the direction perpendicular to the edge (hereafter simply referred to as the surface curvature) determines how much the chosen curve, either linear or quadratic, deviates from the scanned line section. Surface curvatures between 0.002mm^{-1} and 0.125mm^{-1} were tested. This translates to a radius of curvature between 8mm and 500mm.

The scanning amplitude is one of the input parameters to the scanning algorithm and therefore an obvious choice. Amplitudes (A) between 4mm and 50mm were tested.

Noise (η) values between 0.01mm and 0.2mm were tested. This is the noise associated with the points in the cloud. These values are representative of most of the scanning methods described in the Chapter 2.

The line pitch (ρ_L) is the distance between consecutive points on a scan line. This is one of the input parameters to the algorithm. The range tested is from 0.4mm to 4mm.

The edge pitch (ρ_e) is the desired distance between consecutive edge points. This is also an input parameter to the edge scanning algorithm. The range tested is from 0.6mm to 6mm.

The edge curvature (κ_e) can influence the ability of the algorithm to follow the edge and is therefore included in the testing. The range tested is from 0.0033mm^{-1} to 0.033mm^{-1} and translates to a radius of curvature between 30mm and 300mm.

Intuitively the intersection angle (θ) of the two surfaces must have an influence. The values tested are between 90° and 150°.

The scanning tolerance (ε) is another input parameter for the algorithm. Values tested are between 0.05mm and 0.3mm.

One of the objectives of this algorithm is that it must be able to detect the original edge if there is a fillet radius (R_f) between two surfaces. The ability of the algorithm to do this is tested by incorporating fillet radii between 0mm and 4mm.

The algorithm must also be able to scan on a cloud uncompensated with the probe radius (R_p). Probe radii from 0.25mm to 2.5mm were tested. This is representative of the probes in the laboratory.

Lastly, it is hinted in Chapter 4 that the point cloud pitch (ρ_c) can have an influence in the outcome. A point cloud pitch between 0.1mm and 1.4mm was tested.

5.3.3. Discussion of Experimental Results

Twelve experiments were done to test the performance of the edge scanning algorithm within this parameter domain. The specific parameter range for each experiment is given in Appendix E and represented in graph form in this chapter. The outcome of each experiment is also given there. A sensitivity analysis for the parameters was also done, following the method of O'Connor (1991). The results are given in Appendix E and in graph form in this paragraph.

The range of the parameters in the twelve experiments was selected randomly to cover the selected global parameter range. Twelve experiments were enough to draw the conclusions given in this chapter.

In each experiment a series of 16 tests were done. This was done for the zigzag and square scan with linear and quadratic approximations of the scan line sections, i.e. 12×16×4=768 tests in total. However, this is by no means an exhaustive test of the algorithm's ability to detect edges in a point cloud.

5.3.3.1. Experiment 1

5.3.3.1.1. Errors

The parameter range of this experiment is represented in the following graph. The numbers on the top and bottom axis indicate the global parameter ranges for the specific parameter. The black bar represents the range of parameter tested in this experiment. The numerical values of the range is given in Appendix E. This graph is repeated for each of the experiments discussed hereafter without further explanation.

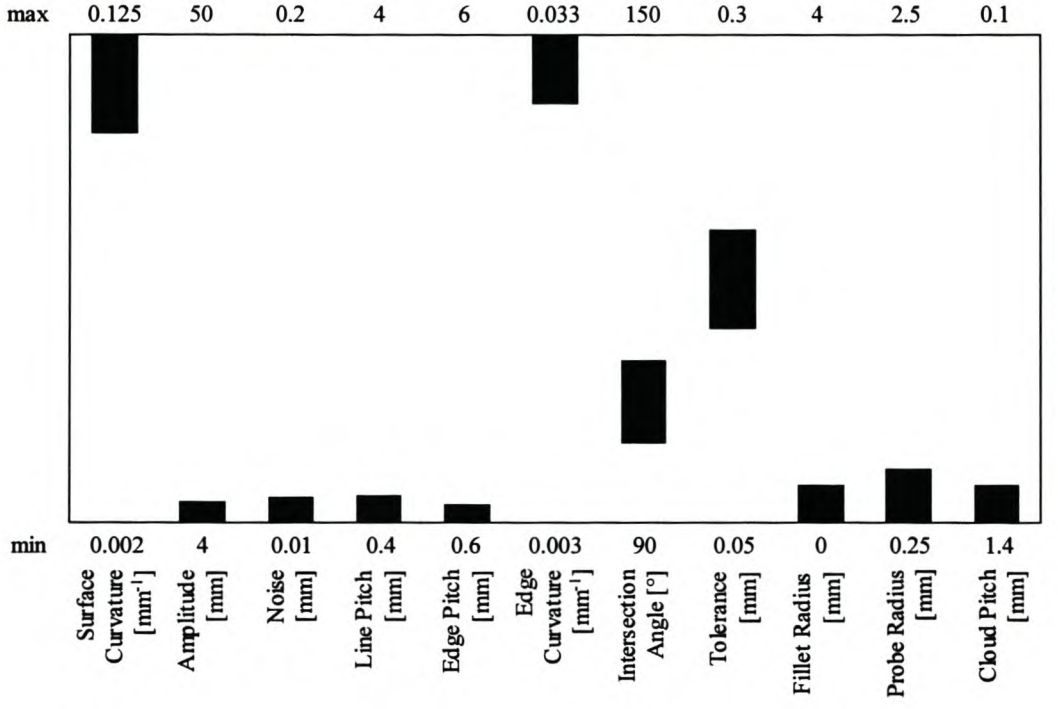


Figure 10 Parameter Range of Experiment 1.

Test number 1S1.12[†] failed to detect the entire edge. The analytical model shows that high ratios between the length of the scan line section and the surface curvature

---

[†] The numbers of the tests have the following meaning. It all contains either an S or Z. It refers to square and zigzag patterns respectively. The number before the S or Z is the experiment number, i.e. it is a number between 1 and 12. The number after the S or Z is either 1 or 2. It refers to linear and quadratic scans respectively. This is followed by a period and then another number indicating which of the 16 tests in the experiment it is.

(hereafter only referred to as $s*\kappa_s$) should not exceed 0.1 for the linear approximations. In this case it is 0.334.

5.3.3.1.2. Sensitivity Analysis

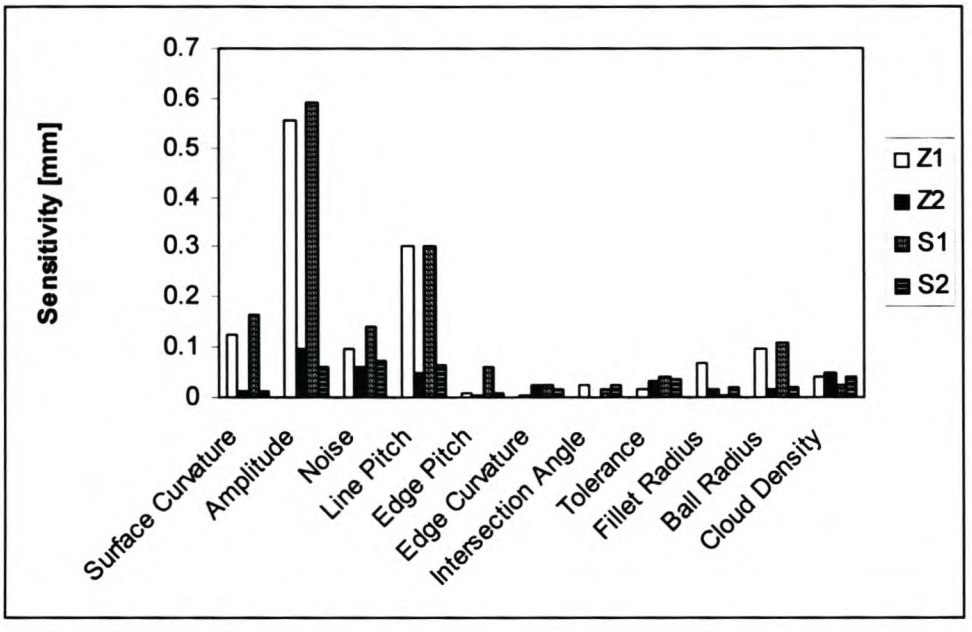


Figure 11 Sensitivity Analysis Results of Experiment 1.[‡]

The errors for the quadratic approximations are significantly smaller. With the high surface curvature for this experiment, it is not surprising.

The error is sensitive to the amplitude and line pitch for both approximations. Only 3 - 7 points were scanned per scan line section, therefore it is not surprising that the amplitude and line pitch are so important. This combination determines how well the curve approximates the line on the surface.

The quadratic approximations are more noise sensitive than linear scans. With so few points per scan line section the noise has a stronger influence on the approximation of

---

[‡] The legend of this and the following graphs has the following meaning. Z1 means a zigzag pattern with linear approximations of the scan line section was done. Z2 refer to a zigzag pattern with quadratic approximations. S1 refer to a square pattern with linear approximations and S2 to a square pattern with quadratic approximations.

quadratic polynomials than on the lines. Therefore it contributes significantly to the errors.

5.3.3.2. Experiment 2

5.3.3.2.1. Errors

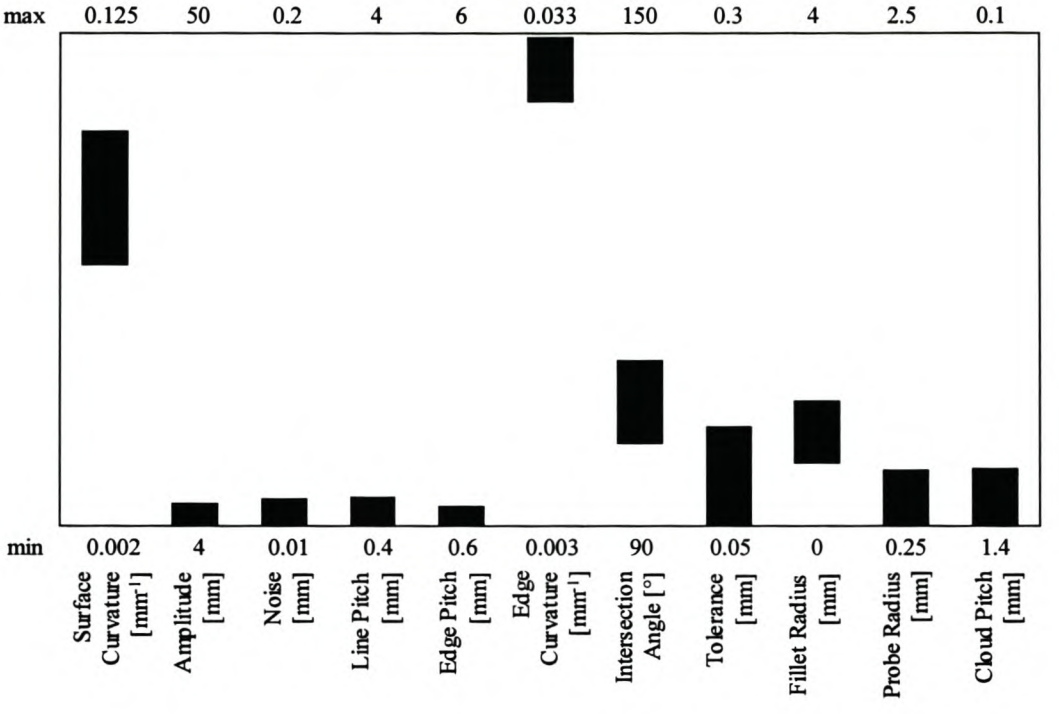


Figure 12 Parameter Range of Experiment 2.

Test 2Z2.1 and 2Z2.6 both failed to complete the edge. The failure can be related to the ratio of the length of the section of the scan line that is scanned on the fillet radius, *df*, to the length of the scan line that is scanned on the surfaces, *s*. The ratio is 1.748 and 1.220 respectively. That means that more points were scanned on the fillet radius than on the actual surface. Depending on the algorithm's ability to ignore the points on the fillet radius, this can result in very bad approximations of the scan line sections and consequently in bad estimates of the edge points. The algorithm needs a reasonable estimate of the edge points in order to follow the edge reliably.

5.3.3.2.2. Sensitivity Analysis

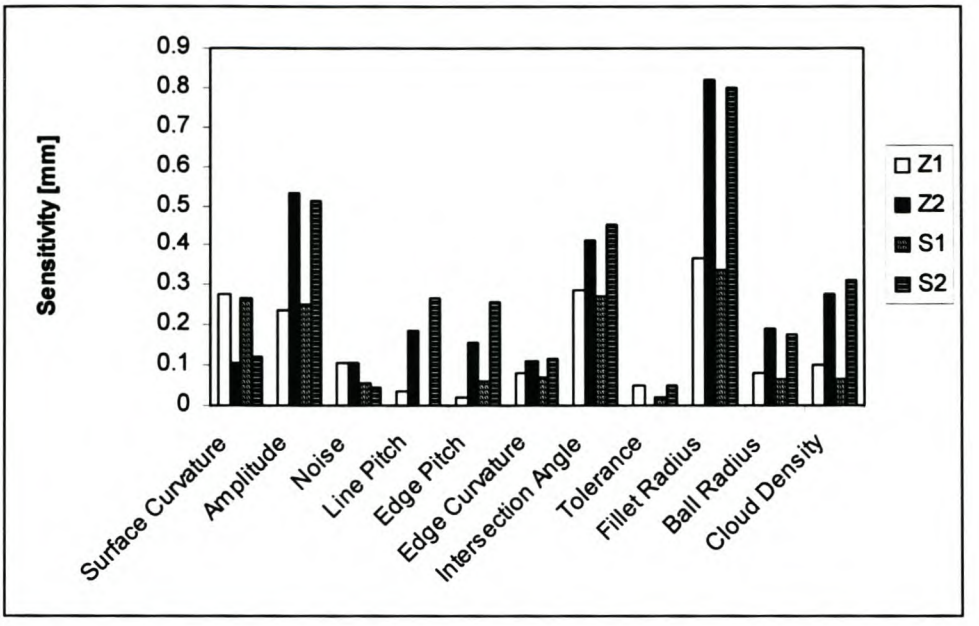


Figure 13 Sensitivity Analysis Results of Experiment 2.

The fillet radius has the most important influence for all scans. The high ratios of *df/s* and the fact that only 3 - 7 points were scanned per scan line section can explain it. This means that the curve approximation is not good enough to extrapolate over the relatively large gap between the surfaces.

The intersection angle has an important influence in all cases. With larger intersection angles, the position of the calculated edge point is sensitive to the approximation of the curve. Small deviations of the approximated curve have a larger effect on the accuracy of the edge point for larger intersection angles. This is perhaps better explained in the figure below. In this figure, the approximated lines are shown as solid lines and the actual surfaces as dashed lines. The deviation of the approximated lines from the surface is exaggerated.

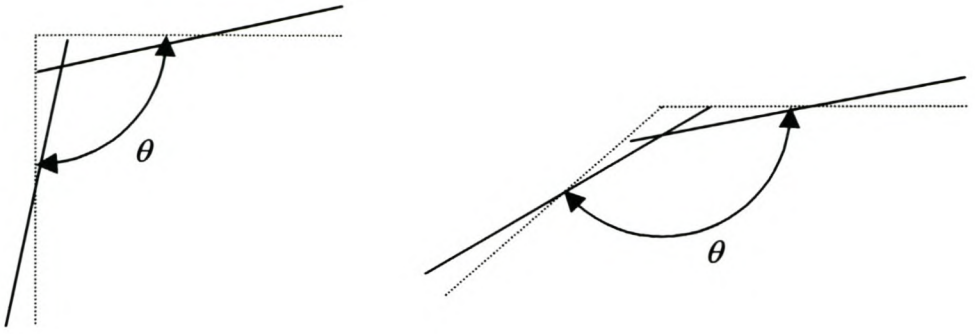


Figure 14 Explanation of the Influence of the Intersection Angle.

The amplitude is important for all cases. It determines whether 3 or 7 points per scan line are scanned. This has a very significant effect on the accuracy of the curve estimates.

The surface curvature is important for the linear scans. Together with the amplitude it determines how well the line represents the scan line section.

5.3.3.3. Experiment 3

5.3.3.3.1. Errors

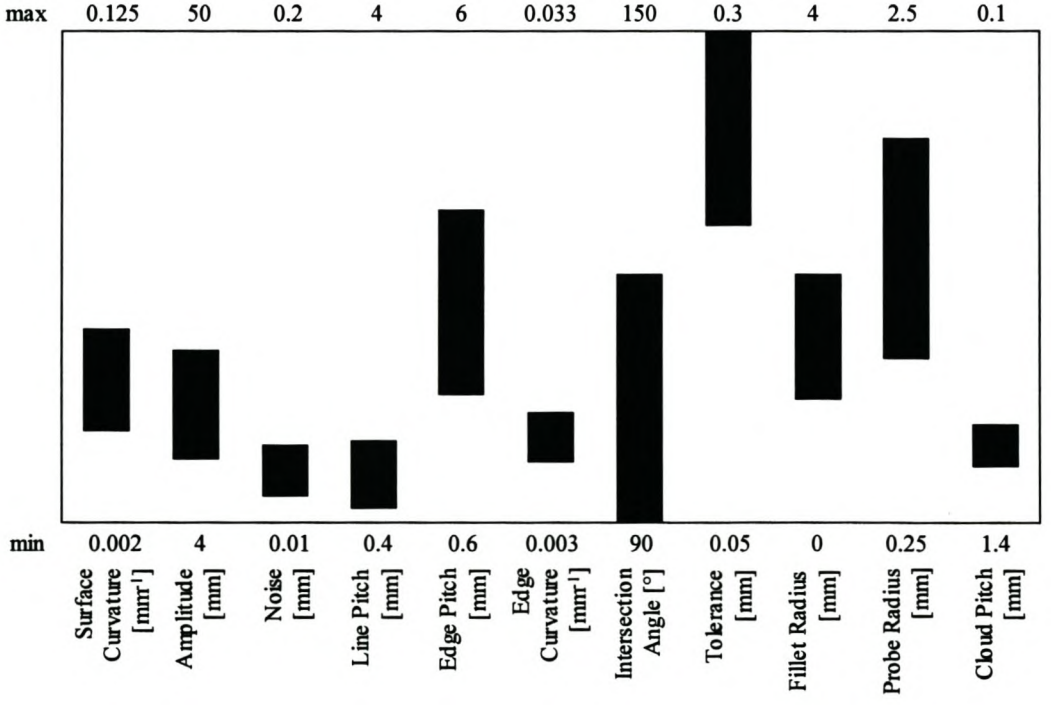


Figure 15 Parameter Range of Experiment 3.

Test 3Z2.1 possibly failed due to a high ratio of *df/s*. In this case *df/s* =1.949.

Tests 3S1.7, 3S1.11 and 3S1.15 most likely failed due to values of $s^*\kappa_s$ in the excess of 0.43, much higher than the recommended value of 0.1 derived for linear scans.

5.3.3.3.2. Sensitivity Analysis

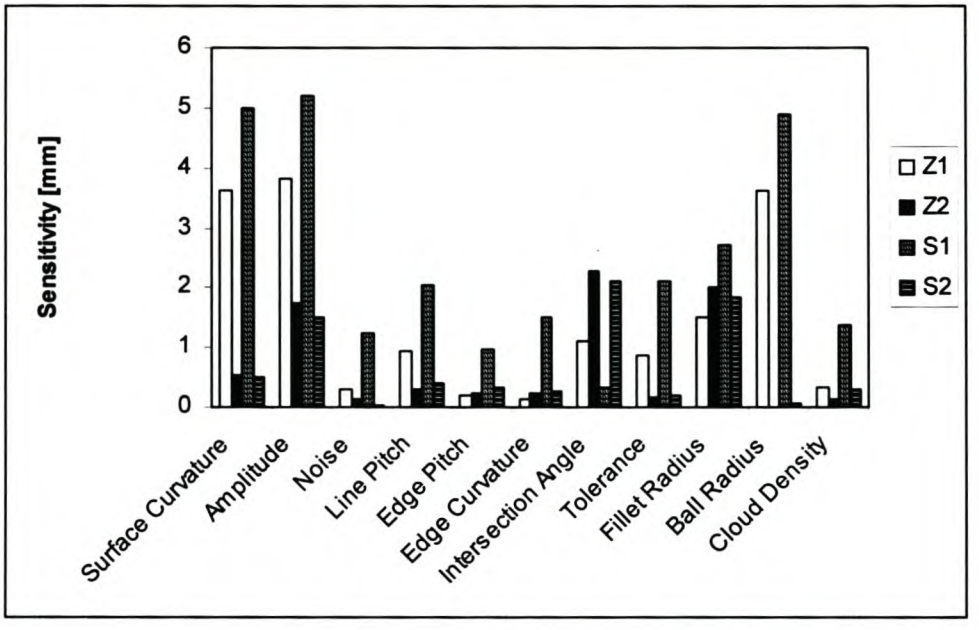


Figure 16 Sensitivity Analysis Results of Experiment 3.

The errors in this example are very high. As already mentioned for the tests that failed, this experiment has high *df/s* and $s^*\kappa_s$ values. The ratio of the line pitch to the cloud pitch also varies between 1.4 and 4. The higher this ratio, the further the scanned points lie from the desired pattern. All these factors contribute to the high average errors for this experiment. The fact that the amplitude, fillet radius and surface curvature stand out in all the tests supports this observation.

The analysis indicates that the linear approximations are sensitive to the ball radius. This is not well understood. It seems to have a very small effect on quadratic approximations. Possibly bad line approximations are amplified during the compensation process.

It is also interesting to note that the effect of the intersection angle, explained for Experiment 2, has a larger effect for quadratic approximations. This is possible since

the angle between the quadratic polynomials at the intersection point is larger than the angle between lines fitted to the same points due to the curvature of the polynomials.

5.3.3.4. Experiment 4

5.3.3.4.1. Errors

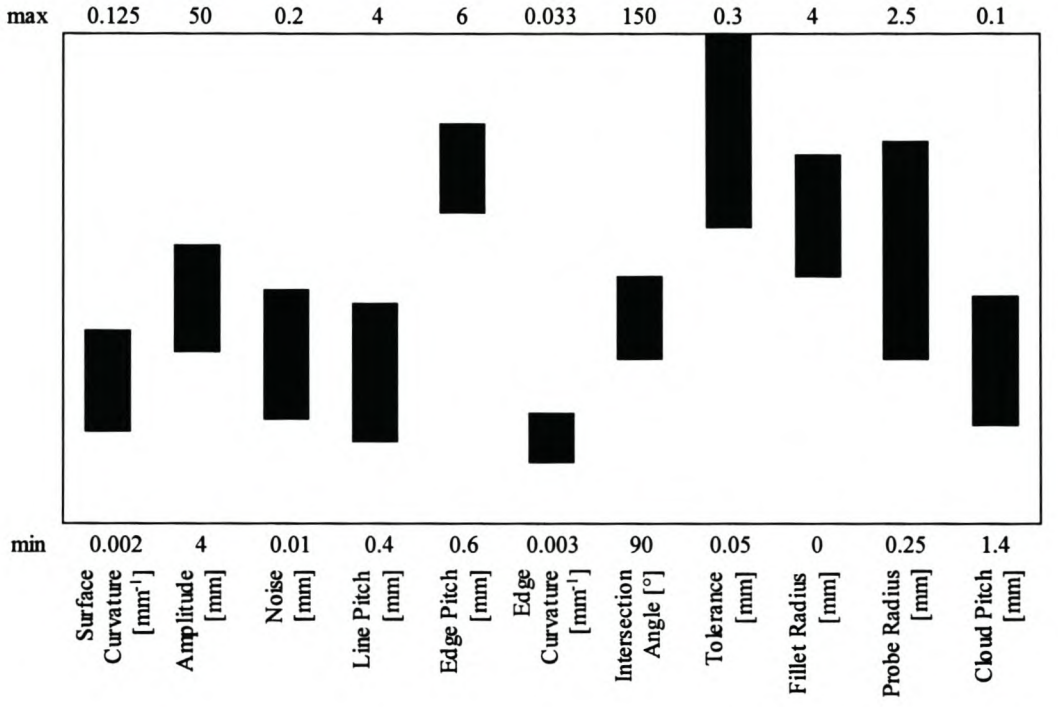


Figure 17 Parameter Range of Experiment 4.

The following tests all failed to scan the complete edge: 4Z1.3; 4Z1.7; 4Z1.11; 4Z1.15; 4Z2.11; 4S1.1; 4S1.3; 4S1.4; 4S1.5; 4S1.7; 4S1.9; 4S1.11; 4S1.12; 4S1.13; 4S1.15; 4S1.16; 4S2.11.

With this experiment, the significance of the ratio $s*\kappa_s$ was first noticed. With values of this ratio between 0.318 and 0.67 it is not surprising that so many of the linear tests failed.

5.3.3.4.2. Sensitivity Analysis

The high errors for the tests that failed completely overwhelm the sensitivity analysis for the linear approximations. The errors are an order of magnitude larger than the

errors of the successful scans. It was therefore decided not to include the sensitivity analysis for the linear approximations in the graph below.

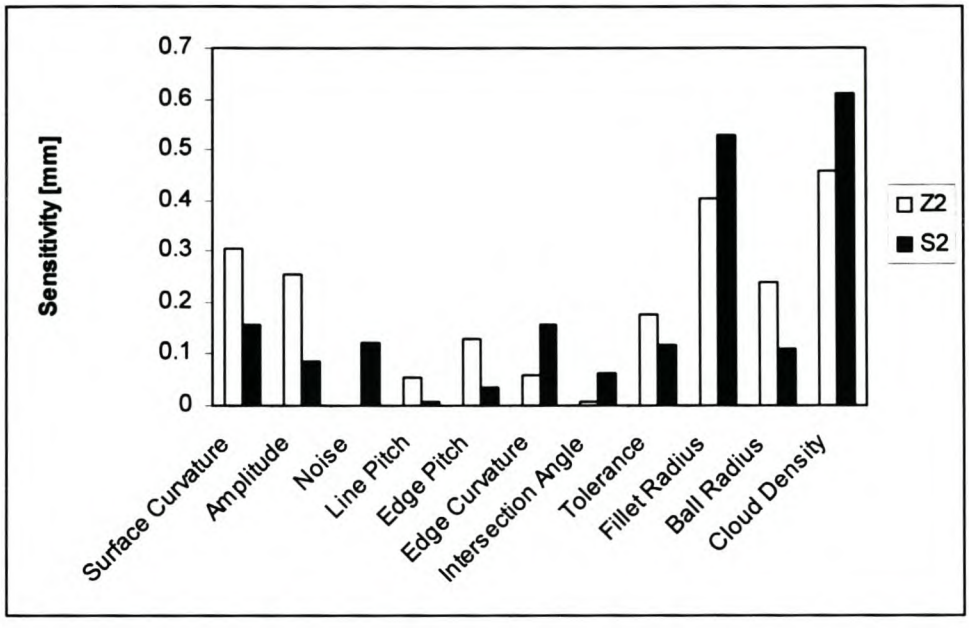


Figure 18 Sensitivity Analysis Results of Experiment 4. (Quadratic Approximations Only.)

The high sensitivity of the quadratic approximations to the cloud pitch stands out. This can be due to the large difference between the minimum and maximum cloud pitch used in the experiment (0.36mm and 0.7mm respectively). The ratio of the line pitch to the cloud pitch is also very important as explained earlier. It determines how close the actual pattern will be to the desired pattern.

Fillet radius stands out for both quadratic scans. Due to the high $s*\kappa_s$ ratios, decreasing the fillet radius has the effect of including more points on the surface and thus helping to improve the results.

5.3.3.5. Experiment 5

5.3.3.5.1. Errors

The following tests all failed to scan the complete edge: 5Z1.9; 5Z1.11; 5Z2.6; 5Z2.8; 5Z2.11; 5Z2.14; 5Z2.15; 5S1.1; 5S1.7; 5S1.11; 5S1.15; 5S2.1; 5S2.5; 5S2.11; 5S2.14; 5S2.15.

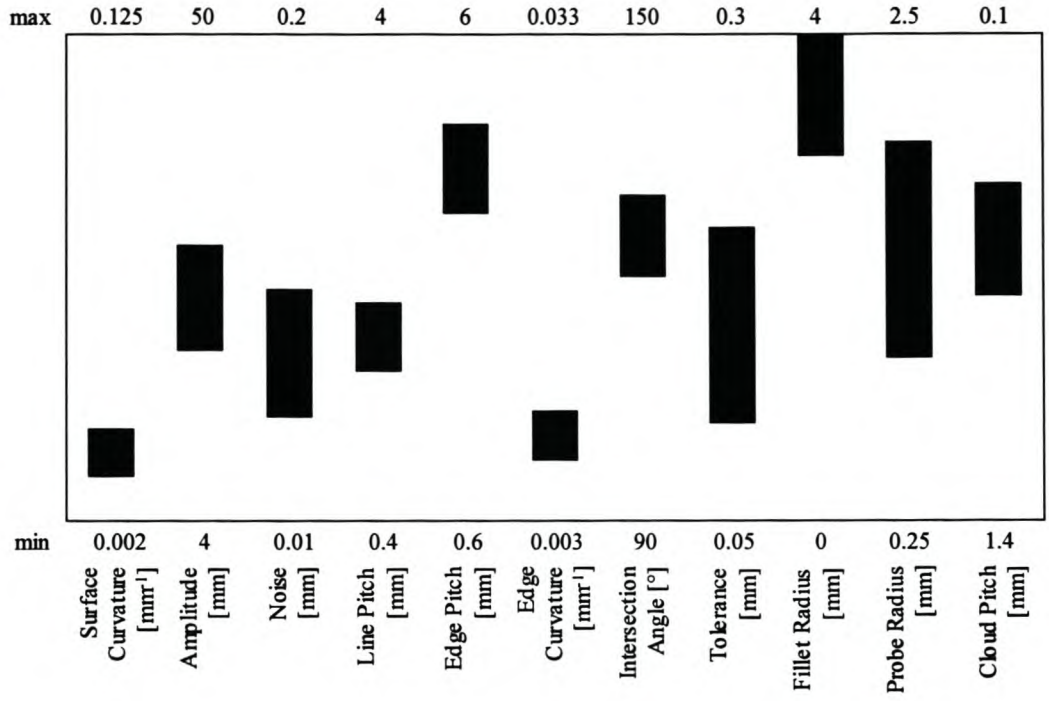


Figure 19 Parameter Range of Experiment 5.

The high number of failures can be explained by an observation that was only made after all the experimental work was completed. It is given here.

The purpose of the scanning tolerance is to exclude points that lie on the fillet radius. Its purpose is not to exclude noisy points that belong to the surface. Since the edge scanning algorithm works with only a few points on each scan line section, it is better to do the calculations with a few noisy point than very few not so noisy points. Therefore the scanning tolerance must be larger than the noise level of the point cloud.

However, it was observed that the point cloud noise is not the only source of noise. The scanned points are projected onto the scanning plane before the curves are fitted. If the scanning plane normal and the local surface normal vector at the point is perpendicular the projection is perfect. However, this is seldom the case. There is an additional component to the noise that is inherent to the edge scanning algorithm. The following picture explains this. The point A must be projected onto the scanning plane prior to fitting the polynomials. This projection is done along the scanning plane normal represented by line AB. If line AB is not parallel to the surface, but deviates by the angle ϕ then additional noise is incurred, i.e. if ϕ =0, then the projected point

will be C and only the cloud noise, η, is incurred. The maximum length of line AB is ρ_c, the point cloud pitch.

The point should not be further away from the scanning plane than the cloud pitch. Through experimentation, it was found that ϕ is seldom more than 10°. A better guideline for specifying the scanning tolerance is then

$$\varepsilon_{min}=\eta+\rho_c\sin(10°) \qquad (5.3.3.5.1.1.)$$

In this experiment, the tolerance is often smaller that the one specified by this guideline. Combined with reasonably high noise values, 0.1mm, and few points per scan line section (4-8 points), it is understandable that many of the curves do not represent the scan line sections well. This in turn leads to poor edge point estimates and therefore to the inability of the algorithm to complete the scanning of the edge.

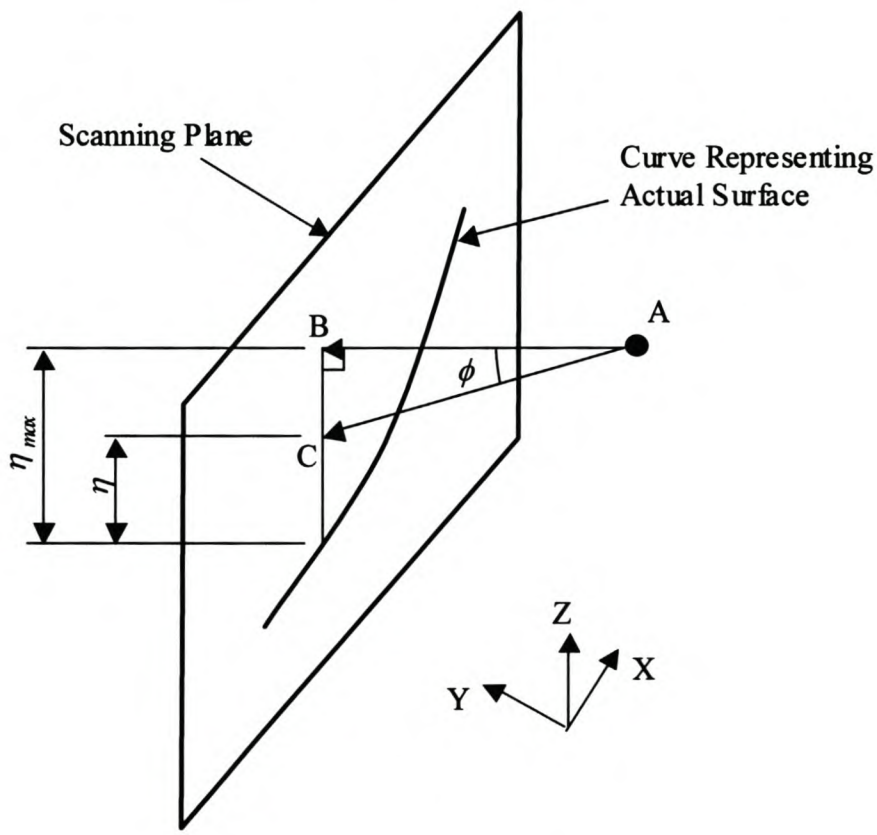


Figure 20 Additional Component of Noise Due to Point Projection.

It is also interesting to note that the linear scans are slightly more robust than the quadratic scans (6 vs. 8 failures). This is the opposite of Experiment 4. The reason is

probably the better $s^*\kappa_s$ ratios in this experiment. Also, the tolerance ratios ($\varepsilon/\varepsilon_{min}$) are worse than in Experiment 4, which may mean that the quadratic scans are more sensitive to the noise in the data.

5.3.3.5.2. Sensitivity Analysis

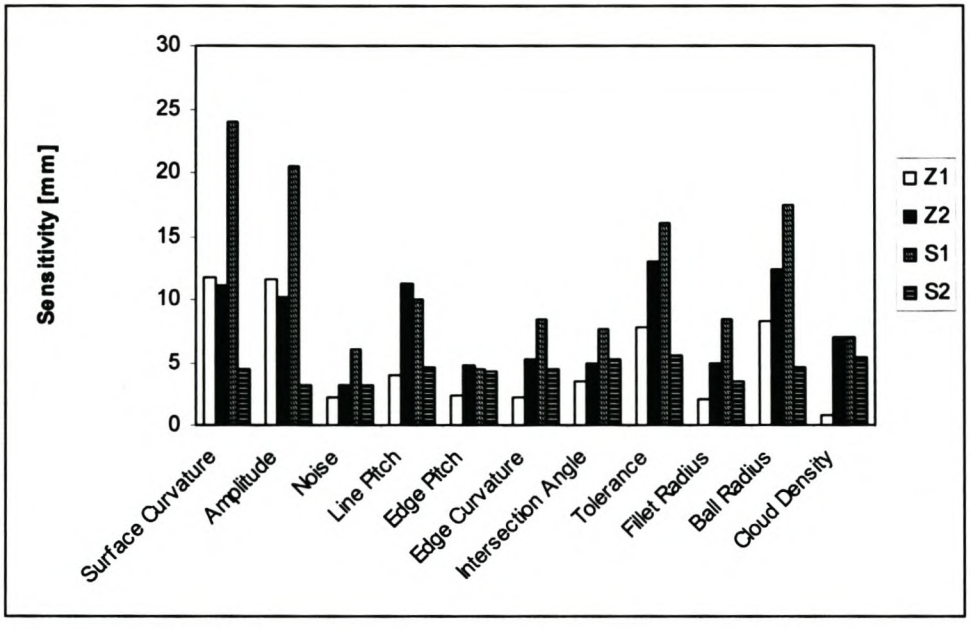


Figure 21 Sensitivity Analysis Results of Experiment 5.

Due to many failures the errors are high and it is therefore dangerous to draw too many conclusions from the sensitivity analysis. It is significant that the tolerance stands out for the all tests, especially the quadratic scans – this supports the theory for the many failures.

5.3.3.6. Experiment 6

5.3.3.6.1. Errors

All tests were successful.

5.3.3.6.2. Sensitivity Analysis

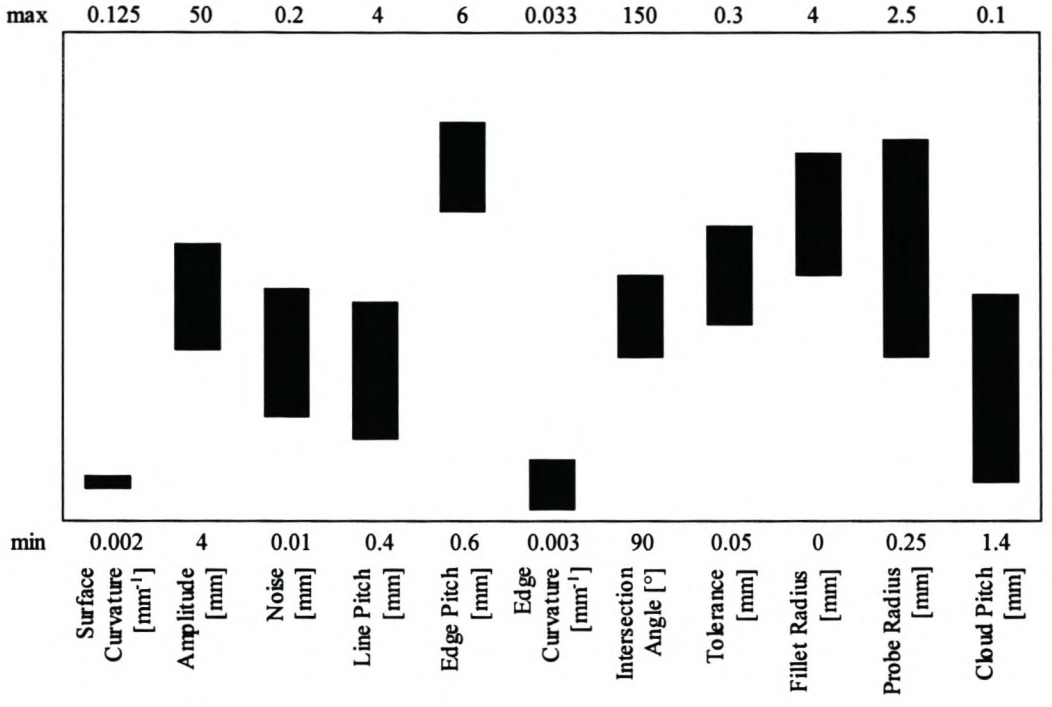


Figure 22 Parameter Range of Experiment 6.

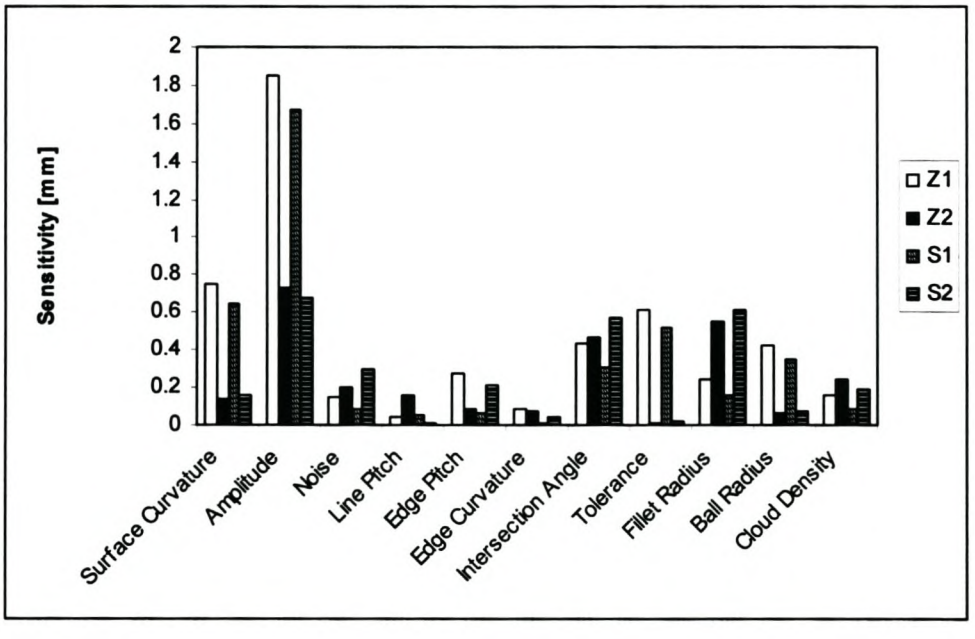


Figure 23 Sensitivity Analysis Results of Experiment 6.

The amplitude, surface curvature and intersection angle stand out for the linear scans. This shows the importance of choosing these parameters so that a good linear

approximation can be made when the linear model does not really represent the surface.

The quadratic scans are sensitive to the choice of the amplitude, intersection angle and fillet radius. The amplitude to fillet radius ratio determines how well the curve extrapolates. It was already shown that the edge point calculation is sensitive to intersection angle, especially for the quadratic scans. Therefore these parameters have an important influence on the calculation of the edge points.

5.3.3.7. Experiment 7

5.3.3.7.1. Errors

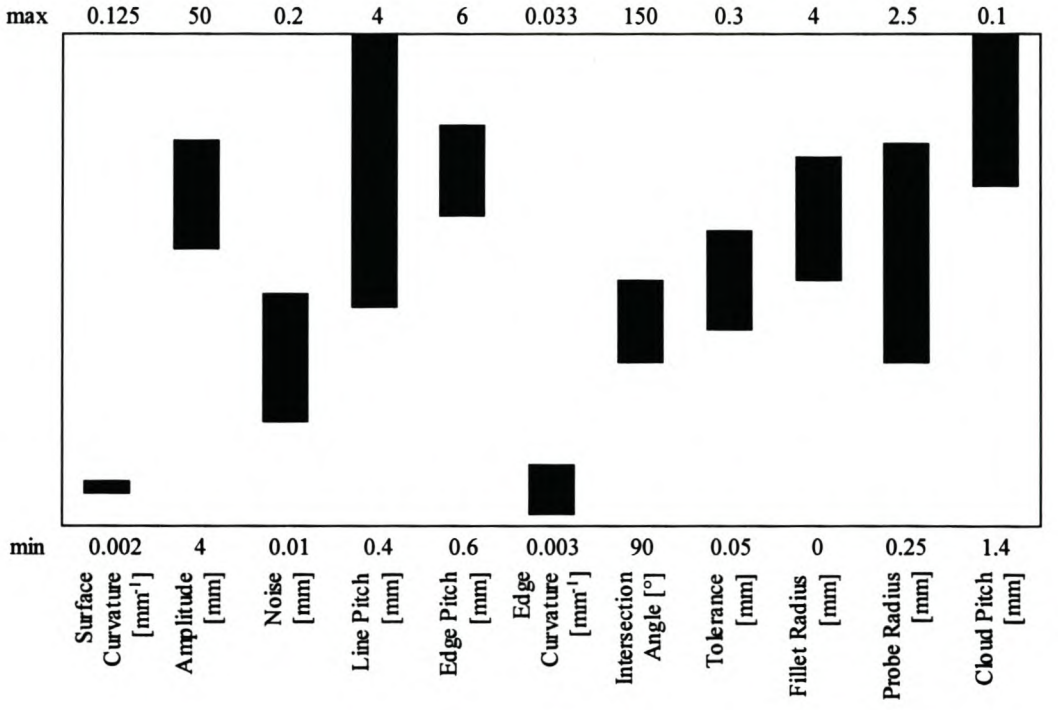


Figure 24 Parameter Range of Experiment 7.

The following tests all failed to scan the complete edge: 7Z1.1; 7Z1.6; 7Z1.8; 7Z2.8; 7S2.14.

This experiment has much the same input values as Experiment 6, but the cloud pitch is higher. This leads to a higher ε_{min}. It can explain the failure of some tests to complete the scan.

5.3.3.7.2. Sensitivity Analysis

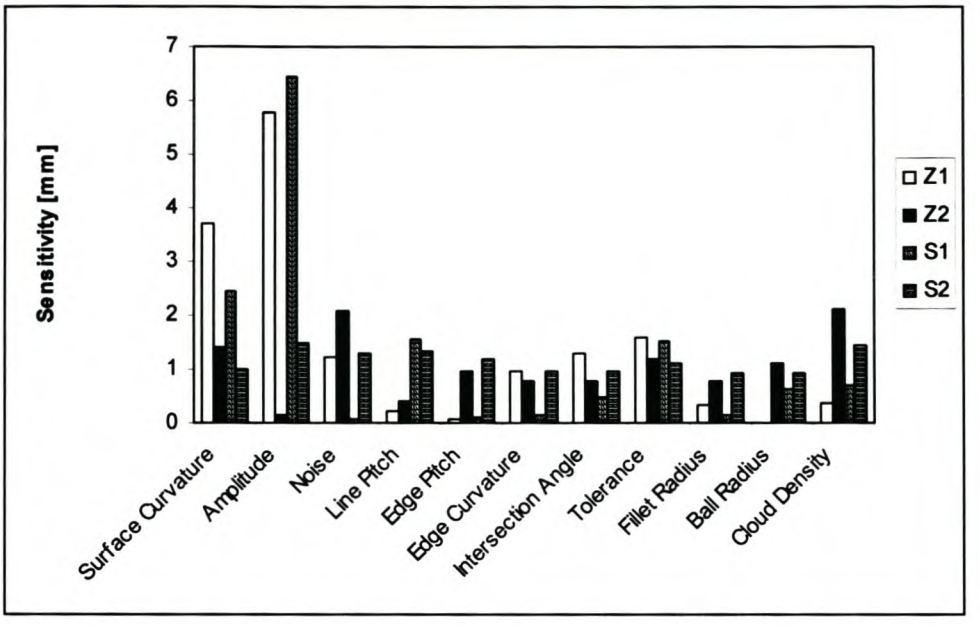


Figure 25 Sensitivity Analysis Results of Experiment 7.

The linear model follows the same trend as Experiment 6; therefore the same observations can be made.

Due to the high error of test 7S2.14, which failed, there is not a significant difference between the sensitivities of the different parameters.

The cloud noise and cloud pitch are prominent for the quadratic zigzag. The observations already made about the influence of these parameters are again applicable here. The absolute value of the pitch is about twice that of the pitch in Experiment 6, thus the noise effect is much higher than in Experiment 6.

5.3.3.8. Experiment 8

5.3.3.8.1. Errors

The following tests all failed to scan the complete edge: 8S1.4; 8S1.11; 8S1.15.

All but the first failed one has tolerance ratios more than 1. The tolerance ratio for 8S1.4 is still high, 0.747.

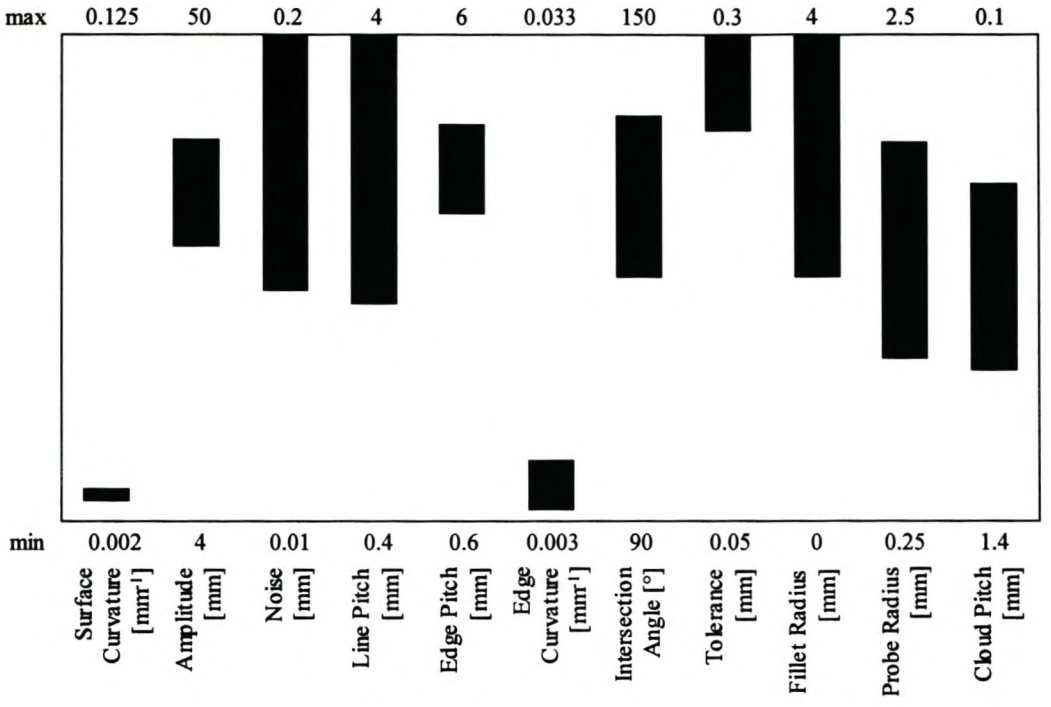


Figure 26 Parameter Range of Experiment 8.

5.3.3.8.2. Sensitivity Analysis

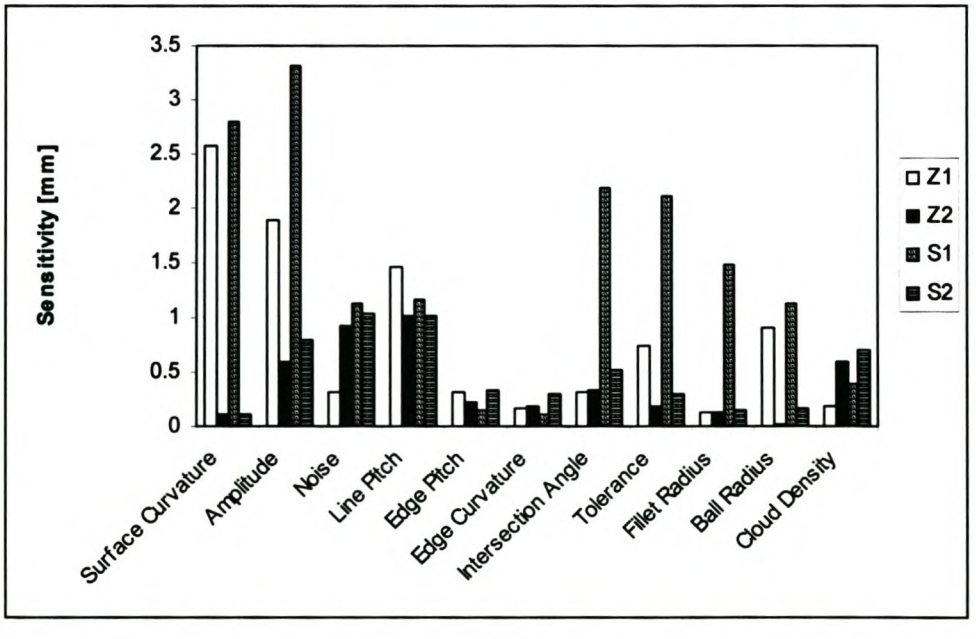


Figure 27 Sensitivity Analysis Results of Experiment 8.

The sensitivity of the linear scans to amplitude, surface curvature and line pitch supports what have been said for previous experiments.

Line pitch, amplitude and noise are again important for the quadratic scans. The reasons are the same as mentioned for earlier experiments. It is probably sensitive to the line pitch, because it helps determine the number of points per line section, which is between 3 and 9 for this case. (Three being rather low for quadric scans.)

5.3.3.9. Experiment 9

5.3.3.9.1. Errors

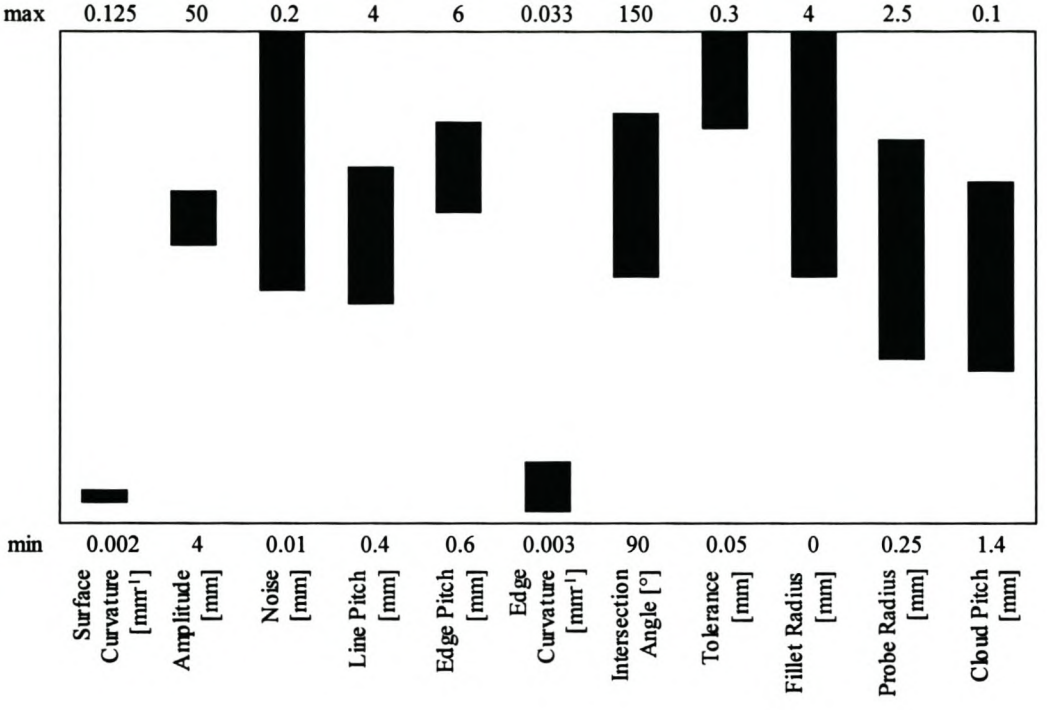


Figure 28 Parameter Range of Experiment 9.

The following tests all failed to scan the complete edge: 9Z1.16; 9Z2.11; 9Z2.14; 9S2.14. The tolerance ratios for all these cases are more than 1.

5.3.3.9.2. Sensitivity Analysis

The same observations made for Experiment 8 can be repeated for this experiment.

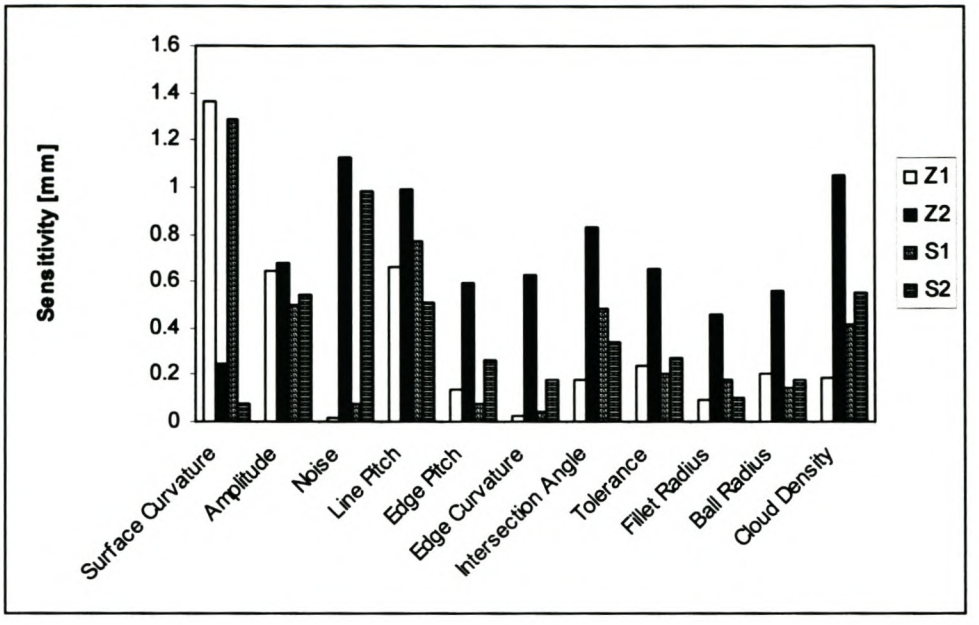


Figure 29 Sensitivity Analysis Results of Experiment 9.

5.3.3.10. Experiment 10

5.3.3.10.1. Errors

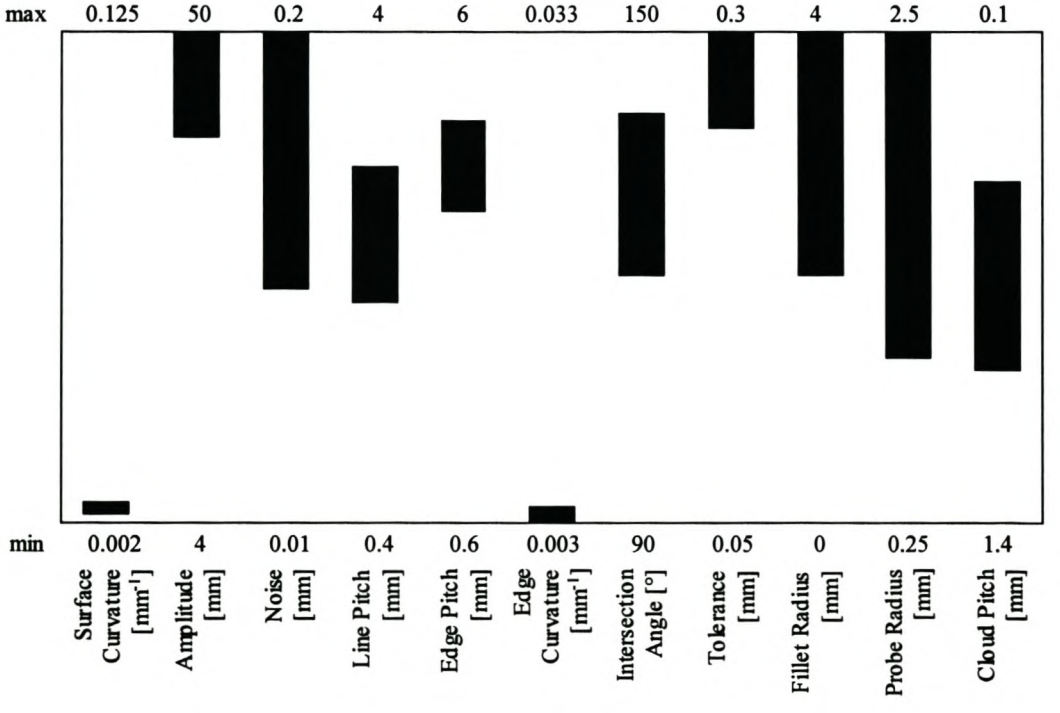


Figure 30 Parameter Range of Experiment 10.

The following tests all failed to scan the complete edge: 10Z2.8; 10Z2.14; 10S1.4; 10S1.7; 10S1.14; 10S1.15; 10S2.14. The tolerance ratios for these cases range from 0.747 to 1.495.

5.3.3.10.2. Sensitivity Analysis

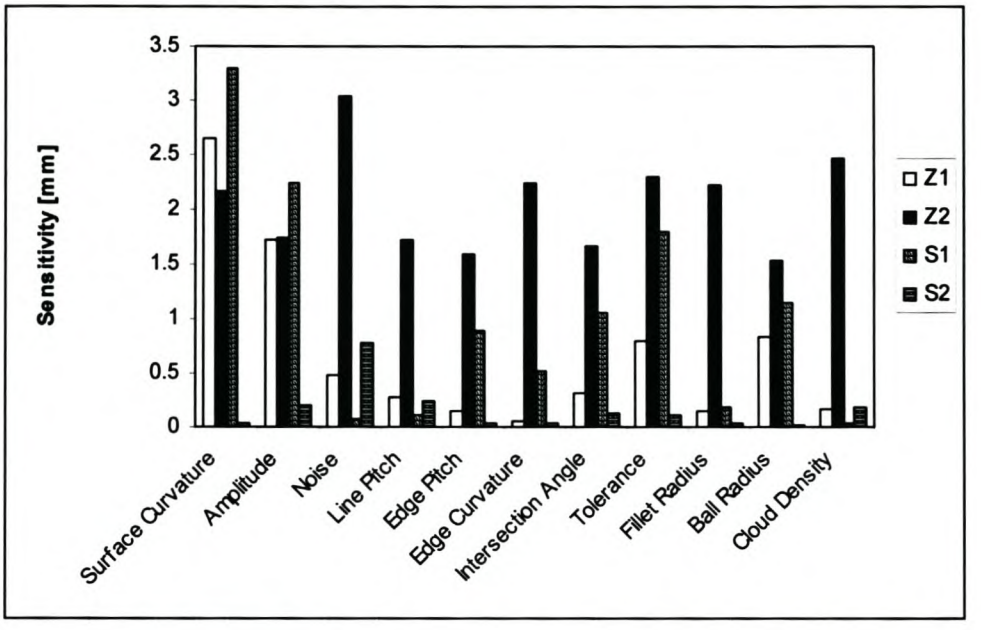


Figure 31 Sensitivity Analysis Results of Experiment 10.

Again, the surface curvature and amplitude are very prominent for the linear scans.

There is nothing of real prominence for the quadratic zigzag. The noise is significant for the quadratic square. With high tolerance ratios and few points per line section (5 – 11 points per scan line section) it is not surprising. It influences the accuracy of the curve approximation.

5.3.3.11. Experiment 11

5.3.3.11.1. Errors

The following tests all failed to scan the complete edge: 11Z2.14; 11S2.9; 11S2.11. The tolerance ratios for all these cases are in the excess of 1.

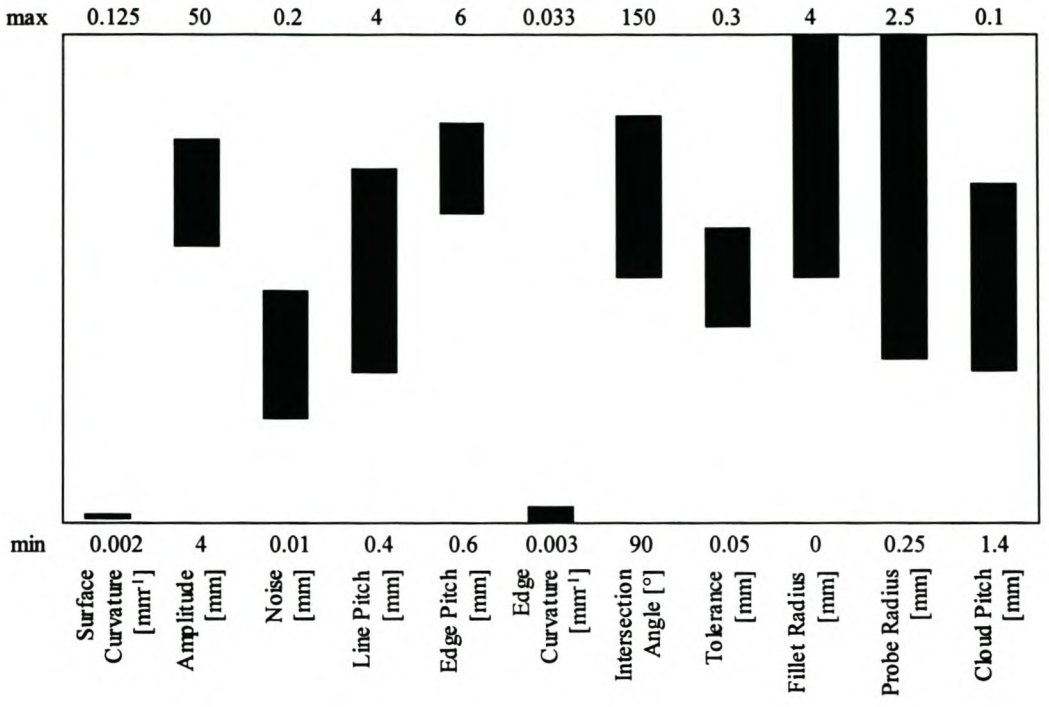


Figure 32 Parameter Range of Experiment 11.

5.3.3.11.2. Sensitivity Analysis

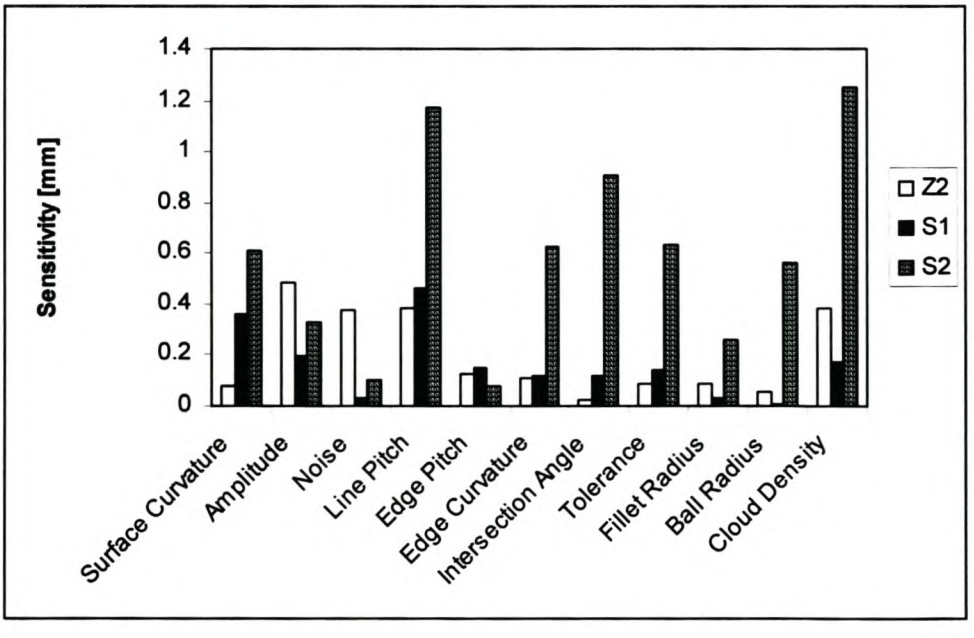


Figure 33 Sensitivity Analysis Results of Experiment 11.

The high error for test 11Z1.1 overwhelms the results for the linear zigzag. The large error is due to the inadequate tolerance ratio (1.491). This led to bad edge point and

tangent estimations, which in turn led to a bad estimation of the next point that the 27th and 28th scan lines aimed for. This is why the error is so high, see Figure 34. What is surprising is not the high error, but the fact that the scan was completed at all. The bad scans at lines 26-28 caused the offset direction to flip. That is why the error is the same order of magnitude as the probe diameter.

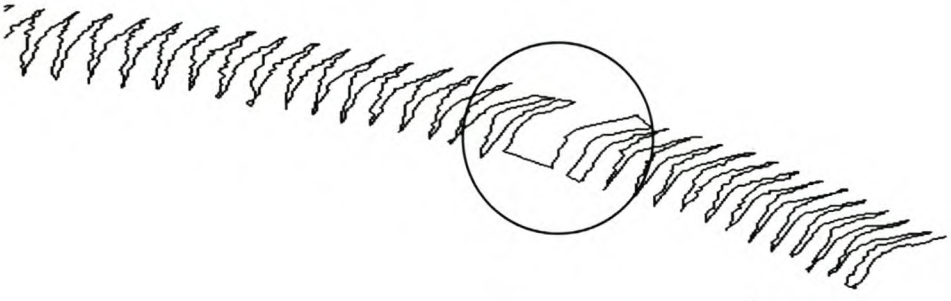


Figure 34 Zigzag Pattern for Test 11Z1.1. (One part of the scan is shown. Notice the circled area.)

For this reason, the sensitivity results for the linear zigzag is not included in Figure 33. It makes it hard to see the trends for the rest of the scans. The same observations made in the previous experiment can be repeated for the results in Figure 33.

5.3.3.12. Experiment 12

5.3.3.12.1. Errors

Due to the lower cloud pitch in this experiment than in the previous experiment, the specified tolerance is adequate to complete the scan.

5.3.3.12.2. Sensitivity Analysis

This is the only experiment where the edge pitch has a significant influence on the linear scans. Seen with the high prominence of the ball radius, it seems that it influences the quality of the offset vector. The line pitch and fillet radius are also important, probably for reasons mentioned earlier.

The amplitude, fillet radius and intersection angle override the influence of the edge pitch for the quadratic scans. The influence of these parameters was discussed earlier.

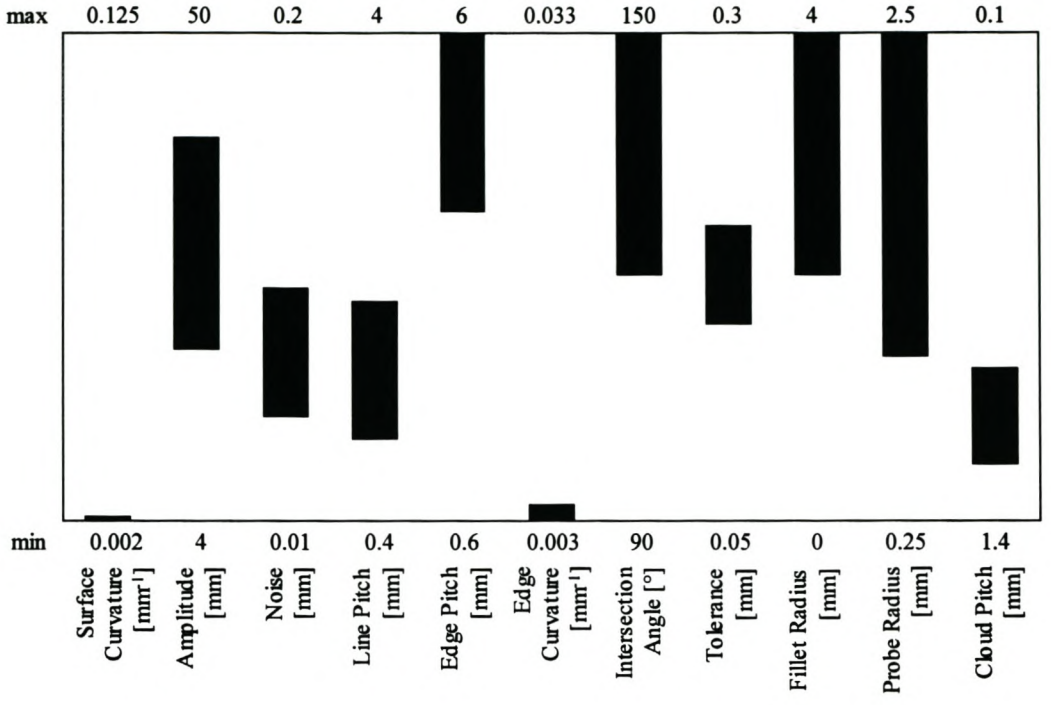


Figure 35 Parameter Range for Experiment 12.

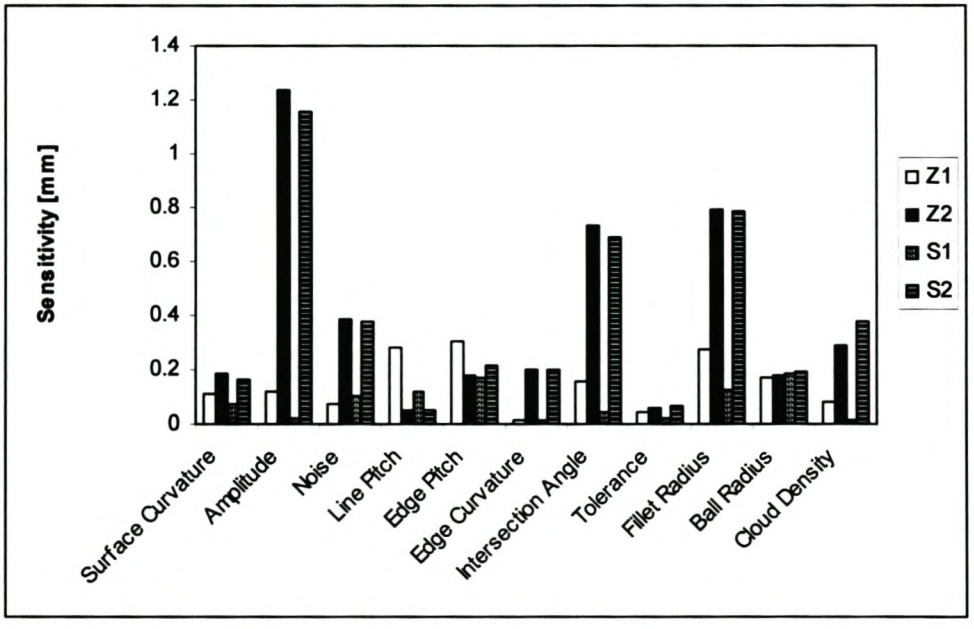


Figure 36 Sensitivity Analysis Results of Experiment 12.

5.3.4. General Conclusions

5.3.4.1. Sensitivity Analysis

The dominance of the surface curvature and amplitude in all the linear scans is clear. This is not surprising since the lines are a much worse approximation of the scan line sections than the quadratic polynomials. It seems that this effect is so strong that it completely overwhelms the influence that any other parameter might have.

The amplitude again features as the most important parameter for the quadratic scans, but the surface curvature seldom features at all. Since the line pitch is prominent with the amplitude, it seems that the effect of the amplitude is on the number of points in each scan line section rather than how well a quadratic polynomial can approximate a circular segment. In fact, the analytical model shows that the quadratic polynomials approximate the circular segments very well for the arc angles used in the experimental testing. The amplitude and line pitch determines the number of points in the scan line section and this has a more important influence on the quality of the quadratic polynomial. This observation is supported by the comparison of the analytical model with actual scans done earlier in this chapter.

The scanning tolerance and point cloud noise also feature prominently for the quadratic scans. Mostly it seems that the approximation is very noise sensitive when working with a small number of points. The scanning tolerance influences the number of points that is included in the approximation.

The intersection angle's effect is more significant for quadratic scans than the linear scans. This was explained in the discussion of Experiment 2.

The effects of the fillet radius and the point cloud pitch were discussed earlier. It seems that they only have an important effect on the quadratic scans. It is possible that the strong influence of the surface curvature and amplitude suppresses the importance of these two parameters for the linear scans, because there does not seem to be any reason why they should not also be significant for the linear scans. The effect of the fillet radius is further diminished for linear scans because they are more robust in extrapolation than the quadratic scans.

5.3.4.2. Range of the Intersection Angle

The range of the intersection angle that was tested is not very large. Some initial tests showed that failures often resulted if angles much larger than 150° were used. However, there is certainly room for further investigation of the practical range of this parameter.

5.3.4.3. Ratio of *df/s*

The significance of the ratio of *df/s* was alluded to in the discussion of Experiment 2. The experiments in general showed that this ratio should be kept below 1 whenever possible. There are some tests that completed the edge scan despite ratios more than 1. It is, however, a good practical guideline to keep $df/s<1$. It also influences the accuracy of the results.

If it is not possible to scan with $df/s<1$, matters can be improved by selecting the scanning parameters such that the maximum number of points per scan line section can be achieved. This improves the curve approximation, which improves the chance that the edge point calculated after extrapolating the curve is close to the actual edge.

5.3.4.4. Success Rate

62 out of the 768 tests were unable to complete the scanning of the edge. This is a success rate of 92%. This is however under laboratory conditions where all the object parameters are known and the distribution of the points in the cloud is also ideal.

It is interesting to note that the square pattern has more failures than the zigzag (37 vs. 25). The reason for this possibly lie in the way with that the scanning plane is determined for the two patterns. The square pattern uses the estimated edge tangent as the scanning plane orientation. The plane origin is at the estimated next edge point. The zigzag pattern uses the edge's tangent as well as the vector from the end point of the last scan line to the estimated next edge point to determine the orientation vector. Therefore, the next scan line starts at the last scanned point. Essentially this means that there is a better chance to minimise the effect of a bad estimate of the edge's tangent when determining the scanning plane orientation for the zigzag pattern than for the square pattern. This leads to a more robust scan.

5.3.4.5. Selection of Scanning Tolerance

Another guideline that was derived as a result of the experimental investigation is the guideline for selecting the scanning tolerance given in equation (5.3.3.5.1.1.). The significance of this guideline is illustrated by its ability to explain many of the failed scans.

It leads to the question whether it is good to project the points to the scanning plane before doing the curve approximation. As explained in paragraph 4.3.1. this is done to improve the robustness of the algorithm.

5.3.4.6. Selection of Scanning Amplitude

The analytical investigation earlier in this chapter indicated that there is an optimum amplitude to scan at. It also hinted, and the experimental investigation further supports the observation, that there is a maximum acceptable amplitude for a specific object. The experimental investigation further shows that there is a minimum amplitude, first of all due to the number of points required on each scan line section and secondly to satisfy $df/s<1$. With this it is possible to bracket the range of practical amplitudes for a specific object.

First, since df/s must be smaller than 1, s must at least be equal to df. Thus, the total length of the scan line, i.e. the amplitude, must be greater than 3 times df. According to the analytical model, $df=2\phi(R_p+R_f)$, where ϕ is found from equation 5.2.1.1. Therefore

$$A \geq 3 \cdot \left(2\phi\left(R_f + R_p\right)\right) = 6\phi\left(R_f + R_p\right) \qquad (5.3.4.6.1.)$$

Also, there must be at least 3 points per scan line section. Since

$$A=s+s+2\phi(R_p+R_f)$$

$$s=A/2+\phi(R_p+R_f)$$

If there must be 3 points on the scan line section, $s/\rho_L=3$. Therefore

$$A/2+\phi(R_p+R_f)=3\rho_L$$

Thus

$$A \geq 6\rho_L - 2\phi(R_p - R_f) \qquad (5.3.4.6.2.)$$

The maximum amplitude obtained with equation (5.3.4.6.1.) and (5.3.4.6.2.) is the minimum practical amplitude.

By comparing the analytical error with some experimental results in Figure 6 it seems that the s/r must not exceed 0.6 for quadratic scans. With $s=0.6r$ the maximum amplitude for the quadratic scan can be derived. Again, since $A=2s+df$

$$A \leq 2(0.6r)+2\phi(R_p+R_f)$$

$$A \leq 1.2r+2\phi(R_p+R_f) \qquad (5.3.4.6.3.)$$

With $(s/r)_{max}=0.1$ from Figure 7 for linear scans, a maximum amplitude for linear scans can be derived in the same way.

$$A \leq 0.2r+2\phi(R_p+R_f) \qquad (5.3.4.6.4.)$$

In Figure 6 it appears that the optimum amplitude is reached at about $s/r=0.4$. Thus, in the same way that the maximum amplitude is derived, the optimum amplitude for quadratic scans is

$$A_{optimum}=0.8r+2\phi(R_p+R_f) \qquad (5.3.4.6.5.)$$

For linear scan it appears from Figure 7 that the optimum s/r is 0.08. Thus, for the linear scans the optimum amplitude is

$$A_{optimum}=0.16r+2\phi(R_p+R_f) \qquad (5.3.4.6.6.)$$

It can happen that inconsistent values of the maximum and minimum amplitudes are found with these equations. It does not mean that it is impossible to scan the edge if this happens, but it does mean that the results can be very inaccurate.

5.3.4.7. Selecting a Good Point Cloud Pitch

The considerations that made it possible to determine the maximum and optimum amplitude can also be used to determine good values for the point cloud pitch.[§]

First of all, since there must be at least 3 points per scan line section and remembering from paragraph 4.3.2. that the point at which the scan line is divided is not used thereafter, the length of the scan line section must be at least $3\rho_L$. In paragraph 4.4.4. it is also determined that $\rho_L \geq 1.5\rho_c$. It is shown that $s \leq 0.6r$ for quadric scans. Therefore the maximum allowable cloud pitch is

$$3(1.5\rho_c) \leq 0.6r$$

$$\rho_c \leq 0.133r \qquad (5.3.4.7.1.)$$

The ideal cloud pitch for quadratic scans is determined by the fact that the best value of s/r is 0.4 from Figure 6. For this test about 20 points where scanned.

$$20(1.5\rho_c) = 0.4r$$

$$(\rho_c)_{ideal} = 0.013r \qquad (5.3.4.7.2.)$$

Similarly a maximum and ideal pitch can be derived for linear scans.

$$\rho_c \leq 0.022r \qquad (5.3.4.7.3.)$$

$$(\rho_c)_{ideal} = 0.003r \qquad (5.3.4.7.4.)$$

The equations for the ideal pitch for the quadratic scans can already result in very dense point clouds. If, for example, the edge of a 60mm diameter pipe must be found, the ideal cloud pitch is 0.4mm. This can result in a massive point cloud that can be very cumbersome to handle on a computer. It can also be very expensive and time consuming to generate such a cloud. The maximum pitch for this example is 4mm.

[§] Of course, often the designers will have to work with what they have. It can happen that they cannot pre-select the cloud pitch. In this case the guidelines developed here can only be useful to decide whether it is feasible to use the edge scanning algorithm at all.

Depending on the cost, time and accuracy constraints, together with the available facilities, a cloud pitch between these two values must be used.

The above equations show that it can also be more practical to use a quadratic scan from the point of view of the size of the point cloud and the cost and time needed to generate it.

It must also be noted that nothing is gained by scanning at a cloud pitch below the ideal scanning pitch.

5.3.4.8. Summary of Guidelines

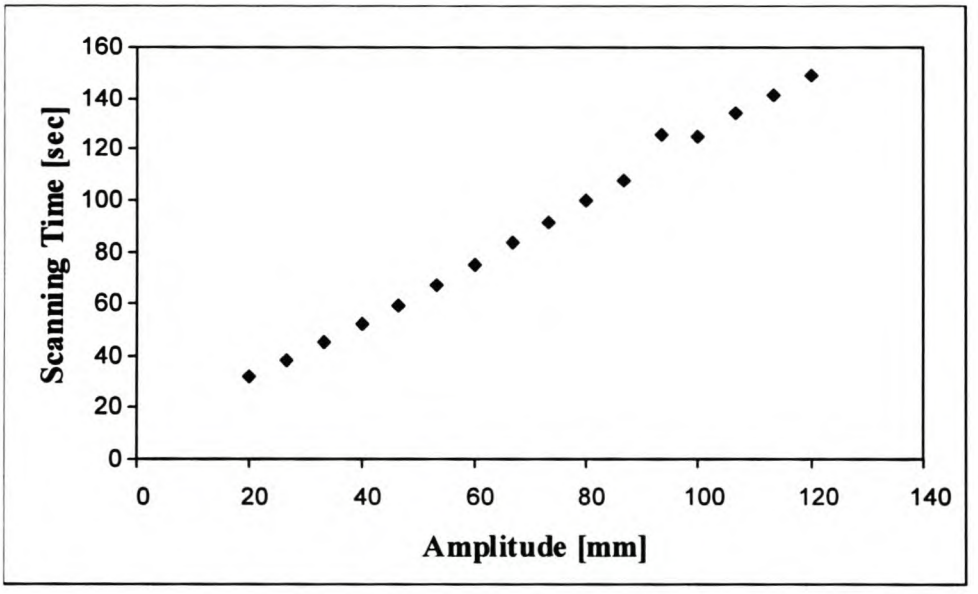
A summary of the guidelines developed in this and the previous chapter are presented in Table 1. These guidelines must be used to select effective scanning parameters.

Table 1 Summary of Guidelines.

Parameter	Scan Type	Guideline	Equation Number
Point Cloud Pitch	Linear	$\rho_c \leq 0.022r$	5.3.4.7.3.
	Linear	$(\rho_c)_{ideal}=0.003r$	5.3.4.7.4.
	Quadratic	$\rho_c \leq 0.133r$	5.3.4.7.1.
	Quadratic	$(\rho_c)_{ideal}=0.013r$	5.3.4.7.2.
Line Pitch	All	$\rho_L \geq 1.5\rho_c$	Paragraph 4.4.4.
Edge Pitch	No guideline was developed. The best is to make it equal to the line pitch.		
Amplitude	All	$A \geq 6\phi(R_f + R_p)$ and $A \geq 6\rho_L - 2\phi(R_p - R_f)$	5.3.4.6.1. 5.3.4.6.2.
	Linear	$A \leq 0.2r+2\phi(R_p+R_f)$	5.3.4.6.4.
	Quadratic	$A \leq 1.2r+2\phi(R_p+R_f)$	5.3.4.6.3.
	Linear	$A_{optimum}=0.16r+2\phi(R_p+R_f)$	5.3.4.6.6.
	Quadratic	$A_{optimum}=0.8r+2\phi(R_p+R_f)$	5.3.4.6.5.
Scanning Tolerance	All	$\varepsilon_{min}=\eta+\rho_c\sin(10°)$	5.3.3.5.1.1.

5.4. Scanning Time

Figure 37 shows the scanning time for 16 scans done on a point cloud containing 123426 points. The experiment was done on a Pentium III 733MHz computer with 128MB RAM. The increase in amplitude means that the number of points per scan line increased linearly for each scan. A quadratic square pattern was used for all 16 scans. Clearly the scanning time is proportional to the amplitude if all other parameters remain the same. The times shown exclude the time for building the octree.

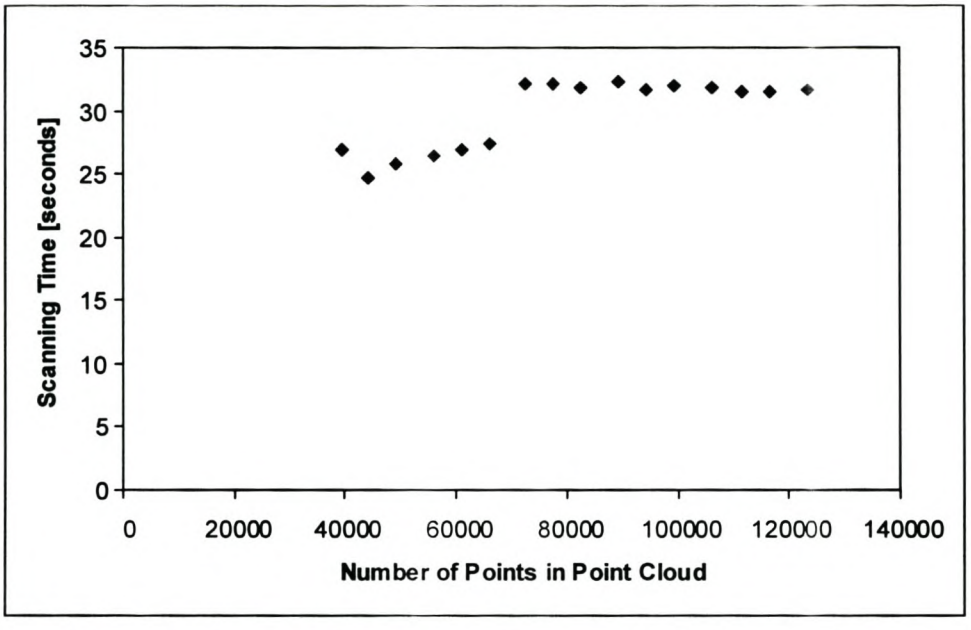


$\rho_L = \rho_e = 2\text{mm}$ $R_p = 0.5\text{mm}$ $\varepsilon = 0.2\text{mm}$
$\kappa_e = 0.004\text{mm}^{-1}$ $R_f = 0\text{mm}$ $\kappa_s = 0.01\text{mm}^{-1}$
$\theta = \pi/2$ $\rho_c = 1\text{mm}$ $\eta = 0.005\text{mm}$

Figure 37 Scanning Time with Increasing Amplitude.

The rectangular box representing the root node of the octree is such that it will just enclose all the points in the point cloud. As explained in Chapter 3 the root node is refined until the length of the smallest side of any node is below a specified minimum length. This will determine the number of levels in the octree. An investigation into the search time for point clouds with increasing number of points was done and the results are presented in Figure 38. The first six tests shows an almost constant time to scan the edge. Then there is a small step and the next 10 tests again took almost the same time. By keeping all the other scanning parameters the same while increasing the number of points in the cloud, the size of the box containing the points increased

for each test. Therefore the root node in the first 6 tests were refined 4 times while that of the last 10 tests were refined 5 times. It is interesting to note that the scanning time does not depend on the number of points in the cloud, but rather on the number of levels in the octree. This is a great advantage of storing the points in an octree data structure. Note that in Figure 38 the octree build time is not included.



$\rho_L = \rho_e = 2\text{mm}$ $R_p = 0.5\text{mm}$ $\varepsilon = 0.2\text{mm}$
$\kappa_e = 0.004\text{mm}^{-1}$ $R_f = 0\text{mm}$ $\kappa_s = 0.01\text{mm}^{-1}$
$\theta = \pi/2$ $\rho_c = 1\text{mm}$ $\eta = 0.005\text{mm}$

Figure 38 Scanning Time with Increasing Point Cloud Size.

Chapter 6.

Surface Modelling

6.1. Introduction

The edge scanning algorithm developed in the preceding chapters is aimed at expediting the approximation of point clouds with parametric surfaces. The surface modelling techniques discussed in this chapter are used in the next chapter. There it is shown how the edge information can be combined with these techniques in order to accelerate the Reverse Engineering process.

This chapter does not contain new work. It should rather be seen as a literature review. The only exception is the section on surface-surface intersections and possibly the section on surface extensions. In the latter, an interesting result is presented on the surface extension method used in AutoCAD (2000). In the former, a comparison is made with the new edge detection algorithm.

The chapter introduces the terminology and notation used in this thesis. Various NURBS surface techniques are discussed, such as least squares fitting, lengthening surfaces, finding surface-surface intersections, etc. The chapter concludes with a paragraph on the modelling and approximation of swept surfaces.

6.2. Basic B-spline and NURBS Theory

The purpose of this paragraph is simply to introduce the notation used in the rest of this chapter. This is done by giving the well known equations for B-spline and NURBS curves and surfaces. In the process, some terminology is also clarified. As far as possible, this thesis follows the notation adopted by Piegl and Tiller (1997).

6.2.1. B-Spline Curves and Surfaces

B-spline curves are defined by the following well known parametric equation:

$$\boldsymbol{C}(u)=\sum_{i=1}^{n} N_{i,k}(u)\boldsymbol{P}_i \qquad (6.2.1.1.)$$

In this equation, $\boldsymbol{P}_i$ are the n three dimensional control points of the B-spline. The parameter, u, is defined between fixed real numbers a and b, such that $a \leq u \leq b$ and $a < b$. $N_{i,k}(u)$ is the polynomial basis function of degree k (order k+1) on the variable u. The basis polynomials are completely defined by the degree k and a knot vector, $\{u_j\}_{j=0}^{n+k}$ such that $u_j \leq u_{j+1}$ for $j=0...n+k$. In this thesis, it is assumed that the knot vector is defined as follows:

$$a = u_1 = u_2 = \ldots = u_k < u_{k+1} \leq u_{k+2} \leq \ldots \leq u_n < u_{n+1} = \ldots = u_{n+k} = b$$

Knots of multiplicity greater than one are allowed.

The basis polynomials, $N_{i,k}(u)$, for a fixed u are calculated with the recurrence relation for B-splines given by De Boor (1978) and shown in the following equation:

$$N_{j,k}(u) = 1, \quad u_j \leq u < u_{j+1} \qquad \text{for } k=0$$

$$= 0, \quad \textit{otherwise}$$

$$N_{j,k}=\frac{u-u_j}{u_{j+k-1}-u_j}N_{j,k-1}(u)+\frac{u_{j+k}-u}{u_{j+k}-u_{j+1}}N_{i+1,k-1}(u) \qquad (6.2.1.2.)$$

The B-spline tensor product surface is given by the following equation:

$$\boldsymbol{S}(u,v)=\sum_{i=1}^{n}\sum_{j=1}^{m} N_{i,k}(u)N_{j,l}(v)\boldsymbol{P}_{ij} \qquad (6.2.1.3.)$$

There are n times m control points for the surface.

6.2.2. NURBS Curves and Surfaces

NURBS (or Non–Uniform Rational B-spline) was formulated by Tiller (1983). Rational B-spline curves are defined by

$$\boldsymbol{C}(u)=\frac{\sum_{i=1}^{n} N_{i,p}(u) w_i \boldsymbol{P}_i}{\sum_{i=1}^{n} N_{i,p}(u) w_i} \qquad (6.2.2.1.)$$

$N_{i,p}(u)$ is the i^{th} B-spline basis function of degree p on the parameter u, as defined by equation 6.2.1.2. NURBS are defined in 4D homogeneous space. The control points in 4D space are $\boldsymbol{P}^w=(wx,wy,wz,w)^T$. The i^{th} homogeneous coordinate, or weight, is w_i. $\boldsymbol{P}_i$ is the i^{th} 3D control point, $\boldsymbol{P}_i=(x,y,z)^T$.

A NURBS surface, as given by equation (6.2.2.2.), is a bi-variate extension of the NURBS curve.

$$\boldsymbol{S}(u,v)=\frac{\sum_{i=1}^{n}\sum_{j=1}^{m} N_{i,p}(u) N_{j,q}(v) w_{i,j} \boldsymbol{P}_{i,j}}{\sum_{i=1}^{n}\sum_{j=1}^{m} N_{i,p}(u) N_{j,q}(v) w_{i,j}} \qquad (6.2.2.2.)$$

The knot vector definition and the calculation of the B-spline basis functions are the same as the B-spline curves and surfaces. Note that when $w_i=1$ for all $i=1,\ldots,n$, the original B-spline representation is again obtained.

6.3. NURBS Surface Approximation

NURBS surface approximation is a very common method of reconstructing a surface in Reverse Engineering. It has several advantages over triangulating the data (also a very common reconstruction approach). For one it gives a much more compact representation of the surface. It gives a surface of arbitrary continuity. (Cubic surfaces are at least C^2 continuous.) The nature of the approximation process is such that it will filter out noise in the data. It gives a designer more opportunity to modify the model if that is necessary as well as better control of the final shape. Approximation is however a very time consuming process requiring constant involvement by the designer. A lot has been published about the problem in the last decade. This paragraph briefly refers to the methods used in this project.

The distance between the NURBS surface and the point cloud must be minimised.

$$\min of \sum_{j=1}^{m} \left\| \boldsymbol{Q}_j - \boldsymbol{C}(u_j, v_j) \right\|^2 \qquad (6.3.1.)$$

$\boldsymbol{Q}_j$, $j=1,...,m$ are the data points. $\boldsymbol{C}(u_j,v_j)$ is the corresponding point on the NURBS surface and (u_j,v_j) are the parameters assigned to each point. This is a non-linear problem since the parameters for each point, (u_j,v_j), as well as the control points of the surface must be determined. The knot vectors of the surface must also be found.

It appears that when working with unstructured point clouds that the base surface parameterisation method suggested by Ma (1994) (see also Ma and Kruth, 1995a) is the best option. Essentially, the designer must construct a surface that roughly approximates the point cloud. The points are then projected onto this surface to obtain their parameters. The better the base surface approximates the point cloud, the better the parameterisation will be.

The base surface method seems cumbersome since the designer must construct a separate surface simply to parameterise the cloud, but Piegl and Tiller (2001) argue that the process cannot be automated and they agree that this method is the best available to date. A number of researchers have tried various methods to iteratively optimise the parameterisation, see for example Hoschek (1988), Rogers and Fog (1989), Sarkar and Menq (1991b) and Lai and Lu (1996). However, it is this author's experience that the improvement made during parameter optimisation is not worth the computational time. Rogers and Fog (1989) agree that the initial parameterisation is often good enough.

After parameterisation, the knot vectors must be selected. Here, the method proposed by Piegl and Tiller (1997, pp. 412) is used. This results in a uniform distribution of the knots. However, this is not always ideal. In regions of higher surface curvature more control over the surface's shape can be obtained by inserting more knots in that region, provided that the cloud is dense enough. This is why Sarkar and Menq (1991a) and Ma (1994) suggest that in a good Reverse Engineering system, the designer must be able to refine the knot vectors interactively.

Once the parameterisation and knot vectors are known, equation 6.3.1. is rewritten in matrix form, equation 6.3.2., and the control points can be solved with Gauss elimination. In equation 6.3.2. $\boldsymbol{N}$ is the matrix containing the B-spline basis functions,

$\boldsymbol{P}$ is the vector of control points and $\boldsymbol{Q}$ is the vector containing the point cloud's points. Dierckx (1993) and Piegl and Tiller (1997), amongst others, discuss the derivation of this equation.

$$\boldsymbol{N^T N P = N^T Q} \qquad (6.3.2.)$$

Strictly speaking, the weights of the NURBS surface's control points must also be calculated. However, this is seldom done, probably because little is gained at considerable additional computational cost. Most researchers assume that the weights are known. In practice this means that the weights will most likely all be 1, in which case the problem of NURBS surface approximation reduces to the problem of B-spline surface approximation. This author knows only about the attempts by Ma and Kruth (1995b) and Yau and Chen (1997) to calculate the weights. In this work, the weights are assumed to be 1 for all control points.

6.4. Lengthening NURBS Surfaces

As mentioned earlier in this thesis, there often are gaps between the approximated surface due to the point cloud segmentation. If no boundary conditions are applied, gaps will exist regardless of the segmentation. One way of avoiding the gaps, as already implied, is to apply boundary conditions during surface approximation. Kruth and Kerstens (1998) show how this can be done, but they warn that problems occur if the edges where the surfaces meet are not of equal length. Lai and Lu (1996) create blending surfaces between the fitted surfaces.

This author's experience is that extending the surfaces and calculating the intersection curve is a very robust method. Only the paper by Shetty and White (1991) was found on this topic. They present a method for extending surfaces using tangential or curvature continuity. Their method is presented in Appendix H.

Their method was implemented for linear extrapolations. The results obtained with their method were compared to surface extensions done with AutoCAD (2000). It is interesting to note that in at least one case, the extension done with AutoCAD (2000) did not result in the promised tangential continuity. It is not clear why this error occurred. It is also not easy to see, because the tessellation of the surface in the AutoCAD (2000) graphical interface hides the discrepancy. The error was noticed

when the derivatives across the boundary were checked. The extension with Shetty and White's (1991) method produced a perfect extension for the same surface. (The results and surface definitions for this case are given in Appendix H.) AutoCAD (2000) apparently produce no problem when the weights of the surface's control points are all one. In the case mentioned here, the weights were not all equal to one.

6.5. Surface-surface Intersections

6.5.1. Intersection of NURBS Surfaces

There are essentially two approaches in the literature for calculating intersections of NURBS, or B-spline, surfaces.

The first approach is the divide-and-conquer algorithm implemented by Peng (1984) for B-spline surfaces. The convex hull property of B-spline surfaces is used. The surface is subdivided (see Böhm, 1981) until the patch is small enough so that it can be considered a flat surface. The convex hull property is used to do the checking. This is done for both surfaces. The intersection points are found from the intersection of the small flat surfaces. Numerous refinements are possible to limit the subdivision to the region of the intersection. Of course, NURBS surfaces have the same convex hull property as B-spline surfaces and can be subdivided in the same way as the B-spline surfaces.

This type of algorithm has the advantage that it is computationally robust. The trouble is that it only gives a number of intersection points without any indication whether the points belong to the same intersection curve. Peng (1984) implemented an extrapolation method that traces the entire length of the intersection curve.

The alternative method of finding the intersection is an iteration method. After selecting starting points, the distance between surfaces are minimised until the intersection points are found. Chen and Ozsoy (1988) implemented such a method for parametric surfaces in general. One of the major problems with this method is finding good starting values for the iteration scheme. Therefore, convergence of the algorithm is not guaranteed. Abdel-Malek and Yeh (1997) address this issue. Interestingly they remark that divide-and-conquer methods are most often used.

Due to the comments of Adbel-Malek and Yeh (1997) a divide-and-conquer method was implemented for NURBS surfaces for this project. Refer to Peng (1984) for details of the algorithm.

6.5.2. Comparison with Edge Scanner Method

The surface extension method described earlier in this chapter is used with the intersection algorithm to create trimmed NURBS surfaces that ensures a closed surface model. In a simple test the results obtained with this approach are compared with results obtained with the edge scanning algorithm in this paragraph. (This is not intended as a comprehensive comparison of the two methods. Such a comparison is beyond the scope of this project.) Experiment 2S1.11 (Appendix B) is used in the comparison.*

The parameters of the base surfaces, constructed in AutoCAD (2000), are given in Appendix I. The base surfaces and the approximated surfaces are shown below.

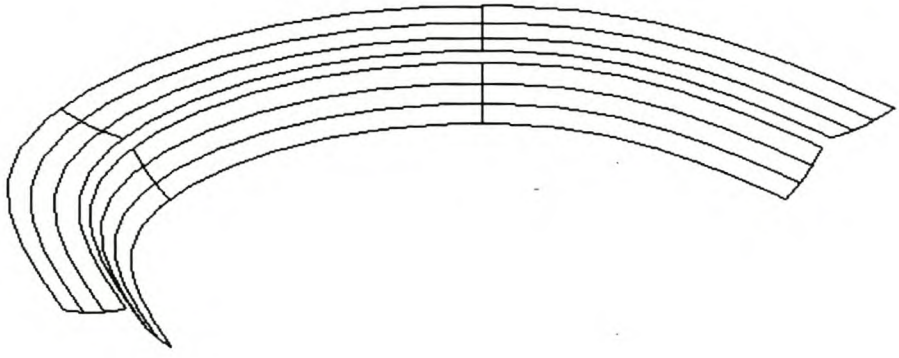


Figure 1 Base Surfaces Used to Parameterise the Point Cloud.

* This specific example was selected because the object has a reasonable fillet radius (1mm) and the size of the object combined with the point cloud density means that the point clouds used in the surface approximation are easier to handle.

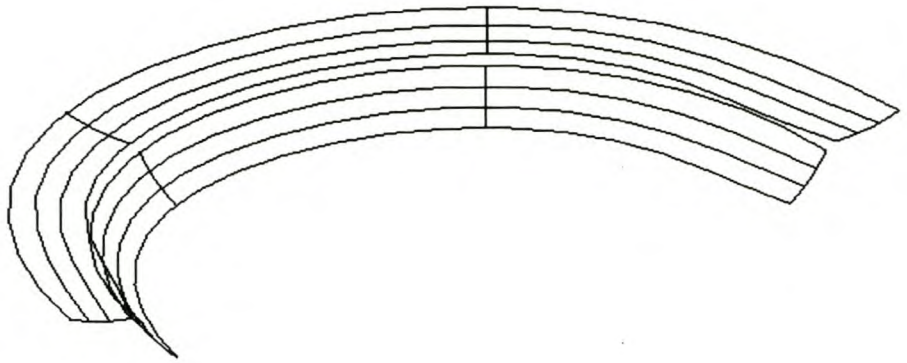


Figure 2 Fitted Surfaces Used to Calculate Intersection Points.

The average errors of the outer and inner surfaces respectively are 0.027mm and 0.046mm. The surfaces were extended and the intersection curve calculated. The intersection curve was compared with the real intersection circle. The average error is 0.086mm. The average error of the edge detected with the edge scanning algorithm is 0.094mm.

The first observation is that the errors are of the same order of magnitude. This is as expected. Both the surface intersection method and the edge scanning method use a linear extrapolation to find the intersection curve.

The surface intersection method's result is a little better than the edge scanning method. In this case it is expected since the surface curvature is significant enough to make the influence of the deviation from the line approximation important.

Lastly, the surface intersection method's result can be improved by improving the surface fit. The fitting result is not the best if the maximum point cloud noise of only 0.01mm is considered. The high errors are probably due to the particular base surfaces used in this example. No attempt was made to improve the surface approximation.

6.6. Swept Surfaces

6.6.1. The Use of Swept Surfaces in Reverse Engineering

In the literature review the point is made that it is often necessary to extract surfaces with engineering "meaning" from the point cloud. When an object is simply copied,

any surface definition that satisfies the requirements for smoothness and accuracy can be used. However, as soon as the designer has to modify parts of an object, it is necessary to have surfaces that can be manipulated within a standard CAD package. This can be done to some extent with free-form surfaces.

However, most often engineering components, such as the core box for the manifold described in the pilot study, are modelled with standard surfaces such as extrusions, surfaces of revolution, ruled surfaces and swept surfaces. These surfaces are well suited to parametric modellers such as AutoCAD Mechanical Desktop (2000).

Under certain circumstances it is therefore useful to extract the original surface type from the point cloud. Elsässer and Hoschek (1996) describe the approximation of point clouds with surfaces of revolution. Ueng and Lai (1998) (also Ueng et al., 1998) discuss approximation by swept surfaces.

6.6.2. Swept Surface Definition

A swept surface is defined as a profile curve that is traversed along a trajectory. It can either be only translated or rotated and translated along the trajectory. In equation 6.6.2.1. $\boldsymbol{T}(u)$ is the trajectory curve, $\boldsymbol{P}(v)$ is the profile curve and $\boldsymbol{M}(u)$ is the transformation matrix representing the translation or translation and rotation. $\boldsymbol{M}(u)$ can also be used to scale the profile curve.

$$\boldsymbol{S}(u,v) = \boldsymbol{T}(u) + \boldsymbol{M}(u)\boldsymbol{P}(v) \qquad (6.6.2.1.)$$

If the profile curve must be rotated as well, some difficulties arise to ensure a consistent rotation. This is described by Piegl and Tiller (1997), amongst others.

Ueng et al. (1998) avoid this problem by only translating the profile curve, but adding blending functions. This makes it possible to use two profile curves and two trajectories. Given two profiles (superscript ^{P1} and ^{P2}) and two trajectories (superscript ^{T1} and ^{T2}) as follows, the swept surface definition of Ueng et al. (1998) is given in equation 6.6.2.6. (The nomenclature of this and the following equation is explained in paragraph 6.2.)

$$\boldsymbol{C}^{P1}(v) = \sum_{j=1}^{m} N_{j,q}(v)\boldsymbol{P}_j^{P1} \tag{6.6.2.2.}$$

$$\boldsymbol{C}^{P2}(v) = \sum_{j=1}^{m} N_{j,q}(v)\boldsymbol{P}_j^{P2} \tag{6.6.2.3.}$$

$$\boldsymbol{C}^{T1}(u) = \sum_{i=1}^{n} N_{i,p}(u)\boldsymbol{P}_i^{T1} \tag{6.6.2.4.}$$

$$\boldsymbol{C}^{T2}(u) = \sum_{i=1}^{n} N_{i,p}(u)\boldsymbol{P}_i^{T2} \tag{6.6.2.5.}$$

$$\boldsymbol{S}(u,v) = \begin{bmatrix} \beta_0(v)\gamma(v)\left[N_{i,p}(u) - \alpha_0(u)N_{i,p}(0) - \alpha_1(u)N_{i,p}(1)\right] \\ \beta_1(v)\gamma(v)\left[N_{i,p}(u) - \alpha_0(u)N_{i,p}(0) - \alpha_1(u)N_{i,p}(1)\right] \\ \alpha_0(u)N_{j,q}(v) \\ \alpha_1(u)N_{j,q}(v) \end{bmatrix}^T \begin{bmatrix} \boldsymbol{P}_i^{T1} \\ \boldsymbol{P}_i^{T2} \\ \boldsymbol{P}_j^{P1} \\ \boldsymbol{P}_j^{P2} \end{bmatrix} \tag{6.6.2.6.}$$

or

$$\boldsymbol{S}=\boldsymbol{NP} \tag{6.6.2.7.}$$

with

$$\alpha_0(u) = 1-u \tag{6.6.2.8.}$$

$$\alpha_1(u) = u \tag{6.6.2.9.}$$

$$\beta_0(v) = 1-v \tag{6.6.2.10.}$$

$$\beta_1(v) = v \tag{6.6.2.11.}$$

$$\gamma(v) = \frac{\left\|\boldsymbol{C}^{P1}(v) - \boldsymbol{C}^{P2}(v)\right\|}{\beta_0(v)\left\|\boldsymbol{C}^{P1}(0) - \boldsymbol{C}^{P2}(0)\right\| + \beta_1(v)\left\|\boldsymbol{C}^{P1}(1) - \boldsymbol{C}^{P2}(1)\right\|} \tag{6.6.2.12.}$$

An important disadvantage of swept surfaces is that equation 6.6.2.1. cannot be written in the compact form for NURBS surfaces given in equation 6.2.2.2. The implication is that separate algorithms for such operations as lengthening and finding intersections (as presented earlier in this chapter), must be written. This is not a

problem for some of the other standard surface types such as ruled or revolved surfaces (Piegl and Tiller, 1997).

Due to this shortcoming a number of researchers (Bronsvoort and Waarts, 1992, Piegl and Tiller, 1997 and Jüttler and Wagner, 1999) have made a skinning approximation of the swept surface. A skinned surface is a surface that interpolates a number of cross sectional curves. Sometimes a spine curve can be added to aid in the orientation of the skinned surface. When approximating a swept surface with a skinned surface, the task is to find a number of cross sectional curves, essentially the profile curve of the swept surface translated and rotated a number of times, that best represent the swept surface.

6.6.3. Approximating Point Clouds with Swept Surfaces

From equation 6.6.2.7. the usual least squares equation can be written.

$$\boldsymbol{N^T Q = N^T N P}$$

$\boldsymbol{Q}$ contains the point cloud points and $\boldsymbol{N}$ contains the B-spline basis function for the parameters (u,v) of a specific point in the cloud. This part is essentially the same as the approximation of tensor product NURBS surfaces discussed earlier in this chapter. Again, it is a non-linear problem since the parameterisation of the point cloud is unknown. Ueng et al. (1998) use points in a structured grid. Thus, they are able to get away with the usual chord length parameterisation. Since no assumption is made about the structure of the cloud in this thesis, a base surface parameterisation is more appropriate.

There is an additional problem with fitting the swept surfaces. Note the equation for $\gamma(v)$, equation 6.6.2.12. An estimate of the two profile curves is needed before the approximation can start. Ueng et al. (1998), working with a structured grid, approximated the first and last row of points with B-spline curves. They use these curves to start the fitting process.

When using a base surface parameterisation such initial estimates of the profile curves are harder to make. In this project the profile curves of the base surfaces are used if the base surface is a swept surface. If a tensor product surface is used, the curves on

the two opposite ends of the surface can be extracted and used to start the approximation.

Ueng et al. (1998) go on to iteratively improve the parameterisation using Powell's method (see for example Press et al. 1997). Their results of five examples show that the approximation does not improve much beyond three iterations. In this work the parameters are not improved after the initial parameterisation for the reasons discussed earlier in this chapter.

6.7. Segmenting a Point Cloud

If a good base surface can be constructed, it can also be used to cut the point cloud. The method is suggested by Bradley and Chan (2001). They suggest that the point cloud can be segmented by finding all the points that lie within a certain distance from the base surface. The same algorithm that is used to parameterise a patch of the point cloud can be used to do this segmentation.

This is a very intuitive way of segmenting a point cloud. However, it can require considerable work from the designer depending on the complexity of the base surface that is needed to segment the cloud.

Chapter 7.

Case Study

7.1. Description

The core box of an IC engine's inlet manifold shown below was used to test the edge scanning strategy, as well as the swept surface approximation method, in a practical example. The surfaces of the pipes appear to be swept surfaces. They are bounded by well defined C^1 edges that appear to be ideally suited for extraction with the edge scanning algorithm.

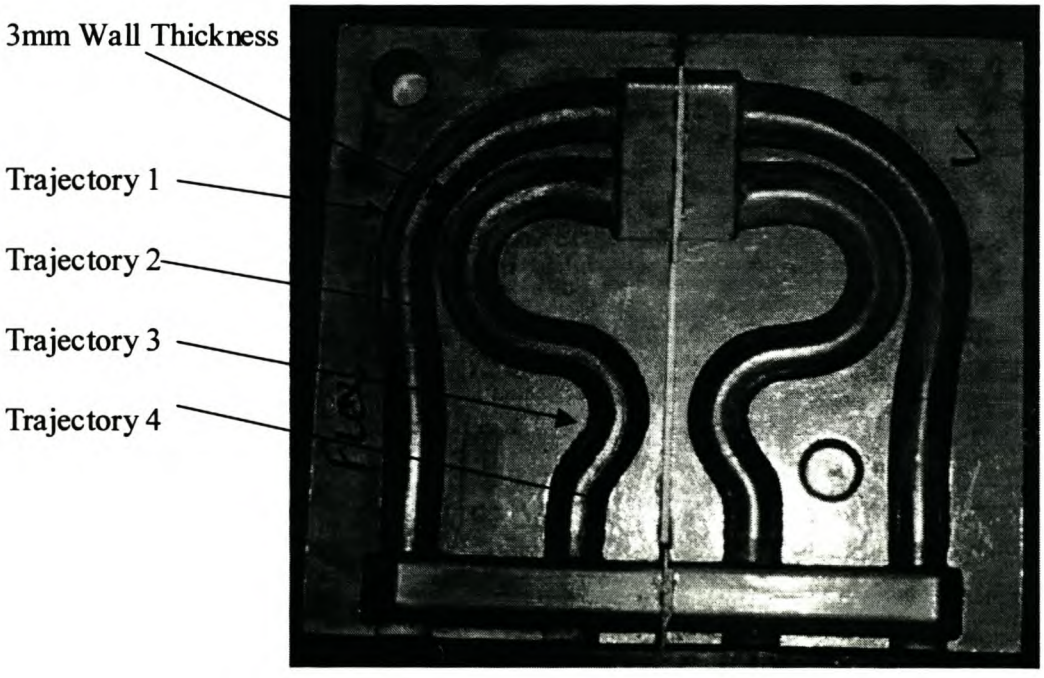


Figure 1 Core Box Used in Case Study.

The diameter at the narrowest section of the pipes is about 34mm. A scanning pitch of 0.5mm was chosen in order to remain within the limits of the edge scanning algorithm set in Chapter 5. The scanning was done on the Renishaw Cyclone. The Cyclone scans a regular gird pattern. It ensures that the pitch along the scan lines is no more than 0.5mm despite the curvature of the object. Unfortunately, the step over distance

between the scan lines, also 0.5mm in this example, is a step along the coordinate axis only and does not take the object's curvature into account.

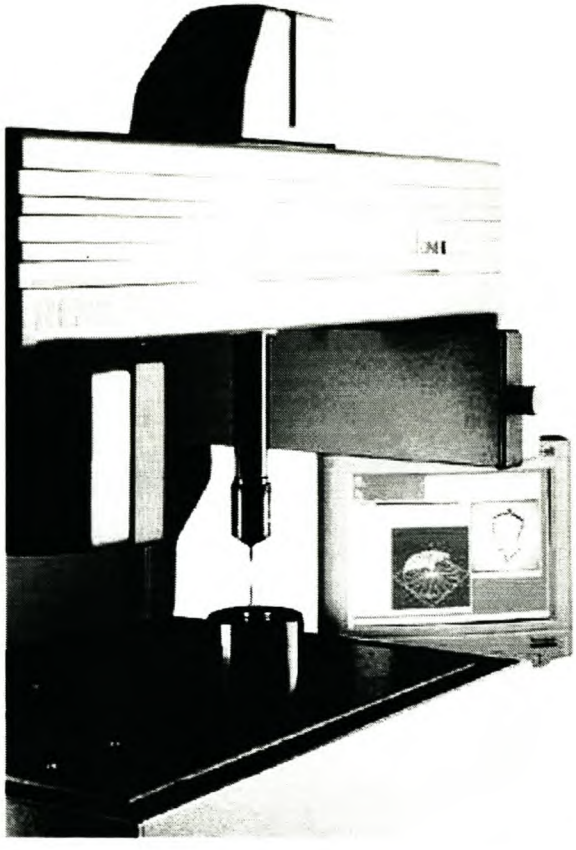


Figure 2 Renishaw Cyclone. (Anonymous, 2001b)

The choice of scanning parameters is further limited by the narrowest gap between neighbouring edges. This gap is shown in Figure 1. The gap is 3mm. Enough points must be scanned in this region to ensure that a line or polynomial can be fitted to calculate the edge.

This scan was completed in 5 hours, excluding set up time. (The time study results are given in Appendix F.) The result is a 22MB text file containing the scanned points. These points are not compensated with the probe diameter of 2.011mm.

The ideal scanning pitch, according to equation 5.3.4.7.2. is 0.221mm. However, a 0.5mm scanning pitch resulted in a point cloud of 22MB. This is already rather large. For practical purposes it was decided not to scan at a higher resolution. It must also be remembered that the scanning time on the Cyclone increases dramatically, and so too the scanning cost, if scanning at a resolution of 0.221mm rather than 0.5mm. The

0.5mm pitch is still safely below the maximum allowable pitch of 2.261mm (equation 5.3.4.7.1.).

7.2. Edge Detection

The edge scanning algorithm developed in this project was used to find the four trajectories indicated in Figure 1. Scanning parameters were selected according to the guidelines developed in Chapter 5. The intersection angle between the pipe's surface and the split plane was estimated as 90° and the minimum surface radius of curvature as 34mm. The edges are sharp enough to ignore the fillet radius caused by wear. These estimates of the surface parameters indicate that the scanning amplitude must be between 6.75mm and 22mm (according to equations 5.3.4.6.2. and 5.3.4.6.3.).

Initially the line pitch and edge pitch were 1.5 times the point cloud pitch. However, there was one unforeseen obstacle. The edge scanning started where the grid direction is perpendicular to the pipes' centrelines. In this region it can be assumed that the cloud pitch is sufficiently uniform in all directions. However, towards the end of the edge the centrelines of the pipes are in the same direction as the grid. This resulted in large gaps between points in the cloud, as can be seen in the figure below. For clarity, not all the scan lines are shown in this figure. Note the large gaps where the scan lines move up the sides of the pipes.

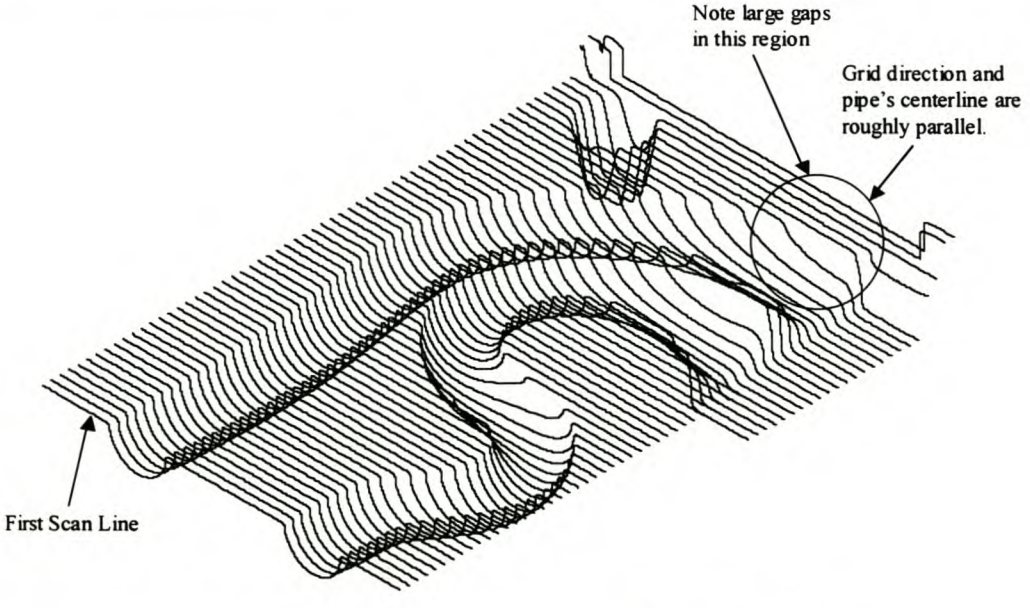


Figure 3 Scan Lines. (Only every 10th line is shown.)

The virtual CMM assumes that these large gaps are real gaps in the surface. Thus, it is not possible to scan in that region with an assumed cloud pitch of 0.5mm. The way to overcome this problem is to increase the pitch that the virtual CMM uses; in this case the pitch was increased to 2mm. The line pitch used by the edge scanning algorithm must also be increased so that it can step over the gaps.

In other words, there are two pitches of concern here. The first is the cloud pitch that the virtual CMM uses to select points in the cloud. If there are gaps in the cloud larger that this pitch, the virtual CMM assumes that it represent true gaps in the surface. The cloud pitch must therefore be larger than the largest gap in the cloud that the virtual CMM will encounter. The negative effect of increasing the cloud pitch is that the voxels of the octree contain more points since they are larger. Thus the time it takes to find a point in the cloud increases.

The second pitch of concern is the line pitch. (Actually the edge pitch as well, but it is assumed that the two are normally equal.) The line pitch is used by the edge scanning algorithm and determines how far consecutive points on a scan line are apart. This pitch cannot be less than the cloud pitch used by the virtual CMM. More guidelines on selecting this pitch are given in Chapter 5.

In this case study the edges were scanned in two or three stages with different scanning parameters for each stage simply to overcome the problem of the variable cloud pitch. The scanning parameters are given in Appendix F. The scanned edges are shown below. Note the erratic nature of the edges near their right ends. This is the area where larger amplitudes and pitches were used. Clearly this resulted in less accurate edges. This then is why the larger scanning parameters were not used to scan the entire edge.

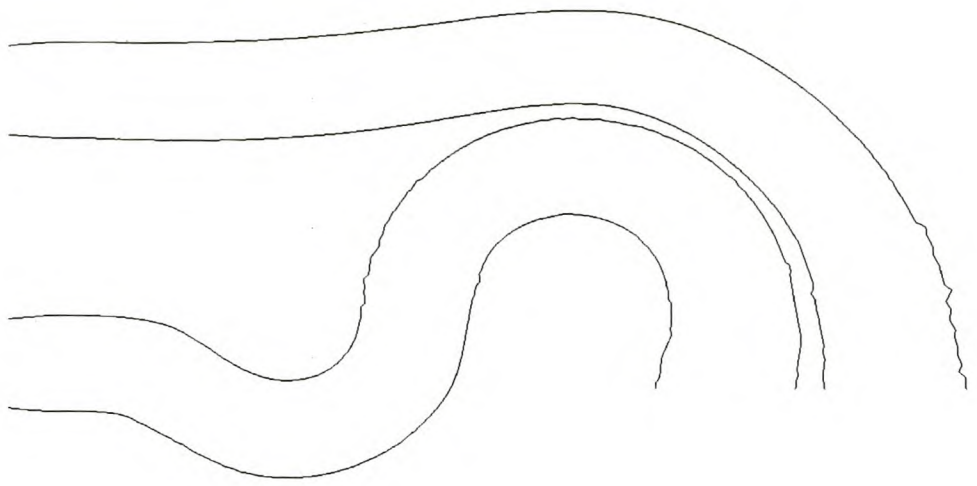


Figure 4 Scanned Edges.

The scanning tolerance is also an important parameter to select. A guideline is given in equation (5.3.3.5.1.1.). The Cyclone in the laboratory's accuracy is 0.05mm. This is taken as the noise level of the point cloud. The noise component attributed to the projection of the scanned points onto the scanning plane must be added to this according to the mentioned equation. The scanning tolerance must be larger than this value. The scanning tolerance used here is given with the other scanning parameters in Appendix F.

The experiments reported in Chapter 5 seem to indicate that the zigzag pattern is a little more robust than the square pattern. Thus the zigzag was used in this case study.

There is a region in the core box, indicated in Figure 1, where the gap between the pipes is only 3mm. This means that at best 4 points can be used to calculate the edge points there. It is probably less due to points that are ignored because of noise and because the pattern does not necessarily start exactly at the edge. Due to the small number of points it was decided to use a linear approximation to calculate the edge points.

7.3. Surface Reconstruction

7.3.1. Base Surfaces

Four profile curves were extracted by selecting points at the start and end of the pipes. These were used with the scanned edges to construct base surfaces. Splines were fitted to the edges and used to construct a swept surface according to Ueng et al.'s (1998) model. These are shown below.

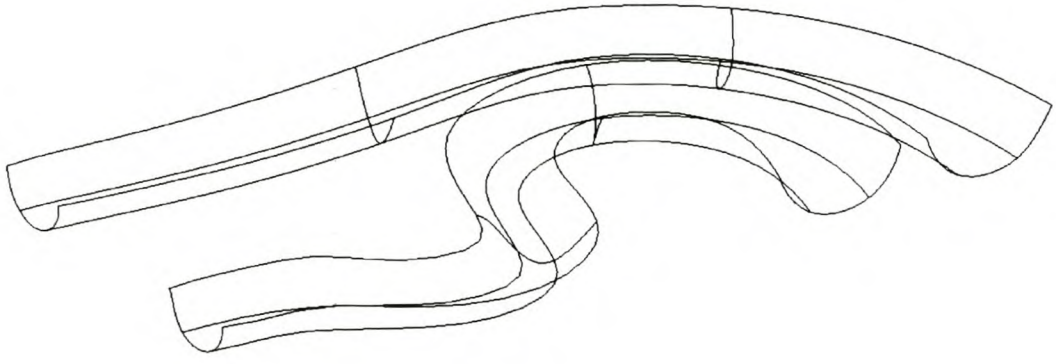


Figure 5 Base Surfaces.

7.3.2. Segmenting the Point Cloud

Once the edges were extracted the very dense point cloud was no longer necessary. This cloud was filtered using the Cyclone's modelling software. The resulting point cloud is a 5mm by 5mm grid. These points are also compensated with the probe radius.

The base surfaces were used in a way suggested by Bradley and Chan (2001) to segment the point cloud. The distance from each point in the cloud to the surface is calculated. If this is less than a prescribed value it is assumed that the point belongs to that specific surface. The extracted cloud patches are shown with the base surfaces in

the figure below. All points that are closer than 3mm to the base surfaces were selected.

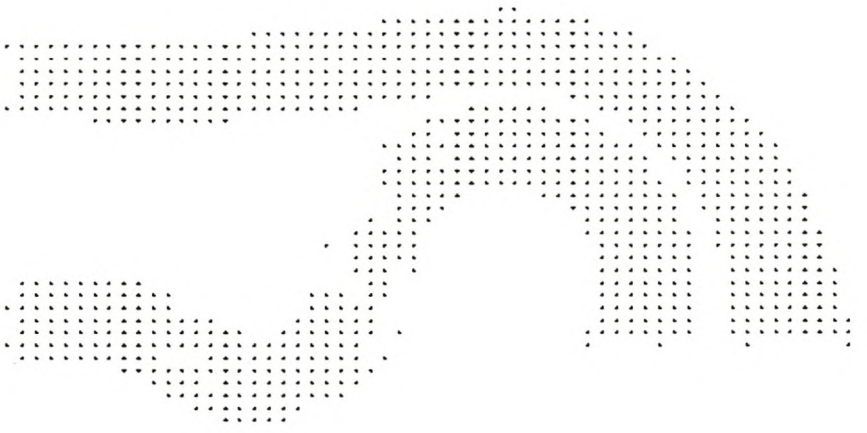


Figure 6 Cloud Segmentation.

Notice that in the middle of the bottom part a number of points are missing. This is an indication that the swept surface of Ueng et al. (1998) does not represent the points very well. The reason is that their model does not include rotation of the profile curve, but rather a blending function of the profile curves at both ends of the surface.

7.3.3. Fitting Swept Surfaces

Lastly a least squares approximation of the surfaces were made. The base surfaces were used to parameterise the cloud patches. The approximation results are given in Table 1.

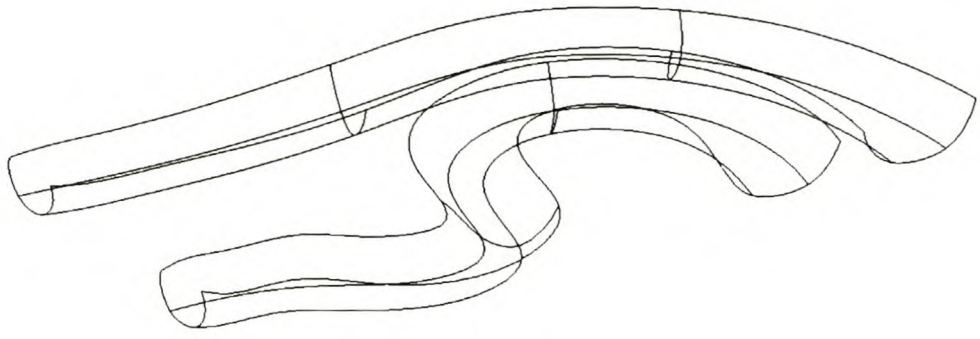


Figure 7 Approximated Surfaces.

The average errors are rather large. Increasing the number of control points does not seem to have a significant effect except for Pipe 2. It is possible that Ueng et al.'s (1998) model is not a good representation of the surfaces. The profiles curves of the pipes are definitely rotated as they are swept along the trajectories. Ueng et al.'s (1998) swept surface model does not include rotation of the profile curve; it rather blends the curve at the beginning and end of the swept surface. The fact that the approximation results do not improve much when the number of control points is increased supports this observation.

Table 1 Approximation Results.

		Pipe 1	Pipe 2
12×6 Grid	ε_{avg}	0.224mm	0.426mm
	ε_{max}	1.183mm	2.114mm
14×7 Grid	ε_{avg}	0.224mm	0.384mm
	ε_{max}	1.058mm	1.632mm

7.4. Conclusions

Clearly the edge scanning algorithm has problems dealing with point clouds of non-uniform density. Most of the scanning methods mentioned in the literature review give clouds with a structure similar to that of the Cyclone. This means that the grid direction must be carefully selected. In many cases, of which this core box is one

example, multiple scans might be necessary to ensure that the maximum gap size in the point cloud is at an acceptable level. Such small stumbling blocks can cause considerable frustration for the designer even if he/she is aware of them. When working with large point clouds, such as the one in this example, it takes a minute or two to scan the edge. The success or failure of the method is therefore not immediately apparent, thus the frustration.

Point clouds can quickly become very unmanageable if the scanning is done at the best, recommended resolution. In some cases, where the curvature is too high, it is simply not practical to scan a dense enough cloud. In fact, in some cases this algorithm requires a point cloud much denser than one required for following the traditional route of fitting surfaces, extending them and calculating the intersections.

Small surface features such as the 3mm gap between the edges indicated in Figure 1 can further limit the practical use of the edge scanning algorithm.

It further seems that Ueng et al.'s (1998) surface model is not appropriate for this example. This is a problem for any feature based Reverse Engineering system. If the original feature used to construct the object is not included in the set of available features in the Reverse Engineering system a bad surface approximation is always possible. The alternative is to use a more general model such as a tensor product NURBS surface. This is of course done at the loss of feature information that might be useful for geometry manipulation in a parametric modeller.

On the positive side it is encouraging the note that good base surfaces can be constructed by edge scanning and swept surface construction. The guidelines developed in Chapter 5 also gave good direction in selecting the scanning parameters – at least once the stumbling block of the non-uniform density was noted!

Chapter 8.

Conclusion

8.1. Have the Goals been Achieved?

The pilot study reported in Chapter 1 highlighted the need for robust edge detection and segmentation methods for Reverse Engineering. In that study a significant amount of time was spent at finding the intersection of the surface patches. The literature review showed that there are many good techniques for segmenting point clouds if they contain only primitive entities. There are also many techniques to segment regular grids of points. However, few methods are available for segmenting point clouds of arbitrary structure. It is in fact an unresolved problem. A further problem touched on by the literature is that of design intent. Many Reverse Engineering systems blindly approximate the point cloud with an arbitrary surface definition. This makes it very difficult to edit the model if that is necessary.

The edge scanning algorithm is an attempt at bridging this gap. It is designed to work on unstructured point clouds. The resulting curves can be used as generator curves in a feature based CAD system to reconstruct the surface. This was demonstrated in a case study reported in Chapter 7. Chapter 5 showed that the average accuracy of the edge points can be of the same order of magnitude as the noise in the point cloud provided that good scanning parameters are specified. Guidelines for doing this are given in that chapter.

Another requirement was that the algorithm must be able to detect the original edge if it was replaced by a fillet radius or simply if it is worn or damaged. This was achieved and makes this edge detection algorithm unique.

This method, however, has a number of shortcomings. An important condition is that the accuracy of the detected edge is limited by the largest gap between points in the cloud. Arguably this will be a limiting factor in any edge detection algorithm. The

problems that occur if this is not the case are illustrated in the case study in Chapter 7. The investigation reported in Chapter 5 showed that in many cases a very dense point cloud would be needed. These two conditions mean that careful attention must be paid when scanning the point cloud. The point cloud can be very huge, making it very cumbersome to use. It may also be very expensive and time consuming to scan a point cloud that is suitable for use by the edge scanning algorithm.

In the beginning of this thesis it was mentioned that one consideration during the development of the edge scanning algorithm was to implement it on a real CMM. It was not done simply because the access to the communication protocol between the PC and CMM was not available. Although it was not tested on a real CMM, it is the opinion of this author that there is no reason why this algorithm can not be implemented on a real CMM. The scanning strategy with the virtual CMM carefully follows the principals of a real CMM, complete with collision detection. The algorithm is divided in separate objects in C++, therefore the virtual CMM object can be replaced with an object that controls a real CMM. It is expected that no changes will be necessary to the scanning algorithm once this replacement is done.

If the edges are scanned directly on a real CMM, the need for very dense point clouds is eliminated. In many cases it will only be necessary to scan the edges in order to reconstruct a surface. This can lead to considerable time savings both during scanning and modelling.

An attempt was made to combine the edge scanning algorithm with feature based surface reconstruction by trying to fit swept surfaces to the data. The result from the edge scanning algorithm produced very good base surfaces, but the swept surface definition used here proved to be inadequate for the example used in Chapter 7. This highlighted the need for a larger library of features.

8.2. Future Work

Obviously the library of features must be extended in a practical Reverse Engineering system.

The most immediate need is that the algorithm must be extended that it can scan the entire boundary of a surface patch. It will be useful if the algorithm can scan around

the corners of a surface patch and thereby scan all the edges bounding a specific surface. Special attention should be paid so that the designer only has to specify one set of input parameters. It is also important that a good definition of the edges must be obtained near the corners. The current algorithm cannot scan effectively in that region.

The edge scanning algorithm can be extended to scan edges formed by surfaces joined with C^2 continuity.

Chapter 9.

References

Abdel-Malek, K. and Yeh, H.-J., 1997, "On the Determination of Starting Points for Parametric Surface Intersections", Computer-Aided Design, Vol. 29, No. 1, pp. 21-35.

Anonymous, 2000, "Exacting Standards", Mechanical Engineering, Vol. 112, No. 12, pp. 71-72.

Anonymous, 2001a, "3-D Window on World of the Unborn", Cape Times, 7 August, pp. 6.

Anonymous, 2001b, "How it All Began?", www.renishaw.com, (27 September 2001).

Armstrong, P.J. and Antonis, J., 2000, "The Development of an Active Computer Vision System for Reverse Engineering", Journal of Engineering Manufacture, Vol. 214, No. B7, pp. 615-618.

Au, C.K. and Yuen, M.M.F., 1999, "Feature-based Reverse Engineering of Mannequin for Garment Design", Computer-Aided Design, Vol. 31, No. 12, pp. 751-759.

Bao, H.P., Soundar, P. and Yang, T., 1994, "Integrated Approach to Design and Manufacture of Shoe Lasts for Orthopaedic Use", Computers in Industrial Engineering, Vol. 26, No. 2, pp. 411-421.

Bernardini, F., Mittleman, J., Rushmeier, H., Silva, C. and Taubin, G., 1999, "The Ball-pivoting Algorithm for Surface Reconstruction", IEEE Transactions on Visualisation and Computer Graphics, Vol. 5, No. 4, pp. 349-359.

Besl, P.J. and Jain, R.C., 1988, "Segmentation through Variable-order Surface Fitting", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 10, No. 2, pp. 167-192.

Bidanda, B. and Hosni, Y.A., 1994, "Reverse Engineering and its Relevance to Industrial Engineering: A Critical Review", Computers in Industrial Engineering, Vol. 26, No. 2, pp. 343-348.

Böhm, W., 1981, "Generating the Bézier Points of B-spline Curves and Surfaces", Computer-Aided Design, Vol. 13, No. 6, pp. 365-366.

Bonitz, P. and Krzystek, P., 1997, "Reverse Engineering in Combination with Digital Photogrammetry (ICEM SURF/ICEM PHOTO)", In: Pratt, M.J., Sriram, R.D. and Wozny, M.J., **Product Modelling for Computer Integrated Design and Manufacture**, Chapman & Hall, London.

Bradley, C. and Chan, V., 2001, "A Complementary Sensor Approach to Reverse Engineering", Journal of Manufacturing Science and Engineering, Vol. 123, No. 1, pp. 74-82.

Bronsvoort, W.F. and Waarts, J.J., 1992, "A Method for Converting the Surface of a Generalized Cylinder into a B-spline Surface", Computers & Graphics, Vol. 16, No. 2, pp. 175-178.

Chan, V.H., Bradley, C. and Vickers, G.W., 2001, "A Multi-sensor Approach to Automating Co-ordinate Measuring Machine-based Reverse Engineering", Computers in Industry, Vol. 44, No. 2, pp. 105-115.

Chant, A., Wilcock, D. and Costello, D., 1998, "The Determination of IC Engine Inlet Port Geometries by Reverse Engineering", The International Journal of Advanced Manufacturing Technology, Vol. 14, No. 6, pp. 65-69.

Chapdelaine, N., 1998, "From Art to Port", Manufacturing Engineering, Vol. 120, No. 6, pp. 66-72.

Chen, J.J. and Ozsoy, T.M., 1988, "Predictor-corrector Type of Intersection Algorithm for C^2 Parametric Surfaces", Computer-Aided Design, Vol. 20, No. 6, pp. 347-352.

Chen, L.-C. and Lin, G.C.I., 1997, "An Integrated Reverse Engineering Approach to Reconstructing Free-form Surfaces", Computer Integrated Manufacturing Systems, Vol. 10, No. 1, pp. 49-60.

Chen, Y.H. and Liu, C.Y., 1997, "Robust Segmentation of CMM Data Based on NURBS", International Journal of Advanced Manufacturing Technology, Vol. 13, No. 8, pp. 530-534.

Chiang, Y.M. and Chen, F.L., 1999, "Sculptured Surface Reconstruction from CMM Measurement Data by a Software Iterative Approach", International Journal of Production Research, Vol. 37, No. 8, pp. 1679-1695.

Choi, B.K., Shin, H.Y., Yoon, Y.I. and Lee, J.W., 1998, "Triangulation of Scattered Data in 3D Space", Computer-Aided Design, Vol. 20, No. 5, pp. 239-248.

Cooke, D., Craven, A.H. and Clarke, G.M., 1985, **Statistical Computing in Pascal**, Edward Arnold, London.

Danckaerts, T. and Yudhira, L., 1997, **Het Ontwikkelen van Meetstrategieën voor CAD-Modellering van Vrije-vorm Oppervlakken met behulp van een Laserscanner**, Eindwerk, Katholieke Universiteit Leuven, Belgium.

De Berg, M., Van Kreveld, M., Overmars, M. and Schwarzkopf, O., 1997, **Computational Geometry Algorithms and Applications**, Springer, Berlin.

De Boor, C., 1978, **A Practical Guide to Splines**, Springer, New York.

Del Taglia, A., Paolucci, A. and Santochi, M., 1995, "The Shadow-Moiré Method Applied to 3D Model Copying", Annals of the CIRP, Vol. 44, No. 1, pp. 497-500.

Dierckx, P., 1993, **Curve and Surface Fitting with Splines**, Oxford University Press, New York.

Do Carmo, M. P., 1976, **Differential Geometry of Curves and Surfaces**, Prentice-Hall, Upper Saddle River.

Ebenstein, S.E., Kiridena, V.S., Rodin, Y.M. and Smith, G.H., 1999, "A Method for Qualifying High Density Data Acquisition systems for Free Form Metrology",

Proceedings of the ASME Design Engineering Technical Conferences, 12-15 September, Paper No. CIE-9102, Las Vegas, USA.

Edelsbrunner, H. and Mücke, E.P., 1994, "Three-dimensional Alpha Shapes", ACM Transactions on Graphics, Vol. 13, No. 1, pp. 43-72.

ElKott, D.F., ElMaraghy, H.A. and Nassef, A.O., 1999, "Sampling for Free Form Surfaces Inspection Planning", Proceedings of the ASME Design Engineering Technical Conferences, 12-15 September, Paper No. CIE-9136, Las Vegas, USA.

Elsässer, B. and Hoschek, J., 1996, "Approximation of Digitized Points by Surfaces of Revolution", Computer & Graphics, Vol. 20, No. 1, pp. 85-94.

Fan, T.-J., Medioni, G. and Nevatia, R., 1987, "Segmented Descriptions of 3-D Surfaces", IEEE Journal of Robotics and Automation, Vol. RA-3, No. 6, pp. 527-538.

Fitzgibbon, A.W., Eggert, D.W. and Fisher, R.B., 1997, "High-level CAD Model Acquisition from Range Images", Computer-Aided Design, Vol. 29, No. 4, pp. 321-330.

Forbes, A.B., 1991, "Least-squares Best-fit Geometric Elements", NPL Report DITC 140/89, Teddington, United Kingdom.

Goussard, C.L., 2001, **Semi-automatic Extraction of Primitive Geometric Entities from Point Clouds**, Masters Thesis, University of Stellenbosch, South Africa.

Goussard, C.L. and Basson, A.H., 2001, "Semi-automatic Extraction of Primitive Geometric Entities from Point Clouds", Proceedings of the International CIRP Design Seminar, 6-8 June, KTH Stockholm, Sweden, pp. 359-364.

Hegazi, H.A. and Metwalli. S.M., 1999, "Reverse Engineering of Standard Mechanical Elements", Proceedings of the ASME Design Engineering Technical Conference, 12-15 September, Paper No. CIE-9137, Las Vegas, USA.

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J. and Stuetzle, W., 1992, "Surface Reconstruction from Unorganised Points", Computer Graphics, SIGGRAPH Proceedings, pp. 71-78.

Horváth, I. and Vergeest, J.S.M., 1998, "Natural Representation of Shapes with Singularities", International Journal of Shape Modelling, Vol. 3, No. 4&4, pp. 127-140.

Hoschek, J., 1988, "Intrinsic Parametrization for Approximation", Computer Aided Geometric Design, Vol. 5, pp. 27-31.

Hosni, Y. and Ferreira, L., 1994, "Laser Based System for Reverse Engineering", Computers in Industrial Engineering, Vol. 26, No. 2, pp. 387-394.

Huang, C.-N. and Motavalli, S., 1994, "Reverse Engineering of Planar Parts Using Machine Vision", Computers in Industrial Engineering, Vol. 26, No. 2, pp. 369-379.

Ip, W.L.-R. and Loftus, M., 1996, "Shaded Image Machining: A Reverse Engineering Approach Using Computer Vision Techniques and Analytical Method to Reproduce Sculptured Surfaces", International Journal of Production Research, Vol. 34, No. 7, pp. 1895-1916.

Ip, W.L.-R. and Hou, K.C.-C., 1999, "Range Image Measurement and Shaded Image Measurement: Two Non-contact Coordinate Measurement Approaches Designed for Special Application Tasks", Journal of Engineering Manufacture, Vol. 213, No. B6, pp. 501-531.

Janssens, M., 1998, **A Triangular Approach to Digitising Free-Form Objects for Reverse Engineering**, Ph.D. Thesis, Katholieke Universiteit Leuven, Belgium.

Jones, L., 1999, "Octree Source Code", http://www.marlboro.edu/~mahoney /support/alg/node41.html, (28 April 1999).

Jun, Y., Chang, M., Rho, H.-M. and Park, S., 2001, "A Surface Alignment Algorithm in a Reverse Engineering System for Reproducing Human Head in a 3D Bust", Proceedings of the International CIRP Design Seminar, 6-8 June, KTH Stockholm, Sweden, pp. 347-352.

Jüttler, B. and Wagner, M.G., 1999, "Rational Motion-based Surface Generation", Computer-Aided Design, Vol. 31, No. 3, pp. 203-213.

Kim, K.I. and Kim, K., 1996, "A New Measuring Strategy for Sculptured Surfaces Using Offset Surfaces", Journal of Manufacturing Science and Engineering, Vol. 118, No. 1, pp. 646-651.

Kreyszig, E., 1988, **Advanced Engineering Mathematics**, Sixth Edition, John Wiley & Sons, New York.

Kruth, J.-P., Janssens, M. and Kerstens, A., 1997, "Automatic Reverse Engineering of Free-form Objects", In: Ikawa, N., Kishinami, T. and Kimura, F., Rapid **Prototyping Development**, Chapman and Hall, London.

Kruth, J.-P. and Kerstens, A., 1998, "Reverse Engineering Modelling of Free-form Surfaces from Point Clouds Subject to Boundary Conditions", Journal of Materials Processing Technology, Vol. 76, No. 1-3, pp. 120-127.

Lai, J.-Y. and Lu, C.-Y., 1996, "Reverse Engineering of Composite Sculptured Surfaces", The International Journal of Advanced Manufacturing Technology, Vol. 12, No. 3, pp. 180-189.

Lawson, C.L. and Hanson, R.J., 1974, **Solving Least Squares Problems**, Prentice-Hall, Englewood Cliffs.

Liu, S. and Ma, W., 1999, "Seed-growing Segmentation of 3-D Surfaces from CT-contour Data", Computer-Aided Design, Vol. 31, No. 8, pp. 517-536.

Lin, Z.-C. and Chen, C.-C., 2001, "Collision-free Path Planning for Coordinate Measurement Machine Probe", International Journal of Production Research, Vol. 39, No. 9, pp. 1969-1992.

Ma, W., 1994, **NURBS-based Modelling from Measured Points of Physical Models**, Ph.D. Thesis, Katholieke Universiteit Leuven, Belgium.

Ma, W. and He, P., 1998, "B-spline Surface Local Updating with Unorganized Points", Computer-Aided Design, Vol. 30, No. 11, pp. 853-862.

Ma, W. and Kruth, J.-P., 1995a, "Parameterisation of Randomly Measured Points for Least Squares Fitting of B-splines Curves and Surfaces", Computer-Aided Design, Vol. 27, No. 9, pp. 663-675.

Ma, W. and Kruth, J.-P., 1995b, "NURBS Curve and Surface Fitting and Interpolation", In: Daehlen, M., Lyche, T. and Schumaker, L.L., Mathematical **Methods for Curves and Surfaces**, Vanderbilt University Press, pp. 315-322.

Meagher, D., 1982, "Geometric Modeling Using Octree Encoding", Computer Graphics and Image Processing, Vol. 19, pp.129-147.

Menon, J., Desai, R. and Buckley, J., 1997, "Constraint-based Reverse Engineering From Ultrasound Cross-sections", Proceedings of the ASME Design Engineering Technical Conference, 14-17 September, Paper No. DFM-4365, Sacramento, USA.

Metwalli, S.M., Radwan, M.A.E., Abdel-Wehab, O., Shalaby, O., Moussa, Y.A. and Hosni, Y.A., 1999, "Maintenance and Parts Fabrication by Reverse Engineering", Proceedings of the ASME Design Engineering Technical Conference, 12-15 September, Paper No. CIE-9134, Las Vegas, USA.

Milroy, M.J., Bradley, C. and Vickers, G.W., 1997, "Segmentation of a Wrap-around Model using an Active Contour", Computer-Aided Design, Vol. 29, No. 4, pp. 299-320.

Milroy, M.J., Weir, D.J., Bradley, C. and Vickers, G.W., 1996, "Reverse Engineering Employing a 3D Laser Scanner: A Case Study", The International Journal of Advanced Manufacturing Technology, Vol. 12, No. 2, pp. 111-121.

Miyake, Y., Kondo, T., Kaneko, S., Igarashi, S. and Narahara, H., 1997, "Reconstruction of Three Dimensional Surface from Slice Positional Data", In: Ikawa, N., Kishinami, T. and Kimura, F., **Rapid Product Development**, Chapman & Hall, London.

Motavalli, S. and Bidanda, B., 1994, "Modular Software Development for Digitizing Systems Data Analysis in Reverse Engineering Applications: Case of Concentric Rotational Parts", Computers in Industrial Engineering, Vol. 26, No. 2, pp. 395-410.

Motavalli, S., Suharitdamrong, V. and Alrashdan, A., 1998, "Design Model Generation for Reverse Engineering Using Multi-sensors", IIE Transactions, Vol. 30, No. 4, pp. 357-366.

O'Connor, P.D.T., 1991, **Practical Reliability Engineering**, Third Edition, John Wiley & Sons, Chichester.

Park, S. and Jun, Y., 2000, "A Face-based Reverse Engineering Approach to Primitive Analytic Surfaces", Proceedings of the International CIRP Design Seminar, 16-18 May, Haifa, Israel, pp. 423-428.

Park, S. and Jun, Y., 2001, "Automated Segmentation of Scanned Point Data in Feature-based Reverse Engineering System", Proceedings of the International CIRP Design Seminar, 6-8 June, KTH Stockholm, Sweden, pp. 353-358.

Peng, Q.-J. and Loftus, M,. 1998a, "A New Approach to Reverse Engineering Based on Vision Information", International Journal of Machine Tools and Manufacture, Vol. 38, No. 8, pp. 881-899.

Peng, Q.-J. and Loftus, M., 1998b, "Rapid Prototyping Based on Image Information in Reverse Design Applications", In: Sivaloganathan S. and Shahin, T.M.M., **Design Reuse**, Engineering Design Conference, Professional Engineering Publishing, pp. 175-182.

Peng, Q.-J. and Loftus, M., 2001, "Using Image Processing Based on Neural Networks in Reverse Engineering", International Journal of Machine Tools and Manufacture, Vol. 41, No. 5, pp. 625-640.

Peng, Q.S., 1984, "An Algorithm for Finding the Intersection Lines between Two B-spline Surfaces", Computer-Aided Design, Vol. 16, No. 4, pp. 191-196.

Perreault, D. and Ward, M., 1999, "The Portable CMM in Reverse Engineering", Proceedings of the ASME Design Engineering Technical Conferences, 12-15 September, Paper No. CIE-9135, Las Vegas, USA.

Piegl, L. and Tiller, W., 1997, **The NURBS Book**, Second Edition, Springer, Berlin.

Piegl, L. and Tiller, W., 2001, "Parametrization for Surface Fitting in Reverse Engineering", Computer-Aided Design, Vol. 33, No. 8, pp. 593-603.

Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.R., 1997, **Numerical Recipies in C**, Second Edition, Cambridge University Press, Cambridge.

Puttré, M., 1994, "Capturing Design Data with Digitizing Systems", Mechanical Engineering, Vol. 116, No. 4, pp. 62-65.

Raab, S., 1994, "Coordinate Measurements Accelerate Reverse Engineering", Machine Design, Vol. 66, No. 22, pp. 50-53.

Rogers, D.F. and Fog, N.G., 1989, "Constrained B-spline Curve and Surface Fitting", Computer-Aided Design, Vol. 21, No. 10, pp. 641-648.

Rolls, C.J., ElMaraghy, W. and Elmaraghy, H., 1999, "Towards Combining Digitization Techniques in the Generation of Reverse Engineering Data Sets", Proceedings of the ASME Design Engineering Technical Conferences, 12-15 September, Paper No. CIE-9130, Las Vegas, USA.

Rüther, H. and Craigie, D.H., 1998, "The Development of a Contact-free 3D Co-ordinate Measurement Machine", R&D Journal, Vol. 14, No. 1, pp. 8-14.

Sarkar, B. and Menq, C.-H., 1991a, "Smooth-surface Approximation and Reverse Engineering", Computer-Aided Design, Vol. 23, No. 9, pp. 623-628.

Sarkar, B. and Menq, C.-H., 1991b, "Parameter Optimization in Approximating Curves and Surfaces to Measurement Data", Computer Aided Geometric Design, Vol. 8, pp. 267-290.

Schneider, F., 2001, "Footwear Insole Mould Design with 3D Point Sampling", Proceedings of the International CIRP Design Seminar, 6-8 June 2001, KTH Stockholm, Sweden, pp. 371-373.

Schreve, K. and Basson, A.H., 2000, "Edge Detection in Reverse Engineering Using a Virtual CMM", Proceedings of ASME Design Engineering Technical Conferences, 10-13 September, Paper No. DAC-14540, Baltimore, USA.

Seiler, A., Balendran, V., Sivayoganathan, K. and Sackfield, A., 1996, "Reverse Engineering from Uni-directional CMM Scan Data", The International Journal of Advanced Manufacturing Technology, Vol. 11, No. 4, pp. 276-284.

Shen, Y. and Springer, M.E., 1998, "A Robust Pretravel Model for Touch Trigger Probes in Coordinate Metrology", Journal of Manufacturing Science and Engineering, Vol. 120, pp. 532-539.

Shetty, S. and White, P.R., 1991, "Curvature-continuous Extensions for Rational B-spline Curves and Surfaces", Computer-Aided Design, Vol. 23, No. 7, pp. 484-491.

Sinha, S.S. and Seneviratne, P., 1996, "Part To Art", Proceedings of the ASME Design Engineering Technical Conferences, 18-22 August, Paper No. DFM-1293, Irvine, USA.

Smith, K.B. and Zheng, Y.F., 1998, "Accuracy Analysis of Point Laser Triangulation Probes Using Simulation", Journal of Manufacturing Science and Engineering, Vol. 120, No. 11, pp. 736-745.

Song, C.-K. and Kim, S.-W., 1997, "Reverse Engineering: Autonomous Digitization of Free-formed Surfaces on a CNC Coordinate Measuring Machine", International Journal of Machine Tools and Manufacture, Vol. 37, No. 7, pp. 1041-1051.

Spiegel, M.R., 1968, **Mathematical Handbook of Formulas and Tables**, McGraw-Hill, New York.

Suzuki, Y. and Aoyama, H., 1997, "Autonomous Measurement of Position Vectors and Unit Normal Vectors of Discrete Points on Unknown Object", In: Ikawa, N., Kishinami, T. and Kimura, F., **Rapid Product Development**, Chapman & Hall, London.

Taubin, G., 1991, "Estimation of Planar Curves, Surfaces, and Nonplanar Space Curves Defined by Implicit Equations with Applications to Edge and Range Image Segmentation", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 13, No. 11, pp. 1115-1138.

Thompson, W.B., Owen, J.C., De St. Germain, H.J., Stark, S.R., Henderson, T.C., 1999, "Feature-based Reverse Engineering of Mechanical Parts", IEEE Transactions on Robotics and Automation, Vol. 15, No. 1, pp. 57-66.

Tiller, W., 1983, "Rational B-splines for Curve and Surface Representation", IEEE Computer Graphics and Applications, Vol. 3, No. 6, pp. 61-69.

Trucco, E. and Fisher, R.B., 1995, "Experiments in Curvature-based Segmentation of Range Data", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 17, No. 2, pp. 177-182.

Üçoluk, G. and Toroslu, I.H., 1999, "Automatic Reconstruction of Broken 3-D Surface Objects", Computers & Graphics, Vol. 23, No. 4, pp. 573-582.

Ueng, W.-D. and Lai, J.-Y., 1998, "A Sweep-surface Fitting Algorithm for Reverse Engineering", Computers in Industry, Vol. 35, No. 3, pp. 261-273.

Ueng, W.-D., Lai, J.-Y. and Doong, J.-L., 1998, "Sweep-surface Reconstruction from Three-dimensional Measured Data", Computer-Aided Design, Vol. 20, No. 10, pp. 791-805.

Várady, T., Martin, R.R. and Cox, J., 1997, "Reverse Engineering of Geometric Models – An Introduction", Computer-Aided Design, Vol. 29, No. 4, pp. 255-268.

Vergeest, J.S.M., Spanjaard, S., Horváth, I. and Jelier, J.J.O., 2000, "Fitting Freeform Shape Patterns to Scanned 3D Objects", Proceedings of the ASME Design Engineering Technical Conferences, 10-13 September, Paper No. CIE-14656, Baltimore, USA.

Vörös, J., 2000, "A Strategy for Repetitive Neighbor Finding in Octree Representations", Image and Vision Computing, Vol. 18, pp. 1085-1091.

Wang, Y.Z. and Chen, Y.H., 1999, "Optimized STL File Generation from Scattered Data", Proceedings of ASME Design Engineering Technical Conferences, 12-15 September, Paper No. DAC-8626, Las Vegas, USA.

Weckenmann, A. and Knauer, M., 1998, "The Influence of Measurement Strategy on the Uncertainty of CMM-Measurements", Annals of the CIRP, Vol. 47, No. 1, pp. 451-454.

Wykes, C. and Morshedizadeh, R., 1995, "Surface Topography Measurement Using Digital Moiré Contouring – Errors and Limitations", Proceedings of the Institution of Mechanical Engineers, Vol. 209, Part B, Journal of Manufacture, pp. 317 - 325.

Yang, M. and Lee, E., 1999, "Segmentation of Measured Point Data Using a Parametric Quadric Surface Approximation", Computer-Aided Design, Vol. 31, No. 7, pp. 449-457.

Yau, H.-T., 1997, "Reverse Engineering of Engine Intake Ports by Digitization and Surface Approximation", International Journal of Machine Tools and Manufacture, Vol. 37, No. 6, pp. 855-871.

Yau, H.-T. and Chen, J.-S., 1997, "Reverse Engineering of Complex Geometry Using Rational B-splines", The International Journal of Advanced Manufacturing Technology, Vol. 13, No. 8, pp. 548-555.

Yau, H.-T., Chen, C.-Y. and Wilhelm, R.G., 2000, "Registration and Integration of Multiple Laser Scanned Data for Reverse Engineering of Complex 3D Models", International Journal of Production Research, Vol. 38, No. 2, pp. 269-285.

Zhao, B.-Y., Okubo, S., Wang, Y.-L. and Tang, H.-J., 1997, "Model Construction by Using the Coordinate Measurement Machine as a Three Dimensional Digitizer", In: Ikawa, N., Kishinami, T. and Kimura, F., **Rapid Prototyping Development**, Chapman and Hall, London.

Software

AutoCAD Mechanical Desktop, 2000, Release 5, AutoDesk, Inc., USA.

Statistica, 2000, Release 5.5, StatSoft, Inc., USA.

Appendix A

Geometric Formulae

A1. Introduction

This appendix contains general geometric formulae used in this thesis. Mostly it consists of various calculations with lines and planes. Most of the formulae are derived with the familiar scalar product and cross product. Readers unfamiliar with these definitions can refer to Kreyszig (1988) pp. 322 and pp 332.

A2. Definitions

A2.1. Lines

In this thesis the parametric definition of a line in R^3 is used as defined in the figure below.

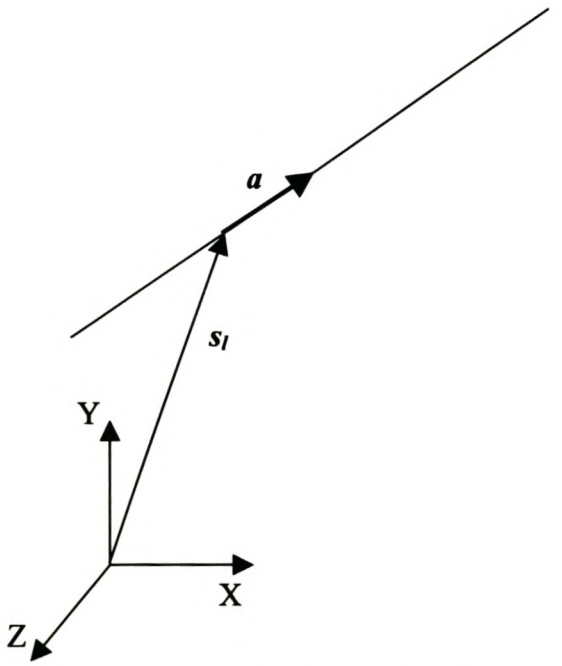


Figure A1 Definition of a Line in R^3.

The origin of the line is at s_l and a, a unit vector, indicates the direction of the line. Any point on the line is then given by the equation below.

$$x=s_l+\lambda a \qquad \text{(A2.1.1.)}$$

A2.2. Planes

The scalar product is used to define a plane. Let s_p be a point in a plane and n be a unit vector normal to the plane. Then the vector from s_p to any point in the plane must be perpendicular to n and so the scalar product of this vector with n must be zero. The plane is then defined by the equation below.

$$n\cdot(x-s_p)=0 \qquad \text{(A2.2.1.)}$$

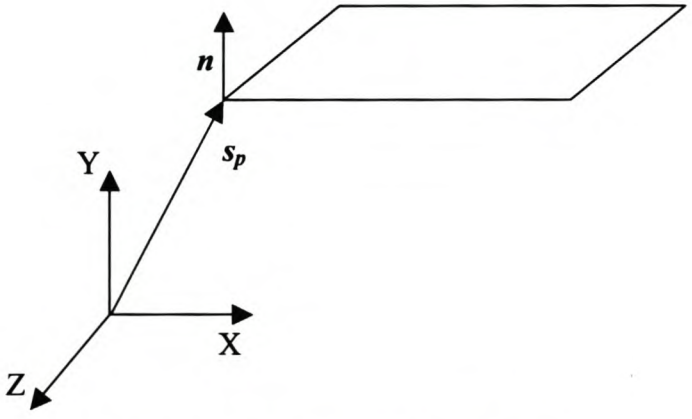


Figure A2 Definition of a Plane in R^3.

A3. Intersection of Two Lines

Two lines in R^3 do not necessarily intersect each other. The method described here can be used to find out if an intersection exists and, if it does, it will give the intersection point. There is also the possibility that the two lines lie on top of each other. A separate check for this case is necessary. First check if the two lines are parallel, i.e. the scalar product is either 1 or –1. If this is the case, check if the origin of the one line lies on the other line. (To do this, the distance from the one origin to the other line can be calculated. A method to calculate the distance from a point to a

line is given later in this appendix.) If the origin lies on the other line and the lines are parallel, then there are an infinite number of solutions.

If the lines are not co-linear on each other, then the intersection point, if it exists, can be found as follows.

The two lines are

$$\boldsymbol{x}_1 = \boldsymbol{s}_{l1} + \lambda_1 \boldsymbol{a}_1$$

and $\boldsymbol{x}_2 = \boldsymbol{s}_{l2} + \lambda_2 \boldsymbol{a}_2$

Thus $\boldsymbol{s}_{l1} + \lambda_1 \boldsymbol{a}_1 = \boldsymbol{s}_{l2} + \lambda_2 \boldsymbol{a}_2$ (A3.1.)

By breaking the above vector equation into it's components, the parameter, λ_2, at the intersection point can be found.

$$\lambda_1 = \frac{s_{l2x} + \lambda_2 a_{2x} - s_{l1x}}{a_{1x}} \text{ if } a_{1x} \neq 0 \qquad \text{(A3.2.)}$$

By substituting equation A3.2. into A3.1. and breaking the vector equation into it's components λ_2 can be found.

$$s_{l1y} + \frac{s_{l2x} + \lambda_2 a_{2x} - s_{l1x}}{a_{1x}} a_{1y} = s_{l2y} + \lambda_2 a_{2y}$$

$$\lambda_2 = \frac{a_{1x}\left(s_{l2y} - s_{l1y}\right) - a_{1y}\left(s_{l2x} - s_{l1x}\right)}{a_{2x}a_{1y} - a_{2y}a_{1x}} \text{ if } a_{2x}a_{1y} - a_{2y}a_{1x} \neq 0 \qquad \text{(A3.3.)}$$

The intersection point is the point on the second line corresponding to λ_2.

Since the denominator in equations A3.2. and A3.3. can be zero, other equations must be found in these instances. This is done by selecting other combinations of the vector components in equation A3.1. Without going through the derivation, these equations are given below.

$$\lambda_2 = \frac{a_{1x}\left(s_{l2z} - s_{l1z}\right) - a_{1z}\left(s_{l2x} - s_{l1x}\right)}{a_{2x}a_{1z} - a_{2z}a_{1x}} \text{ if } a_{2x}a_{1z} - a_{2z}a_{1x} \neq 0 \qquad \text{(A3.4.)}$$

$$\lambda_2 = \frac{a_{1y}(s_{l2z} - s_{l1z}) - a_{1z}(s_{l2y} - s_{l1y})}{a_{2y}a_{1z} - a_{2z}a_{1y}} \text{ if } a_{2y}a_{1z} - a_{2z}a_{1y} \neq 0 \quad \text{(A3.5.)}$$

$$\lambda_2 = \frac{a_{1y}(s_{l2x} - s_{l1x}) - a_{1x}(s_{l2y} - s_{l1y})}{a_{2y}a_{1x} - a_{2x}a_{1y}} \text{ if } a_{2y}a_{1x} - a_{2x}a_{1y} \neq 0 \quad \text{(A3.6.)}$$

$$\lambda_2 = \frac{a_{1z}(s_{l2x} - s_{l1x}) - a_{1x}(s_{l2z} - s_{l1z})}{a_{2z}a_{1x} - a_{2x}a_{1z}} \text{ if } a_{2z}a_{1x} - a_{2x}a_{1z} \neq 0 \quad \text{(A3.7.)}$$

$$\lambda_2 = \frac{a_{1z}(s_{l2y} - s_{l1y}) - a_{1y}(s_{l2z} - s_{l1z})}{a_{2z}a_{1y} - a_{2y}a_{1z}} \text{ if } a_{2z}a_{1y} - a_{2y}a_{1z} \neq 0 \quad \text{(A3.8.)}$$

A4. Distance from a Point to a Plane

From the definition of the scalar product, the cosine of the angle between the plane's unit normal, $\boldsymbol{n}$, and the vector from the plane's origin, $\boldsymbol{s_p}$, to an arbitrary point in R^3, $\boldsymbol{p}$, can be found.

$$\cos(\theta) = \frac{\boldsymbol{n} \cdot (\boldsymbol{p} - \boldsymbol{s}_p)}{\|\boldsymbol{n}\|\|\boldsymbol{p} - \boldsymbol{s}_p\|}$$

This cosine multiplied by the length of the vector from $\boldsymbol{s_p}$ to $\boldsymbol{p}$ is the shortest distance, d, from the point $\boldsymbol{p}$ to the plane.

$$d = \|\boldsymbol{p} - \boldsymbol{s}_p\|\cos(\theta)$$

$$d = \boldsymbol{n} \cdot (\boldsymbol{p} - \boldsymbol{s}_p) \quad \text{(A4.1.)}$$

A5. Projection of a Point onto a Plane

The projection of a point, $\boldsymbol{p}$, onto a plane is found by first finding the shortest vector from the point $\boldsymbol{p}$ to the plane. Of course, this vector will be parallel to the plane's unit normal, $\boldsymbol{n}$. The length of this vector is the distance from $\boldsymbol{p}$ to the plane, given by equation A4.1. So, the shortest vector is $\boldsymbol{n}(\boldsymbol{n} \cdot (\boldsymbol{p} - \boldsymbol{s}_p))$.

The projected point, $\boldsymbol{p}'$, is $\boldsymbol{p}$ minus the shortest vector.

$$\boldsymbol{p}' = \boldsymbol{p} - \boldsymbol{n}(\boldsymbol{n} \cdot (\boldsymbol{p} - \boldsymbol{s}_p)) \qquad \text{(A5.1.)}$$

This is illustrated by the figure below.

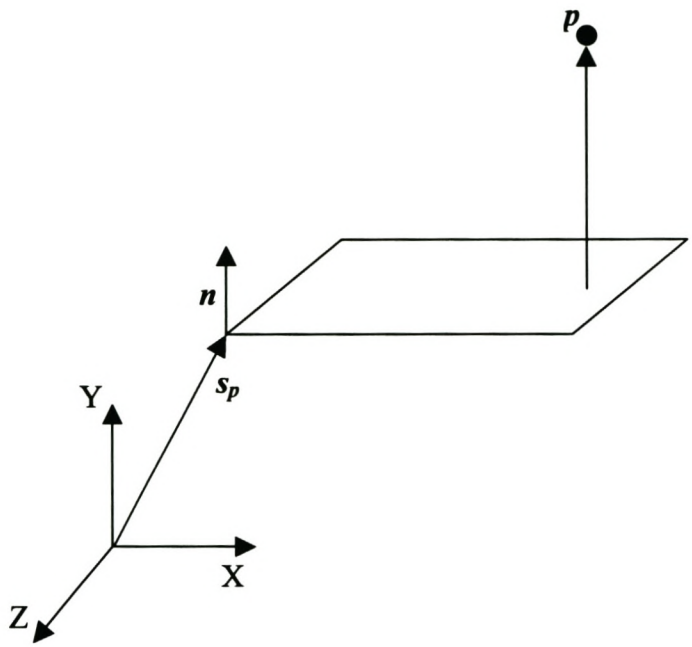


Figure A3 Projecting a Point onto a Plane.

A6. Distance from a Point to a Line

The definition of the vector product is used to find the sine of the angle θ in the figure below.

$$\sin(\theta) = \frac{\|\boldsymbol{a} \otimes (\boldsymbol{p} - \boldsymbol{s}_l)\|}{\|\boldsymbol{a}\| \|\boldsymbol{p} - \boldsymbol{s}_l\|}$$

The distance, d, from $\boldsymbol{p}$ to the line, is the sine of θ multiplied by the length of the vector from $\boldsymbol{s}_l$ to $\boldsymbol{p}$.

$$d = \|\boldsymbol{p} - \boldsymbol{s}_l\| \sin(\theta)$$

$$d = \|\boldsymbol{a} \otimes (\boldsymbol{p} - \boldsymbol{s}_l)\| \qquad \text{(A6.1.)}$$

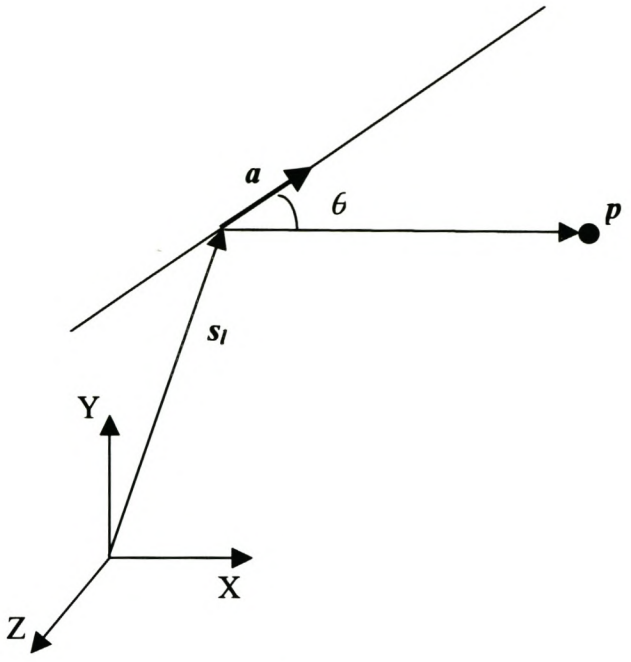


Figure A4 Distance from a Point to a Line.

A7. Projection of a Point onto a Line

Referring to Figure A4 the projection of $\boldsymbol{p}$ onto the line can be found. The value of λ corresponding to the projected point, $\boldsymbol{p}$', is the cosine of θ multiplied by length of the vector from $\boldsymbol{s}_l$ to $\boldsymbol{p}$.

$$\cos(\theta) = \frac{\boldsymbol{a} \cdot (\boldsymbol{p} - \boldsymbol{s}_l)}{\|\boldsymbol{a}\| \|\boldsymbol{p} - \boldsymbol{s}_l\|}$$

Thus $\lambda = \|\boldsymbol{p} - \boldsymbol{s}_l\| \cos(\theta)$

$$\lambda = \boldsymbol{a} \cdot (\boldsymbol{p} - \boldsymbol{s}_l) \qquad \text{(A7.1.)}$$

By substituting the value of λ in the equation of the line, the projected point, $\boldsymbol{p}$', can be found.

A8. Intersection of a Line and a Plane

To find the intersection of a line and a plane, substitute the equation of a line, equation A2.1.1. into the equation of a plane, equation A2.2.1. This gives the following equation.

$$\boldsymbol{n} \cdot \left(\boldsymbol{s}_l + \lambda \boldsymbol{a} - \boldsymbol{s}_p \right) = 0$$

Then, solve for λ.

$$\boldsymbol{n} \cdot \boldsymbol{s}_l + \lambda \boldsymbol{n} \cdot \boldsymbol{a} - \boldsymbol{n} \cdot \boldsymbol{s}_p = 0$$

$$\lambda = \frac{\boldsymbol{n} \cdot \left(\boldsymbol{s}_l - \boldsymbol{s}_p \right)}{\boldsymbol{n} \cdot \boldsymbol{a}} \text{ if } \boldsymbol{n} \cdot \boldsymbol{a} \neq 0 \qquad \text{(A8.1.)}$$

If $\boldsymbol{n} \cdot \boldsymbol{a} = 0$ it means than the line is parallel to the plane and there is either no solution or an infinite number of solutions if the line lies in the plane. The intersection point, if there is one, can be found by substituting the solution for λ in the equation for the line, equation A2.1.1.

A9. Projecting a Line onto a Plane

The origin point of the line, $\boldsymbol{s}_l$, is projected onto the plane using equation A5.1. The direction vector of the line is orientated by taking cross products of $\boldsymbol{a}$ and $\boldsymbol{n}$ as follows.

$$\boldsymbol{a}' = \frac{(\boldsymbol{n} \otimes \boldsymbol{a}) \otimes \boldsymbol{n}}{\left\| (\boldsymbol{n} \otimes \boldsymbol{a}) \otimes \boldsymbol{n} \right\|} \qquad \text{(A9.1.)}$$

The cross product of $\boldsymbol{n}$ with $\boldsymbol{a}$ defines a plane (or rather the normal vector there of) that, at the same time, contains the original line and is perpendicular to the plane. Thus, the cross product of this new normal vector with $\boldsymbol{n}$ must give the direction vector of the line in the original plane.

If the line is perpendicular to the plane, the cross product of $\boldsymbol{n}$ with $\boldsymbol{a}$ will be null. In this case, the projection of the line onto the plane results only in a point in the plane, which can be found as stated above.

A10. Intersection Line of Two Planes

The algorithm that finds the intersection curve of two B-spline surfaces uses the intersection line of two planes during the subdivision process.

Given two planes as, $\boldsymbol{n}_1 \cdot (\boldsymbol{x}_1 - \boldsymbol{s}_{p1}) = 0$ and $\boldsymbol{n}_2 \cdot (\boldsymbol{x}_2 - \boldsymbol{s}_{p2}) = 0$, with $\boldsymbol{n}_1 \cdot \boldsymbol{n}_2 \neq 1$ then the intersection line is found as follows.

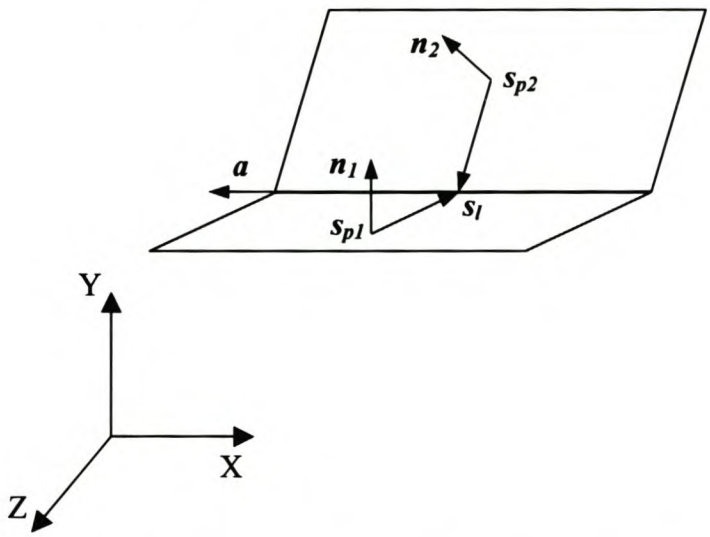


Figure A5 Definitions for finding the intersection line between two

The direction of the intersection line is simply

$$\boldsymbol{a} = \frac{\boldsymbol{n}_1 \otimes \boldsymbol{n}_2}{\|\boldsymbol{n}_1 \otimes \boldsymbol{n}_2\|} \qquad \text{(A10.1.)}$$

Now, a point on the intersection line is needed to complete the definition of the line. The perpendicular direction from $\boldsymbol{s}_{p1}$ to the intersection line is the cross product of the direction $\boldsymbol{a}$ and normal vector $\boldsymbol{n}_1$. Therefore the line from $\boldsymbol{s}_{p1}$ to the intersection line is $\boldsymbol{s}_{p1} + \lambda(\boldsymbol{a} \otimes \boldsymbol{n}_1)$. The point where this line intersects plane 2 is also a point on the intersection line. Thus, it is only necessary to determine λ where this line intersects plane 2. Using equation A8.1., the value of λ is easily determined. That gives the following equation for the intersection line of two planes. (Note that in this equation λ is now the independent parameter of the intersection line of the two planes.)

$$\boldsymbol{x} = \left(\boldsymbol{s}_{p1} + \frac{\boldsymbol{n}_2 \left(\boldsymbol{s}_{p2} - \boldsymbol{s}_{p1} \right)}{\boldsymbol{n}_2 \left(\boldsymbol{a} \otimes \boldsymbol{n}_1 \right)} \left(\boldsymbol{a} \otimes \boldsymbol{n}_1 \right) \right) + \lambda \boldsymbol{a} \qquad \text{(A10.2.)}$$

A11. Line Offsets

Line offsets are simple to calculate since only the line origin must be translated along the offset vector. The orientation vector is not changed; otherwise the new line will no longer be parallel to the original line. The offset vector is the cross product of the orientation vector of the line with the orientation vector of the plane in which the line must be compensated. If the line must be offset a distance d, the new line is

$$\boldsymbol{x}' = \left(\boldsymbol{s}_1 + \frac{\boldsymbol{a} \otimes \boldsymbol{n}}{\left\| \boldsymbol{a} \otimes \boldsymbol{n} \right\|} d \right) + \lambda \boldsymbol{a} \qquad \text{(A11.1.)}$$

A12. Distance from a Point to a Circle

What is the shortest distance from a point $\boldsymbol{p}$ in R^3 to a circle with centre at the origin of the coordinate system, radius r and lying in the XY plane?

The distance of $\boldsymbol{p}$ to the centre of the circle, projected onto the XY plane is

$$d = \sqrt{p_x^2 + p_y^2}$$

The distance of the point to the circle is found from the Z height of the $\boldsymbol{p}$ and the difference between d and r. Therefore the distance from the point to the circle is

$$d = \sqrt{\left(r - \sqrt{p_x^2 - p_y^2} \right)^2 + p_z^2} \qquad \text{(A12.1.)}$$

If the circle does not lie in the XY plane, but in a plane parallel to this plane, and the origin is still on the Z axis, the above equation is modified as follows to find the distance. In this equation c_z is the Z coordinate of the circle's origin.

$$d = \sqrt{\left(r - \sqrt{p_x^2 - p_y^2} \right)^2 + \left(p_z - c_z \right)^2} \qquad \text{(A12.2.)}$$

Appendix B

Pilot Time Study Results

Study No: 1	TIME STUDY FORM		
Observer: K. Schreve	Date: 06-Jan-99	Machine Code:	Mitutoyo Bright 710
Operator: K. Schreve	Part Name: Core Box Bottom	Part No:	CBB
Operation Description:	Measurements for Reverse Engineering	Time Units:	h:min

Date	Element Description	Time Started	Time Finished	Time
1/6	Set up comupter; Clamp part	08:30	08:35	00:05
1/6	Clean core box	08:35	08:41	00:06
1/6	Calibrate probe; Measure coordinate system	08:41	08:50	00:09
1/6	Set up point cloud measurement #1.0	08:50	09:12	00:22
1/6	Start measurement #1.0	09:12	10:00	00:48
1/6	(Stop #1.0) Set up cloud #2.0	10:32	10:40	00:08
1/6	Start measurement #2.0	10:40	10:48	00:08
1/6	Restart after crash #2.1	10:48	10:51	00:03
1/6	Start again #2.1	10:51	11:04	00:13
1/6	Add #2.1 to #2.0	11:04	11:06	00:02
1/6	Set up refinement of #2.0	11:06	11:08	00:02
1/6	Start refinement of #2.0	11:08	11:48	00:40
1/6	Restart in new Geopak	11:48	11:56	00:08
1/6	Measure inlet ports	11:56	11:59	00:03
1/6	Remeasure coordinate system + inlet ports	12:06	12:59	00:53
1/6	Set up measurement of left pipe, 33.0	14:03	14:15	00:12
1/6	Start measurement #3.0	14:15	15:01	00:46
1/6	Resume #3.0	15:36	16:30	00:54
1/7	Resume #3.0	07:20	09:52	02:32
1/7	Calibrate probe; Measure coordinate system	09:59	10:17	00:18
1/7	Measure connection pin origens and other miscellaneous	10:17	11:03	00:46
1/7	Remeasure coordinate system for METRIS	11:03	11:09	00:06
1/7	Set up point cloud measurement #4.0	11:09	11:21	00:12
1/7	Start measurement #4.0	11:21	11:48	00:27
1/7	Set up point cloud measurement #5.0	11:48	11:58	00:10
1/7	Start measurement #5.0	11:58	12:35	00:37
1/7	Resume #5.0	13:48	14:47	00:59
1/7	Resume #5.0	15:59	16:49	00:50
1/7	Set up point cloud measurement #6.0	16:49	16:55	00:06
1/7	Start measurement #6.0	16:55	18:11	01:16
1/8	Resume #6.0	07:10	08:18	01:08
1/8	Resume #6.0	09:00	09:29	00:29
1/8	Set up measurement #7.0 (#3 port)	09:29	09:35	00:06
1/8	Start measurement #7.0	09:35	10:19	00:44
1/8	Set up measurement #8.0 (left plenum blend)	13:23	13:30	00:07
1/8	Start measurement #8.0	13:30	14:30	01:00
1/11	Calibrate probe; Measure coordinate system	07:15	07:23	00:08
1/11	Set up measurement #9.0 (#1 port)	07:23	07:39	00:16
1/11	Start measurement #9.0	07:39	11:03	03:24
1/11	Set up measurement #10.0 (#4 port)	11:12	11:18	00:06
1/11	Start measurement #10.0	11:18	11:20	00:02
1/11	Redo set up of measurement #10.0 (Crashed across border)	11:20	11:28	00:08

1/11	Restart measurement #10.0	11:28	11:30	00:02
1/11	Redo set up of measurement #10.0 (Crashed across border)	11:30	11:37	00:07
1/11	Restart measurement #10.0 (Now do only one half of the pipe)	11:37	12:27	00:50
1/11	Set up measurement #10.1 (the other half)	15:09	15:19	00:10
1/11	Start measurement #10.1	15:19	15:54	00:35
1/11	Set up measurement #11.0 (Plenum top wall)	15:56	16:00	00:04
1/11	Start measurement #11.0	16:00	16:04	00:04
1/11	Redo set up of measurement #11.0 (Crashed across border) (Discard)	16:08	16:34	00:26
1/12	Measure splines on plenum top and bottom wall with METRIS	07:19	07:34	00:15
1/12	Set up measurement #12.0 (Plenum right blend)	07:34	07:39	00:05
1/12	Start measurement #12.0	07:39	08:14	00:35
1/12	Set up measurement #6.1 (Gap in cloud #6.0)	09:18	09:23	00:05
1/12	Start measurement #6.1	09:23	09:37	00:14
1/12	Set up measurement #3.1 (Gap in cloud #3.0)	09:37	09:40	00:03
1/12	Start measurement #3.1	09:40	09:55	00:15
1/12	Splines measurement #12.1 (Splines for plenum right blend)	09:55	10:02	00:07
1/15	Set up measurement #15.0 (more points for plenum)	14:16	14:19	00:03
1/15	Measure coordinate system	14:19	14:27	00:08
1/15	Set up measurement #15.0	14:27	14:33	00:06
1/15	Start measurement #15.0	14:33	14:38	00:05
1/15	Set up measurement #15.1	14:38	14:40	00:02
1/15	Start measurement #15.1	14:40	14:46	00:06
1/15	Set up measurement #15.2	14:46	14:48	00:02
1/15	Start measurement #15.2	14:48	15:00	00:12
1/15	Set up measurement #15.3	15:07	15:09	00:02
1/15	Start measurement #15.3	15:09	15:14	00:05
1/18	Set up measurement #15.4	07:10	07:20	00:10
1/18	Start measurement #15.4	07:20	07:36	00:16

Total (hours): 26.88

Study No: 2	TIME STUDY FORM		
Observer: K. Schreve	Date: 06-Jan-99	Machine Code:	Mitutoyo Bright 710
Operator: K. Schreve	Part Name: Core Box Bottom	Part No:	CBB
Operation Description: CAD Modelling		Time Units:	h:min

Date	Element Description	Time Started	Time Finished	Time
1/6	Start modelling left interior split plane	11:27	11:48	00:21
1/6	Model inlet ports + inlet step	14:18	15:01	00:43
1/6	Resume	15:36	15:46	00:10
1/6	Split plane	15:46	16:14	00:28
1/7	Wedge cavity	07:22	08:44	01:22
1/7	Plenum	11:21	11:42	00:21
1/7	Inlet port shoulders	11:42	11:48	00:06
1/7	Resume	12:00	12:07	00:07
1/7	Redo split plane outside	12:07	12:35	00:28
1/7	Thumbnails	13:55	14:32	00:37
1/7	Find interior split surface's edge	17:36	18:11	00:35
1/8	Resume	07:12	08:17	01:05
1/8	Resume	09:00	09:27	00:27
1/8	Resume	09:35	10:06	00:31
1/8	Interior split surface	10:06	10:19	00:13
1/8	Resume	13:30	13:47	00:17
1/8	Resume	13:52	14:23	00:31
1/11	Resume	07:39	08:30	00:51
1/11	Resume	11:18	12:27	01:09
1/11	Resume	15:19	15:56	00:37
1/11	Resume	16:00	16:08	00:08
1/12	Resume	07:39	08:14	00:35
1/12	Resume	09:23	09:34	00:11
1/12	Resume	09:40	09:55	00:15
1/12	Resume	10:02	10:27	00:25
1/12	Resume	10:54	11:31	00:37
1/12	Redo front Ends	11:31	12:04	00:33
1/12	Resume	14:50	15:20	00:30
1/12	Resume	16:02	17:07	01:05
1/13	Resume	07:18	07:49	00:31
1/13	Finish split plane	08:36	08:57	00:21
1/13	Resume	09:28	10:10	00:42
1/13	Resume	10:43	12:04	01:21
1/13	Resume	14:20	14:43	00:23
1/13	Resume	14:53	15:34	00:41
1/14	Left pipe	09:40	09:42	00:02
1/14	Resume	09:54	10:39	00:45
1/14	Resume	10:46	10:56	00:10
1/14	Resume	11:50	12:22	00:32
1/14	Right pipe	12:22	12:25	00:03
1/14	Resume	14:23	15:34	01:11
1/14	Resume	16:05	16:55	00:50

1/15	Resume	11:47	12:09	00:22
1/15	Plenum blends	12:09	13:00	00:51
1/15	Resume	13:36	14:15	00:39
1/18	Plenum blends	07:20	08:06	00:46
1/18	Resume	08:30	10:00	01:30
1/18	Resume	10:20	11:09	00:49
1/18	Resume	11:34	12:32	00:58
1/18	Intersection between ports and pipes	13:46	15:00	01:14
1/18	Resume	15:30	15:52	00:22
1/18	Draft on pipes	15:52	16:19	00:27

Total (hours): 30.8

Study No: 3	TIME STUDY FORM		
Observer: K. Schreve	Date: 06-Jan-99	Machine Code:	Mitutoyo Bright 710
Operator: K. Schreve	Part Name: Core Box Top	Part No:	CBT
Operation Description:	Measurements for Reverse Engineering	Time Units:	h:min

Date	Element Description	Time Started	Time Finished	Time
1/14	Set up part on machine	07:44	07:51	00:07
1/14	Calibrate probe and measure coordinate system	07:51	07:56	00:05
1/14	Set up measurement #13.0 (Plenum)	07:56	08:07	00:11
1/14	Start measurement #13.0	08:07	08:11	00:04
1/14	Stort out problem with probe contact	08:11	08:21	00:10
1/14	Calibrate probe and measure coordinate system	08:21	08:31	00:10
1/14	Redo set up of measurement #13.0	09:22	09:27	00:05
1/14	Start measurement #13.0 -> unexplained error	09:27	09:30	00:03
1/14	Set up measurement #13.1 (first half of plenum)	09:30	09:38	00:08
1/14	Start measurement #13.1	09:38	09:42	00:04
1/14	Redo set up of measurement #13.1 (Top patch)	09:42	09:52	00:10
1/14	Start measurement #13.1	09:52	10:39	00:47
1/14	Set up measurement #14.0 (left pipe)	10:39	10:44	00:05
1/14	Start measurement #14.0	10:44	10:56	00:12
1/19	Calibrate probe and measure coordinate system	11:11	11:18	00:07
1/19	Measure origens	11:18	11:22	00:04
1/19	Measure front step (5 planes)	11:22	11:28	00:06
1/19	Inlet ports contour	11:28	11:44	00:16
1/19	Print data	11:44	11:53	00:09
1/19	Calibrate probe and measure coordinate system	11:53	12:03	00:10
1/19	Set up measurement #16.0 (left pipe)	12:03	12:12	00:09
1/19	Start measurement #16.0	12:12	12:30	00:18
1/19	Set up measurement #17.0	12:30	12:34	00:04
1/19	Start measurement #17.0	12:34	12:44	00:10
1/19	Restart measurement #17.0	14:29	15:11	00:42
1/19	Calibrate probe and measure coordinate system	15:29	15:39	00:10
1/19	Redo set up of measurement #16.0	15:39	15:44	00:05
1/19	Restart measurement #16.0	15:44	16:01	00:17
1/19	Redo set up of measurement #17.0	16:06	16:09	00:03
1/19	Restart measurement #17.0	12:44	16:50	04:06
1/19	Set up measurement #16.1 (more points on left pipe)	19:51	19:57	00:06
1/19	Start measurement #16.1	19:57	20:02	00:05
1/19	Set up measurement #16.2 (more points on left pipe)	20:02	20:05	00:03
1/19	Start measurement #16.2	20:05	20:07	00:02
1/19	Set up refinement #16.3 (left pipe)	20:07	20:10	00:03
1/19	Start refinement #16.3 (operator left at 21:01)	20:10	00:59	04:49
1/20	Set up measurement #18.0 (Right pipe)	07:18	07:31	00:13
1/20	Start measurement #18.0	07:31	07:46	00:15
1/20	Set up measurement #18.1	07:46	07:51	00:05
1/20	Start measurement #18.1	07:51	08:08	00:17
1/20	Set up measurement #18.2 (Refinement)	08:43	08:50	00:07
1/20	Start measurement #18.2	08:50	11:30	02:40

1/20	Set up measurement #19.0 (#2 Port)	11:30	11:38	00:08
1/20	Start measurement #19.0	11:38	14:47	03:09
1/21	Intersection between inlet ports and pipes; Measure coordinate system	07:29	07:39	00:10
1/21	Set up measurement #20.0 (#3 Port)	07:39	07:45	00:06
1/21	Start measurement #20.0	07:45	09:25	01:40
1/21	Set up measurement #21.0 (#4 Port)	09:25	09:30	00:05
1/21	Start measurement #21.0	09:30	11:49	02:19
1/21	Set up measurement #22.0 (Wedge top)	12:08	12:12	00:04
1/21	Start measurement #22.0	12:12	12:45	00:33
1/21	Set up measurement #23.0 (Plenum middle section)	12:48	12:54	00:06
1/21	Start measurement #23.0	12:54	13:34	00:40
1/21	Set up measurement #18.4 (More points for gap in right pipe)	13:45	13:48	00:03
1/21	Start measurement #18.4	13:48	13:54	00:06
1/21	Redo set up measurement #18.4 (sticking colision)	17:08	17:13	00:05
1/21	Restart measurement #18.4	17:13	17:20	00:07
1/21	Calibrate probe and measure coordinate system	17:20	17:32	00:12
1/21	Measure more points for right pipe in new Geopak software	17:32	17:47	00:15
1/21	Measure plenum in Geopak	17:47	18:23	00:36

Total (hours): 28.43

Study No: 4	TIME STUDY FORM		
Observer: K. Schreve	Date: 19-Jan-99	Machine Code:	Mitutoyo Bright 710
Operator: K. Schreve	Part Name: Core Box Top	Part No:	CBT
Operation Description: CAD Modelling		Time Units:	h:min

Date	Element Description	Time Started	Time Finished	Time
1/19	Inlet port step	14:52	15:11	00:19
1/19	Transform coordinate system of Port102.txt, LeftPipe02.txt (Did not work)	15:11	15:29	00:18
1/19	Model inlet ports	16:12	16:54	00:42
1/19	Import left pipe clouds	19:47	19:51	00:04
1/19	Model inlet ports (Resume)	20:10	21:01	00:51
1/20	Import left pipe clouds	07:14	07:18	00:04
1/20	Import split plane	07:31	07:46	00:15
1/20	Resume	07:51	08:08	00:17
1/21	Intersection between inlet ports and pipes	07:49	08:11	00:22
1/21	Model inlet ports step	08:33	08:40	00:07
1/21	Model left pipes (Middle triangle)	08:49	09:24	00:35
1/21	Resume	09:30	09:43	00:13
1/21	Model #1 Port	09:43	09:55	00:12
1/21	Model #2 Port	10:32	10:43	00:11
1/21	Model main left pipe	12:12	12:34	00:22
1/21	Import clouds for right pipe	12:34	12:48	00:14
1/21	Resume	13:39	13:45	00:06
1/21	Import plenum clouds	18:23	18:32	00:09
1/21	Manipulate right pipe clouds	18:32	18:40	00:08
1/21	Model Right pipe (Middle triangle)	18:40	18:51	00:11
1/22	Model #3 Port	07:23	07:34	00:11
1/22	Model #4 Port	07:34	07:46	00:12
1/22	Model main right pipe	07:46	07:55	00:09
1/22	Resume	08:50	09:12	00:22
1/22	Redo middle section (A good stich could not be achieved)	09:12	09:35	00:23
1/22	Resume	11:48	11:55	00:07
1/22	Redo main right pipe	11:55	12:16	00:21
1/22	Model plenum	12:16	12:50	00:34
1/22	Model wedge top	13:47	14:00	00:13
1/22	Model inlet step	14:00	14:17	00:17
1/22	Fil gaps	14:25	14:57	00:32
1/22	Resume	15:36	15:39	00:03

Total (hours): 9.07

Appendix C

Polynomial Approximation of a Circle

In order to find the polynomial that best approximates any circle segment, the distance between the polynomial and the circle must be minimised. The circle and the polynomial use the same parameterisation so that the problem can be simplified. Here the circle segment will always start where the parameter value is zero and the parameter will never exceed 2π. The parameterisation of the circle segment is shown in the figure below. The radius of the circle is r and u is the arc length.

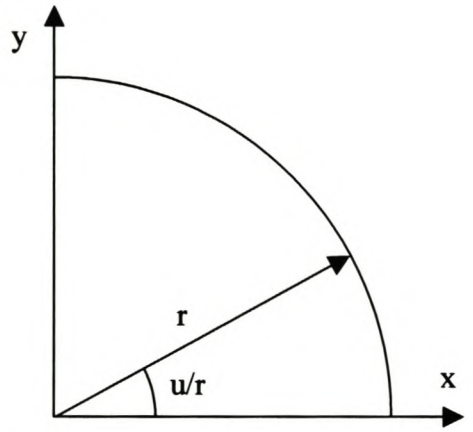


Figure 1 Parameterisation of a Circle.

The equation of the circle is

$$f_c(u) = \begin{pmatrix} r \cdot \cos\left(\frac{u}{r}\right) \\ r \cdot \sin\left(\frac{u}{r}\right) \end{pmatrix} \qquad \text{(C1.)}$$

and the equation of the quadratic polynomial is

$$f_p(u) = \begin{pmatrix} a_x + b_x \cdot u + c_x \cdot u^2 \\ a_y + b_y \cdot u + c_y \cdot u^2 \end{pmatrix} \qquad \text{(C2.)}$$

The objective is to find the six coefficients of the polynomial that will minimise the distance between the polynomial and the circle, i.e. the following equation must be minimised. In this equation s is the total arc length of the circle segment.

$$F = \int_0^s \left(\left|f_c(u) - f_p(u)\right|\right)^2 du$$

$$F = \int_0^s \left(f_c(u)\cdot f_c(u) - 2\cdot f_c(u)\cdot f_p(u) + f_p(u)\cdot f_p(u)\right) du \qquad \text{(C3.)}$$

If the partial derivatives of the above equation with respect to the polynomial coefficients are taken, six equations are obtained that will be used to find the polynomial coefficients. With this in mind, the first term in the above equation can be eliminated because it does not contain any of the polynomial coefficients and will therefore play no role in the differentiation process. This makes the rest of the discussion a little easier to follow. So, the equation is rewritten as follows.

$$F = \int_0^s \left(-2\cdot f_c(u)\cdot f_p(u) + f_p(u)\cdot f_p(u)\right) du \qquad \text{(C4.)}$$

The result of the integration is too long to present here. After integration, the partial derivatives are taken.

$\frac{\partial F}{\partial a_x} = 0$ yields

$$-2\cdot\sin\left(\frac{s}{r}\right)\cdot r^2 + \frac{2}{3}\cdot s^3\cdot c_x + b_x\cdot s^2 + 2\cdot a_x\cdot s = 0 \qquad \text{(C5.)}$$

$\frac{\partial F}{\partial b_x} = 0$ yields

$$-2\cdot r^3\cdot\left(\cos\left(\frac{s}{r}\right) + \frac{s}{r}\cdot\sin\left(\frac{s}{r}\right)\right) + \frac{1}{2}\cdot c_x\cdot s^4 + \frac{2}{3}\cdot s^3\cdot b_x + a_x\cdot s^2 + 2\cdot r^3 = 0 \qquad \text{(C6.)}$$

$\frac{\partial F}{\partial c_x} = 0$ yields

$$-2\cdot r^4\cdot\left(\frac{s^2}{r^2}\cdot\sin\left(\frac{s}{r}\right)-2\cdot\sin\left(\frac{s}{r}\right)+2\cdot\frac{s}{r}\cdot\cos\left(\frac{s}{r}\right)\right)+\frac{2}{5}\cdot c_x\cdot s^5+\frac{1}{2}\cdot b_x\cdot s^4+\frac{2}{3}\cdot a_x\cdot s^3=0 \qquad \text{(C7.)}$$

$\frac{\partial F}{\partial a_y}=0$ yields

$$2\cdot\cos\left(\frac{s}{r}\right)\cdot r^2+\frac{2}{3}\cdot s^3\cdot c_y+b_y\cdot s^2+2\cdot a_y\cdot s-2\cdot r^2=0 \qquad \text{(C8.)}$$

$\frac{\partial F}{\partial b_y}=0$ yields

$$-2\cdot r^3\cdot\left(\sin\left(\frac{s}{r}\right)-\frac{s}{r}\cdot\cos\left(\frac{s}{r}\right)\right)+\frac{1}{2}\cdot c_y\cdot s^4+\frac{2}{3}\cdot s^3\cdot b_y+a_y\cdot s^2=0 \qquad \text{(C9.)}$$

$\frac{\partial F}{\partial c_y}=0$ yields

$$-2\cdot r^4\cdot\left(\frac{-s^2}{r^2}\cdot\cos\left(\frac{s}{r}\right)+2\cdot\cos\left(\frac{s}{r}\right)+2\cdot\frac{s}{r}\cdot\sin\left(\frac{s}{r}\right)\right)+\frac{2}{5}\cdot c_y\cdot s^5+\frac{1}{2}\cdot b_y\cdot s^4+\frac{2}{3}\cdot a_y\cdot s^3+4\cdot r^4=0$$

(C10.)

The above six equations are now used to find the six coefficients of the polynomial curve that best approximates the circle segment of arc length s. Algebraic manipulation of these equations yield the following six equations for the coefficients.

$$c_x=30\cdot r^2\cdot\frac{\left(s^2\cdot\sin\left(\frac{s}{r}\right)-12\cdot\sin\left(\frac{s}{r}\right)\cdot r^2+6\cdot r\cdot s\cdot\cos\left(\frac{s}{r}\right)+6\cdot r\cdot s\right)}{s^5} \qquad \text{(C11.)}$$

$$b_x=\frac{\left(12\cdot r^3\cdot\cos\left(\frac{s}{r}\right)+6\cdot r^2\cdot s\cdot\sin\left(\frac{s}{r}\right)-c_x\cdot s^4-12\cdot r^3\right)}{s^3} \qquad \text{(C12.)}$$

$$a_x=\frac{1}{6}\cdot\frac{\left(-3\cdot b_x\cdot s^2-2\cdot s^3\cdot c_x+6\cdot\sin\left(\frac{s}{r}\right)\cdot r^2\right)}{s} \qquad \text{(C13.)}$$

$$c_y = 30 \cdot r^2 \cdot \frac{\left(s^2 - s^2 \cdot \cos\left(\frac{s}{r}\right) + 6 \cdot r \cdot s \cdot \sin\left(\frac{s}{r}\right) - 12 \cdot r^2 + 12 \cdot \cos\left(\frac{s}{r}\right) \cdot r^2\right)}{s^5} \qquad \text{(C14.)}$$

$$b_y = \frac{\left(-c_y \cdot s^4 - 6 \cdot r^2 \cdot s - 6 \cdot r^2 \cdot s \cdot \cos\left(\frac{s}{r}\right) + 12 \cdot r^3 \cdot \sin\left(\frac{s}{r}\right)\right)}{s^3} \qquad \text{(C15.)}$$

$$a_y = \frac{-1}{6} \cdot \frac{\left(3 \cdot b_y \cdot s^2 + 2 \cdot s^3 \cdot c_y - 6 \cdot r^2 + 6 \cdot \cos\left(\frac{s}{r}\right) \cdot r^2\right)}{s} \qquad \text{(C16.)}$$

The polynomial curve is plotted against the circle segment for four different cases in the figures below. The curves are compared with circle segments with arc lengths of $2\pi r$, πr, $\pi r/2$ and $\pi r/6$.

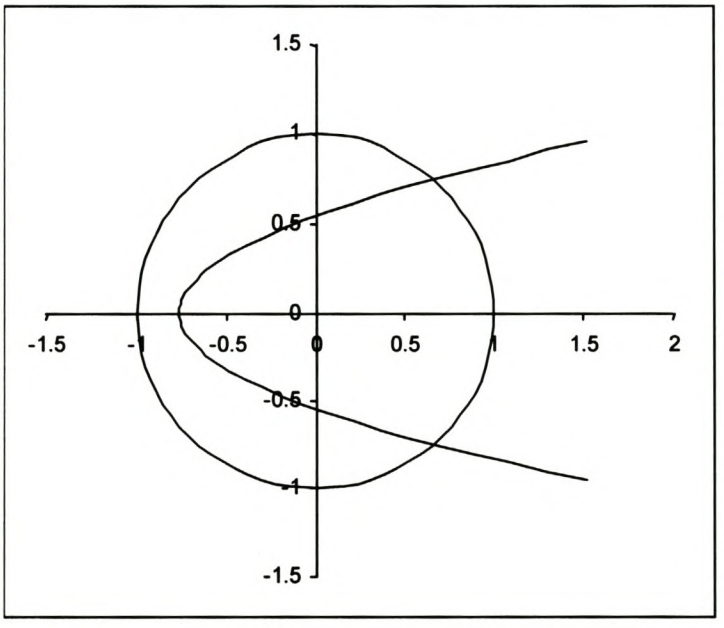


Figure 2 Quadratic Polynomial Approximation of a Full Circle.

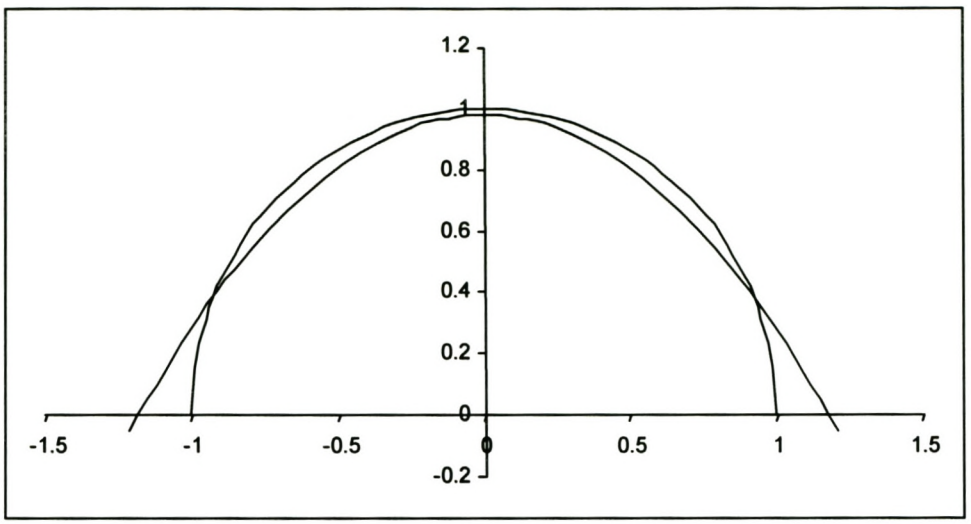


Figure 3 Quadratic Polynomial Approximation of a Half Circle.

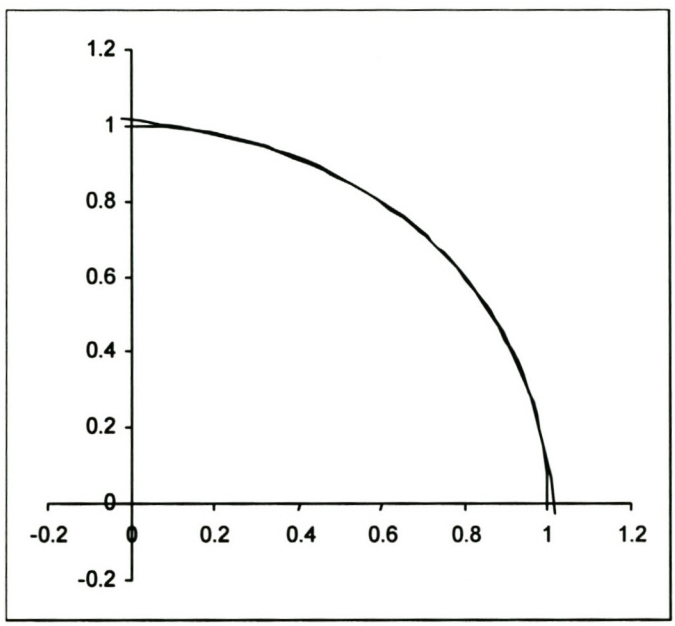


Figure 4 Quadratic Polynomial Approximation of a Quarter Circle.

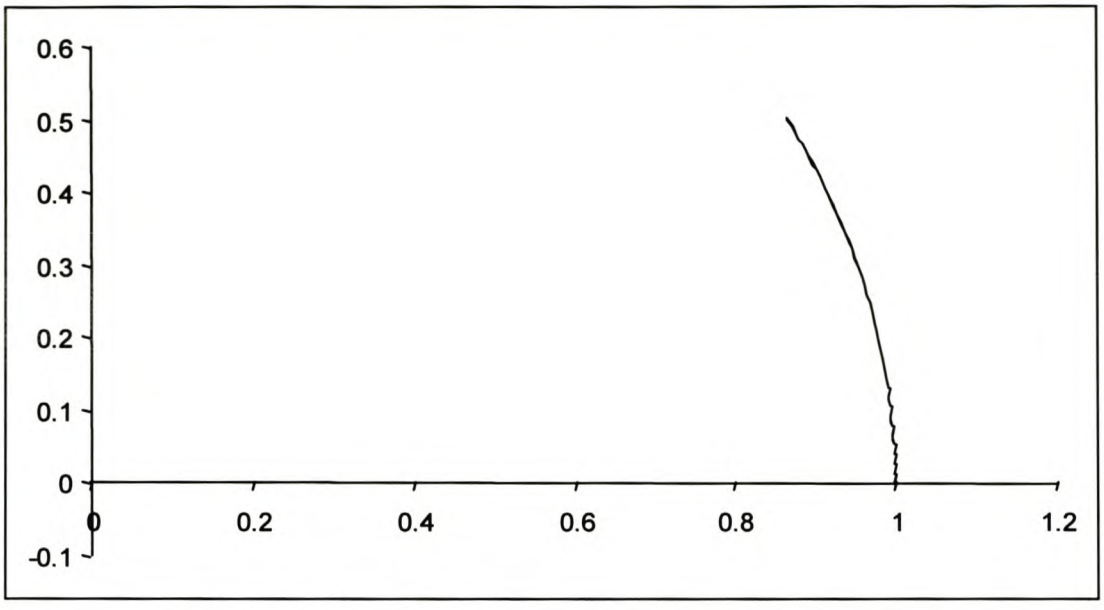


Figure 5 Quadratic Polynomial Approximation of a Twelfth Circle.

As expected the approximation improves as the arc length decreases. In the last figure, the difference between the polynomial and the circle segment is barely detectable. The coefficients for the above polynomials are given in the table below.

Table 1 Quadratic Polynomial Coefficients for Circle Approximation.

	Full Circle	**Half Circle**	**Quarter Circle**	**Twelfth Circle**
a_x	1.519818	1.215854	1.019373	1.000265
b_x	-1.451319	-0.774037	-0.133133	-0.005406
c_x	0.230985	0	-0.33824	-0.480603
a_y	0.95493	-0.050466	-0.024325	-0.001161
b_y	-0.303964	1.312236	1.195745	1.026749
c_y	0	-0.417698	-0.33824	-0.128777

The distance between the polynomial curve and the circle segments for the four cases given above are plotted in the figures below. Clearly they all have the same shape, with the maximum distance at the beginning and end of the circle segments.

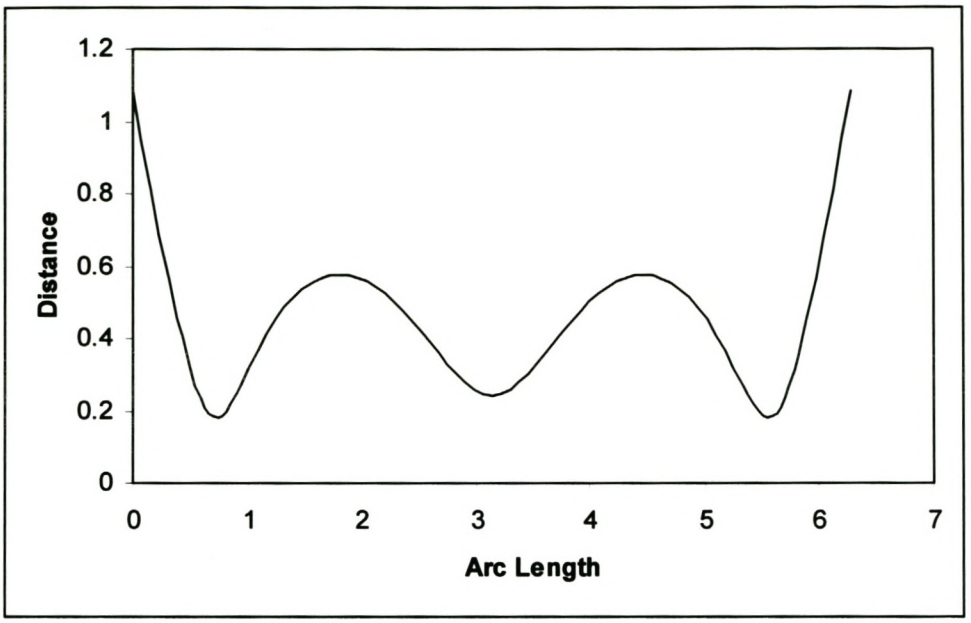


Figure 6 Distance between Best Fit Quadratic Polynomial and a Full Circle.

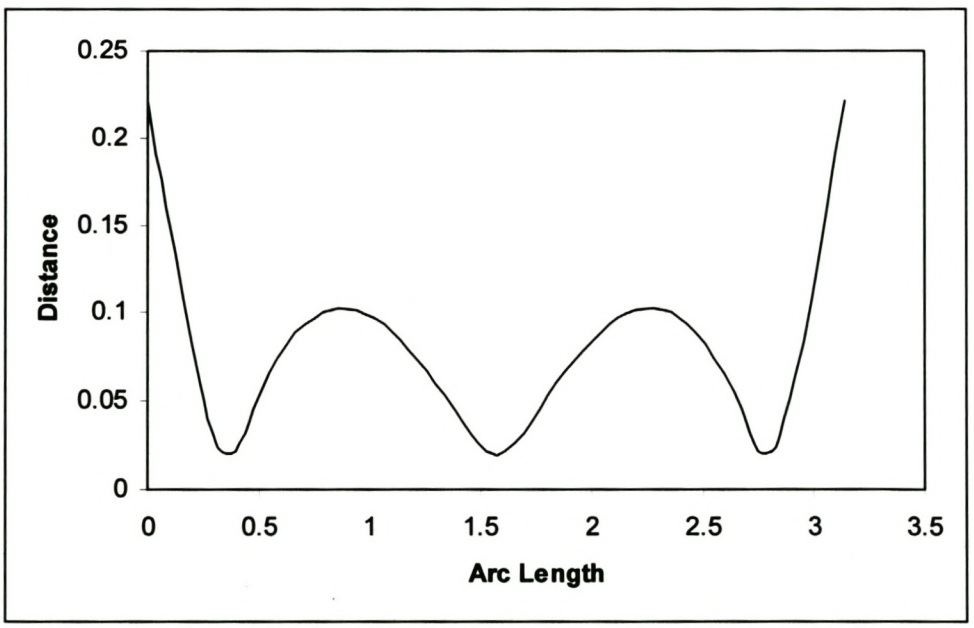


Figure 7 Distance between Best Fit Quadratic Polynomial and a Half Circle.

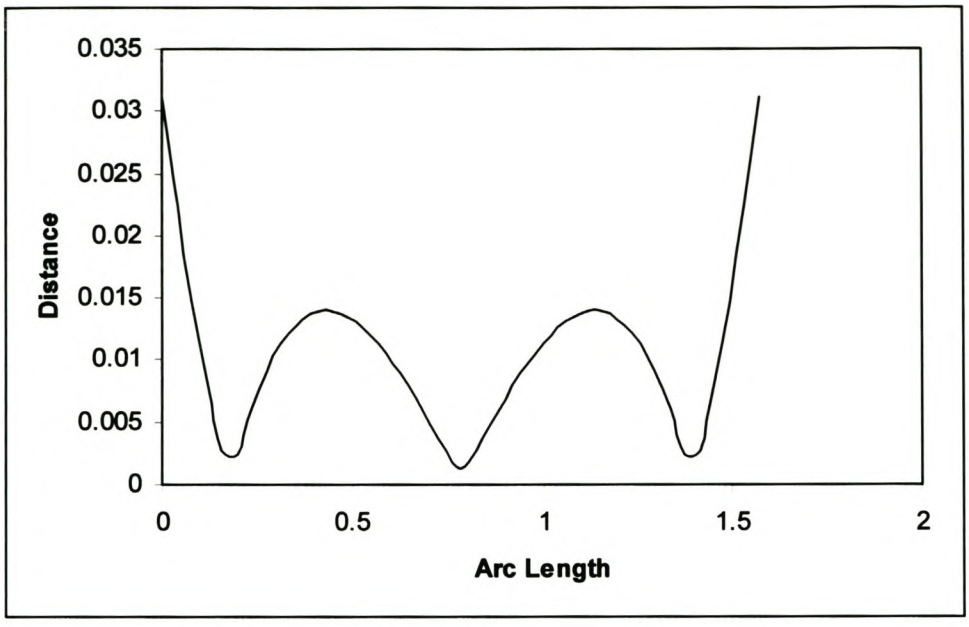


Figure 8 Distance between Best Fit Quadratic Polynomial and a Quarter Circle.

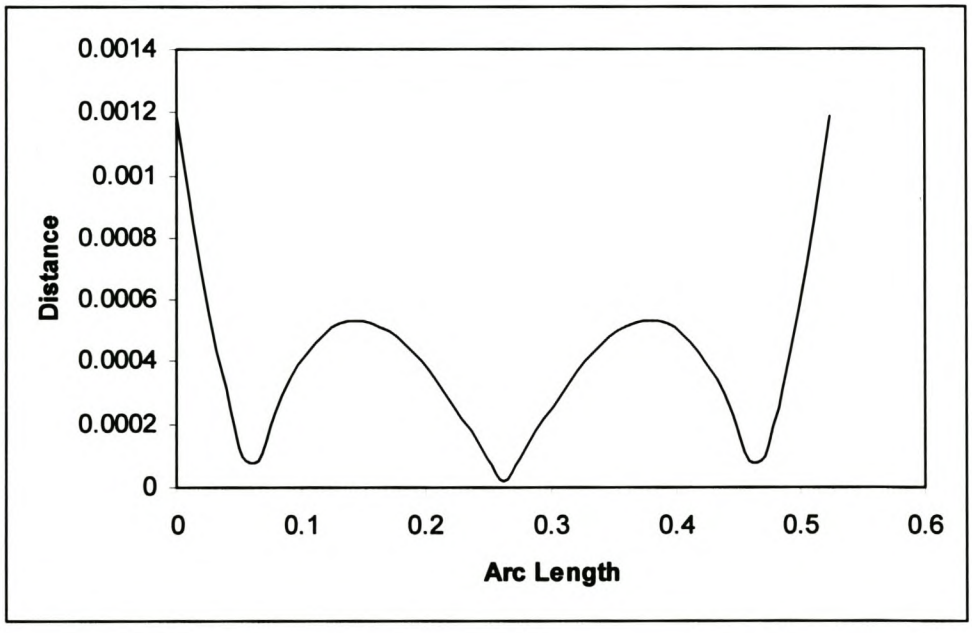


Figure 9 Distance between Best Fit Quadratic Polynomial and a Twelfth Circle.

The distance between the polynomial and the circle is given in the following equation.

$$d(u) = \left| f_c(u) - f_p(u) \right| \qquad \text{(C17.)}$$

The distance at u=0, the maximum distance, is then given by the following expression.

$$d_{max} = r^2 - 2 \cdot r \cdot a_x + a_x^2 + a_y^2 \qquad \text{(C18.)}$$

It turns out that this maximum distance of the polynomial from the circle can be non-dimensionalised by dividing it by the radius, r. The non-dimensional distance is plotted against the ratio of the arc length to the radius in the graph below.

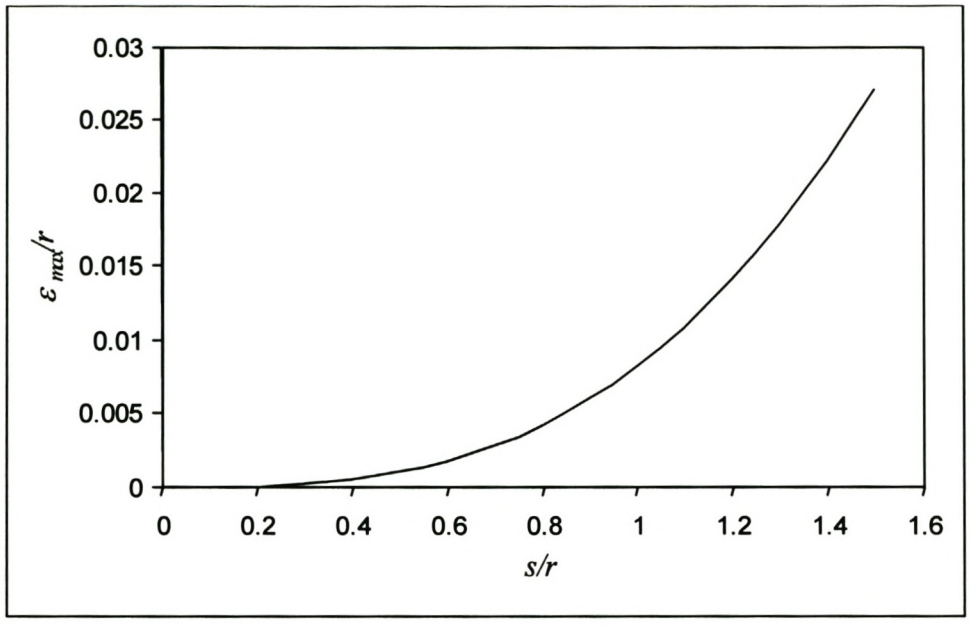


Figure 10 Maximum Deviation from a Circle Segment of a Best Fit Quadratic Polynomial.

Appendix D

Line Approximation of a Circle

D1. Introduction

In this appendix, a theoretical model of the expected error of the edge scanning algorithm is presented assuming that a line approximation of the points is made. The line that best approximates a given circular segment is first derived. Then, the maximum distance from the line to the circular segment is determined. The best line is used to determine the maximum theoretical error for the given scanning parameters.

Of course, the scanned points will seldom lie on a circular segment. This is assumption that is made to simplify the analysis. It assumes that the curvature in the direction perpendicular to the edge is constant.

D2. Best Fit Line

In order to find the line that best approximates any circle segment, the distance between the line and the circle must be minimised. The circle and the line use the same parameterisation so that the problem can be simplified. The unity restriction that is often placed on the direction vector of the line is also dropped since it only complicates the analysis and does not contribute any insight. Here the circle segment starts where the parameter value is zero and the parameter never exceeds 2π. The parameterisation of the circle segment is shown in the figure below. The radius of the circle is r and u is the arc length.

The equation of the circle is

$$\boldsymbol{f}_c(u) = \begin{pmatrix} r\cos\left(\frac{u}{r}\right) \\ r\sin\left(\frac{u}{r}\right) \end{pmatrix} \qquad \text{(D2.1.)}$$

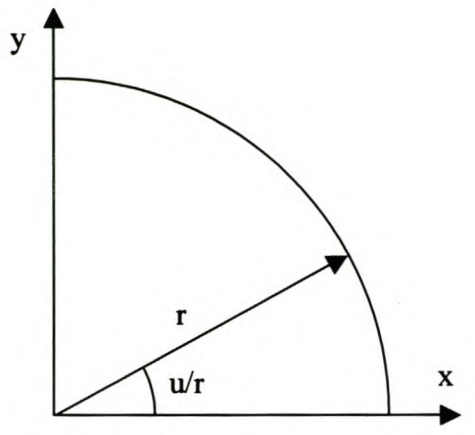


Figure 1 Parameterisation of a Circle.

The equation of the line, using the same parameterisation, is

$$\boldsymbol{f}_p = \begin{pmatrix} a_x + b_x u \\ a_y + b_y u \end{pmatrix} \tag{D2.2.}$$

The four coefficients of the line that will minimise the distance between the line and the circular segment must be found. The following equation must be minimised. In this equation s is the total arc length of the circular segment.

$$F = \int_0^s \left\| \boldsymbol{f}_c(u) - \boldsymbol{f}_p(u) \right\|^2 du$$

$$F = \int_0^s \left(\boldsymbol{f}_c(u)\boldsymbol{f}_c(u) - 2\boldsymbol{f}_c(u)\boldsymbol{f}_p(u) + \boldsymbol{f}_p(u)\boldsymbol{f}_p(u) \right) du \tag{D2.3.}$$

If the partial derivatives of the above equation with respect to the line coefficients are taken, four equations are obtained that is used to find the coefficients. With this in mind, the first term in the above equation can be eliminated since it does not contain any of the coefficients and therefore it is anyway eliminated during differentiation. This makes the rest a little easier to follow. So, the equation is now rewritten as follows.

$$F = \int_0^s \left(-2\boldsymbol{f}_c(u)\boldsymbol{f}_p(u) + \boldsymbol{f}_p(u)\boldsymbol{f}_p(u) \right) du \tag{D2.4.}$$

The partial derivatives with respect to the line coefficients are taken to obtain four equations.

$\frac{dF}{da_x} = 0$ yields

$$-2r^2 \sin\left(\frac{s}{r}\right) + 2a_x s + b_x s^2 = 0 \qquad \text{(D2.5.)}$$

$\frac{dF}{db_x} = 0$ yields

$$-2r^3\left(\cos\left(\frac{s}{r}\right) + \frac{s}{r}\sin\left(\frac{s}{r}\right)\right) + 2r^3 + a_x s^2 + \frac{2}{3}b_x s^3 = 0 \qquad \text{(D2.6.)}$$

$\frac{dF}{da_y} = 0$ yields

$$2r^2 \cos\left(\frac{s}{r}\right) - 2r^2 + 2a_y s + b_y s^2 = 0 \qquad \text{(D2.7.)}$$

$\frac{dF}{db_y} = 0$ yields

$$-2r^3\left(\sin\left(\frac{s}{r}\right) - \frac{s}{r}\cos\left(\frac{s}{r}\right)\right) + a_y s^2 + \frac{2}{3}b_y s^3 = 0 \qquad \text{(D2.8.)}$$

The above four equations are now used to find the line's coefficients that best approximates a circle segment of arc length s. Algebraic manipulation of these equations yield the following four equations for the coefficients.

$$a_x = \frac{r^2}{s}\left[-2\sin\left(\frac{s}{r}\right) - \frac{6r}{s}\left(\cos\left(\frac{s}{r}\right) - 1\right)\right] \qquad \text{(D2.9.)}$$

$$b_x = \frac{6r^2}{s^3}\left(2r\cos\left(\frac{s}{r}\right) + s\sin\left(\frac{s}{r}\right) - 2r\right) \qquad \text{(D2.10.)}$$

$$a_y = \frac{-r^2}{s}\left(-2\cos\left(\frac{s}{r}\right) - 4 + \frac{6r}{s}\sin\left(\frac{s}{r}\right)\right) \qquad \text{(D2.11.)}$$

$$b_y = \frac{6r^2}{s^3}\left(2r\sin\left(\frac{s}{r}\right) - s\cos\left(\frac{s}{r}\right) - s\right) \qquad \text{(D2.12.)}$$

The line is plotted against a circle segment for three different cases below. The lines are compared with circle segments with arc length of πr, πr/2 and πr/6. The line representing a full circle lies on the Y-axis. The coefficients of this and the other three lines are given in Table 1.

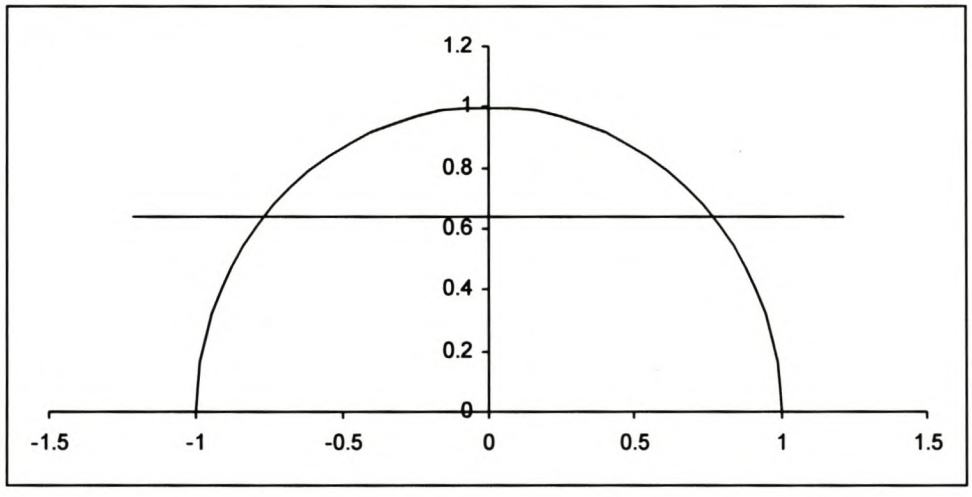


Figure 2 Line Approximation of a Half Circle.

The coefficients for the line in figures 2 to 4 are given in the table below.

Table 1 Line Coefficients for Circle Approximation.

	Full Circle	**Half Circle**	**Quarter Circle**	**Twelfth Circle**
a_x	0	1.215854	1.158469	1.022225
b_x	0	-0.77404	-0.66444	-0.25705
a_y	0.95493	0.63662	0.114771	0.004723
b_y	-0.30396	0	0.66444	0.959322

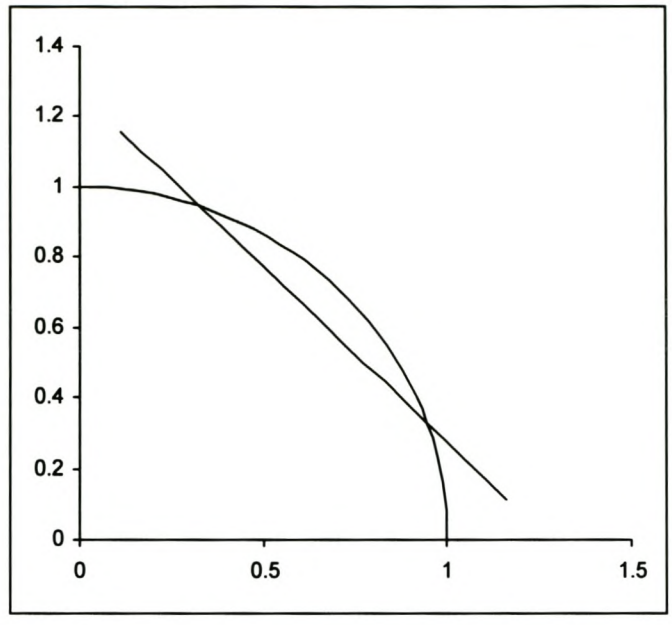


Figure 3 Line Approximation of a Quarter Circle.

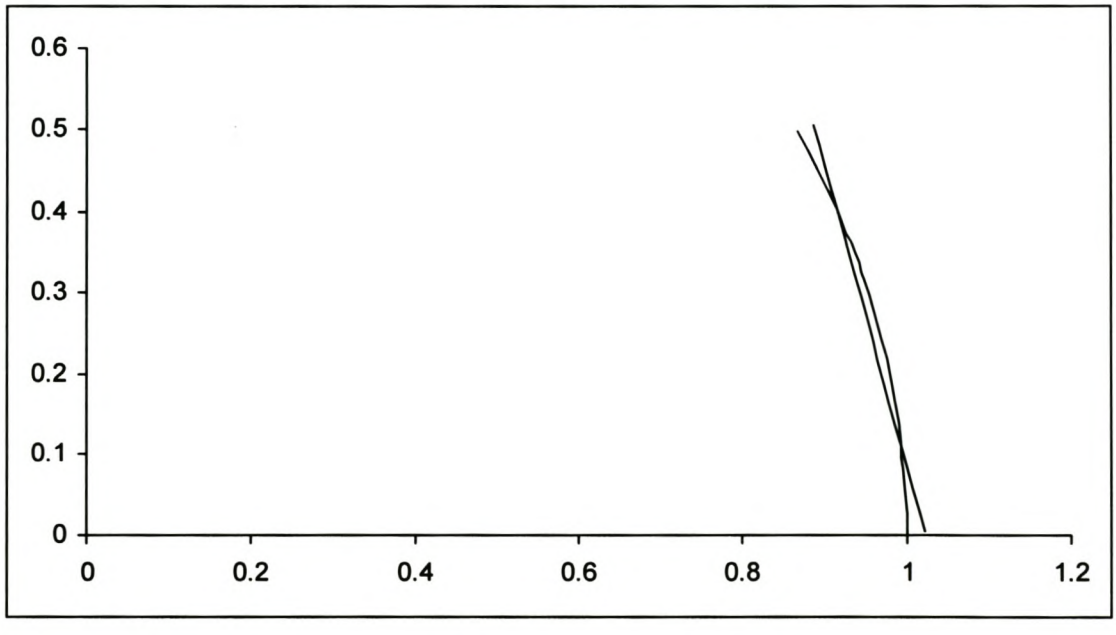


Figure 4 Line Approximation of a Twelfth Circle.

D3. Distance between Line and Circle

The distance between the line and the circle segments for the four cases given above are plotted in the figures below. Clearly they all have the same shape, with the maximum distance at the beginning and end of the circle segments. It must be noted

that the distance is actually not the shortest distance between the line and arc, but rather the between points on the line and circle with the same parameter.

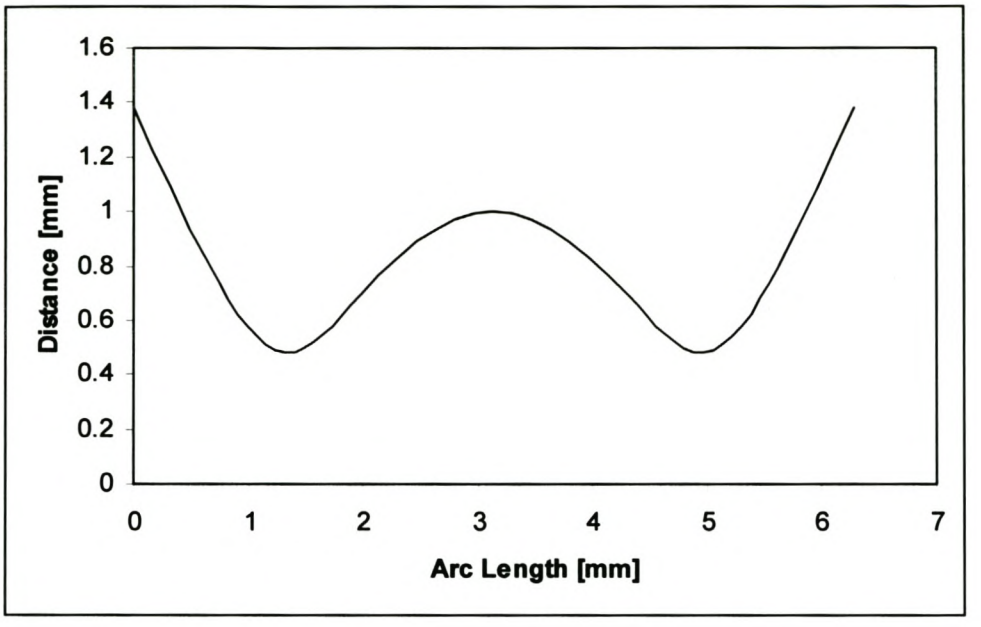


Figure 5 Distance between Best Fit Line and a Full Circle.

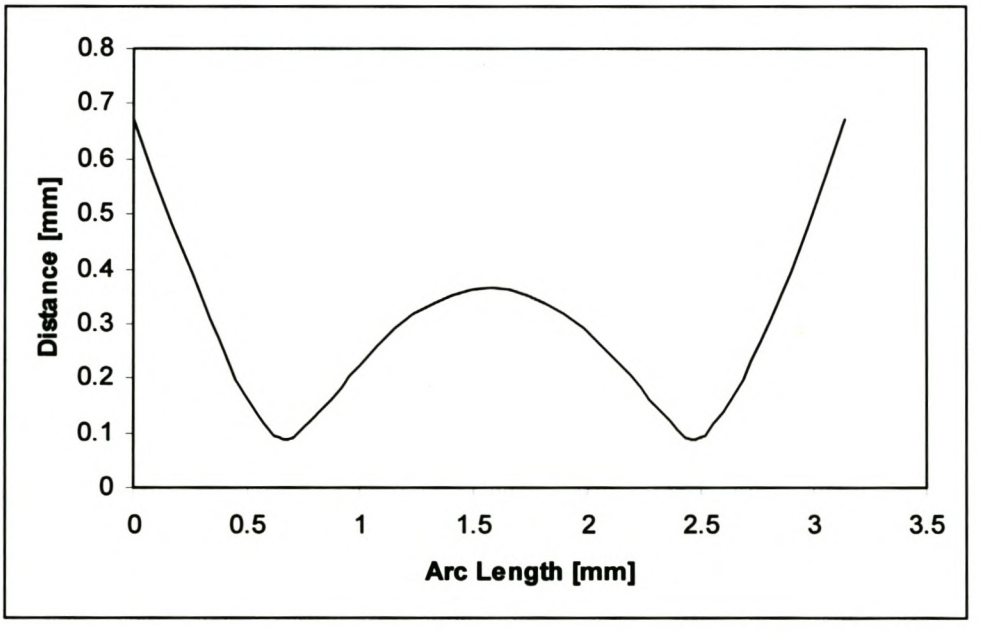


Figure 6 Distance between a Best Fit Line and a Half Circle.

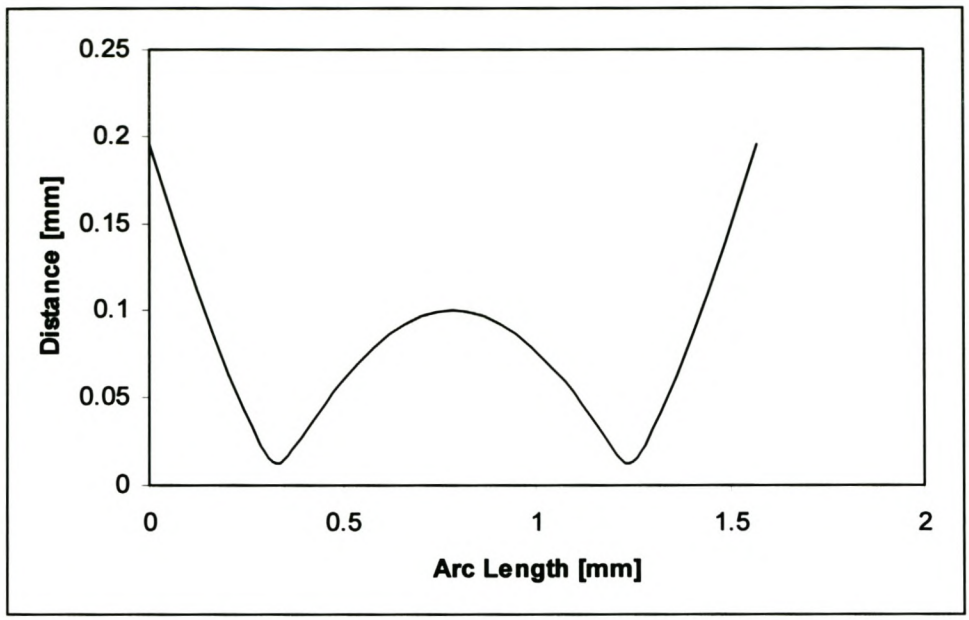


Figure 7 Distance between a Best Fit Line and a Quarter Circle.

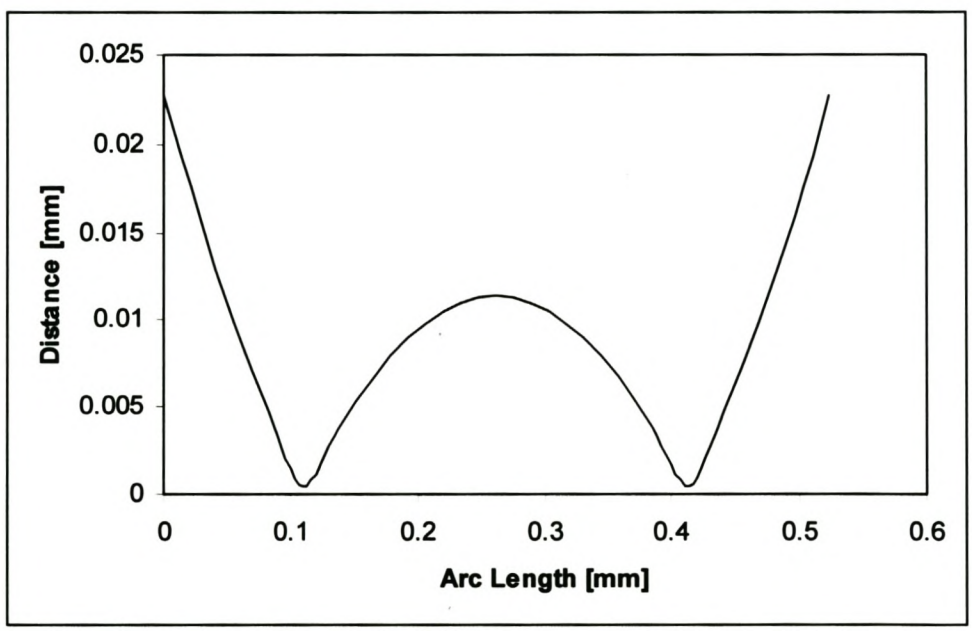


Figure 8 Distance between a Best Fit Line and a Twelfth Circle.

The distance between the line and the circle is given in the following equation.

$$d(u) = \left\| \boldsymbol{f}_c(u) - \boldsymbol{f}_p(u) \right\| \qquad \text{(D3.1.)}$$

From the above figures it is clear that the maximum distance deviation is at the beginning and end of the line. The distance at $u=0$, the maximum distance, is then given by the following expression.

$$d_{\max} = \sqrt{(r - a_x)^2 + a_y{}^2} \qquad \text{(D3.2.)}$$

It turns out that the maximum distance can be non-dimensionalised by dividing it by the radius, r. The non-dimensional distance is plotted against the ratio of the arc length to the radius in the graph below.

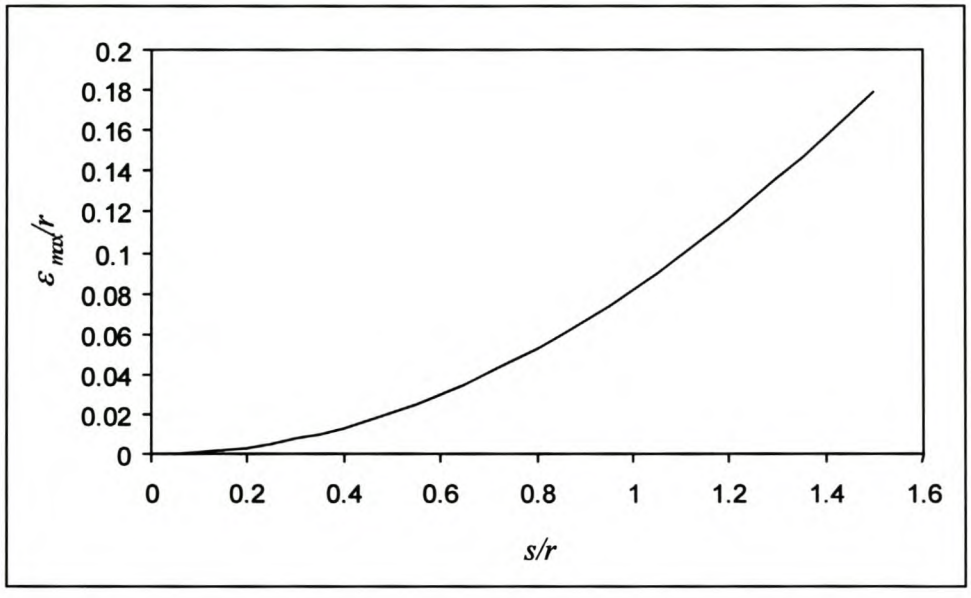


Figure 9 Maximum Deviation from a Circle Segment of a Best Fit Line.

D4. Influence of the Process Parameters on the Theoretical Error

The equation for the theoretical error derived in Chapter 5 can be used without change by simply substituting in the equation of the line derived above rather than the equation of a quadratic polynomial.

There are five parameters that determine the theoretical error, that is the probe radius, R_p, the radius of curvature of the surface in the direction perpendicular to the edge, R, the fillet radius between the surfaces or the gap, R_f, the intersection angle of the two surfaces, θ, and the length of the scan line on the surface, s.

In this section three graphs are presented that show how changes to these parameters influence the theoretical error. For a discussion of these results, please refer back to the main body of this thesis.

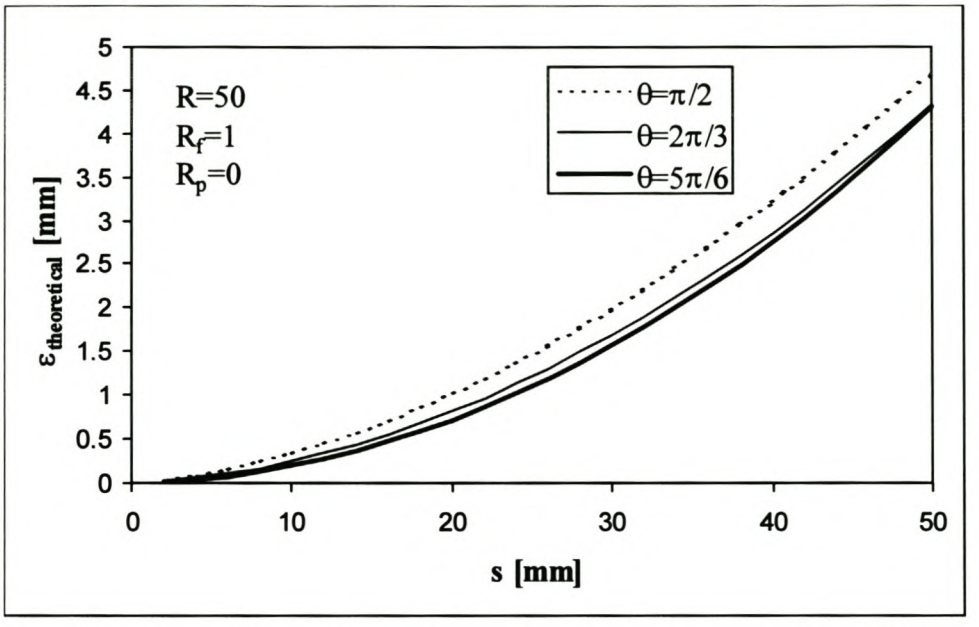


Figure 10 Influence of Intersection Angle on Theoretical Error.

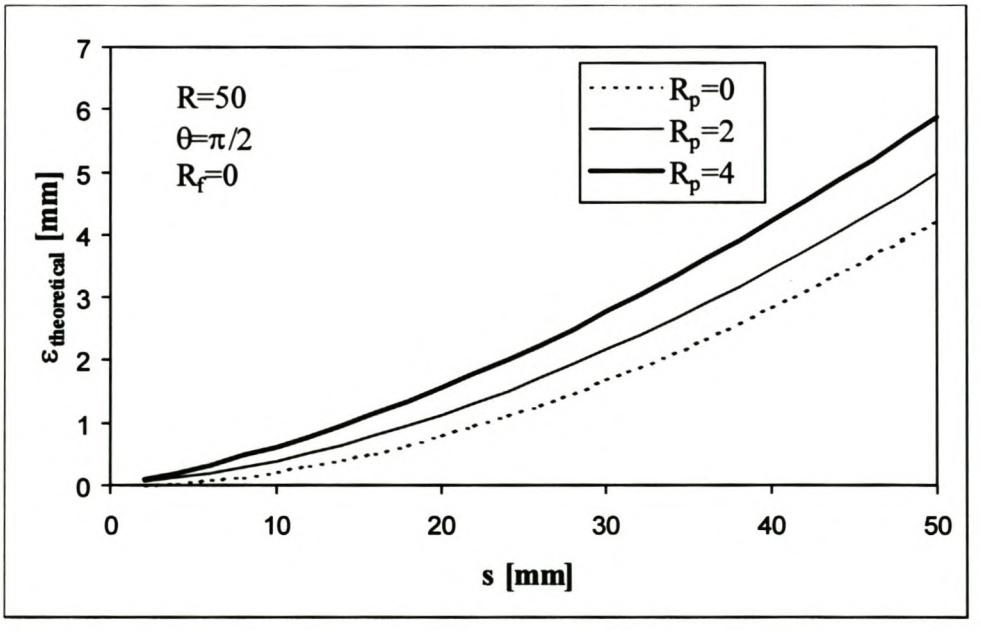


Figure 11 Influence of the Probe Radius on the Theoretical Error.

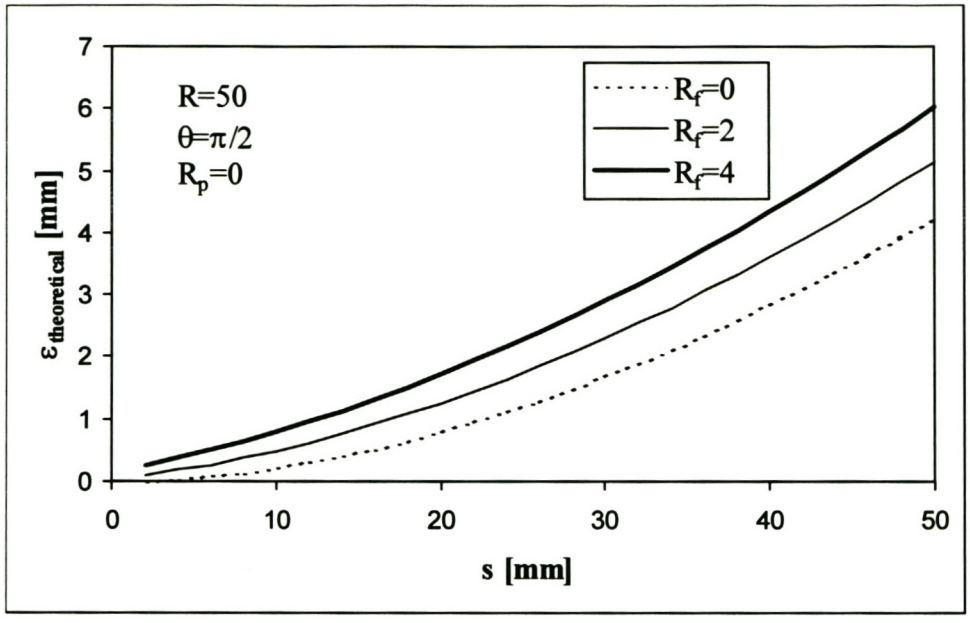


Figure 12 Influence of the Fillet Radius on the Theoretical Error.

Appendix E

Design of Experiments Results

This Appendix contains the results of an experimental investigation into the performance of the edge scanning algorithm. A level 16 experiment was chosen from the Statistica (2000) software package. This is shown in Table 1. Eleven parameters were identified that influence the algorithm's performance. They are given in the same table.

Table 1 Level 16 Experiment.

	Surface Curvature	Ampli-tude	Noise	Line Pitch	Edge Pitch	Edge Curvature	Intersec-tion Angle	Toler-ance	Fillet Radius	Probe Radius	Cloud Pitch
Test	[mm^{-1}]	[mm]	[mm]	[mm]	[mm]	[mm^{-1}]	[radians]	[mm]	[mm]	[mm]	[mm]
1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1
2	1	-1	-1	-1	1	-1	1	1	-1	-1	-1
3	-1	1	-1	-1	1	1	-1	1	-1	-1	1
4	1	1	-1	-1	-1	1	1	-1	1	1	-1
5	-1	-1	1	-1	1	1	1	-1	-1	1	-1
6	1	-1	1	-1	-1	1	-1	1	1	-1	1
7	-1	1	1	-1	-1	-1	1	1	1	-1	-1
8	1	1	1	-1	1	-1	-1	-1	-1	1	1
9	-1	-1	-1	1	-1	1	1	1	-1	1	1
10	1	-1	-1	1	1	1	-1	-1	1	-1	-1
11	-1	1	-1	1	1	-1	1	-1	1	-1	1
12	1	1	-1	1	-1	-1	-1	1	-1	1	-1
13	-1	-1	1	1	1	-1	-1	1	1	1	-1
14	1	-1	1	1	-1	-1	1	-1	-1	-1	1
15	-1	1	1	1	-1	1	-1	-1	-1	-1	-1
16	1	1	1	1	1	1	1	1	1	1	1

Experiment 1

Input Parameters

	Surface Curvature	Ampli-tude	Noise	Line Pitch	Edge Pitch	Edge Curvature	Intersec-tion Angle	Toler-ance	Fillet Radius	Probe Radius	Cloud Pitch
Low	0.1	4	0.01	0.4	0.6	0.029	1.745	0.15	0	0.25	0.1
High	0.125	6	0.02	0.6	0.8	0.033	1.920	0.2	0.3	0.5	0.2

Experimental Results - Zigzag

	Linear			Quadratic		
Test	Average Error	Maximum Error	Number of Points	Average Error	Maximum Error	Number of Points
1	0.041	0.072	162	0.027	0.082	160
2	0.060	0.091	123	0.017	0.034	122
3	0.098	0.142	104	0.015	0.039	103
4	0.107	0.138	142	0.012	0.032	142
5	0.032	0.065	105	0.021	0.049	103
6	0.054	0.088	141	0.024	0.056	136
7	0.096	0.126	170	0.018	0.043	167
8	0.085	0.121	124	0.020	0.039	120
9	0.060	0.129	143	0.023	0.111	142
10	0.091	0.143	102	0.028	0.091	99
11	0.136	0.202	125	0.018	0.058	125
12	0.136	0.210	159	0.011	0.037	162
13	0.060	0.104	119	0.030	0.085	113
14	0.075	0.136	165	0.043	0.154	160
15	0.119	0.177	138	0.018	0.037	137
16	0.134	0.212	107	0.023	0.067	103

Experimental Results - Square

	Linear			Quadratic		
Test	Average Error	Maximum Error	Number of Points	Average Error	Maximum Error	Number of Points
1	0.041	0.051	166	0.019	0.068	165
2	0.059	0.086	126	0.016	0.034	126
3	0.103	0.163	106	0.014	0.028	106
4	0.103	0.134	144	0.012	0.035	145
5	0.030	0.074	107	0.020	0.041	106
6	0.058	0.094	142	0.017	0.046	142

7	0.095	0.121	171	0.018	0.043	172
8	0.083	0.123	125	0.020	0.044	126
9	0.058	0.136	144	0.022	0.080	144
10	0.087	0.141	106	0.023	0.060	107
11	0.135	0.224	123	0.016	0.040	124
12	0.162	0.888	Error	0.011	0.026	170
13	0.046	0.078	126	0.025	0.087	125
14	0.079	0.163	170	0.043	0.205	168
15	0.118	0.162	142	0.023	0.056	145
16	0.125	0.174	104	0.024	0.050	105

Design of Experiments Sensitivity Analysis

	Zigzag				Square			
	Linear		Quadratic		Linear		Quadratic	
	Average Error	Maximum Error	Average Error	Maximum Error	Average Error	Maximum Error	Average Error	Maximum Error
Surface Curvature	0.125	0.153	0.011	0.007	0.163	1.003	0.011	0.075
Amplitude	0.553	0.634	0.097	0.392	0.593	1.473	0.061	0.379
Noise	0.096	0.124	0.061	0.058	0.142	1.058	0.071	0.255
Line Pitch	0.300	0.592	0.048	0.337	0.301	1.419	0.066	0.335
Edge Pitch	0.009	0.003	0.004	0.112	0.060	0.865	0.010	0.220
Edge Curvature	0.004	0.044	0.023	0.062	0.023	0.830	0.016	0.189
Intersection Angle	0.022	0.054	0.002	0.101	0.017	0.744	0.023	0.144
Tolerance	0.015	0.060	0.032	0.088	0.039	0.845	0.035	0.196
Fillet Radius	0.069	0.018	0.016	0.016	0.002	0.985	0.019	0.108
Ball Radius	0.095	0.066	0.017	0.011	0.109	0.639	0.022	0.101
Cloud Pitch	0.041	0.072	0.049	0.250	0.024	0.704	0.038	0.228

Experiment 2

Input Parameters

	Surface Curvature	Ampli-tude	Noise	Line Pitch	Edge Pitch	Edge Curvature	Intersec-tion Angle	Toler-ance	Fillet Radius	Probe Radius	Cloud Pitch
Low	0.067	4	0.01	0.4	0.6	0.029	1.745	0.05	0.5	0.25	0.1
High	0.100	6	0.02	0.6	0.8	0.033	1.920	0.1	1	0.5	0.25

Experimental Results - Zigzag

	Linear			Quadratic		
Test	Average Error	Maximum Error	Number of Points	Average Error	Maximum Error	Number of Points
1	0.103	0.117	163	0.224	0.401	Error
2	0.052	0.077	123	0.025	0.061	123
3	0.067	0.077	102	0.031	0.045	100
4	0.119	0.147	143	0.064	0.089	138
5	0.036	0.067	103	0.040	0.137	103
6	0.128	0.170	138	0.171	0.301	Error
7	0.092	0.108	167	0.063	0.171	163
8	0.132	0.264	114	0.040	0.072	123
9	0.035	0.066	140	0.038	0.378	138
10	0.112	0.138	100	0.091	0.170	98
11	0.099	0.219	119	0.055	0.109	123
12	0.113	0.157	166	0.014	0.058	161
13	0.111	0.226	117	0.155	0.345	118
14	0.072	0.225	159	0.051	0.510	158
15	0.089	0.121	137	0.021	0.079	134
16	0.124	0.207	103	0.085	0.171	103

Experimental Results - Square

	Linear			Quadratic		
Test	Average Error	Maximum Error	Number of Points	Average Error	Maximum Error	Number of Points
1	0.103	0.117	164	0.236	0.402	170
2	0.053	0.073	126	0.019	0.050	126
3	0.070	0.083	103	0.030	0.081	103
4	0.116	0.139	146	0.067	0.116	142
5	0.030	0.054	107	0.034	0.089	107
6	0.125	0.178	141	0.177	0.319	141

7	0.090	0.104	172	0.059	0.166	169
8	0.124	0.222	122	0.038	0.089	125
9	0.036	0.059	138	0.034	0.104	142
10	0.102	0.135	106	0.080	0.167	105
11	0.094	0.142	123	0.054	0.113	122
12	0.113	0.151	170	0.017	0.065	172
13	0.095	0.144	125	0.134	0.290	124
14	0.065	0.189	Error	0.044	0.321	160
15	0.085	0.130	145	0.022	0.053	147
16	0.118	0.143	103	0.065	0.187	104

Design of Experiments Sensitivity Analysis

	Zigzag				Square			
	Linear		Quadratic		Linear		Quadratic	
	Average Error	Maximum Error	Average Error	Maximum Error	Average Error	Maximum Error	Average Error	Maximum Error
Surface Curvature	0.279	0.485	0.107	0.293	0.268	0.503	0.122	0.019
Amplitude	0.237	0.271	0.534	1.911	0.253	0.211	0.514	1.101
Noise	0.105	0.492	0.107	0.603	0.058	0.333	0.046	0.525
Line Pitch	0.034	0.422	0.186	0.687	0.002	0.156	0.265	0.016
Edge Pitch	0.022	0.207	0.158	1.110	0.060	0.089	0.256	0.606
Edge Curvature	0.081	0.505	0.109	0.456	0.071	0.278	0.117	0.480
Intersection Angle	0.287	0.196	0.412	0.194	0.272	0.325	0.451	0.405
Tolerance	0.049	0.265	0.002	0.047	0.022	0.242	0.049	0.112
Fillet Radius	0.367	0.349	0.821	0.528	0.337	0.181	0.800	1.149
Ball Radius	0.079	0.147	0.190	0.259	0.067	0.005	0.175	0.093
Cloud Pitch	0.103	0.117	0.278	1.111	0.066	0.257	0.310	0.786

Experiment 3

Input Parameters

	Surface Curvature	Ampli-tude	Noise	Line Pitch	Edge Pitch	Edge Curvature	Intersec-tion Angle	Toler-ance	Fillet Radius	Probe Radius	Cloud Pitch
Low	0.050	10	0.02	0.5	2	0.007	1.571	0.2	1	1	0.25
High	0.025	20	0.04	1	4	0.010	2.094	0.3	2	2	0.36

Experimental Results - Zigzag

	Linear			Quadratic		
Test	Average Error	Maximum Error	Number of Points	Average Error	Maximum Error	Number of Points
1	0.327	0.370	215	0.460	1.179	Error
2	0.069	0.111	105	0.058	0.158	106
3	0.762	1.168	67	0.110	0.181	67
4	0.221	0.253	146	0.122	0.169	145
5	0.102	0.141	67	0.065	0.156	68
6	0.367	0.427	136	0.581	0.799	145
7	0.766	1.252	176	0.075	0.172	219
8	0.221	0.319	105	0.120	0.172	104
9	0.099	0.143	146	0.058	0.153	146
10	0.353	0.498	64	0.490	0.847	67
11	1.230	2.008	109	0.051	0.120	107
12	0.259	0.277	218	0.171	0.245	214
13	0.317	0.414	105	0.395	0.685	108
14	0.062	0.113	220	0.053	0.160	223
15	1.028	1.187	141	0.051	0.114	141
16	0.224	0.277	67	0.088	0.189	67

Experimental Results - Square

	Linear			Quadratic		
Test	Average Error	Maximum Error	Number of Points	Average Error	Maximum Error	Number of Points
1	0.331	0.375	222	0.441	0.560	223
2	0.067	0.105	106	0.043	0.152	106
3	0.693	0.938	67	0.111	0.190	67
4	0.219	0.239	146	0.124	0.176	146
5	0.102	0.177	68	0.068	0.170	67
6	0.385	0.475	146	0.574	0.770	146

7	0.822	1.082	Error	0.075	0.140	224
8	0.211	0.260	108	0.129	0.176	107
9	0.096	0.131	145	0.059	0.135	146
10	0.263	0.401	67	0.417	0.627	67
11	2.203	4.364	Error	0.052	0.112	106
12	0.257	0.279	225	0.167	0.242	224
13	0.282	0.353	108	0.342	0.475	107
14	0.063	0.115	222	0.049	0.160	224
15	1.083	1.252	Error	0.047	0.113	146
16	0.210	0.249	68	0.102	0.205	68

Design of Experiments Sensitivity Analysis

	Zigzag				Square			
	Linear		Quadratic		Linear		Quadratic	
	Average Error	Maximum Error	Average Error	Maximum Error	Average Error	Maximum Error	Average Error	Maximum Error
Surface Curvature	3.613	5.578	0.532	0.024	4.982	8.282	0.518	0.776
Amplitude	3.813	5.723	1.734	3.511	5.198	8.266	1.503	2.144
Noise	0.295	0.884	0.119	0.764	1.227	3.630	0.037	0.020
Line Pitch	0.934	1.104	0.296	0.596	2.060	4.418	0.417	0.334
Edge Pitch	0.186	1.155	0.244	0.612	0.981	3.668	0.344	0.239
Edge Curvature	0.119	0.974	0.231	0.356	1.498	3.884	0.258	0.470
Intersection Angle	1.090	0.458	2.288	3.725	0.349	2.692	2.095	2.409
Tolerance	0.860	1.038	0.157	0.424	2.104	4.517	0.185	0.272
Fillet Radius	1.523	2.581	1.995	3.569	2.713	5.416	1.839	2.185
Ball Radius	3.626	5.779	0.010	0.502	4.897	8.437	0.080	0.159
Cloud Pitch	0.327	0.370	0.119	0.514	1.389	3.818	0.298	0.267

Experiment 4

Input Parameters

	Surface Curvature	Ampli-tude	Noise	Line Pitch	Edge Pitch	Edge Curvature	Intersec-tion Angle	Toler-ance	Fillet Radius	Probe Radius	Cloud Pitch
Low	0.050	20	0.05	1	4	0.007	1.920	0.2	2	1	0.36
High	0.025	30	0.1	2	5	0.010	2.094	0.3	3	2	0.7

Experimental Results - Zigzag

	Linear			Quadratic		
Test	Average Error	Maximum Error	Number of Points	Average Error	Maximum Error	Number of Points
1	1.312	2.639	103	0.159	0.209	106
2	0.245	0.297	83	0.079	0.191	83
3	3.679	5.734	Error	0.124	0.268	52
4	1.361	1.649	67	0.086	0.226	68
5	1.887	4.546	50	0.144	0.342	53
6	0.293	0.402	66	0.210	0.485	66
7	7.194	10.073	Error	0.096	0.246	106
8	1.307	1.847	83	0.096	0.200	82
9	1.293	3.645	64	0.131	0.379	66
10	0.362	0.465	51	0.133	0.274	51
11	8.933	17.014	Error	0.244	1.751	Error
12	0.899	1.288	107	0.059	0.153	104
13	0.733	1.444	82	0.137	0.391	82
14	0.371	1.053	106	0.122	0.363	105
15	8.641	15.335	Error	0.101	0.216	67
16	1.205	2.483	51	0.110	0.299	52

Experimental Results - Square

	Linear			Quadratic		
Test	Average Error	Maximum Error	Number of Points	Average Error	Maximum Error	Number of Points
1	1.598	2.284	Error	0.177	0.312	104
2	0.228	0.284	83	0.069	0.181	83
3	14.291	26.554	Error	0.111	0.196	51
4	1.385	4.081	Error	0.103	0.184	69
5	2.278	3.986	Error	0.133	0.314	53
6	0.294	0.378	68	0.232	0.429	67

7	4.407	5.941	Error	0.078	0.257	107
8	1.230	1.684	82	0.081	0.167	82
9	2.281	4.424	Error	0.099	0.289	66
10	0.336	0.487	52	0.083	0.194	53
11	12.465	20.578	Error	0.232	1.894	Error
12	0.918	3.666	Error	0.060	0.159	107
13	1.445	5.183	Error	0.108	0.312	83
14	0.332	0.629	105	0.115	0.323	106
15	3.663	4.050	Error	0.105	0.250	68
16	1.561	3.705	Error	0.177	0.403	52

Design of Experiments Sensitivity Analysis

	Zigzag				Square			
	Linear		Quadratic		Linear		Quadratic	
	Average Error	Maximum Error	Average Error	Maximum Error	Average Error	Maximum Error	Average Error	Maximum Error
Surface Curvature	34.949	64.440	0.306	2.038	45.718	73.472	0.156	2.256
Amplitude	33.802	51.774	0.254	0.919	39.372	66.539	0.087	1.464
Noise	4.487	5.629	0.001	1.149	23.139	46.550	0.120	1.208
Line Pitch	6.528	19.655	0.054	2.099	3.427	3.123	0.008	2.259
Edge Pitch	3.811	2.853	0.131	1.823	23.978	46.814	0.034	1.841
Edge Curvature	2.874	1.764	0.061	1.286	4.384	9.381	0.157	1.700
Intersection Angle	6.656	14.682	0.009	2.025	1.471	0.833	0.064	2.309
Tolerance	10.921	24.263	0.175	1.480	2.703	15.629	0.119	1.785
Fillet Radius	3.884	3.064	0.402	2.237	2.189	3.341	0.527	2.664
Ball Radius	24.945	39.000	0.238	2.019	29.499	37.806	0.108	2.004
Cloud Pitch	1.312	2.639	0.456	2.422	24.527	41.182	0.612	2.732

Experiment 5

Input Parameters

	Surface Curvature	Ampli-tude	Noise	Line Pitch	Edge Pitch	Edge Curvature	Intersec-tion Angle	Toler-ance	Fillet Radius	Probe Radius	Cloud Pitch
Low	0.025	20	0.05	1.5	4	0.007	2.094	0.1	3	1	0.7
High	0.013	30	0.1	2	5	0.010	2.269	0.2	4	2	1

Experimental Results - Zigzag

	Linear			Quadratic		
Test	Average Error	Maximum Error	Number of Points	Average Error	Maximum Error	Number of Points
1	0.742	8.228	99	0.306	4.952	102
2	0.162	0.394	83	0.106	0.290	83
3	1.545	2.396	48	0.172	0.357	51
4	0.791	1.522	63	0.120	0.715	65
5	0.837	1.743	51	0.474	7.436	50
6	0.238	1.366	65	0.756	12.816	Error
7	1.518	3.367	98	0.098	0.287	105
8	0.852	2.074	79	0.439	4.669	Error
9	0.603	2.174	Error	0.158	0.569	66
10	0.251	0.476	52	0.175	0.484	51
11	4.722	15.202	Error	7.024	11.198	Error
12	0.319	0.600	104	0.102	0.294	105
13	0.337	0.721	82	0.226	0.656	81
14	0.500	4.654	101	0.682	7.955	Error
15	2.631	38.112	56	2.816	17.104	Error
16	0.499	1.561	49	0.163	0.297	52

Experimental Results - Square

	Linear			Quadratic		
Test	Average Error	Maximum Error	Number of Points	Average Error	Maximum Error	Number of Points
1	1.598	19.233	Error	0.191	0.732	Error
2	0.148	0.249	83	0.098	0.300	82
3	1.573	3.928	50	0.164	0.247	51
4	0.631	1.039	66	0.104	0.271	67
5	0.978	2.730	49	0.515	8.896	Error
6	0.202	0.477	66	0.306	0.742	66

7	2.562	5.724	Error	0.104	0.308	107
8	0.736	1.890	82	0.235	1.891	78
9	0.548	2.262	64	0.228	2.102	63
10	0.210	0.393	52	0.126	0.373	52
11	9.778	42.882	Error	3.744	8.208	Error
12	0.286	0.490	106	0.114	0.255	106
13	0.306	0.499	83	0.197	0.639	83
14	0.330	1.333	102	0.640	4.653	Error
15	4.498	8.569	Error	0.166	0.702	Error
16	0.399	1.567	48	0.147	0.416	52

Design of Experimehts Sensitivity Analysis

	Zigzag				Square			
	Linear		Quadratic		Linear		Quadratic	
	Average Error	Maximum Error	Average Error	Maximum Error	Average Error	Maximum Error	Average Error	Maximum Error
Surface Curvature	11.792	75.006	11.043	19.022	23.906	99.153	4.477	16.360
Amplitude	11.648	57.018	10.183	0.300	20.421	49.220	3.135	7.766
Noise	2.178	28.596	3.173	40.933	6.022	60.319	3.112	7.284
Line Pitch	4.020	53.644	11.225	8.899	10.026	28.747	4.611	5.010
Edge Pitch	2.358	44.849	4.732	24.418	4.391	18.987	4.265	14.172
Edge Curvature	2.224	17.851	5.249	11.985	8.480	64.932	4.511	4.095
Intersection Angle	3.437	29.546	4.849	15.918	7.544	28.216	5.160	24.756
Tolerance	7.725	75.176	12.973	49.264	16.106	79.527	5.520	26.206
Fillet Radius	2.086	24.924	4.958	9.196	8.336	63.705	3.492	9.304
Ball Radius	8.333	59.885	12.446	39.089	17.480	42.811	4.578	0.418
Cloud Pitch	0.742	8.228	7.060	19.666	7.013	68.152	5.352	9.166

Experiment 6

Input Parameters

	Surface Curvature	Ampli-tude	Noise	Line Pitch	Edge Pitch	Edge Curvature	Intersec-tion Angle	Toler-ance	Fillet Radius	Probe Radius	Cloud Pitch
Low	0.010	20	0.05	1	4	0.004	1.920	0.15	2	1	0.2
High	0.013	30	0.1	2	5	0.007	2.094	0.2	3	2	0.7

Experimental Results - Zigzag

	Linear			Quadratic		
Test	Average Error	Maximum Error	Number of Points	Average Error	Maximum Error	Number of Points
1	0.154	0.197	178	0.222	0.356	178
2	0.129	0.174	144	0.071	0.221	144
3	0.247	0.597	81	0.083	0.199	81
4	0.422	0.573	106	0.068	0.154	106
5	0.093	0.176	82	0.089	0.273	82
6	0.178	0.220	104	0.204	0.387	104
7	0.215	0.301	184	0.071	0.176	183
8	0.511	1.777	144	0.071	0.190	143
9	0.105	0.265	105	0.107	0.264	105
10	0.217	0.267	80	0.154	0.318	80
11	0.330	0.577	144	0.095	0.237	144
12	0.302	0.402	181	0.063	0.170	179
13	0.199	0.316	140	0.236	0.500	140
14	0.155	0.497	182	0.146	1.093	178
15	0.325	0.552	104	0.094	0.208	103
16	0.346	0.593	82	0.112	0.314	82

Experimental Results - Square

	Linear			Quadratic		
Test	Average Error	Maximum Error	Number of Points	Average Error	Maximum Error	Number of Points
1	0.155	0.182	182	0.249	0.350	183
2	0.116	0.158	146	0.065	0.206	146
3	0.210	0.249	83	0.082	0.219	82
4	0.386	0.508	107	0.070	0.133	107
5	0.092	0.203	83	0.104	0.228	83
6	0.172	0.255	106	0.232	0.433	107

7	0.211	0.308	186	0.085	0.208	186
8	0.421	0.723	145	0.061	0.162	146
9	0.102	0.152	106	0.078	0.205	105
10	0.171	0.228	83	0.138	0.304	83
11	0.276	0.424	144	0.078	0.229	144
12	0.293	0.363	186	0.076	0.200	186
13	0.133	0.250	146	0.224	0.502	146
14	0.141	0.307	184	0.129	0.517	183
15	0.307	0.555	106	0.116	0.263	107
16	0.299	0.564	83	0.117	0.269	83

Design of Experiments Sensitivity Analysis

	Zigzag				Square			
	Linear		Quadratic		Linear		Quadratic	
	Average Error	Maximum Error	Average Error	Maximum Error	Average Error	Maximum Error	Average Error	Maximum Error
Surface Curvature	0.749	1.925	0.137	0.802	0.649	0.990	0.161	0.026
Amplitude	1.857	4.123	0.727	2.232	1.671	2.478	0.677	1.341
Noise	0.149	1.746	0.201	1.544	0.087	1.141	0.293	0.930
Line Pitch	0.040	0.691	0.162	1.450	0.053	0.329	0.009	0.697
Edge Pitch	0.271	1.859	0.082	0.703	0.061	0.215	0.213	0.239
Edge Curvature	0.080	1.264	0.079	1.044	0.007	0.000	0.039	0.405
Intersection Angle	0.430	1.484	0.467	0.510	0.302	0.231	0.573	0.555
Tolerance	0.615	2.213	0.010	0.755	0.521	1.052	0.020	0.069
Fillet Radius	0.244	1.765	0.554	0.224	0.154	0.011	0.610	0.542
Ball Radius	0.425	1.409	0.063	0.781	0.347	0.581	0.071	0.419
Cloud Pitch	0.154	0.197	0.242	1.289	0.085	0.359	0.187	0.431

Experiment 7

Input Parameters

	Surface Curvature	Ampli-tude	Noise	Line Pitch	Edge Pitch	Edge Curvature	Intersec-tion Angle	Toler-ance	Fillet Radius	Probe Radius	Cloud Pitch
Low	0.010	30	0.05	2	4	0.004	1.920	0.15	2	1	1
High	0.013	40	0.1	4	5	0.007	2.094	0.2	3	2	1.4

Experimental Results - Zigzag

	Linear			Quadratic		
Test	Average Error	Maximum Error	Number of Points	Average Error	Maximum Error	Number of Points
1	0.380	4.631	Error	0.204	4.016	172
2	0.428	0.794	145	0.080	0.185	145
3	0.877	1.190	78	0.100	0.378	80
4	1.325	1.866	94	0.086	0.255	102
5	0.441	1.257	80	0.243	6.349	79
6	1.067	20.404	Error	0.347	6.404	92
7	0.768	1.169	170	0.085	0.241	182
8	1.630	17.255	Error	0.985	15.873	Error
9	0.411	1.179	99	0.223	2.001	101
10	0.840	2.824	82	0.172	0.471	79
11	1.100	1.939	130	0.188	0.786	140
12	1.165	1.694	172	0.092	0.353	177
13	0.396	0.855	139	0.231	1.385	138
14	0.757	2.690	169	0.410	2.537	175
15	1.161	5.845	96	0.145	0.635	101
16	1.263	1.994	74	0.347	9.457	78

Experimental Results - Square

	Linear			Quadratic		
Test	Average Error	Maximum Error	Number of Points	Average Error	Maximum Error	Number of Points
1	0.344	1.514	174	0.160	0.612	175
2	0.396	0.713	143	0.081	0.204	143
3	0.796	1.216	80	0.070	0.142	78
4	1.279	1.753	105	0.077	0.191	104
5	0.302	0.594	82	0.114	0.272	82
6	0.402	0.831	104	0.186	0.618	101

7	0.772	1.088	179	0.084	0.226	184
8	1.291	2.171	141	0.138	0.860	143
9	0.444	0.914	98	0.197	1.124	99
10	0.633	1.239	83	0.136	0.367	83
11	1.186	7.224	136	0.126	1.155	140
12	1.148	1.570	179	0.078	0.208	184
13	0.376	0.749	143	0.153	0.643	144
14	0.752	3.183	169	0.985	19.813	Error
15	1.014	1.839	103	0.132	0.332	106
16	1.265	2.307	75	0.144	0.328	81

Design of Experiments Sensitivity Analysis

	Zigzag				Square			
	Linear		Quadratic		Linear		Quadratic	
	Average Error	Maximum Error	Average Error	Maximum Error	Average Error	Maximum Error	Average Error	Maximum Error
Surface Curvature	3.720	39.788	1.391	24.975	2.442	1.736	1.001	22.875
Amplitude	5.779	2.126	0.145	5.856	6.455	11.930	1.470	25.564
Noise	1.207	44.716	2.085	43.559	0.066	4.278	1.280	24.146
Line Pitch	0.227	37.374	0.408	20.335	1.564	11.569	1.316	26.369
Edge Pitch	0.075	14.381	0.955	23.330	0.114	4.452	1.186	24.225
Edge Curvature	0.965	6.997	0.773	0.726	0.163	9.510	0.946	25.737
Intersection Angle	1.294	52.884	0.779	9.742	0.496	8.407	0.953	24.705
Tolerance	1.593	11.420	1.173	13.302	1.521	12.814	1.109	25.437
Fillet Radius	0.341	4.777	0.781	6.700	0.145	5.697	0.920	23.797
Ball Radius	0.014	7.746	1.118	35.482	0.631	7.288	0.935	23.550
Cloud Pitch	0.380	4.631	2.114	39.946	0.709	12.413	1.456	28.094

Experiment 8

Input Parameters

	Surface Curvature	Ampli-tude	Noise	Line Pitch	Edge Pitch	Edge Curvature	Intersec-tion Angle	Toler-ance	Fillet Radius	Probe Radius	Cloud Pitch
Low	0.007	30	0.1	2	4	0.004	2.094	0.25	2	1	0.5
High	0.010	40	0.2	4	5	0.007	2.443	0.3	4	2	1

Experimental Results - Zigzag

	Linear			Quadratic		
Test	Average Error	Maximum Error	Number of Points	Average Error	Maximum Error	Number of Points
1	0.184	0.282	182	0.180	0.468	181
2	0.268	0.575	145	0.106	0.279	145
3	0.256	0.350	82	0.125	0.210	82
4	0.577	1.030	99	0.102	0.305	104
5	0.164	0.416	83	0.187	0.326	83
6	0.217	0.363	106	0.182	0.570	106
7	0.237	0.596	182	0.168	0.457	182
8	0.699	2.291	141	0.177	1.237	143
9	0.280	0.684	102	0.275	0.781	104
10	0.474	0.795	82	0.169	0.478	80
11	0.412	0.806	143	0.192	0.941	145
12	0.603	0.991	181	0.114	0.331	177
13	0.256	0.618	140	0.309	0.986	137
14	0.589	13.847	166	0.457	3.643	175
15	0.371	0.746	104	0.242	0.586	102
16	0.775	1.791	77	0.276	0.679	79

Experimental Results - Square

	Linear			Quadratic		
Test	Average Error	Maximum Error	Number of Points	Average Error	Maximum Error	Number of Points
1	0.174	0.267	185	0.199	0.529	185
2	0.251	0.578	145	0.117	0.275	146
3	0.219	0.302	83	0.128	0.211	82
4	1.256	5.538	Error	0.109	0.382	107
5	0.169	0.665	82	0.208	0.464	83
6	0.212	0.370	106	0.200	0.788	107

7	0.232	0.695	185	0.172	0.446	186
8	0.590	1.494	144	0.150	0.394	146
9	0.246	0.600	102	0.243	0.729	105
10	0.394	0.624	84	0.175	0.557	85
11	0.890	6.358	Error	0.184	0.689	143
12	0.578	0.863	183	0.117	0.284	184
13	0.209	0.455	146	0.296	0.794	145
14	0.600	2.091	161	0.562	4.260	167
15	0.322	0.473	Error	0.212	0.516	106
16	0.786	1.883	77	0.292	0.953	83

Design of Experiments Sensitivity Analysis

	Zigzag				Square			
	Linear		Quadratic		Linear		Quadratic	
	Average Error	Maximum Error	Average Error	Maximum Error	Average Error	Maximum Error	Average Error	Maximum Error
Surface Curvature	2.582	21.736	0.118	3.499	2.790	4.587	0.102	4.443
Amplitude	1.895	11.356	0.596	3.523	3.311	15.124	0.804	5.718
Noise	0.319	19.167	0.931	5.931	1.123	8.860	1.038	6.272
Line Pitch	1.465	18.181	1.023	5.788	1.166	4.350	1.010	6.695
Edge Pitch	0.309	13.782	0.228	2.537	0.145	1.849	0.331	4.550
Edge Curvature	0.169	17.495	0.185	5.575	0.102	2.968	0.295	3.885
Intersection Angle	0.307	16.835	0.332	3.220	2.191	17.151	0.520	5.219
Tolerance	0.732	18.022	0.190	4.668	2.103	14.881	0.297	4.187
Fillet Radius	0.124	17.226	0.133	3.173	1.490	11.542	0.140	2.525
Ball Radius	0.905	12.618	0.028	2.593	1.123	0.343	0.172	4.066
Cloud Pitch	0.184	0.282	0.590	6.048	0.384	4.393	0.698	6.115

Experiment 9

Input Parameters

	Surface Curvature	Ampli-tude	Noise	Line Pitch	Edge Pitch	Edge Curvature	Intersec-tion Angle	Toler-ance	Fillet Radius	Probe Radius	Cloud Pitch
Low	0.007	30	0.1	2	4	0.004	2.094	0.25	2	1	0.5
High	0.010	35	0.2	3	5	0.007	2.443	0.3	4	2	1

Experimental Results - Zigzag

	Linear			Quadratic		
Test	Average Error	Maximum Error	Number of Points	Average Error	Maximum Error	Number of Points
1	0.184	0.282	182	0.180	0.468	181
2	0.268	0.575	145	0.106	0.279	145
3	0.199	0.338	82	0.132	0.313	82
4	0.326	0.587	104	0.103	0.236	105
5	0.164	0.416	83	0.187	0.326	83
6	0.217	0.363	106	0.182	0.570	106
7	0.190	0.453	185	0.181	0.572	183
8	0.377	1.044	141	0.170	0.801	142
9	0.222	0.546	101	0.212	1.892	101
10	0.346	0.553	82	0.122	0.313	81
11	0.269	0.829	143	0.227	14.451	Error
12	0.377	0.589	182	0.103	0.297	180
13	0.166	0.314	141	0.212	0.620	140
14	0.365	0.898	171	0.699	4.453	Error
15	0.250	0.579	103	0.203	0.728	103
16	0.450	4.546	Error	0.243	0.568	82

Experimental Results - Square

	Linear			Quadratic		
Test	Average Error	Maximum Error	Number of Points	Average Error	Maximum Error	Number of Points
1	0.174	0.267	185	0.199	0.529	185
2	0.251	0.578	145	0.117	0.275	146
3	0.184	0.298	83	0.133	0.269	82
4	0.307	0.465	106	0.102	0.424	107
5	0.169	0.665	82	0.208	0.464	83
6	0.212	0.370	106	0.200	0.788	107

7	0.188	0.594	184	0.174	0.421	185
8	0.319	0.777	144	0.151	0.353	145
9	0.243	0.667	106	0.165	0.534	105
10	0.295	0.438	84	0.137	0.375	84
11	0.268	0.548	144	0.144	0.480	146
12	0.358	0.506	184	0.099	0.255	185
13	0.144	0.397	145	0.226	0.693	145
14	0.424	2.232	172	0.446	7.057	Error
15	0.231	0.635	106	0.205	0.483	106
16	0.450	1.523	81	0.264	0.690	83

Design of Experiments Sensitivity Analysis

	Zigzag				Square			
	Linear		Quadratic		Linear		Quadratic	
	Average Error	Maximum Error	Average Error	Maximum Error	Average Error	Maximum Error	Average Error	Maximum Error
Surface Curvature	1.366	6.827	0.247	14.994	1.284	3.566	0.078	8.024
Amplitude	0.639	6.347	0.681	11.442	0.496	0.341	0.540	9.287
Noise	0.014	5.456	1.130	12.157	0.073	4.332	0.983	9.875
Line Pitch	0.657	6.068	0.986	24.989	0.770	3.708	0.509	8.909
Edge Pitch	0.136	5.462	0.590	10.694	0.074	0.648	0.262	8.718
Edge Curvature	0.029	3.722	0.626	21.499	0.043	1.058	0.180	7.634
Intersection Angle	0.176	6.056	0.828	23.612	0.484	4.533	0.341	8.350
Tolerance	0.241	3.205	0.655	21.080	0.200	1.383	0.272	7.892
Fillet Radius	0.092	3.721	0.457	11.019	0.179	2.219	0.100	6.694
Ball Radius	0.206	4.729	0.559	20.833	0.140	0.540	0.181	7.850
Cloud Pitch	0.184	0.282	1.049	25.480	0.417	3.040	0.549	9.247

Experiment 10

Input Parameters

	Surface Curvature	Ampli-tude	Noise	Line Pitch	Edge Pitch	Edge Curvature	Intersec-tion Angle	Toler-ance	Fillet Radius	Probe Radius	Cloud Pitch
Low	0.004	40	0.1	2	4	0.003	2.094	0.25	2	1	0.5
High	0.007	50	0.2	3	5	0.004	2.443	0.3	4	2.5	1

Experimental Results - Zigzag

	Linear			Quadratic		
Test	Average Error	Maximum Error	Number of Points	Average Error	Maximum Error	Number of Points
1	0.167	0.221	259	0.135	0.311	258
2	0.273	0.413	208	0.095	0.237	207
3	0.213	0.330	143	0.106	0.195	143
4	0.582	0.837	172	0.089	0.298	180
5	0.148	0.437	144	0.182	0.723	145
6	0.245	0.541	182	0.171	0.462	180
7	0.206	0.533	257	0.167	0.400	261
8	0.682	1.309	194	0.434	13.970	Error
9	0.166	0.417	180	0.135	0.547	182
10	0.335	0.467	143	0.109	0.281	142
11	0.264	0.500	205	0.112	0.263	205
12	0.441	0.627	257	0.081	0.182	256
13	0.136	0.374	203	0.190	0.573	201
14	0.479	7.649	240	1.732	16.385	Error
15	0.283	0.744	176	0.179	0.484	177
16	0.635	9.490	137	0.207	0.552	140

Experimental Results - Square

	Linear			Quadratic		
Test	Average Error	Maximum Error	Number of Points	Average Error	Maximum Error	Number of Points
1	0.169	0.256	265	0.145	0.377	263
2	0.259	0.439	208	0.096	0.241	208
3	0.209	0.294	145	0.104	0.211	144
4	1.091	8.000	Error	0.092	0.284	186
5	0.146	0.383	146	0.181	0.424	146
6	0.247	0.477	185	0.154	0.546	184

7	0.205	0.777	Error	0.162	0.391	262
8	0.674	1.471	202	0.146	0.520	207
9	0.176	0.422	184	0.118	0.547	183
10	0.328	0.507	146	0.115	0.267	145
11	0.267	0.407	207	0.106	0.305	207
12	0.431	0.686	262	0.088	0.202	262
13	0.120	0.334	208	0.192	0.512	208
14	0.629	12.134	Error	0.252	3.637	Error
15	0.361	4.343	Error	0.176	0.352	186
16	0.603	1.589	137	0.217	0.644	145

Design of Experiments Sensitivity Analysis

	Zigzag				Square			
	Linear		Quadratic		Linear		Quadratic	
	Average Error	Maximum Error	Average Error	Maximum Error	Average Error	Maximum Error	Average Error	Maximum Error
Surface Curvature	2.644	22.488	2.165	36.517	3.302	22.876	0.029	4.076
Amplitude	1.718	4.869	1.738	4.015	2.237	3.306	0.207	4.606
Noise	0.475	21.840	3.038	39.509	0.072	13.279	0.778	5.805
Line Pitch	0.283	19.791	1.725	3.378	0.107	10.529	0.232	4.390
Edge Pitch	0.147	2.217	1.586	2.877	0.891	27.410	0.034	4.064
Edge Curvature	0.051	2.074	2.234	36.405	0.516	0.618	0.038	3.680
Intersection Angle	0.318	19.811	1.663	3.724	1.058	19.964	0.131	4.411
Tolerance	0.789	0.710	2.303	37.398	1.792	28.439	0.103	3.632
Fillet Radius	0.146	1.310	2.230	37.420	0.184	9.897	0.031	3.552
Ball Radius	0.834	3.205	1.539	1.961	1.144	7.887	0.016	3.087
Cloud Pitch	0.167	0.221	2.456	37.325	0.042	2.001	0.176	5.203

Experiment 11

Input Parameters

	Surface Curvature	Ampli-tude	Noise	Line Pitch	Edge Pitch	Edge Curvature	Intersec-tion Angle	Toler-ance	Fillet Radius	Probe Radius	Cloud Pitch
Low	0.003	30	0.05	1.5	4	0.003	2.094	0.15	2	1	0.5
High	0.004	40	0.1	3	5	0.004	2.443	0.2	4	2.5	1

Experimental Results - Zigzag

	Linear			Quadratic		
Test	Average Error	Maximum Error	Number of Points	Average Error	Maximum Error	Number of Points
1	5.394	10.560	256	0.148	0.305	258
2	0.078	0.139	208	0.059	0.177	208
3	0.104	0.214	144	0.101	0.196	145
4	0.139	0.171	184	0.060	0.201	184
5	0.061	0.217	145	0.096	0.294	145
6	0.089	0.183	181	0.121	0.335	180
7	0.085	0.191	262	0.082	0.266	262
8	0.140	0.688	202	0.070	0.397	205
9	0.092	0.498	187	0.123	0.435	183
10	0.146	0.256	142	0.107	0.325	141
11	0.133	0.690	204	0.090	0.347	205
12	0.170	0.263	258	0.055	0.155	257
13	0.123	0.336	201	0.178	0.507	200
14	0.163	0.716	257	0.250	3.299	Error
15	0.129	0.279	180	0.101	0.260	178
16	0.157	0.463	144	0.138	0.338	145

Experimental Results - Square

	Linear			Quadratic		
Test	Average Error	Maximum Error	Number of Points	Average Error	Maximum Error	Number of Points
1	0.087	0.159	263	0.154	0.246	262
2	0.079	0.146	208	0.056	0.130	208
3	0.095	0.132	144	0.076	0.187	145
4	0.135	0.182	185	0.055	0.223	186
5	0.069	0.179	146	0.099	0.283	146
6	0.094	0.167	184	0.134	0.318	184

7	0.084	0.159	263	0.084	0.223	263
8	0.114	0.250	207	0.066	0.162	207
9	0.102	0.396	181	0.258	7.220	Error
10	0.120	0.175	145	0.085	0.247	146
11	0.144	0.403	205	0.495	20.725	Error
12	0.168	0.236	263	0.050	0.116	264
13	0.101	0.262	209	0.157	0.481	207
14	0.211	5.245	256	0.374	11.343	391
15	0.116	0.230	185	0.109	0.230	185
16	0.162	0.630	145	0.125	0.298	146

Design of Experiments Sensitivity Analysis

	Zigzag				Square			
	Linear		Quadratic		Linear		Quadratic	
	Average Error	Maximum Error	Average Error	Maximum Error	Average Error	Maximum Error	Average Error	Maximum Error
Surface Curvature	6.373	12.783	0.076	3.311	0.360	6.467	0.613	21.194
Amplitude	6.438	12.580	0.487	4.446	0.197	5.703	0.327	2.399
Noise	6.715	12.291	0.373	4.496	0.028	6.693	0.102	19.930
Line Pitch	6.296	11.211	0.386	4.421	0.463	7.848	1.174	49.190
Edge Pitch	6.728	12.468	0.127	3.383	0.147	5.816	0.075	3.281
Edge Curvature	6.790	14.294	0.111	3.881	0.120	6.033	0.624	30.888
Intersection Angle	6.812	12.261	0.023	3.642	0.115	7.245	0.905	48.648
Tolerance	6.841	14.283	0.084	3.819	0.140	5.940	0.630	30.972
Fillet Radius	6.739	12.443	0.087	3.276	0.033	5.917	0.256	3.906
Ball Radius	6.765	13.317	0.057	3.253	0.005	5.520	0.566	30.829
Cloud Pitch	5.394	10.560	0.381	4.386	0.173	7.353	1.249	48.782

Experiment 12

Input Parameters

	Surface Curvature	Ampli-tude	Noise	Line Pitch	Edge Pitch	Edge Curvature	Intersec-tion Angle	Toler-ance	Fillet Radius	Probe Radius	Cloud Pitch
Low	0.002	20	0.05	1	4	0.003	2.094	0.15	2	1	0.25
High	0.003	40	0.1	2	6	0.004	2.618	0.2	4	2.5	0.5

Experimental Results - Zigzag

	Linear			Quadratic		
Test	Average Error	Maximum Error	Number of Points	Average Error	Maximum Error	Number of Points
1	0.081	0.178	256	0.324	0.546	260
2	0.057	0.140	172	0.072	0.195	172
3	0.080	0.113	119	0.068	0.148	119
4	0.093	0.148	185	0.054	0.226	183
5	0.096	0.194	120	0.130	0.346	120
6	0.078	0.168	180	0.272	0.490	183
7	0.058	0.146	263	0.079	0.266	262
8	0.088	0.180	171	0.076	0.196	171
9	0.060	0.215	185	0.104	0.369	186
10	0.152	0.303	117	0.192	0.439	116
11	0.092	0.187	171	0.071	0.186	171
12	0.096	0.139	258	0.060	0.174	258
13	0.168	0.438	166	0.307	0.583	170
14	0.096	0.324	263	0.186	0.677	263
15	0.061	0.145	182	0.087	0.179	181
16	0.127	0.329	120	0.111	0.386	120

Experimental Results - Square

	Linear			Quadratic		
Test	Average Error	Maximum Error	Number of Points	Average Error	Maximum Error	Number of Points
1	0.073	0.136	263	0.340	0.587	264
2	0.056	0.148	172	0.080	0.193	172
3	0.079	0.121	120	0.070	0.158	120
4	0.086	0.127	186	0.059	0.193	186
5	0.098	0.217	120	0.126	0.294	120
6	0.083	0.189	185	0.275	0.503	185

7	0.050	0.131	263	0.080	0.234	264
8	0.078	0.142	173	0.082	0.196	172
9	0.058	0.259	185	0.096	0.628	185
10	0.082	0.218	120	0.168	0.420	120
11	0.085	0.171	172	0.071	0.177	172
12	0.092	0.138	263	0.056	0.172	264
13	0.125	0.362	172	0.281	0.718	172
14	0.086	0.287	262	0.179	0.723	262
15	0.055	0.129	186	0.090	0.172	186
16	0.116	0.307	120	0.127	0.375	120

Design of Experiments Sensitivity Analysis

	Zigzag				Square			
	Linear		Quadratic		Linear		Quadratic	
	Average Error	Maximum Error	Average Error	Maximum Error	Average Error	Maximum Error	Average Error	Maximum Error
Surface Curvature	0.111	0.147	0.187	0.203	0.073	0.038	0.160	0.243
Amplitude	0.118	0.726	1.240	2.382	0.025	0.692	1.152	3.018
Noise	0.077	0.635	0.383	1.063	0.100	0.564	0.381	0.868
Line Pitch	0.280	1.028	0.054	0.733	0.120	0.835	0.054	1.299
Edge Pitch	0.301	0.532	0.180	0.565	0.168	0.366	0.213	0.861
Edge Curvature	0.017	0.148	0.199	0.305	0.017	0.068	0.200	0.325
Intersection Angle	0.158	0.022	0.734	0.132	0.043	0.268	0.689	0.139
Tolerance	0.043	0.036	0.061	0.233	0.020	0.290	0.064	0.275
Fillet Radius	0.274	0.565	0.795	1.061	0.124	0.251	0.786	0.849
Ball Radius	0.168	0.377	0.176	0.310	0.189	0.371	0.193	0.739
Cloud Pitch	0.081	0.178	0.291	0.746	0.018	0.178	0.380	1.203

Appendix F

Second Case Study Results

F1. Time Study

The time study measurements for the Reverse Engineering of the core box in Figure 1 are given in the following table.

Figure 1 Core Box.

Study No: 1	**TIME STUDY FORM**		
Observer: K. Schreve	**Date:** 03-Sept-2001	**Machine Code:** Cyclone	
Operator: K. Schreve	**Part Name:** Core Box	**Part No:** CB02	
Operation Description:	Reverse Engineering of core box	**Time Units:** h:min	

Date	**Element Description**	**Time Started**	**Time Finished**	**Time**
3/9	Set up measurement	16:02	16:28	0:26
	Calibrate probe and set up coordinate system	16:28	16:42	0:14
4/9	Start Scanning	12:31	17.34	5:03
17/9	Copy data into CADLAB and set up CAD database	09:30	09:45	0:15
	Scan edges	09:45	10:20	0:35
		11:00	12:30	1:30
		14:00	15:10	1:10
	Join edge sections	16:10	16:30	0:20
	Construct base surfaces	16:30	16:50	0:20
	Cut point cloud	20:30	20:43	0:13
19/9	Fit swept surfaces	17:40	17:56	0:16

F2. Input Parameters for Edge Scanning

The edges were scanned in different sections. The scanning parameters used to detect the edges are given in the following tables.

Table 1 Edge 1 Parameters

		Section 1	Section 2	Section 3
Cloud Pitch	[mm]	0.5	0.5	2
Prehit Distance	[mm]	2	2	8
Search Distance	[mm]	30	30	30
Probe Direction	[]	(0,0,-1)	(0,0,-1)	(0,0,-1)
Probe Diameter	[mm]	1	1	1
Line Pitch	[mm]	0.75	1	3
Edge Pitch	[mm]	1	2	3
Amplitude	[mm]	7	10	20
Tolerance	[mm]	0.15	0.15	0.15
Start Point	[mm]	(52,167,0.5)	(312,133,0.5)	(335.7,102.5,0.5)
Start Direction	[]	(0,-1,0)	(-0.777,-0.629,0)	(-0.906,-0.424,0)
End Point	[mm]	(342,42,0.5)	(342,42,0.5)	(342,42,0.5)
End Direction	[mm]	(-1,0,0)	(-1,0,0)	(-1,0,0)

Table 2 Edge 2 Parameters.

		Section 1	Section 2
Cloud Pitch	[mm]	0.5	2
Prehit Distance	[mm]	2	8
Search Distance	[mm]	30	30
Probe Direction	[]	(0,0,-1)	(0,0,-1)
Probe Diameter	[mm]	1	1
Line Pitch	[mm]	0.75	3
Edge Pitch	[mm]	1	3
Amplitude	[mm]	7	20
Tolerance	[mm]	0.15	0.15
Start Point	[mm]	(52,128,0.5)	(290,42,0.5)
Start Direction	[]	(0,1,0)	(1,0,0)
End Point	[mm]	(295,42,0.5)	(287,83,0.5)
End Direction	[mm]	(1,0,0)	(1,0,0)

Table 3 Edge 3 Parameters.

		Section 1	Section 2	Section 3
Cloud Pitch	[mm]	0.5	1	2
Prehit Distance	[mm]	2	4	8
Search Distance	[mm]	30	30	30
Probe Direction	[]	(0,0,-1)	(0,0,-1)	(0,0,-1)
Probe Diameter	[mm]	1	1	1
Line Pitch	[mm]	0.75	1.5	3
Edge Pitch	[mm]	1	2	3
Amplitude	[mm]	7	10	20
Tolerance	[mm]	0.15	0.15	0.15
Start Point	[mm]	(52,69,0.5)	(148.6,56.7,0.5)	(298,42,0.5)
Start Direction	[]	(0,-1,0)	(0.921,-0.391,0)	(-1,0,0)
End Point	[mm]	(292,42,0.5)	(292,42,0.5)	(296,70,0.5)
End Direction	[mm]	(-1,0,0)	(-1,0,0)	(-1,0,0)

Table 4 Edge 4 Parameters.

		Section 1	Section 2
Cloud Pitch	[mm]	1	2
Prehit Distance	[mm]	4	8
Search Distance	[mm]	30	30
Probe Direction	[]	(0,0,-1)	(0,0,-1)
Probe Diameter	[mm]	1	1
Line Pitch	[mm]	1.5	3
Edge Pitch	[mm]	2	3
Amplitude	[mm]	10	20
Tolerance	[mm]	0.15	0.15
Start Point	[mm]	(52,30,0.5)	(237,42,0.5)
Start Direction	[]	(0,1,0)	(1,0,0)
End Point	[mm]	(242,42,0.5)	(249,60,0.5)
End Direction	[mm]	(1,0,0)	(1,0,0)

Appendix G

Torus Model Detail

G1. Introduction

A simple model constructed of two torii was used for testing the scanning algorithm. It is described here. The model consists of two segments of two torii that intersect each other. The one torus has a smaller major radius than the other and both have the same section radius. An example of the model is shown in the figure below. As shown in the figure, a fillet radius can easily be added as a third torus. It is also very easy to simulate the effect of uncompensated points by increasing or decreasing the section radius of the torii depending on the situation.

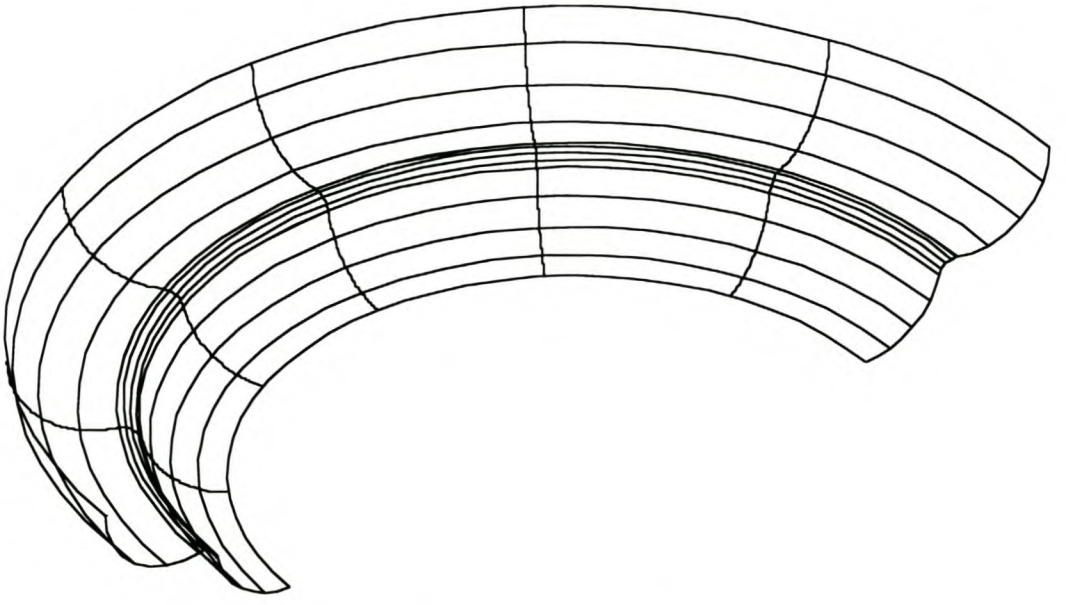


Figure 1 Example of the Torus Model.

The rest of this Appendix describes the method by which the torus parameters are derived so that the object parameters governing the scanning algorithm can be studied. These parameters are the surface curvature in the direction perpendicular to the edge (i.e. the section radius of the torus), the intersection angle between the two surfaces and the gap that might be formed by something like a fillet radius, chamfer or

damaged edge. It is also necessary to know the parameters of the circle that represents the edge so that the accuracy of the edge points can be calculated.

Scanning on a point cloud is much more difficult than scanning on a continuous surface. Thus, it is wise to make the algorithm to work on a continuous surface before going on to a discrete surface. The torus model was used for this purpose as well as for generating point clouds with which the algorithm can be tested. The last section of this appendix presents the method that is used to find the intersection of the rays that represent the movement of the CMM with the torii.

G2. Model Description

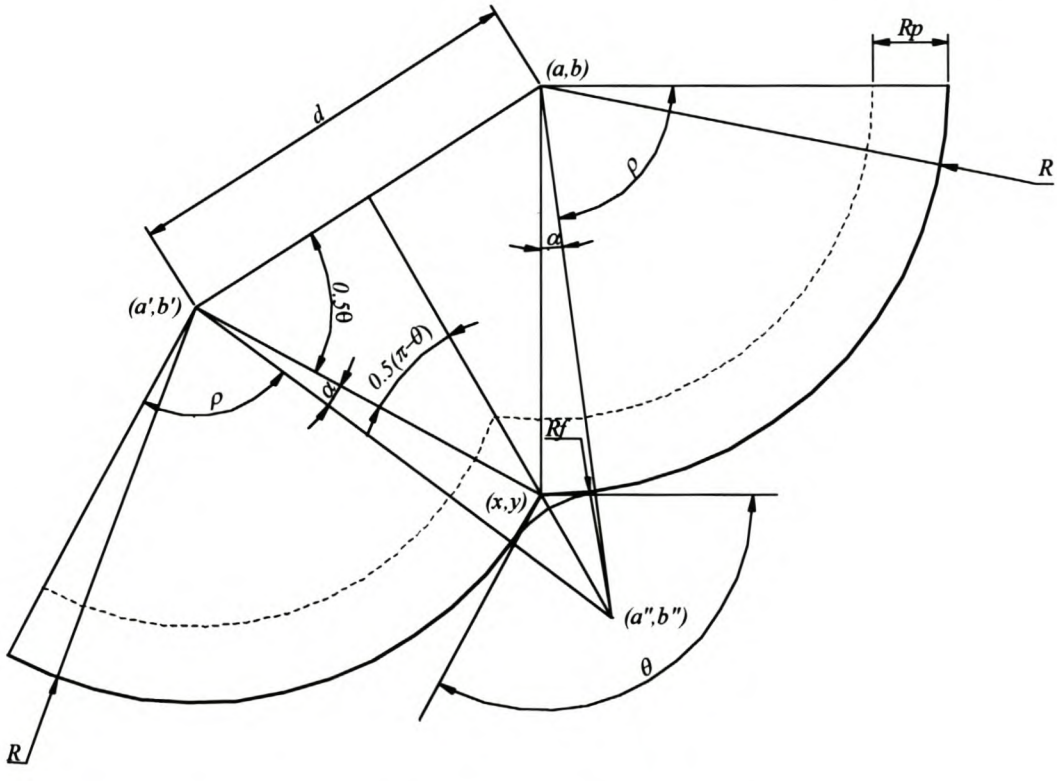


Figure 2 Description of Torus Model.

The model is derived as follows (the symbols are described in Figure 2).

Given:

θ the intersection angle between the two torii

R section radius of the torii

x radius of the intersection circle

R_f section radius of the fillet torus

R_p probe ball raius

The major radius of the torii must be determined. These are a, a' and a'' as shown in the Figure 2. The torii are orientated such that the axis of revolution is the z-axis (Figure 3). Thus, the position of the centre of the torii must also be determined. They are b, b' and b'' in the Figure 2. It is assumed that b is 0 and also that the edge circle has the same radius as major radius of the largest torus. Therefore, $x=a$ in Figure 2.

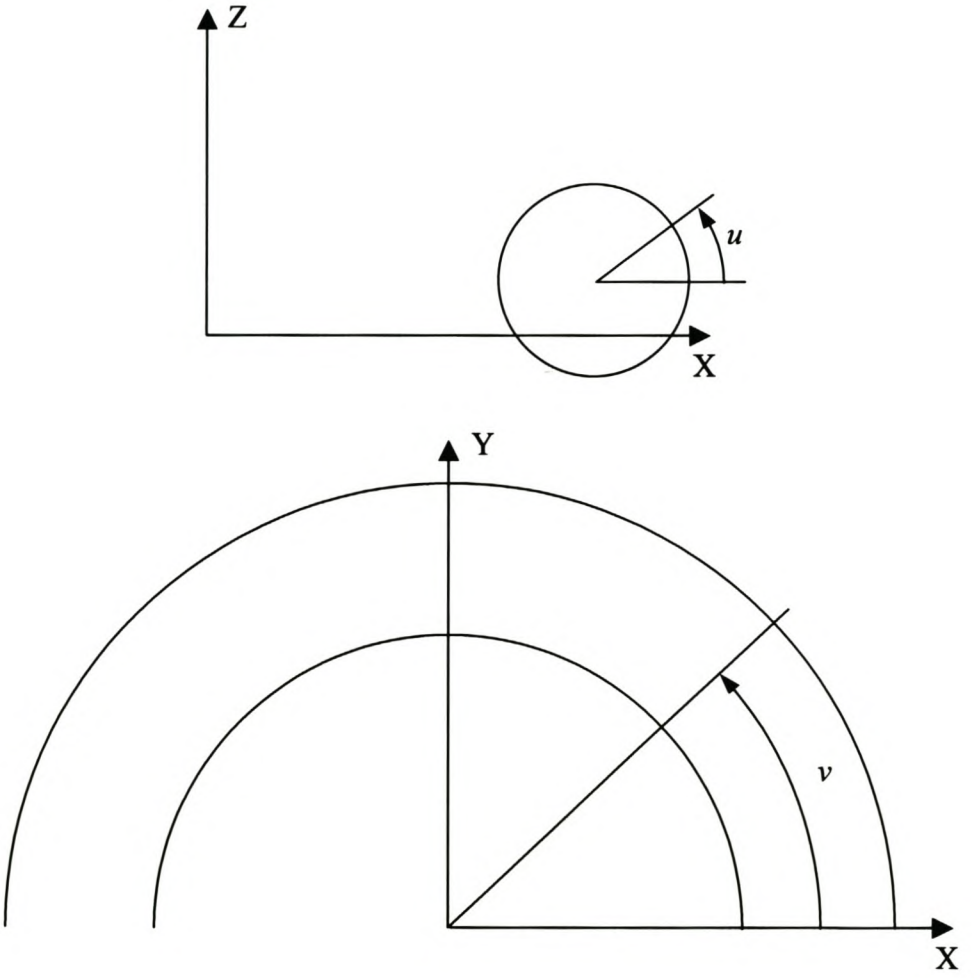


Figure 3 Torus Parameterisation.

Only the half of the torii with non-negative y-value is used in the model. Also, only an arc of length s (arc angle ρ) is used to create the section of the torus on which will be

scanned. This is used to determine the parameterisation of the torii. The parameterisation is shown below.

The torii parameters are found as follows using simple geometric and trigonometric rules. The meaning of these parameters is clear from Figure 2. (All angles are in radians.)

$$d = 2R\sin\left(\frac{\pi-\theta}{2}\right) \tag{G2.1.}$$

$$\alpha = \arccos\left(\frac{0.5d}{R+R_f}\right)-\frac{\theta}{2} \tag{G2.2.}$$

$$b' = R\cos(\pi-\theta)-R \tag{G2.3.}$$

$$a' = x - R\sin(\pi-\theta) \tag{G2.4.}$$

$$a'' = a'+(R+R_f)\sin(\pi-\theta-\alpha) \tag{G2.5.}$$

$$b'' = b'-(R+R_f)\cos(\pi-\theta-\alpha) \tag{G2.6.}$$

The position of the edge circle's centre on the z-axis is

$$y = b'-R\cos(\pi-\theta) \tag{G2.7.}$$

The parameterisation for the torus with major radius a is

$$u_0 = 1.5\pi+\alpha \tag{G2.8.}$$

$$u_1 = u_0+\rho \tag{G2.9.}$$

The parameterisation of the torus with major radius a' is

$$u_1' = 1.5\pi+(\pi-\theta)-\alpha = 2.5\pi-\theta-\alpha \tag{G2.10.}$$

$$u_0' = u_1-\rho \tag{G2.11.}$$

The parameterisation of the fillet torus with major radius a'' is

$u_0'' = 0.5\pi + \alpha$ (G2.12.)

$u_1'' = u_0'' + \pi - \theta - 2\alpha = 1.5\pi - \alpha - \theta$ (G2.13.)

G3. Analytic Line Intersections

The torus model described in the previous section was also used to test the edge scanning method on a continuous surface. A method of finding the intersection of the ray representing the CMM movement with the torus model is needed to do this. In this section a method is presented to find the intersection between a torus and a line. This method is used twice (or thrice if a fillet torus is part of the model) to find all the possible intersections. The point that the CMM will touch first is then easily determined.

The parametric model of a torus as given by Do Carmo (1976) is used. This is repeated in the equation below. The parameter b is added so that the torus's origin can be anywhere on the Z-axis.

$$\boldsymbol{t}(u,v) = \big((a + r\cos u)\cos v,\ (a + r\cos u)\sin v,\ r\sin u + b\big) \quad \text{(G3.1.)}$$

The intersection points of the line and torus will be where the torus equals the line. This gives three equation with which the three unknowns, u, v and λ can be solved. (λ is the parameter of the line.)

$$s_x + \lambda a_x = (a + r\cos u)\cos v \quad \text{(G3.2.a)}$$

$$s_y + \lambda a_y = (a + r\cos u)\sin v \quad \text{(G3.2.b)}$$

$$s_z + \lambda a_z = r\sin u + b \quad \text{(G3.2.c)}$$

Taking equation G3.2.a and remembering that $\cos^2 x + \sin^2 x = 1$ a new expression for $\sin(v)$ is

$$\sin v = \sqrt{1 - \left(\frac{s_x + \lambda a_x}{a + r\cos u}\right)^2} \quad \text{(G3.3.)}$$

The same is done to find another expression for $\cos(u)$ from equation G3.2.c.

$$\cos u = \sqrt{1-\left(\frac{s_z + \lambda a_z - b}{r}\right)^2} \qquad \text{(G3.4.)}$$

Now, substitute equations G3.3. and G3.4. in equation G3.2.b and simplify the equation.

$$\left(s_y + \lambda a_y\right)^2 + \left(s_x + \lambda a_x\right)^2 - a^2 - r^2 + \left(b - s_z - \lambda a_z\right)^2 = 2ar\sqrt{1-\left(\frac{b - s_z - \lambda a_z}{r}\right)^2}$$

(G3.5.)

From now on it is to cumbersome to work with equation G3.5. as a hole. The square of the left hand side (*LH*) can be simplified a bit if it remembered that the direction vector of the line, $\boldsymbol{a}$, is a unit vector.

$$LH^2 = \left(\|\boldsymbol{s}\|^2 - a^2 - r^2 + b^2 - 2bs_z + 2(\boldsymbol{s}\cdot\boldsymbol{a})\lambda - 2ba_z\lambda + \lambda^2\right)^2 \qquad \text{(G3.6.)}$$

Define the constants K_1 and K_2 to find the polynomial expression of equation G3.6. as given in equation G3.9.

$$K_1 = \|\boldsymbol{s}\|^2 - a^2 - r^2 + b^2 - 2bs_z \qquad \text{(G3.7.)}$$

$$K_2 = 2(\boldsymbol{s}\cdot\boldsymbol{a}) - 2ba_z \qquad \text{(G3.8.)}$$

$$LH^2 = K_1^2 + 2K_1K_2\lambda + (2K_1 + K_2^2)\lambda^2 + 2K_2\lambda^3 + \lambda^4 \qquad \text{(G3.9.)}$$

A similar process is followed for the square of the right hand side (*RH*) of equation G3.5.

$$RH^2 = K_3 + K_4\lambda + K_5\lambda^2 \qquad \text{(G3.10.)}$$

$$K_3 = 4a^2(r^2 - b^2 + 2bs_z - s_z^2) \qquad \text{(G3.11.)}$$

$$K_4 = 8a^2a_z(b - s_z) \qquad \text{(G3.12.)}$$

$$K_5 = -4a^2a_z^2 \qquad \text{(G3.13.)}$$

Equations G3.9. and G3.10. can now be combined to give the following quartic polynomial with which λ can be solved.

$$0 = (K_1^2 - K_3) + (2K_1K_2 - K_4)\lambda + (2K_1 + K_2^2 - K_5)\lambda^2 + 2K_2\lambda^3 + \lambda^4 \qquad \text{(G3.14.)}$$

In the implementation of the algorithm, an eigenvalue method (Laguerre's method) given by Press et al. (1997) is used. With hindsight, it is probably better to use the direct solution of the quartic equation as given by Spiegel (1968).

If the real roots of this equation are substituted into the equation for the line, all the intersection points are easily obtained. The next step is to determine if the points lie on the section of the torus that is used for the model. (See the parameter range as defined in the previous section.) Thus, the parameters of the points on the torus (u,v) must be determined and compared with the parameter range of the model.

Let $\boldsymbol{p}$ be an intersection point. If $p_x \neq 0$ then the parameters on the torus are obtained as follows. The v parameter is obtained from the X and Y components of the point $\boldsymbol{p}$.

$$v = \arctan\left(\frac{p_y}{p_x}\right) \text{ if } p_x \neq 0 \text{ (making sure that } v \text{ is in the correct quadrant)} \qquad \text{(G3.15.a)}$$

$$v = \frac{\pi}{2} \text{ if } p_x = 0 \text{ and } p_y > 0 \qquad \text{(G3.15.b)}$$

$$v = \frac{3\pi}{2} \text{ if } p_x = 0 \text{ and } p_y < 0 \qquad \text{(G3.15.c)}$$

The u parameter is simply

$$u = \arcsin\left(\frac{p_z - b}{r}\right) \text{(making sure that } u \text{ is in the correct quadrant)} \qquad \text{(G3.16.)}$$

Appendix H

Lengthening NURBS Surfaces

H1. Shetty and White's Method*

It is often necessary to lengthen NURBS surfaces. The problem is that their extension is not intuitive as is the case with cylinders and planes for example. The reason for this is that to lengthen a NURBS surface without changing the shape of the existing part of the surface means that the surface must be extended beyond the existing parameter space. In other words, some assumption must be made about how the surface will behave outside its existing parameter domain. An assumption must also be made about the degree of continuity at the boundary where the surface will be extended.

Literature on the topic is hard to find. The method described below is based on the work of Shetty and White (1991). The method makes the assumption that the extension will be a ruled surface with tangent-plane continuity on the boundary.

In Figure 1, $\boldsymbol{S}(u,v)$ is the original surface. It will be extended on the boundary where $u=0$. $\boldsymbol{R}(u^r,v^r)$ is a ruled surface. On the boundary $v=v^r$ and $\boldsymbol{S}(u,v)=\boldsymbol{R}(u^r,v^r)=\boldsymbol{R}(u^r,v)$. $\boldsymbol{S}$ and $\boldsymbol{R}$ are defined on the following knot vectors.

$$\boldsymbol{U}=\{u_i\}_0^{n+p+1}$$

such that $a=u_0=u_1=\ldots=u_p$ and $u_{n+p+1}=u_{n+p}=\ldots=u_{n+1}=b$ and $a<b$

$$\boldsymbol{V}=\{v_i\}_0^{m+q+1}$$

such that $c=v_0=v_1=\ldots=v_q$ and $v_{m+q+1}=v_{m+q}=\ldots=v_{m+1}=d$ and $c<d$

---

* Please note that further explanations of the nomenclature can be found in paragraph 6.2.

$$U^r = \{ur_i\}_0^{nr+p+1}$$

such that $e=ur_0=ur_1=\ldots=ur_p$ and $ur_{nr+p+1}=ur_{nr+p}=\ldots=ur_{nr+1}=a$ and $e<a$

$$V^r = V$$

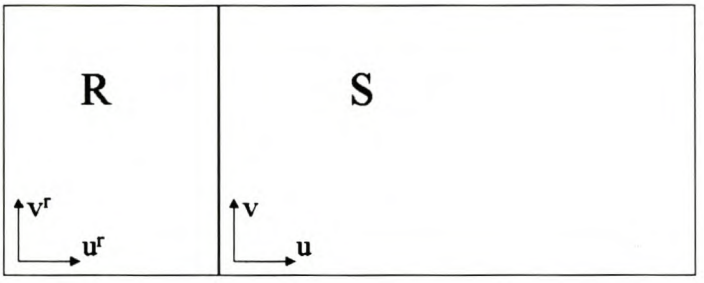


Figure 1 Surface Extension.

In order to maintain geometric tangent plane continuity the following equality must hold.

$$\boldsymbol{S}_u(a,v)+\lambda(v)\boldsymbol{R}_u(a,v)+\alpha(v)\boldsymbol{S}_v(a,v)=\boldsymbol{0} \qquad \text{(H1.1.)}$$

$\lambda(v)$ and $\alpha(v)$ can be any function of v and $\lambda(v)\neq 0$. For brevity, $\boldsymbol{S}_u(a,v)$ will be written just as $\boldsymbol{S}_u$ in the rest of this discussion and similarly for $\lambda(v)$, $\alpha(v)$, $\boldsymbol{R}_u(a,v)$ and $\boldsymbol{S}_v(a,v)$. The equations for the derivatives are (where W is the homogenous coordinate)

$$S_u=(S^H{}_u W_S - S^H W_{Su})/W_S^2$$

$$S_v=(S^H{}_v W_S - S^H W_{Sv})/W_S^2$$

$$\boldsymbol{R}_u=(\boldsymbol{R}^H{}_u W_R - \boldsymbol{R}^H W_{Rv})/W_R^2$$

Since $\boldsymbol{S}=\boldsymbol{R}$ at the boundary, $\boldsymbol{S}^H=\boldsymbol{R}^H W_S/W_R$. Now, this and the above equations for the derivatives can be substituted in equation H1.1. After simplifying, the following equation can be written. In the equation below $\omega=W_S/W_R$.

$$\boldsymbol{S}_u^H + \lambda\omega\boldsymbol{R}_u^H + \alpha\boldsymbol{S}_v^H = \left(\frac{W_{Su}}{W_S} + \lambda\omega\frac{W_{Ru}}{W_S} + \alpha\frac{W_{Sv}}{W_S}\right)\boldsymbol{S}^H$$

If τ represent the term in brackets, this equation can be decoupled so that one equation is only a function of τ and the weights. The continuity conditions can also be further

restricted so that $\boldsymbol{S}_u^H = k\boldsymbol{R}_u^H$. This means that α in the above equations must be 0. For geometric continuity λ is set to a constant, say λ_0. With all these simplifications, the equations for tangent-plane continuity are:

$$\boldsymbol{S}_u^H + \omega\lambda_0 \boldsymbol{R}_u^H = \tau \boldsymbol{S}^H \qquad \text{(H1.2.a)}$$

$$W_{Su} + \omega\lambda_0 W_{Ru} = \tau W_S \qquad \text{(H1.2.b)}$$

The derivatives in the above equations can be calculated using the equations for the B-spline surface derivatives. Doing this for equation H1.2.a the following equation is obtained. (The superscript ^{S} and ^{R} above the $\boldsymbol{P}$ and w refer to the surfaces **S** and **R** in Figure 1 respectively.)

$$\tau \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(a)N_{j,q}(v) w_{i,j}^S \boldsymbol{P}_{i,j}^S = p\sum_{i=0}^{n-1}\sum_{j=0}^{m} N_{i+1,p-1}(a)N_{j,q}(v)\frac{w_{i+1,j}^S \boldsymbol{P}_{i+1,j}^S - w_{i,j}^S \boldsymbol{P}_{i,j}^S}{u_{p+i+1}-u_{i+1}}$$
$$+\lambda_0 \omega p \sum_{i=0}^{nr-1}\sum_{j=0}^{m} N_{i+1,p-1}(a)N_{j,q}(v)\frac{w_{i+1,j}^R \boldsymbol{P}_{i+1,j}^R - w_{i,j}^R \boldsymbol{P}_{i,j}^R}{ur_{p+i+1}-ur_{i+1}}$$

This equation can be further simplified when it is noted, from definition of the B-spline basis functions and the definition of the knot vectors for the surfaces **R** and **S**, that $N_{0,p}(a)=N_{1,p-1}(a)=1$ and $N_{i,p}(a)=N_{i+1,p-1}(a)=0, \ \forall i \neq 0$. The result is shown here.

$$\tau \sum_{j=0}^{m} N_{j,q}(v) w_{0,j}^S \boldsymbol{P}_{0.j}^S = p\sum_{j=0}^{m} N_{j,q}(v)\frac{w_{1,j}^S \boldsymbol{P}_{1,j}^S - w_{0,j}^S \boldsymbol{P}_{0,j}^S}{u_{p+1}-u_1} + \lambda_0 \omega p \sum_{j=0}^{m} N_{j,q}(v)\frac{w_{n,j}^R \boldsymbol{P}_{n,j}^R - w_{n-1,j}^R \boldsymbol{P}_{n-1,j}^R}{ur_{nr+p}-ur_{nr}}$$

If this equation is written in matrix form, the next simplification becomes obvious. It is done as follows.

$$\begin{bmatrix} \ddots & & \iddots \\ & N_{j,q}(v) & \\ \iddots & & \ddots \end{bmatrix}_{(m+1)\times(m+1)} \begin{bmatrix} \vdots \\ \tau \boldsymbol{P}^S_{0,j} - p\dfrac{w^S_{1,j}\boldsymbol{P}^S_{1,j} - w^S_{0,j}\boldsymbol{P}^S_{0,j}}{u_{p+1}-u_1} \\ \vdots \end{bmatrix}_{(m+1)\times 1} =$$

$$\begin{bmatrix} \ddots & & \iddots \\ & N_{j,q}(v) & \\ \iddots & & \ddots \end{bmatrix}_{(m+1)\times(m+1)} \begin{bmatrix} \vdots \\ \lambda_0 \omega p\dfrac{w^R_{nr,j}\boldsymbol{P}^R_{n,j} - w^R_{n-1,j}\boldsymbol{P}^R_{n-1,j}}{ur_{nr+p}-ur_{nr}} \\ \vdots \end{bmatrix}_{(m+1)\times 1}$$

The matrix containing the B-spline basis functions on both sides of the equation are the same. If the surfaces are defined in such a way that the inverse of these matrices exists, then they can be taken out of the equation by pre-multiplying with the inverse of these matrices. This leads to the following result.

$$\tau w^S_{0,j}\boldsymbol{P}^S_{0,j} = p\frac{w^S_{1,j}\boldsymbol{P}^S_{1,j} - w^S_{0,j}\boldsymbol{P}^S_{0,j}}{u_{p+1}-u_1} + \lambda_0 \omega p\frac{w^R_{nr,j}\boldsymbol{P}^R_{nr,j} - w^R_{nr-1,j}\boldsymbol{P}^R_{nr-1,j}}{ur_{nr+p}-u_{nr}} \qquad \text{(H1.3.)}$$

If the new control net is numbered as is shown in Figure 2, the new control points in (*r-1*)'th column is obtained from the equation above equation. A similar equation for the new weights is found by following the same procedure for deriving equation H1.3. The equations H1.4.a and H1.4.b are the resulting two equations, using the control net numbering of Figure 2. Here, *r* is the number of new control point columns that must be added, it is equal to the degree of the surface in the *u* parametric direction, i.e. *r=p*. The new surface is defined on the new knot vector ***U***, equation H1.5., and the original knot vector ***V***. The equations H1.4.a and H1.4.b are derived using the new knot vector ***U***.

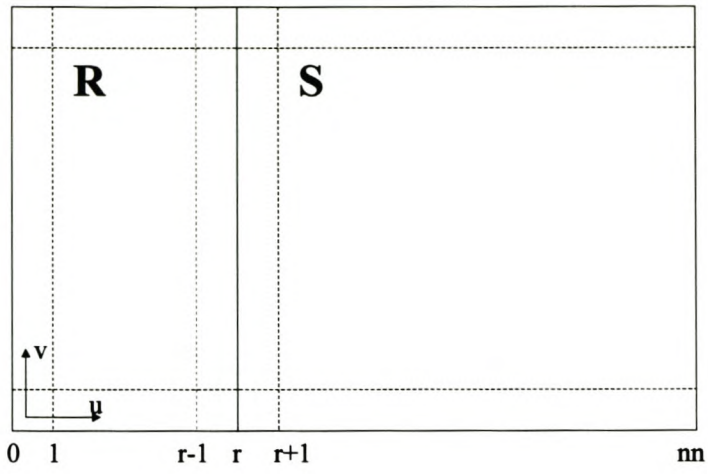


Figure 2 New Control Net Numbering.

$$\tau \boldsymbol{P}_{r,j} = p\frac{w_{r+1,j}\boldsymbol{P}_{r+1,j} - w_{r,j}\boldsymbol{P}_{r,j}}{u_{2p+1} - u_{p+1}} + \lambda_0 \omega p \frac{w_{r,j}\boldsymbol{P}_{r,j} - w_{r-1,j}\boldsymbol{P}_{r-1,j}}{u_{2p} - u_p} \qquad \text{(H1.4.a)}$$

$$\tau w_{r,j} = p\frac{w_{r+1,j} - w_{r,j}}{u_{2p+1} - u_{p+1}} + \lambda_0 \omega p \frac{w_{r,j} - w_{r-1,j}}{u_{2p} - u_p} \qquad \text{(H1.4.b)}$$

$$\boldsymbol{U} = \{u_i\}_0^{nn+p+1} \qquad \text{(H1.5.)}$$

such that $e=u_0 = \ldots = u_p$, $a=u_{p+1}=\ldots=u_{2p}$ and $u_{nn+p+1}=u_{nn+p}=\ldots=u_{nn+1}=b$ and $e<a<b$

The only unknowns in the above equations are $\boldsymbol{P}_{r-1,j}$, $w_{r-1,j}$, and τ. Now, the procedure to extend a surface, using a linear extrapolation and tangential continuity at the boundary, is as follows.

- Choose $w_{r-1,j}=w_{r+1,j}$.
- Select a value for e in the new knot vector. A criterion such as the percentage extension of the knot vector is suitable when a cord length parameterisation was used. This means that the percentage extension of the knot vector directly translates in the lengthening of the surface by the same percentage.

- Using equation H1.4.b, calculate a τ for every j and then select the maximum value of τ. (This procedure for finding the value of τ was suggested by Shetty and White, 1991.)

- Using τ_{max}, recalculate the weights $w_{r-1,j}$.

- Calculate the new control points $\boldsymbol{P}_{r-1,j}$.

- The last step is to find the remaining control points $\boldsymbol{P}_{0,j}...\boldsymbol{P}_{r-2,j}$. A linear extrapolation using the Euclidian control points, equation H1.6., is used. The linear extrapolation can be written as follows in terms of the homogeneous control points.

$$\frac{w_{r-i,j}\boldsymbol{P}_{r-i,j}}{w_{r-i,j}} = i\frac{w_{r-1,j}\boldsymbol{P}_{r-1,j}}{w_{r-1,j}} - (i-1)\frac{w_{r,j}\boldsymbol{P}_{r,j}}{w_{r,j}} \qquad 2 \le i \le r$$

Choose $w_{r-i,j}=w_{r-1,j}$. Then:

$$\boldsymbol{P}_{r-i,j} = i\boldsymbol{P}_{r-1,j} - (i-1)\frac{w_{r-1,j}}{w_{r,j}}\boldsymbol{P}_{r,j} \qquad \text{(H1.6.)}$$

A similar procedure and equation can be derived to extend a NURBS surface along the other three boundaries.

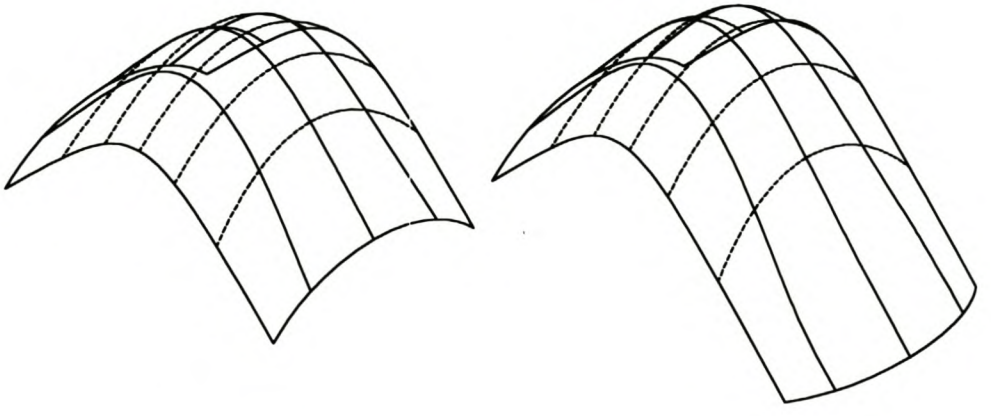


Figure 3 Surface Extension Using Linear Extrapolation and Tangential Continuity. (The original surface is shown on the left.)

H2. Comparison with AutoCAD

Shetty and White's (1991) method was compared with the surface extension of AutoCAD (2000). The surface definition that was used in the comparison is given in the following table.

Table 1 Definition of Original Surface.

Surface Degree	*U-direction*	3	*V-direction*	3
Knot Vectors	**U**=(0,0,0,0,1,1,1,1)		**V**=(0,0,0,0,1,2,2,2,2)	
Control Polygon				

	U-direction			
V-direction	(0,0,0,1)	(0,1,1,0.5)	(0,2,1,0.5)	(0,3,0,1)
	(1,0,1,1)	(1,1,2,1)	(1,2,2,1)	(1,3,1,1)
	(2,0,2,1)	(2,1,3,1)	(2,2,3,1)	(2,3,2,1)
	(3,0,1,1)	(3,1,2,1)	(3,2,2,1)	(3,3,1,1)
	(4,0,0,1)	(4,1,1,0.5)	(4,2,1,0.5)	(4,3,0,1)

The extension of the surface in Table 1 is given in Table 2. This is a linear extension using AutoCAD (2000). Note that the knot vector was normalised after extension.

Table 2 Extension of Original Surface Using AutoCAD (2000).

Surface Degree	*U-direction*	3	*V-direction*	3
Knot Vectors	**U**=(0,0,0,0,1,1,1,1)		**V**=(0,0,0,0,0.2,0.2,0.2,0.6,1,1,1,1)	
Control Polygon				

	U-direction			
V-direction	(-1.125,0,-1.1875,1)	(-1.2,1,-0.267, 0.938)	(-1.2,2, -0.267,0.938)	(-1.125,3,-1.188,1)
	(-0.875,0,-0.875,1)	(-1,1,0,0.875)	(-1,2,0,0.875)	(-0.875,3,-0.875,1)
	(-0.5,0,-0.5,1)	(-0.667,1,0.333, 0.75)	(-0.667,2,0.333, 0.75)	(-0.5,3,-0.5,1)
	(0,0,0,1)	(0,1,1,0.5)	(0,2,1,0.5)	(0,3,0,1)
	(1,0,1,1)	(1,1,2,1)	(1,2,2,1)	(1,3,1,1)
	(2,0,2,1)	(2,1,3,1)	(2,2,3,1)	(2,3,2,1)
	(3,0,1,1)	(3,1,2,1)	(3,2,2,1)	(3,3,1,1)
	(4,0,0,1)	(4,1,1,0.5)	(4,2,1,0.5)	(4,3,0,1)

This extension was repeated with the method of Shetty and White (1991). The result is given in Table 3.

Table 3 Extension of Original Surface Using Shetty and White's (1991) Method.

Surface Degree	*U-direction*	3	*V-direction*	3
Knot Vectors	**U**=(0,0,0,0,1,1,1,1)		**V**=(0,0,0,0,0.2,0.2,0.2,0.6,1,1,1,1)	
Control Polygon				

	U-direction			
V-direction	(-1.286,0, -1.286,1.167)	(-4.5,1,-3.5,0.333)	(-4.5,2,-3.5,0.333)	(-1.286,3, -1.286,1.167)
	(-0.857,0, -0.857,1.167)	(-3,1,-2,0.333)	(-3,2,-2,0.333)	(-0.857,3, -0.857,1.167)
	(-0.429,0, -0.429,1.167)	(-1.5,1,-0.5,0.333)	(-1.5,2,-0.5,0.333)	(-0.429,3, -0.429,1.167)
	(0,0,0,1)	(0,1,1,0.5)	(0,2,1,0.5)	(0,3,0,1)
	(1,0,1,1)	(1,1,2,1)	(1,2,2,1)	(1,3,1,1)
	(2,0,2,1)	(2,1,3,1)	(2,2,3,1)	(2,3,2,1)
	(3,0,1,1)	(3,1,2,1)	(3,2,2,1)	(3,3,1,1)
	(4,0,0,1)	(4,1,1,0.5)	(4,2,1,0.5)	(4,3,0,1)

When the first partial derivatives with respect to v to the left and right of v=0.2 are checked it is seen that the extension generated by AutoCAD (2000) is not continuous. Due to the tessellation of the surface for display by AutoCAD this is not seen on the graphical display. The reason for this discrepancy is not clear. It is possible that it is a programming error. The implementation of Shetty and White's (1991) method is indeed continuous. The partial derivatives are compared in Table 4.

Table 4 Comparison of Continuity of the AutoCAD and Shetty and White (1991) Surface Extension.

	AutoCAD	Shetty and White (1991)
$\left.\frac{\partial \boldsymbol{S}(0.5,0.2)}{\partial v}\right\|_{-}$	(4.508,0,4.508)	(12,0,13.8)
$\left.\frac{\partial \boldsymbol{S}(0.5,0.2)}{\partial v}\right\|_{+}$	(12,0,13.8)	(12,0,13.8)

Appendix I

Tables with Surface Definitions for Surface Intersection Investigation

The following definition of a NURBS surfaces is used

$$\boldsymbol{S}(u,v)=\frac{\sum_{i=1}^{n}\sum_{j=1}^{m}N_{i,p}(u)N_{j,q}(v)w_{i,j}\boldsymbol{P}_{i,j}}{\sum_{i=1}^{n}\sum_{j=1}^{m}N_{i,p}(u)N_{j,q}(v)w_{i,j}} \qquad (1.)$$

The following two tables contain the definition of the base surfaces.

Table 1 Base Surface Used for Parameterising the Outer Surface.

Surface Degree	*U-direction*	3	*V-direction*	3
Knot Vectors	**U**=(0,0,0,0,0.25,0.5,0.75,1,1,1,1)		**V**=(0,0,0,0,0.5,1,1,1,1)	
Control Polygon	*The indices i,j refer the equation* 1.			

(*i,j*)	**Control Point**	(*i,j*)	**Control Point**
(0,0)	(41.968,0,-13.283,1)	(3,3)	(0,40.722,-14.877,1)
(0,1)	(40.961,0,-13.742,1)	(3,4)	(0,39.504,-14.986,1)
(0,2)	(38.946,0,-14.659,1)	(4,0)	(-34.022,32.887,-13.283,1)
(0,3)	(36.746,0,-14.877,1)	(4,1)	(-33.205,32.097,-13.742,1)
(0,4)	(35.647,0,-14.986,1)	(4,2)	(-31.572,30.518,-14.659,1)
(1,0)	(39.319,10.962,-13.283,1)	(4,3)	(-29.789,28.795,-14.877,1)
(1,1)	(38.376,10.699,-13.742,1)	(4,4)	(-28.897,27.933,-14.986,1)
(1,2)	(36.488,10.173,-14.659,1)	(5,0)	(-39.319,10.962,-13.283,1)
(1,3)	(34.427,9.598,-14.877,1)	(5,1)	(-38.376,10.699,-13.742,1)
(1,4)	(33.397,9.311,-14.986,1)	(5,2)	(-36.488,10.173,-14.659,1)
(2,0)	(34.022,32.887,-13.283,1)	(5,3)	(-34.427,9.598,-14.877,1)
(2,1)	(33.205,32.097,-13.742,1)	(5,4)	(-33.397,9.311,-14.986,1)
(2,2)	(31.572,30.518,-14.659,1)	(6,0)	(-41.968,0,-13.283,1)
(2,3)	(29.789,28.795,-14.877,1)	(6,1)	(-40.961,0,-13.742,1)

(2,4)	(28.897,27.933,-14.986,1)	(6,2)	(-38.946,0,-14.659,1)
(3,0)	(0,46.509,-13.283,1)	(6,3)	(-36.746,0,-14.877,1)
(3,1)	(0, 45.393,-13.742,1)	(6,4)	(-35.647,0,-14.986,1)
(3,2)	(0,43.160,-14.659,1)		

Table 2 Base Surface Used for Parameterising the Inner Surface.

Surface Degree	*U-direction*	3	*V-direction*	3
Knot Vectors	**U**=(0,0,0,0,0.25,0.5,0.75,1,1,1,1)		**V**=(0,0,0,0,0.5,1,1,1,1)	
Control Polygon	*The indices i,j refer the equation* 1.			

(*i,j*)	**Control Point**	(*i,j*)	**Control Point**
(0,0)	(34.668,0,-15.833,1)	(3,3)	(0,35.184,-20.205,1)
(0,1)	(34.193,0,-16.787,1)	(3,4)	(0,34.358,-20.961,1)
(0,2)	(33.241,0,-18.694,1)	(4,0)	(-28.104,27.166,-15.833,1)
(0,3)	(31.749,0,-20.205,1)	(4,1)	(-27.719,26.794,-16.787,1)
(0,4)	(31.004,0,-20.961,1)	(4,2)	(-26.947,26.048,-18.694,1)
(1,0)	(32.480,9.055,-15.833,1)	(4,3)	(-25.738,24.879,-20.205,1)
(1,1)	(32.035, 8.931,-16.787,1)	(4,4)	(-25.133,24.295,-20.961,1)
(1,2)	(31.143,8.683,-18.694,1)	(5,0)	(-32.480,9.055,-15.833,1)
(1,3)	(29.746,8.293,-20.205,1)	(5,1)	(-32.035,8.931,-16.787,1)
(1,4)	(29.047,8.098,-20.961,1)	(5,2)	(-31.143,8.683,-18.6941)
(2,0)	(28.104,27.166,-15.833,1)	(5,3)	(-29.746,8.293,-20.205,1)
(2,1)	(27.719,26.794,-16.787,1)	(5,4)	(-29.047,8.098,-20.961,1)
(2,2)	(26.947,26.048,-18.694,1)	(6,0)	(-34.668,0,-15.833,1)
(2,3)	(25.738,24.879,-20.205,1)	(6,1)	(-34.193,0,-16.787,1)
(2,4)	(25.133,24.295,-20.961,1)	(6,2)	(-33.241,0,-18.694,1)
(3,0)	(0,38.419,-15.833,1)	(6,3)	(-31.749,0,-20.205,1)
(3,1)	(0,37.892,-16.787,1)	(6,4)	(-31.004,0,-20.961,1)
(3,2)	(0,36.838,-18.694,1)		

The following two surfaces are the result of the surface approximation.

Table 3 Surface Approximation of Outer Surface.

Surface Degree	*U-direction*	3	*V-direction*	3
Knot Vectors	**U=(0,0,0,0,0.197,0.403,0.598, 0.804,1,1,1,1)**		**V=(0,0,0,0,0.106,0.312, 0.522,0.745,1,1,1,1)**	
Control Polygon	*The indices i,j refer the equation* 1.			

(*i,j*)	**Control Point**	(*i,j*)	**Control Point**
(0,0)	(42.033,-0.117,-13.254,1)	(4,0)	(-14.992,42.531,-13.271,1)
(0,1)	(41.863,0.017,-13.343,1)	(4,1)	(-14.828,42.329,-13.376,1)
(0,2)	(41.222,-0.030,-13.669,1)	(4,2)	(-14.658,41.709,-13.688,1)
(0,3)	(40.155,0.002,-14.111,1)	(4,3)	(-14.247,40.620,-14.127,1)
(0,4)	(38.839,-0.008,-14.533,1)	(4,4)	(-13.789,39.283,-14.541,1)
(0,5)	(37.310,0.012,-14.854,1)	(4,5)	(-13.247,37.773,-14.849,1)
(0,6)	(36.358,-0.005,-14.957,1)	(4,6)	(-12.893,36.737,-14.958,1)
(0,7)	(35.690,-0.046,-14.981,1)	(4,7)	(-12.422,36.134,-14.991,1)
(1,0)	(41.172,9.033-13.700,1)	(5,0)	(-35.909,26.414,-13.232,1)
(1,1)	(39.344,8.631,-14.552,1)	(5,1)	(-36.144,26.029,-13.2071)
(1,2)	(39.447,8.613,-14.399,1)	(5,2)	(-35.399,25.783,-13.607,1)
(1,3)	(37.938,8.303,-14.871,1)	(5,3)	(-34.598,25.034,-14.031,1)
(1,4)	(36.736,7.985,-14.987,1)	(5,4)	(-33.427,24.261,-14.489,1)
(1,5)	(35.137,7.694,-15.136,1)	(5,5)	(-32.168,23.280,-14.822,1)
(1,6)	(34.246,7.340,-15.0321)	(5,6)	(-31.301,22.710,-14.946,1)
(1,7)	(32.565,7.961,-15.196,1)	(5,7)	(-31.168,22.130,-14.954,1)
(2,0)	(35.871,26.501,-13.228,1)	(6,0)	(-41.178,8.980,-13.698,1)
(2,1)	(36.115,26.120,-13.198,1)	(6,1)	(-39.354,8.582,-14.547,1)
(2,2)	(35.367,25.872,-13.602,1)	(6,2)	(-39.455,8.565,-14.397,1)
(2,3)	(34.570,25.121,-14.025,1)	(6,3)	(-37.948,8.256,-14.866,1)
(2,4)	(33.401,24.346,-14.485,1)	(6,4)	(-36.748,7.941,-14.985,1)
(2,5)	(32.130,23.363,-14.810,1)	(6,5)	(-35.141,7.651,-15.135,1)
(2,6)	(31.279,22.790,-14.945,1)	(6,6)	(-34.260,7.300,-15.031,1)
(2,7)	(31.140,22.203,-14.953,1)	(6,7)	(-32.582,7.905,-15.195,1)
(3,0)	(14.842,42.583,-13.274,1)	(7,0)	(-42.031,-0.113,-13.254,1)
(3,1)	(14.678,42.377,-13.381,1)	(7,1)	(-41.859,0.019,-13.346,1)
(3,2)	(14.510,41.757,-13.692,1)	(7,2)	(-41.221,-0.029,-13.669,1)
(3,3)	(14.103,40.667,-14.131,1)	(7,3)	(-40.150,0.003,-14.113,1)

(3,4)	(13.649,39.328,-14.543,1)	(7,4)	(-38.831,-0.008,-14.533,1)
(3,5)	(13.114,37.816,-14.850,1)	(7,5)	(-37.323,0.012,-14.854,1)
(3,6)	(12.761,36.778,-14.959,1)	(7,6)	(-36.349,-0.004,-14.957,1)
(3,7)	(12.295,36.179,-14.991,1)	(7,7)	(-35.683,-0.041,-14.982,1)

Table 4 Surface Approximation of Inner Surface.

Surface Degree	*U-direction*	3	*V-direction*	3
Knot Vectors	**U**=(0,0,0,0,0.197,0.403,0.598, 0.804,1,1,1,1)		**V**=(0,0,0,0,0.138,0.339,0.550, 0.769,1,1,1,1)	
Control Polygon	*The indices i,j refer the equation* 1.			

(*i,j*)	**Control Point**	(*i,j*)	**Control Point**
(0,0)	(34.686,-0.095,-15.813,1)	(4,0)	(-12.377,35.102,-15.816,1)
(0,1)	(34.591,-0.077,-16.052,1)	(4,1)	(-12.296,34.989,-16.097,1)
(0,2)	(34.271,-0.070,-16.727,1)	(4,2)	(-12.210,34.683,-16.745,1)
(0,3)	(33.704,-0.067,-17.741,1)	(4,3)	(-11.988,34.102,-17.772,1)
(0,4)	(32.968,-0.064,-18.830,1)	(4,4)	(-11.737,33.355,-18.860,1)
(0,5)	(32.073,-0.033,-19.919,1)	(4,5)	(-11.389,32.460,-19.947,1)
(0,6)	(31.431,-0.052,-20.571,1)	(4,6)	(-11.201,31.786,-20.608,1)
(0,7)	(31.043,-0.061,-20.952,1)	(4,7)	(-10.911,31.430,-20.978,1)
(1,0)	(34.533,7.684,-16.146,1)	(5,0)	(-29.615,21.739,-15.784,1)
(1,1)	(34.036,7.658,-17.155,1)	(5,1)	(-29.641,21.512,-15.941,1)
(1,2)	(33.890,7.536,-17.413,1)	(5,2)	(-29.306,21.434,-16.666,1)
(1,3)	(33.113,7.415,-18.667,1)	(5,3)	(-28.874,21.013,-17.651,1)
(1,4)	(32.259,7.136,-19.763,1)	(5,4)	(-28.233,20.600,-18.743,1)
(1,5)	(31.142,6.972,-20.915,1)	(5,5)	(-27.539,19.994,-19.821,1)
(1,6)	(30.546,6.609,-21.474,1)	(5,6)	(-26.915,19.662,-20.491,1)
(1,7)	(29.683,7.242,-21.924,1)	(5,7)	(-26.821,19.191,-20.818,1)
(2,0)	(29.579,21.807,-15.781,1)	(6,0)	(-34.537,7.630,-16.144,1)
(2,1)	(29.608,21.580,-15.933,1)	(6,1)	(-34.042,7.605,-17.150,1)
(2,2)	(29.272,21.501,-16.660,1)	(6,2)	(-33.895,7.483,-17.409,1)
(2,3)	(28.841,21.079,-17.644,1)	(6,3)	(-33.119,7.362,-18.662,1)
(2,4)	(28.202,20.666,-18.736,1)	(6,4)	(-32.264,7.088,-19.758,1)

(2,5)	(27.509,20.060,-19.814,1)	(6,5)	(-31.148,6.924,-20.911,1)
(2,6)	(26.888,19.726,-20.482,1)	(6,6)	(-30.552,6.565,-21.468,1)
(2,7)	(26.793,19.251-20.815,1)	(6,7)	(-29.695,7.183,-21.923,1)
(3,0)	(12.262,35.139,-15.819,1)	(7,0)	(-34.684,-0.091,-15.814,1)
(3,1)	(12.181,35.026,-16.102,1)	(7,1)	(-34.588,-0.072,-16.053,1)
(3,2)	(12.095,34.719,-16.749,1)	(7,2)	(-34.268,-0.064,-16.729,1)
(3,3)	(11.876,34.137,-17.776,1)	(7,3)	(-33.702,-0.062,-17.742,1)
(3,4)	(11.627,33.389,-18.865,1)	(7,4)	(-32.966,-0.061,-18.831,1)
(3,5)	(11.281,32.492,-19.951,1)	(7,5)	(-32.072,-0.031,-19.920,1)
(3,6)	(11.096,31.817,-20.614,1)	(7,6)	(-31.428,-0.049,-20.574,1)
(3,7)	(10.807,31.463,-20.980,1)	(7,7)	(-31.040,-0.056,-20.952,1)