Robust Computational Methods for Shape Interrogation

by

Takashi Maekawa

O.E. in Ocean Engineering, M.I.T., June, 1987
M.S. in Mechanical Engineering, Waseda University, March, 1978
B.S. in Mechanical Engineering, Waseda University, March, 1976

Submitted to the Department of Ocean Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1993

Author .....
Department of Ocean Engineering
June 18, 1993

Certified by .....
Nicholas M. Patrikalakis
Associate Professor of Ocean Engineering
Thesis Supervisor

Accepted by .....
A. Douglas Carmichael
Chairman, Departmental Committee on Graduate Students

Robust Computational Methods for Shape Interrogation

by

Takashi Maekawa

Submitted to the Department of Ocean Engineering
on June 18, 1993, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Robustness and accuracy are two of the most fundamental outstanding problems in computational geometry and geometric modeling, despite significant advances of these fields in the last three decades. Since curves and surfaces are typically represented by parametric piecewise polynomial equations, the governing equations for geometric processing and shape interrogation in general reduce to solving systems of nonlinear polynomial equations or irrational equations involving nonlinear polynomials and square roots of polynomials. The square root arises for example from the normalization of the normal vector and from the analytical expressions of curvatures. This thesis addresses the development of a new robust and accurate computational method to compute all real roots of such systems within a finite box. A key component of our method is the reduction of the problem involving irrational equations into solution of systems of nonlinear polynomial equations of higher dimensionality through the introduction of auxiliary variables. A fundamental component of our method for solving general non-linear polynomial equation systems is rounded interval arithmetic in the context of Bernstein subdivision. Rounded interval arithmetic leads to numerical robustness and provides results with numerical certainty and verifiability. Several computational geometry applications of the new nonlinear solver are studied to evaluate the method in a realistic context. These applications include continuous decomposition of parametric polynomial surface patches into a set of trimmed patches each with a specified range of curvature (Gaussian, mean, maximum principal, minimum principal and root mean square curvature). Such surface decomposition has applications in sculptured surface fairing for design, tessellation for analysis, and manufacture. Another application includes the computation of self-intersections of offset curves or of intersections of two offsets of two curves. Such algorithm can be applied to design specification, feature recognition through construction of skeletons of geometric models, and manufacture. Finally, the nonlinear solver is used in computing all umbilics of parametric polynomial surface patches for application in surface recognition and tessellation.

Thesis Supervisor: Nicholas M. Patrikalakis
Title: Associate Professor of Ocean Engineering

Acknowledgements

This dissertation is dedicated to my wife Yuko and my son Takuya. Without their love, support, encouragement and understanding this dissertation would never have been completed.

I express my sincere appreciation to Professor Nicholas M. Patrikalakis for his excellent advice on research and other professional concerns. He has been a wonderful mentor and a long time friend. Regardless of his work load, he always found time to discuss with me, which is a very difficult thing to do. I feel privileged for having the opportunity to work with him.

I am also grateful to thesis committee members Professor Chryssostomos Chryssostomidis, who has established and directs the Design Laboratory and Professor David C. Gossard, director of the CAD Laboratory, for their intellectual feedback and great interest in the progress of this research.

I also enjoyed countless discussions with Dr. Franz -Erich Wolter, who is a highly skilled mathematician and also a member of my thesis committee.

I would also like to thank Mr. Michael S. Drooker, manager of the Design Laboratory, for his great assistance in computer systems and Mr. Stephen L. Abrams, who is an exceptionally competent software engineer.

I wish to thank all the members of the MIT Design Laboratory and friends for their support and friendship during my stay at MIT. Special thanks go to the CAD group future doctors, Chun-Yi Hu, Evan Sherbrooke, Seamus Tuohy, Jingfang Zhou and also a former CAD group future doctor Bradley Moran for their support and stimulating discussions which led to some of the ideas in my dissertation.

I also want thank Hui Nam, who as a UROP student, assisted in some of the early implementations.

This work was supported, in part, by the MIT Sea Grant College Program, the Office of Naval Research and National Science Foundation in the USA under grant numbers NA90AA-D-SG-424, N00014-91-J-1014 and DDM-9215411.

Finally, I would like to acknowledge my father Jiro and my late mother Mitsuko for their love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Robust Shape Interrogation

Computer Aided Design (CAD) and Computer Aided Manufacturing (CAM) made significant advances in the last three decades, however many of the advances are intended to support rather replace human labors. Furthermore CAD and CAM have been developed rather independently and the linkage between them is not automated. Especially for complex sculptured or free-form objects, the linkage relies on the skill and experience of a qualified engineer. We refer to planar free-form curves and free-form surfaces in 3D space as free-form objects. Free-form surfaces, also called sculptured surfaces, are widely used in scientific and engineering applications. For example, the shape of propeller and turbine blades significantly affects performance. Free-form surfaces arise in the bodies of the ships, automobiles and aircraft, which have both functionality and attractive shape requirements. Free-form curves and surfaces are usually represented by parametric equations. The use of the parametric representation provides an efficient way to generate data points explicitly and avoids axis dependence. One of the obstacles, which prevents the implementation of entirely automatic manufacturing of free-form objects, is the lack of robustness of the interrogation tools. Robustness is one of the most important key factors of automated solution procedures. If robustness is not guaranteed, then the design or manufacturing engineer needs to manually or visually verify the results and use exceedingly suboptimal procedures, sometimes in the form of large safety factors to account for non-robust computation.

1.2 Research Objective

The objective of this thesis is to develop a robust shape interrogation methodology of free-form objects to support automation of design and manufacture. The word *interrogation* refers to extraction of all the required differential and global geometric properties of the free-form object and the adjective *robust* means that the extraction is done automatically without failure.

Most of the existing interrogation algorithms are based on *local* and *discrete* methods and the computation involves *numerical uncertainty* which do not guarantee to extract all the necessary information from the free-form object. Since curves and surfaces are usually represented in parametric piecewise polynomial equations, the governing equations for interrogation reduce to systems of nonlinear polynomial equations, frequently involving also square roots of polynomial equations, which arise from normalization of the normal vector and analytical expressions of the principal curvatures of the surface. These systems of equations have been solved in earlier work by *local* numerical techniques such as Newton-type methods which require good initial approximation to all roots, and hence cannot provide full assurance that all roots will be found. On the other hand *global* techniques find all the roots without initial approximation. More than 80% of all mechanical parts which are manufactured by the numerically controlled (NC) machining can be produced by $2\frac{1}{2}D$ pocket machining. The programming procedure includes computing offsets of planar curves. Traditional techniques for computing offsets are based on *local* methods and do not guarantee to eliminate all the possible loops created by self-intersections, while *global* techniques guarantee to find all the possible self-intersections. Propeller and turbine blades are manufactured by $3D$ or $5D$ milling machines. The programmer must know the exact range of the curvature to select the optimal combination of tool path and cutter size for NC machining. Currently *discrete* color coded maps are used to estimate the range of principal curvatures but are not sufficient to provide detailed machining information. *Continuous* decomposition of surfaces on the basis of curvature provides exact range of curvatures and is able to supply detailed machining information. The degrees of some of the governing equations for interrogation are relatively high and if floating point arithmetic is employed for the computation, there exists substantial *numerical uncertainty* in the formulation and solution process. If the computation is conducted with intervals, it is *continuous* within the interval even if it is computed with finite precision arithmetic and can obtain the results with *numerical certainty*.

This thesis concentrates on a different methodology for shape interrogation based on three characteristic properties which lead to robustness:

- **global** versus local
- **continuous** versus discrete
- computation with **numerical certainty** versus numerical uncertainty

1.3 Thesis Organization

The thesis is organized as follows.

Chapter 2 presents the four major applications and a literature review on robust shape interrogation. The four major applications motivating this work are *machining*, *fairing*, *tessellation* and *recognition* of free-form objects.

Chapter 3 reviews global methods for solving a system of nonlinear polynomial equations, and introduces a new robust solver for a system of nonlinear polynomial equations and also discusses the robust implementation of such methods. A variety of numerical experiments are conducted to evaluate the robustness, accuracy and efficiency of the method.

Chapter 4 reviews the differential geometry properties of planar curves and studies the computation of singularities and intersections of offsets of planar polynomial parametric curves and applies this technique to $2\frac{1}{2}$D pocket machining.

Chapter 5 reviews the differential geometry properties of parametric surfaces and proposes a new method for continuous decomposition of polynomial parametric surface patches based on various curvature measures including Gaussian, mean, maximum and minimum principal curvatures.

Chapter 6 extends the method described in Chapter 5 to automatic triangular mesh generation (tessellation) to accurately approximate a free-form parametric polynomial surface using continuous decomposition of polynomial parametric surface patches based on root mean square curvature.

Chapter 7 presents a procedure to locate all umbilics and compute the lines of curvature near an umbilic on a parametric surface for application in surface recognition problems.

Chapter 8 summarizes the contributions of this thesis and provides recommendations for further research in this area.

Appendix A provides the formulas for evaluating the derivatives of curvature which are used in Chapters 5 and 6.

Appendix B reviews the classification of stationary points of functions.

Appendix C proves a condition for the gradient of the mean curvature to become zero which is used in section 7.5.

Chapter 2

Background and Motivation

2.1 Introduction

In this Chapter we will present four engineering applications of robust computational methods for shape interrogation, which have motivated our research. All these four applications contribute towards automation of design and manufacturing systems. These applications are: NC machining of free-form objects, fairing, tessellation of free-form surfaces, and free-form surface recognition in computer vision.

2.2 Machining

2.2.1 Terminology in Machining

The purpose of milling is to remove material from a workpiece. The material is removed in the form of small chips produced by the milling cutter which rotates at a high speed. The milling machine is one of the most versatile machine tools used in industry. The adaptation of the numerical control (NC) technique to milling machines has made this type of machine tool even more versatile [72]. The operation of an NC machine is controlled by a program written in an NC language. The program is executed by the NC machine-controller system. A machine tool is characterized by the motions it can perform. Such motions as changing the relative position of the tool and workpiece consist of linear translations and rotations about different axes. However, they do not include the rotation of the cutter or workpiece for maintaining cutting action. NC machines are classified as follows

[72],[35].

2-D Milling : 2-D milling refers to the contouring capability of the machine tool limited to the xy-plane. By moving the x and y axes simultaneously, while keeping z axis constant, a complete 360 degrees contouring capability can be achieved.

$2\frac{1}{2}$-D Milling : $2\frac{1}{2}$-D milling has a capability between 2-D and 3-D milling. In $2\frac{1}{2}$-D milling, the cutting tool can follow any arbitrary curve in the xy-plane, but can only move stepwise in the z direction. This $2\frac{1}{2}$-D milling is also referred as *pocket machining*.

3-D Milling : 3-D milling refers to a cutting tool moving simultaneously in the x, y and z axes, but is not capable of performing tool rotation with respect to the workpiece.

5-D Milling : A rotation around two of the axes x, y and z is added to the x, y and z translation, hence the tool orientation can vary. The 5-D milling is suitable for large production runs, because the two additional rotations will reduce the required setups significantly [70].

To avoid ambiguities in the following discussion, we introduce some terminology using Figure 2-1.

CC point (Cutter Contact point) : As shown in Figure 2-1, the point $\mathbf{r}$ on the part surface at which the ball end-mill is to touch is called CC point.

CC data (Cutter Contact data) : Let $\mathbf{N}$ be the unit surface normal vector at point $\mathbf{r}$. Then the pair $(\mathbf{r}, \mathbf{N})$ is called CC data.

Offset point : The center of the hemisphere of the end-mill is called an offset point $\mathbf{c}$, and is given by $\mathbf{c} = \mathbf{r} + R\mathbf{N}$, where R is radius of the sphere.

Cutter reference point : The tip of the cutter $\mathbf{b}$ is the cutter reference point and is given by $\mathbf{b} = \mathbf{r} + R(\mathbf{N} - \mathbf{u})$ where $\mathbf{u}$ is the unit vector along the cutter axis. For a three axis machine $\mathbf{u} = (0, 0, 1)$.

CL data (Cutter Location data) : The pair $(\mathbf{b}, \mathbf{u})$ is called CL data.

CC path : A sequence of line segments obtained by connecting CC points are called CC path.

The success of NC milling highly depends on the availability of efficient algorithms of defining the *tool path*. The procedure for defining the tool path is divided into two parts : *path generation* and *cutter location calculation*. The path is a description of the tool sequence such as *zigzag*, *spiral* etc.,

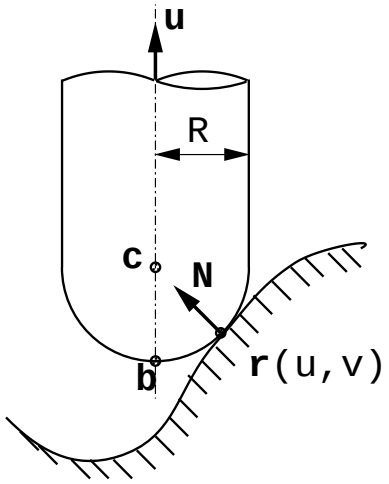

Figure 2-1: CC data and CL data

while the cutter location is the position of the cutter tip. The cutter motion for machining a part consists of *roughing*, *semi-roughing* and *finishing*, and should be considered separately, as illustrated in Figure 2-2. For each process, correct tool size and tool path needs to be determined.

rough machining : It should be as simple as possible and preferably consist of a linear type motion only to minimize machining time. In other words, the cutter path should be as short as possible and the depth of cut and feedrate should be as large as possible.

semi-rough machining : After rough machining, the *shoulders* left on the part should be removed.

finishing machining : The cutter should follow the profile during these operations and the deviations of the cutter from the profile should always be maintained within a designated tolerance.

2.2.2 2D Global Offsets for Pocket Machining

More than 80% of all mechanical parts which are manufactured by milling machines can be cut by NC pocket machining [35]. This is based on the facts that most mechanical parts consist of faces parallel or vertical to xy-plane, and that free-form objects are usually produced from a raw stock by $2\frac{1}{2}$-D roughing and $3D$ or $5D$ finishing. Persson's early work [79] is one of the first to study the spiral pocket machining using Voronoi diagrams. A book by Held [35] reviews all the related work and introduces an algorithm for the determination of tool paths for spiral and zig-zag milling,

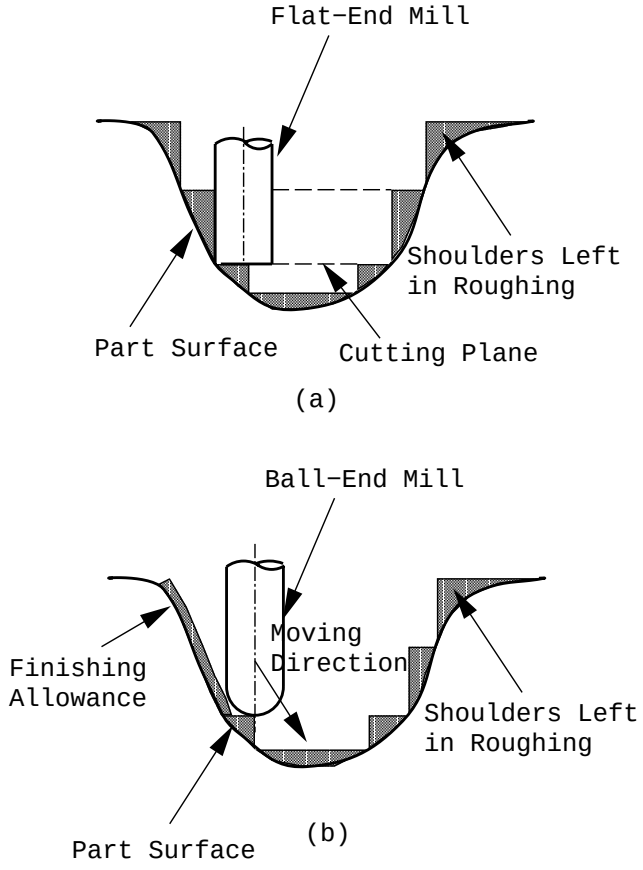

Figure 2-2: (a) Pocket machining with flat end-mill in roughing; (b) semi-roughing with large ball end-mill.(Adapted from [57])

and the optimization of tool paths. Held's spiral algorithm, based on Persson's ideas provides a general concept for fully automated pocket machining. The spiral $2\frac{1}{2}$-D pocket machining is based on offsetting. The algorithms by Persson and Held are restricted to boundary contours of straight line and circular arcs, due to the fact that intersection algorithms for more complex boundaries are complex and difficult [35]. When a cylindrical end-mill cutter in numerically controlled (NC) $2\frac{1}{2}$-D pocket machining is used and the cutter is located on the side of the curve where the center of curvature lies, the cutter radius must be smaller than the smallest radius of curvature of the boundary curve of the part to be machined to avoid local overcut (gouging). Gouging is the one of the most critical problems in NC pocket machining. Therefore, we must determine the distribution of the curvatures along the boundary curve, to select the cutter size [77], [59]. Tool paths are generated by offsetting at a distance equal to the radius of the cutter from the boundary curve. Figure 2-3 shows the tool path of a cylindrical cutter pocket machining a region where the center of curvature of the parabolic boundary curve lies. The parabola $\mathbf{r}(t) = [t, t^2]^T$ has the maximum curvature at $(0, 0)$ with curvature value $\kappa = 2$. Thus if the radius of the cylindrical cutter exceeds 0.5, there will be a region of gouging as depicted in Figure 2-4 (a), where the cutter has a radius 0.8. Also the offset with $d = -0.8$ has one self-intersection and two cusps. The segment of the offset bounded by the self-intersecting points on the offset have distance less than the nominal offset distance 0.8 from the generator and this fact causes the gouging. Therefore, if we trim off the region of the offset bounded by the two parameters associated with the self-intersection, the cutter will not overcut the part but will leave an undercut region, see Figure 2-4 (b). The undercut region must be revisited with the smaller size cutter. This trimmed offset curve is referred to as *global* offset and each point on the curve is at least distance $|d|$ from every point on the progenitor [26]. Therefore computing the self-intersection points of the offset of a progenitor curve is very important. As already mentioned, existing algorithm are restricted to boundary contours made up of straight lines and circles. Therefore computing the self-intersections of the offsets reduce to computing the intersections of a straight line to a straight line, a circle to a circle or a straight line to a circle. A brute force approach takes $\mathcal{O}(n^2)$ time for computation where n is the number of segments plus the number of reflex vertices. Reflex vertices have an interior angle larger than π. Consequently approximating with straight lines and circles not only reduces accuracy but is also computationally expensive when n is large. In Chapter 4, we introduce a new robust and efficient method for computing the singularities of a normal offset of a planar integral polynomial curve and the intersections of two specific normal offsets of planar integral polynomial curves.

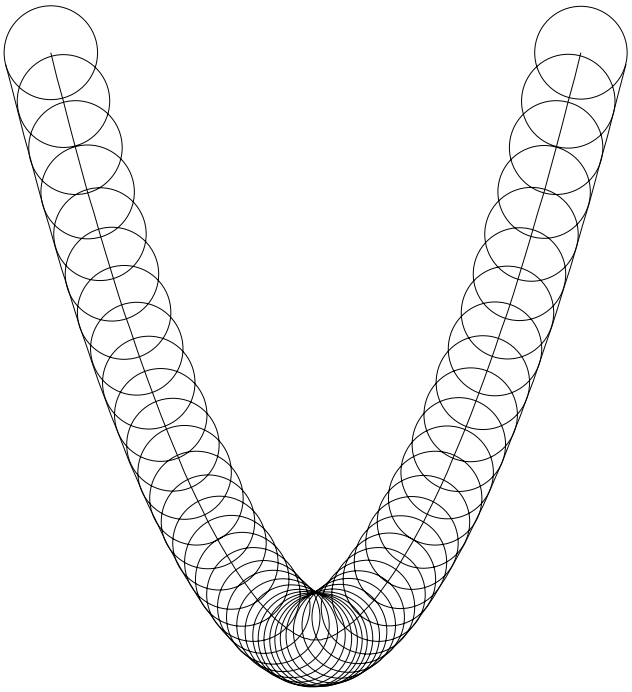

Figure 2-3: Interior offsets to the parabola $\mathbf{r}(t) = [t, t^2]^T$ with d=-0.3 and cutter path

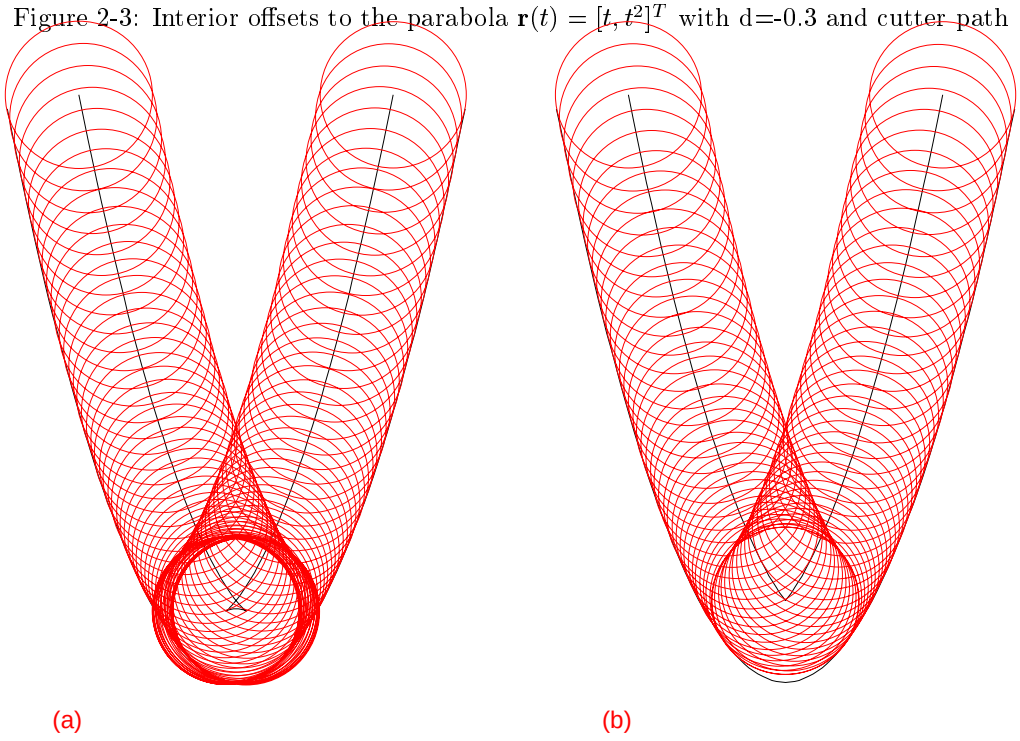

Figure 2-4: (a) Interior offsets to the parabola $\mathbf{r}(t) = [t, t^2]^T$ with d=-0.8 and cutter path; (b) trimmed interior offsets to the parabola $\mathbf{r}(t) = [t, t^2]^T$ with d=-0.8 and cutter path

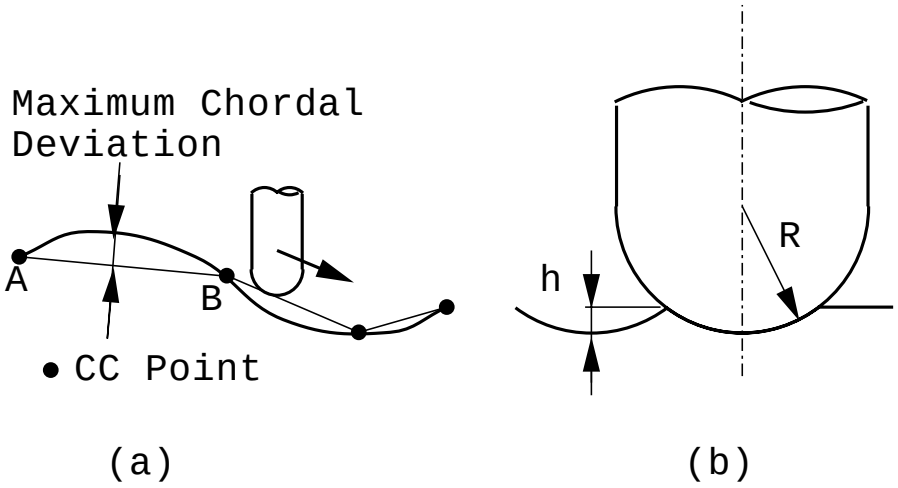

Figure 2-5: Tolerance in free-form surface machining (a) Maximum chordal deviation; (b) Scallop height

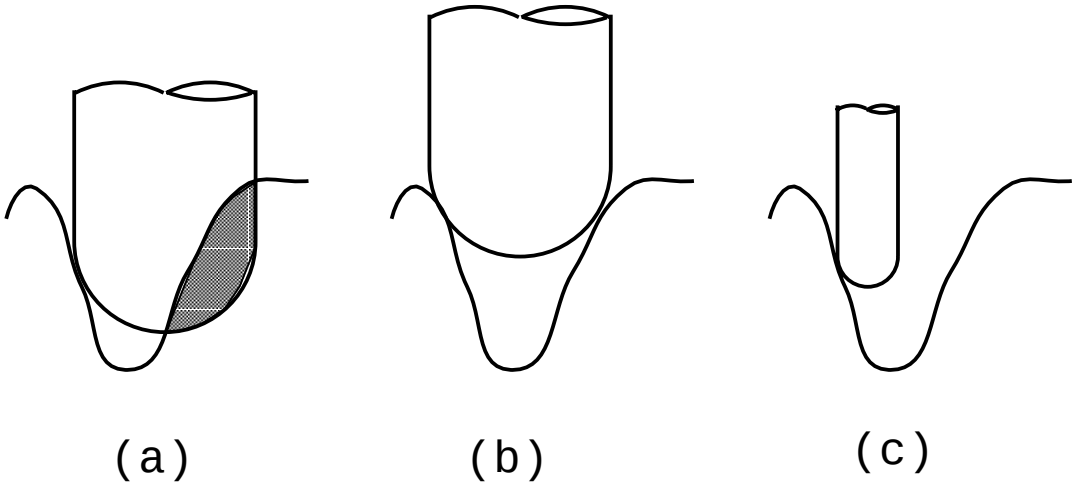

Figure 2-6: Gouging on sculptured surface (a) Gouging; (b) Undercut; (c) Removing the undercut part, using smaller size ball end-mill

2.2.3 Curvature Maps for 3D and 5D Machining

The general method for $3D$ machining a free-form surface by a ball end-mill cutter is as follows:

Step 1: Part surface is intersected with a plane to obtain the intersection curve. Some systems use isoparametric curves instead of intersection curves by fixing one of the parameters of the parametric surface. Although isoparametric curves are easy to generate, this method has a problem when the surface is very narrow on one side compared to the opposite side. There will be a repeated overlapping motion on the narrow side.

Step 2: The intersection curve is approximated by a sequence of linear segments since most NC machines are only capable of executing linear and circular interpolation. A straight line segment is drawn from A to B and the maximum chordal deviation is calculated, see Figure 2-5 (a). If the deviation is greater than the input tolerance, the curve is subdivided until the chordal deviation is within the tolerance. For detail, see Faux and Pratt [28].

Step 3: Side step (path interval) is determined such that the scallop height which is the cusp height of the material removed by the cutter is within the tolerance, as shown in the Figure 2-5 (b).

Step 4: After all the CC data are obtained, CL data are computed.

Step 4 is actually the most difficult task in the whole process. When a ball end-mill cutter is used, the cutter radius must be smaller than the smallest concave radius of curvature of the surface to be machined to avoid local gouging (see Figure 2-6 (a)). Choi and Jun [12] introduces an algorithm which avoids gouging by comparing each CC point with adjacent CC points which locates within the projection of the ball end-mill on the xy-plane. The gouge test is checked by computing the $3D$ distance between the offset point and each CC point within the projected circle of the cutter. If any of the distance is less than the cutter radius, the offset point becomes an gouging point. Kuragano et al [52] generate a polygonal offset surface by connecting the offset points. The intersection between the offset surface with the plane in Step 1 becomes the tool path. When there is a self-intersection in the polygonal offset surface, the portion bounded by the self-intersection lines is trimmed off. Therefore, existing methods rely on discrete point data approximation, which does not guarantee to avoid gouging. Consequently, a robust procedure for decomposing a free-form surface into regions of specific range of principal curvature is desirable. In Chapter 5 a new method to decompose a free-form surface patch into sub-patches with specific range of curvature is introduced.

2.3 Fairing

Fairing is a process of eliminating undesirable shape features, in order to produce a smoother shape. It should be noted here that the purpose of the fairing process is not only to remove shape irregularities by eliminating oscillations and unwanted inflections of the surface but also to preserve the shape and keep it as consistent as possible with the design requirements [1].

Surface inflection exists on a surface at a point **P** if the surface crosses the tangent plane at **P**. The Gaussian curvature gives us information about surface inflections i.e. if the Gaussian curvature is negative at a point on a surface there is a surface inflection at the point. If the surface is developable or the Gaussian curvature is very close to zero, the Gaussian curvature cannot provide us with adequate information about the shape of the surface. In such case the mean curvature is also needed. If the Gaussian curvature is zero in a region then the surface has an inflection point in that region only if the mean curvature changes sign. The Gaussian and mean curvatures provide us with sufficient information in order to identify surface inflections on a surface [67] [66]. Similarly to motivation for 3D and 5D machining, a robust procedure for decomposing a free-form surface into regions of positive and negative Gaussian and mean curvatures is desirable for fairing. This topic will be discussed in Chapter 5.

2.4 Surface Tessellation

2.4.1 Finite Element Meshing

Design engineers spend a large amount of time for analysis. For a preliminary design, most of the analysis is conducted by simple analytical methods and handbooks to obtain a rough specification of the design. But as the design process proceeds, analysis is conducted more precisely using the finite element method if the problems are complex. The finite element method is one of the most useful general analysis tools for solving complex problems in engineering. The finite element analysis consists of three basic steps, the formulation of the problem in variational form, the finite element discretization of the governing equations, and solving the resulting system of algebraic or ordinary differential equations. The success of finite element analysis largely depends on partitioning the problem domain into a finite element mesh. Mesh generation is often the most time consuming process in the analysis, usually involving more time than the solution process itself. The accuracy and cost of the analysis also depends directly on the size, shape and number of mesh elements. Ho-Le

[37], Gursoy [31] review and classify existing algorithms for 2D (planar) and 3D (solid) automatic mesh generation. There are essentially two geometric types of meshes for partitioning a planar domain, triangular and quadrilateral meshes. Small and large angles in a triangular mesh and large aspect ratio in a quadrilateral mesh imply generally bad conditioning. In other words, if the ratio of the radius of the inscribed circle to the radius of the circumscribed circle is very small, then the linear system is usually poorly conditioned. As a rule of thumb, we require that all mesh elements resemble equilateral triangles for a triangular mesh and squares for a quadrilateral mesh. Another key point for successful meshing leading to greater efficiency is that mesh elements should be small in a region where the variables are changing rapidly to achieve accuracy, while larger mesh elements should be provided to in a region where the variables are changing slowly. For many problems in applied mechanics involving free-form surfaces, the variables frequently change rapidly in regions where absolute values of curvature are large, while the variables change slowly in regions where absolute values of curvature are small.

2.4.2 Computer Graphics

Rendering of NURBS and Bézier surfaces is becoming increasingly important in *animation* and *scientific visualization*. The quality of the display of surfaces largely depends on how faithfully the discretized surface represents the geometry and the topology of the original surface. Most of the existing algorithms for rendering surfaces do not reflect the differential geometry of the surface, rather they uniformly tessellate the region into a grid of rectangles in the uv space and map to the 3D space in order to permit computation in real time [87]. But to obtain a high quality image at low cost, high curvature regions must be meshed densely and low curvature regions sparsely.

2.4.3 Solid Free-Form Fabrication

In solid free-form fabrication methods, a tessellated surface needs to be exchanged from the design to the manufacturing system due to the early data format standards used in this method. To permit accurate manufacture, enormous data files need to be exchanged. Tessellating the surface adaptively, can significantly reduce the number of necessary triangular facets and memory needed in this process.

Therefore tessellating free-form surfaces based on the curvature level is desirable for finite element discretization, computer graphics (rendering) and solid free-form fabrication. In Chapter 6, a new method for tessellating a free-form surface based on the curvature level is discussed.

2.5 Surface Recognition

A robot manipulator becomes more intelligent by the aid of a computer vision system, see Figure 2-7. A computer vision system can identify 3-D surfaces belonging to 3-D objects, therefore can deal with variations in part position and orientation.

A surface can be measured by single-view range imaging sensors in a set of coordinates $z_{ij} = f(x_i, y_i)$ [6]. Magnetic resonance imaging (MRI) systems and computerized tomography (CT) allow the direct measurement of 3-D objects. Typically the data obtained by the sensors are noisy and need to be processed using image processing techniques for further computation.

If we can accurately estimate the first and second partial derivatives from these data, we can compute invariant geometric quantities using differential geometry properties. The derivatives can be estimated in two ways, a local approximation and a global approximation. A local approximation is based on estimating the derivatives at a given point with its neighboring points by taking the appropriate finite differences. One way to estimate the appropriate difference is to fit a surface locally and compute the derivatives at the point with the derivatives of the fitted function. One of the disadvantages of local surface fitting is that a discontinuity usually exists between the local patches. On the other hand, a single piecewise polynomial fit using B-spline provides a global approximation, hence consistent differential properties can be obtained. The disadvantages of the global fit is that the fit usually acts as a low-pass filter and the high frequency oscillations of the surface may be lost. Occasionally also, extraneous oscillations not implied by the data may be introduced in both local and global fitting methods. Since the data collected by a sensor contain error and noise and the fit is an approximation, the reconstructed surface is different from the real surface. Therefore the computer needs generic information from the surface. It is well known from the last century that there are three types of generic umbilics [14]. In Chapter 7 we investigate a theoretical and numerical implementation of generic feature extraction of umbilics of free-form surfaces for surface recognition.

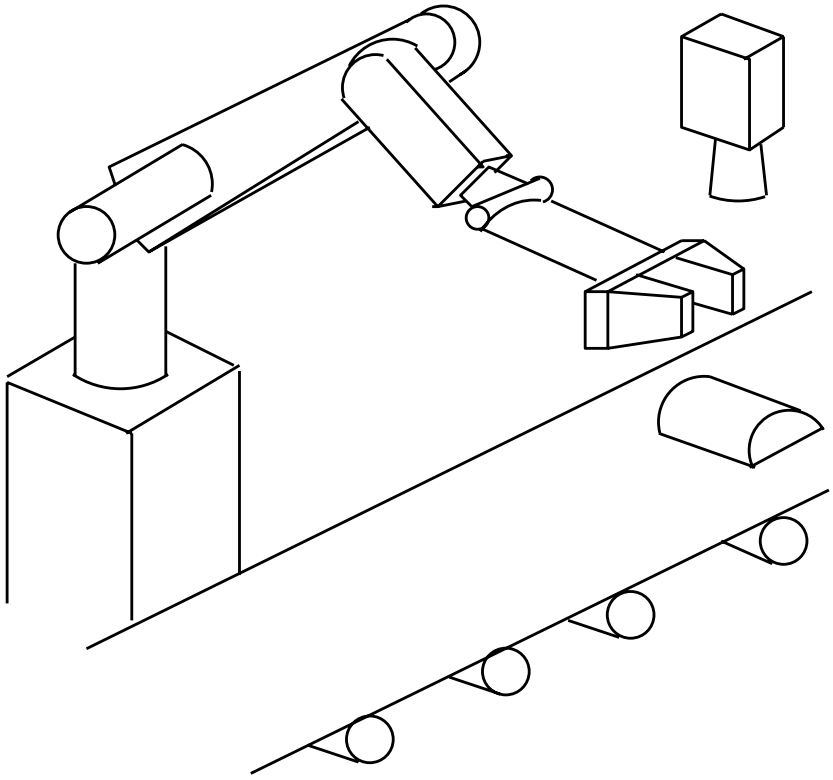

Figure 2-7: A robot manipulator with a computer vision system

Chapter 3

Robust Shape Interrogation

3.1 Introduction and Literature Review

Shape interrogation of free-form objects involves finding *all* solutions of a system of simultaneous, nonlinear equations. Usually the nonlinear equations consist of equations involving polynomials and square roots of polynomials. The square root destroys all the nice properties that a polynomial function possesses eg. convex hull property. Processing of square root functions is currently based on elementary lattice methods, which do not guarantee computation of all roots [23], [67], [66], and squaring methods or elimination methods [105], [25]. Squaring and elimination methods lead to systems of polynomial equations of higher degree. Therefore, we will first briefly review the methods for computation of the roots of a system of simultaneous nonlinear polynomial equations.

In recent CAD-related research, three classes of methods for the computation of solutions of nonlinear polynomial systems have been favored: algebraic techniques, homotopy, and subdivision [93]. Beyond these methods, there are more recent techniques combining the advantages of algebraic and numerical techniques [62]. These methods may be classified as *global* because they are designed to compute all roots in some area of interest. There also exist a number of *local* numerical techniques which employ some variation of Newton-Raphson iteration or numerical optimization [13]. These methods are used in CAD applications requiring high accuracy because they are efficient (usually exhibiting quadratic convergence rates close to simple roots) and are straightforward to program. However, they typically require good initial approximations to roots; such approximations are usually obtained through some sort of global search like sampling, a process which cannot provide full

assurance that all roots have been found. This lack of robustness makes the development of efficient and stable global techniques desirable; in what follows, we briefly review three classes of global methods.

The computation of all complex roots of nonlinear polynomial systems has typically been approached with *algebraic geometry techniques* like elimination or Groebner basis methods [9] [10]. These methods have many advantages; they are theoretically elegant, guaranteed to find all complex roots of a system irrespective of the dimensionality of the solution set, and well-suited for implementation in symbolic mathematical systems [99]. However, they suffer from numerical instability, making implementation in floating point arithmetic difficult. Furthermore, they are inefficient in memory and processing time requirements and therefore unattractive except for low degree or dimensionality systems.

The second category of methods is the class of *homotopy techniques* [29] [112]. These methods may be used to find all complex solutions of a nonlinear polynomial system if the number of roots is finite. Unfortunately, investigation of such methods indicates that they also tend to be numerically ill-conditioned. If we try to get around this problem by implementing the algorithm in exact rational arithmetic, we end up with enormous memory requirements because we have to solve large systems of complex initial value problems. Furthermore, such techniques are excessive in many problems we encounter where we only need real roots within a bounded set.

The third class is the *subdivision-based techniques* which have been used in a wide variety of intersection problems for geometric modeling [73]. Lane and Riesenfeld [53] investigated the application of binary subdivision and the variation diminishing property of polynomials in the Bernstein basis to eliciting the real roots and extrema of a polynomial within an interval. Geisow [30] was among the first to use this technique in a study of surface intersections. Patrikalakis, Prakash, and Kriezis [83], [78], [51] investigated the use of subdivision of algebraic curves in intersecting an implicit algebraic surface with a rational polynomial surface. Their method relies on the computation of real characteristic points of an algebraic curve represented in the tensor-product Bernstein basis within a rectangle, which typically involves intersecting two or three algebraic curves by repeated adaptive subdivision and minimization. Minimization is used to increase the precision of the root quadratically. Sederberg [90] developed an adaptive subdivision algorithm to intersect planar algebraic curves expressed in the barycentric Bernstein basis within triangles. Nishita et al. [69] developed an adaptive subdivision technique to intersect rays with trimmed rational polynomial surface patches, also recasting the problem as the intersection of two algebraic curves expressed in the Bernstein

basis. Vafiadou and Patrikalakis [105] employed a two-dimensional adaptive subdivision algorithm coupled with minimization to ray-trace offset surfaces (outlined in section 3.2). Sherbrooke and Patrikalakis [93] develop the n-dimensional Projected-Polyhedron algorithm, along with the related Linear Programming approach for solving systems of n nonlinear polynomial equations (also outlined in section 3.2) and investigate the convergence and complexity properties of these methods; see, also Patrikalakis et al [77] and Zhou et al [113] for a summary and applications of the n-dimensional algorithm. It should be noted that subdivision techniques have a number of disadvantages. They are not as general as algebraic methods, since they are only capable of isolating zero-dimensional solutions. Furthermore, although the chances, that all roots have been found, increase as the resolution tolerance is lowered, there is no certainty that each root has been extracted. Lastly, subdivision techniques provide no explicit information about root multiplicities without additional computation. However, despite these drawbacks, their speed and stability make them attractive as root-finding schemes, see also Patrikalakis [73] for an overview of their application in intersection problems.

Subdivision methods described above belong to the general class of interval methods. *Rounded interval arithmetic (RIA) methods* guarantee not to miss solutions and are very attractive from the reliability point of view but are known to be expensive. Interval techniques, primarily interval Newton methods combined with bisection to ensure convergence, have been the focus of significant attention, see for example Kearfott [46], Neumaier [68]. Interval methods have been applied in geometric modeling and CAD. For example, Mudur and Koparkar [65], Toth [101], Enger [21], Duff [17] and Snyder [96] applied interval algorithms to geometry processing, whereas Sederberg and Farouki [92], Sederberg and Buehler [91] applied interval methods in approximation problems. Tuohy and Patrikalakis [104] applied interval methods in the representation of functions with uncertainty, such as geophysical property maps. Tuohy et al [103] and Hager [32] applied interval methods in robotics. Bliek [7] studied interval Newton methods for design automation and inclusion monotonicity properties in interval arithmetic for solving the consistency problem associated with a hierarchical design methodology. As shown in this thesis, de Casteljau subdivision methods coupled with rounded interval arithmetic can be effectively adapted to the computation of all real roots of systems of irrational equations involving polynomials and square roots of polynomials.

For the computation of all real roots of systems of irrational equations involving square roots of polynomials, subdivision methods, if they can be effectively adapted to such problems, are likely to be the most successful methods in practice. Algebraic techniques require elimination of the radical either by squaring [105] or by elimination methods [25] which lead to high degree polynomials and

hence are generally inefficient as we will see in section 5.5.2. Homotopy techniques are numerically ill-conditioned for high degree polynomials and similarly inefficient because they are exhaustive. Therefore, we focus on extensions of subdivision techniques for irrational functions. As mentioned earlier the key property of polynomials in the Bernstein basis, ie. the convex hull property, does not apply directly and alternate techniques need to be developed to permit effective application of subdivision [60].

This Chapter is organized as follows. Section 3.2 reviews the Bernstein subdivision method for systems of nonlinear polynomial equations. Section 3.3 presents the auxiliary variable method to handle irrational equations involving polynomials and square roots of polynomials. Section 3.4 reviews interval arithmetic and its algebraic properties and also explains the implementation of RIA. Section 3.5 overviews a robust and accurate method for problem formulation and solution such as RIA and rational arithmetic (RA). Section 3.6 describes a new robust and accurate Bernstein subdivision method coupled with the RIA. Finally section 3.7 conducts a comparison between Bernstein subdivision method coupled with RIA and an extant interval Newton method coupled with RIA.

3.2 Review of Bernstein Subdivision Method for Systems of Nonlinear Polynomial Equations

In this section we review the Bernstein subdivision method for systems of nonlinear polynomial equations, see Patrikalakis et al [77]. Suppose we solve a system of nonlinear polynomial equations $\mathbf{f} = (f_1, f_2, \ldots, f_n) = \mathbf{0}$ over the box $S \in \mathbf{R}^n$ where S is defined by

$$S = [a_1, b_1] \times [a_2, b_2] \times \ldots \times [a_n, b_n]. \tag{3.1}$$

That is, we wish to find all $\mathbf{u} \in S$ such that

$$f_1(\mathbf{u}) = f_2(\mathbf{u}) = \ldots = f_n(\mathbf{u}) = \mathbf{0}. \tag{3.2}$$

By making the *affine parameter transformation* [22] $u_i = a_i + x_i(b_i - a_i)$ for $i = 1, \cdots n$, we simplify the problem to the problem of determining all $\mathbf{x} \in [0, 1]^n$ such that

$$f_1(\mathbf{x}) = f_2(\mathbf{x}) = \ldots = f_n(\mathbf{x}) = \mathbf{0}. \tag{3.3}$$

Now furthermore suppose that each of the f_k is polynomial in the independent parameters $x_1, x_2, \ldots, x_n$. Let $m_i^{(k)}$ denote the degree of f_k in the variable x_i; then f_k can be written in the multivariate Bernstein basis:

$$f_k(\mathbf{x}) = \sum_{i_1=0}^{m_1^{(k)}} \sum_{i_2=0}^{m_2^{(k)}} \cdots \sum_{i_n=0}^{m_n^{(k)}} \bar{f}^{(k)}_{i_1 i_2 \ldots i_n} B_{i_1, m_1^{(k)}}(x_1) B_{i_2, m_2^{(k)}}(x_2) \ldots B_{i_n, m_n^{(k)}}(x_n). \tag{3.4}$$

where $B_{i,m}$ is the ith Bernstein polynomial given by

$$B_{i,m}(t) = \binom{m}{i} t^i (1-t)^{m-i} \tag{3.5}$$

The notation in (3.4) may simplified by letting $I = (i_1, i_2, \ldots i_n)$, $M^{(k)} = (m_1^{(k)}, m_2^{(k)}, \ldots, m_n^{(k)})$ and writing (3.4) in the equivalent form [93].

$$f_k(\mathbf{x}) = \sum_{I}^{M^{(k)}} \bar{f}_I^{(k)} \bar{B}_{I, M^{(k)}}(\mathbf{x}). \tag{3.6}$$

Here we have merely rewritten the product of Bernstein polynomials as a single *Bernstein multinomial* $\bar{B}_{I,M^{(k)}}(\mathbf{x})$. Bernstein polynomials have a useful identity called *linear precision property*, which is given by [22]

$$t = \sum_{i=0}^{m} \frac{i}{m} B_{i,m}(t) \tag{3.7}$$

In other words, the monomial t can be expressed as the weighted sum of Bernstein polynomials with coefficients evenly spaced in the interval $[0, 1]$. Using this property, we can rewrite the equation (3.6) equations as follows:

$$\mathbf{F}_k(\mathbf{x}) = \sum_{I}^{M^{(k)}} \mathbf{v}_I^{(k)} \bar{B}_{I, M^{(k)}}(\mathbf{x}) \tag{3.8}$$

where

$$\mathbf{v}_I^{(k)} = \left(\frac{i_1}{m_1^{(k)}}, \frac{i_2}{m_2^{(k)}}, \ldots, \frac{i_n}{m_n^{(k)}}, \bar{f}_I^{(k)}\right)^T. \tag{3.9}$$

These $\mathbf{v}_I^{(k)}$ are called the *control points* of $\mathbf{F}_k$.

Now the algebraic problem of finding roots of systems of polynomials has been transformed to the geometric problem involving the intersection of the hypersurfaces. Because the problem is now

phrased geometrically, we can use the *convex hull property* of the multivariate Bernstein basis to bound the set of roots.

For a fixed k, the convex hull C_k of the $\mathbf{v}_I^{(k)}$ is the set of points $p \in \mathbf{R}^{n+1}$ which can be expressed in the form

$$p = \sum_I^{M^{(k)}} c_I \mathbf{v}_I^{(k)} \tag{3.10}$$

for some c_I which must be nonnegative and which sum to 1.

Referring to (3.8) and noting that Bernstein multinomials are nonnegative for $\mathbf{x} \in [0,1]^n$ and sum to 1 (and therefore satisfy the restrictions on the c_I) leads immediately to

$$\mathbf{F}_k(\mathbf{x}) \in C_k \tag{3.11}$$

for $\mathbf{x} \in [0,1]^n$ and for $1 \leq k \leq n$.

It is apparent, then, that if $\mathbf{x}$ is a root of (3.3), then because $\mathbf{F}_k(\mathbf{x}) = (\mathbf{x}, 0)$ for each k, $(\mathbf{x}, 0)$ lies within each C_k. Thus if we were able to intersect the C_k with one another and with the hyperplane $x_{n+1} = 0$, the point $(\mathbf{x}, 0)$ would belong to the intersection set.

In practice, this intersection is a tedious task if more than one variable is involved. Fortunately, all we need out of this somewhat complicated intersection set is an n-dimensional rectangular box bounding the set of roots of (3.3), because the simple multivariate De Casteljau subdivision we will perform as a recursive step needs to work on a rectangular box. If we could find such a box, we could structure a root-finding algorithm as follows [77]:

1. Start with an initial box of search.

2. Scale the box and, as we did in converting between equations (3.2) and (3.3), perform an appropriate affine parameter transformation to the functions f_k, so that the box becomes $[0,1]^n$. However, keep track of the scaling relationship between this box and the initial box of search. This transformation can be performed with multivariate De Casteljau subdivision.

3. Using the convex hull property, find a sub-box of $[0,1]^n$ which contains all the roots. The essential idea behind the box generation scheme in this algorithm is to transform a complicated $n+1$-dimensional problem into a series of n two-dimensional problems. Suppose $\mathbf{R}^{n+1}$ can be coordinatized with the $x_1, x_2, \ldots, x_{n+1}$ axes; we can then employ these steps:

   (a) Project the $\mathbf{v}_I^{(k)}$ of all of the $\mathbf{F}_k$ into n different coordinate planes; specifically, the

(x_1, x_{n+1})-plane, the (x_2, x_{n+1})-plane, and so on, up to the (x_n, x_{n+1}) plane.

(b) In each one of these planes,

i. Construct n two-dimensional convex hulls. The first is the convex hull of the projected control points of $\mathbf{F}_1$, the second is from $\mathbf{F}_2$ and so on.

ii. Intersect each convex hull with the horizontal axis (that is, $x_{n+1} = 0$). Because the polygon is convex, the intersection may be either a closed interval (which may degenerate to a point) or empty. If it is empty, then no root of the system exists within the given search box.

iii. Intersect the intervals with one another. Again, if the result is empty, no root exists within the given search box.

(c) Construct an n-dimensional box by taking the Cartesian product of each one of these intervals in order. In other words, the x_1 side of the box is the interval resulting from the intersection in the (x_1, x_{n+1})-plane, and so forth.

4. Using the scaling relationship between our current box and the initial box of search, see if the new sub-box represents a sufficiently small box in $\mathbf{R}^n$. If it does, conclude that there is a root inside, and return it.

5. If any dimensions of this sub-box are not much smaller than 1 unit in length (i.e., the box has not decreased much in size along one or more sides), split the box evenly along each dimension which is causing trouble. Continue on the next iteration with several independent sub-problems.

6. Go back to step 2, once for each new box.

It can be shown [93] that this box does in fact contain all the roots within the given box of search. In [93] it is also shown that this algorithm gives rise to an algorithm which is quadratically convergent in one dimension but only linearly convergent for higher dimensional problems. This lowered rate of convergence arises from the "loss of information" involved in projecting these hypersurfaces. However, the cost per step involved in generating these boxes is sufficiently low to mitigate this problem [93].

We will demonstrate this problem with a single univariate polynomial equation $f(u) = 0$ of degree m over the range $a \leq u \leq b$. By making the affine parameter transformation $u = a + t(b - a)$

so that $0 \leq t \leq 1$, we can write $f(t)$ in Bernstein basis as:

$$f(t) = \sum_{i=0}^{m} \bar{f}_i B_{i,m}(t) \tag{3.12}$$

Using the linear precision property (3.7), we can rewrite the Bézier function $f(t)$ as a parametric Bézier curve $\mathbf{f}(t)$.

$$\mathbf{f}(t) = \begin{pmatrix} t \\ f(t) \end{pmatrix} = \sum_{i=0}^{M} \begin{pmatrix} \frac{i}{M} \\ \bar{f}_i \end{pmatrix} B_{i,M}(t) \tag{3.13}$$

Now the problem of finding roots of the univariate polynomial has been transformed into a problem of finding the intersection of the Bézier curve with the parameter axis which can be solved using the recursive de Casteljau subdivision algorithm. Figure 3-1 shows how the regions which do not contain the intersection points are discarded for a quadratic Bézier curve. The large triangle is the convex hull of the quadratic Bézier curve. This triangle intersects the axis at two points $t = a$ and $t = b$. Applying de Casteljau subdivision algorithm to the Bézier curve with control points, the vertices of the large triangle, at these parameter values, we obtain a small triangle, which is shaded, which also intersects the axis at two points. Such a recursive subdivision process, using the convex hull property, can be continued until the interval width becomes as small as required. But when there are more than one root in the interval, the interval will not be arbitrarily reduced. In such case binary subdivision may be introduced [69]. Binary subdivision is applied when the box size did not reduce more than 20% from the previous step, in accordance with [69]. Accuracy in a subdivision method could be lost for high degree polynomials, if floating point arithmetic is employed.

Now we will illustrate the two dimensional case. The governing equations for finding the starting points for mean curvature along the domain boundary, which is discussed in section 5.4, reduce to univariate irrational functions involving polynomials and square root of polynomials as in equations (5.68) and (5.69). Using the auxiliary variable method, which will be introduced in section 3.3, the univariate equation involving polynomials and square root of polynomials reduce to two polynomial equations with two unknowns, see also [59]. This system of polynomial equations can be converted into two explicit Bézier patches $\mathbf{H}_1(u, v)$ and $\mathbf{H}_2(u, v)$ in (u, v, w) coordinate system using the linear precision property of the Bernstein basis (equation (3.7)). From a geometric point of view, solving for all the roots of two simultaneous bivariate polynomial equations has been replaced by finding the intersections of two Bézier patches with the plane w =0.

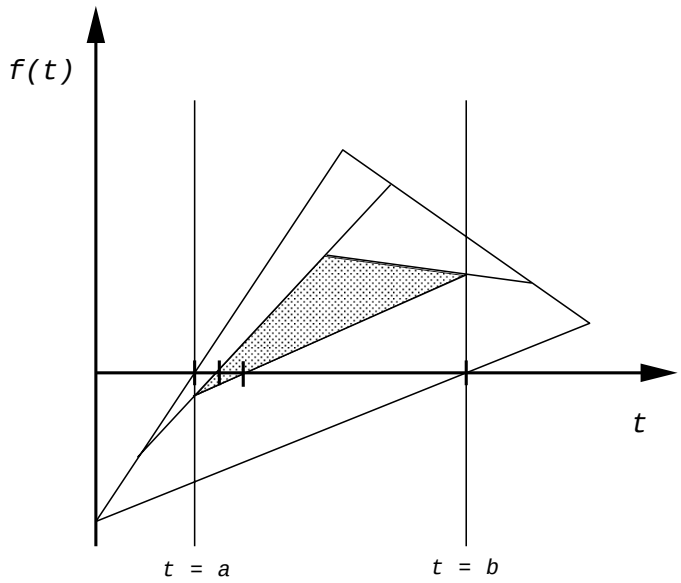

Figure 3-1: de Casteljau Algorithm Applied to the Quadratic Bezier Curve

The convex hull property of Bézier patches $\mathbf{H}_1(u,v)$ and $\mathbf{H}_2(u,v)$ is used to discard parameter regions that do not contain any common solution of the governing equations using recursive de Casteljau subdivision algorithm. Projected convex hulls of $\mathbf{H}_1(u,v)$ and $\mathbf{H}_2(u,v)$ in u and v directions after some subdivision steps are shown in Figures 3-2 and 3-3 for the bicubic surface illustrated more detail in section 5.5. The solid lines correspond to the convex hull of $\mathbf{H}_1(u,v)$ and the dotted lines correspond to the convex hull of $\mathbf{H}_2(u,v)$. The thick solid line are the regions where the roots are contained and the asterisks are the locations of the roots. We will discard the intervals which do not contain the roots by the de Casteljau algorithm. The recursive process can be continued until the resulting rectangles potentially containing roots are as small as required.

3.3 Auxiliary Variable Method for Systems of Nonlinear Polynomial Equations with Square Roots

In this section we will focus on how to compute *all* real roots of systems of irrational equations involving nonlinear polynomials and square roots of nonlinear polynomials within a finite box. Suppose we seek the solution of systems of irrational equations involving nonlinear polynomials and

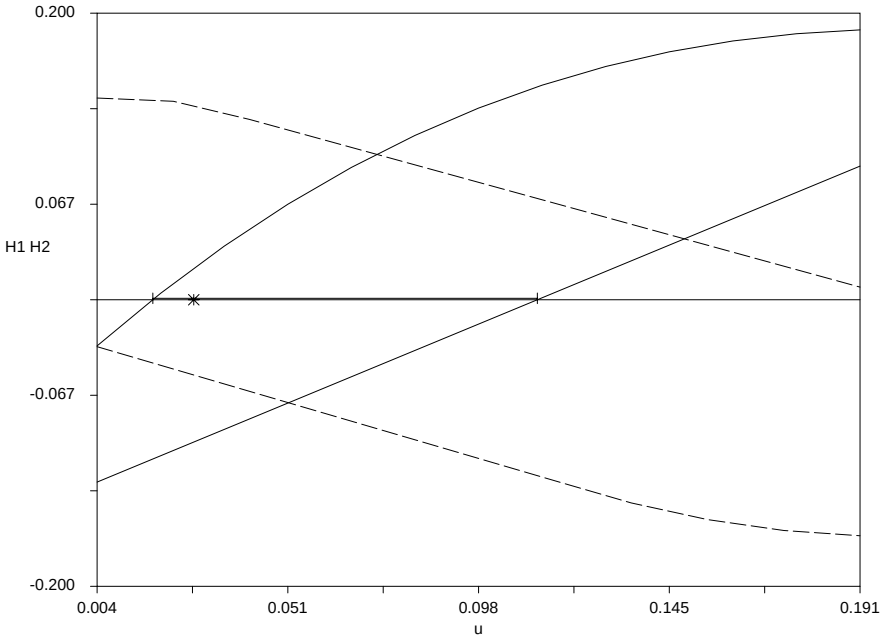

Figure 3.2: Two Convex Hulls of the Projected Control Points in u-Direction

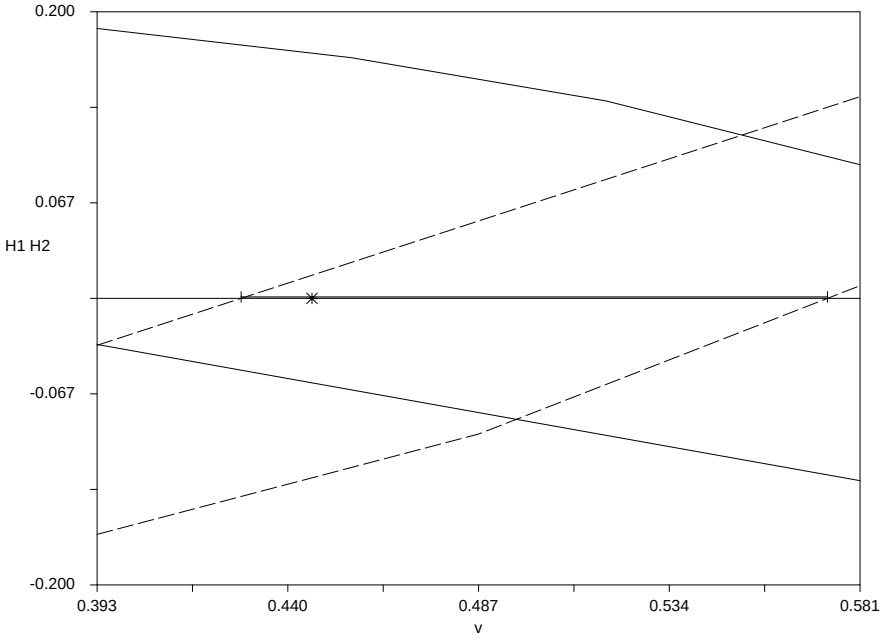

Figure 3-3: Two Convex Hulls of the Projected Control Points in v-Direction

square roots of nonlinear polynomials as follows:

$$f_k(\mathbf{x}) + g_k(\mathbf{x})\sqrt{h_k(\mathbf{x})} \qquad k = 1, 2, \cdots n \tag{3.14}$$

where $\mathbf{x}$ is the unknown vector of n variables, and $f_k(\mathbf{x})$, $g_k(\mathbf{x})$ and $h_k(\mathbf{x})$ are the kth components of known vector fields $\mathbf{f}$, $\mathbf{g}$ and $\mathbf{h}$, over the box $\mathbf{x} \in [0,1]^n$. These components can be expressed in the Bernstein basis as

$$f_k(\mathbf{x}) = \sum_{I}^{M_f^{(k)}} \bar{f}_I^{(k)} \bar{B}_{I,M_f^{(k)}}(\mathbf{x}) \tag{3.15}$$

$$g_k(\mathbf{x}) = \sum_{I}^{M_g^{(k)}} \bar{g}_I^{(k)} \bar{B}_{I,M_g^{(k)}}(\mathbf{x}) \tag{3.16}$$

$$h_k(\mathbf{x}) = \sum_{I}^{M_h^{(k)}} \bar{h}_I^{(k)} \bar{B}_{I,M_h^{(k)}}(\mathbf{x}) \tag{3.17}$$

Since the square root is involved we can not use the convex hull property of the Bernstein polynomial directly. One might consider a *squaring method* to square out the square root, so that the equation becomes

$$f_k^2(\mathbf{x}) - g_k^2(\mathbf{x}) h_k(\mathbf{x}) = 0 \qquad k = 1, 2, \cdots n \tag{3.18}$$

This leads to a higher degree equation, also providing extraneous roots which are not typically necessary. The disadvantages of this squaring method are discussed in section 5.5.2. We investigated two methods based on subdivision, the *bounding method* which is described in [77] and the *auxiliary variable method* which will be discussed in this section. We compared the computational time for both methods and found that the auxiliary variable method is faster than the bounding method by an order of magnitude. The drawback of the bounding method is that we need to recompute the governing equations each time we apply the de Casteljau subdivision to the original Bézier patch, while for the auxiliary variable method the governing equations are formulated once. Therefore we will focus on the auxiliary variable method which will transform the problem into a problem of higher dimensionality. The higher dimensional formulation has been studied by Hoffmann [39] for

surface interrogation problems. First we will introduce the auxiliary variables T_k such that

$$T_k^2 = h_k(\mathbf{x}) \qquad k = 1, 2, \cdots n \tag{3.19}$$

The bounds $a_k \leq T_k \leq b_k$ can be obtained by

$$a_k = \sqrt{\min_I \bar{h}_I^{(k)}} \tag{3.20}$$

$$b_k = \sqrt{\max_I \bar{h}_I^{(k)}} \tag{3.21}$$

when $\min_I \bar{h}_I^{(k)}$ is negative, we just set a_k=0. For convenience, we also scale T_k such that $\tau_k \equiv \frac{T_k - a_k}{b_k - a_k}$, so that $0 \leq \tau_k \leq 1$. Consequently, the system of irrational equations involving nonlinear polynomials and square roots of nonlinear polynomials (3.14), which consists of n equations with n unknowns, has been transformed to a system of nonlinear polynomial equations which consists of $2n$ equations with $2n$ unknowns as follows.

$$f_k(\mathbf{x}) + g_k(\mathbf{x})\left[a_k + \tau_k(b_k - a_k)\right] = 0 \tag{3.22}$$

$$\left[a_k + \tau_k(b_k - a_k)\right]^2 - h_k(\mathbf{x}) = 0 \tag{3.23}$$

for $k = 1, 2, \cdots n$, where $0 \leq \tau_k \leq 1$ and $\mathbf{x} \in [0,1]^n$. Note that even though we transformed the problem into a problem of higher dimensionality, the degree of the new variables τ_k $(k = 1, \cdots n)$ is only two. System (3.23) of $2n$ polynomial equations can be solved in principle using the algorithm introduced in section 3.2.

3.4 Interval Arithmetic and Rounded Interval Arithmetic

3.4.1 Definition

An *interval* is a set of real numbers defined below [64]:

$$[a, b] = \{x | a \leq x \leq b\} \tag{3.24}$$

Two intervals $[a, b]$ and $[c, d]$ are said to be *equal* if $a = c$ and $b = d$. The *intersection* of two intervals is *empty* or $[a, b] \cap [c, d] = \emptyset$, if either $a > d$ or $c > b$. Otherwise, $[a, b] \cap [c, d] = [max(a, c), min(b, d)]$.

The *union* of the two intersecting intervals is $[a,b] \cup [c,d] = [min(a,c), max(b,d)]$. An *order* of intervals is defined by $[a,b] < [c,d]$ if and only if $b < c$. The width of an interval $[a,b]$ is $b-a$ and the *absolute value* is $|[a,b]| = max(|a|,|b|)$.

3.4.2 Interval Arithmetic and its Algebraic Properties

The interval arithmetic operations are defined by [64]

$$[a,b] \circ [c,d] = \{x \circ y \mid x \in [a,b] \; and \; y \in [c,d]\}. \tag{3.25}$$

where $\circ$ represents an arithmetic operation $\circ \in \{+,-,\cdot,/\}$. Using the end points of the two intervals, we can rewrite equation (3.25) as follows

$$\begin{aligned}
&[a,b] + [c,d] = [a+c, b+d] \\
&[a,b] - [c,d] = [a-d, b-c] \\
&[a,b] \cdot [c,d] = [min(ac,ad,bc,bd), max(ac,ad,bc,bd)] \\
&[a,b]/[c,d] = [min(a/c, a/d, b/c, b/d), max(a/c, a/d, b/c, b/d)]
\end{aligned} \tag{3.26}$$

provided $0 \notin [c,d]$ in the division relation.

Interval arithmetic is *commutative* and *associative*.

$$\begin{aligned}
&[a,b] + [c,d] = [c,d] + [a,b] \\
&[a,b] \cdot [c,d] = [c,d] \cdot [a,b] \\
&[a,b] + ([c,d] + [e,f]) = ([a,b] + [c,d]) + [e,f] \\
&[a,b] \cdot ([c,d] \cdot [e,f]) = ([a,b] \cdot [c,d]) \cdot [e,f]
\end{aligned}$$

But it is not *distributive*, however, it is *subdistributive*.

$$[a,b] \cdot ([c,d] + [e,f]) \subseteq [a,b] \cdot [c,d] + [a,b] \cdot [e,f]$$

3.4.3 Rounded Interval Arithmetic and its Implementation

If floating point arithmetic is used to evaluate the interval arithmetic equations (3.26), there is no guarantee that the roundings of the bounds are conducted conservatively. Floating numbers are

represented in the computer by a fixed length. The number of bytes to represent a floating point number depends on the precision of the variable. For example, the IEEE standard for a *double-precision* has 64 bits, 8 bytes wordsize, and is stored in a binary form $(\pm)m \cdot 2^{exp}$, where m is the *mantissa* $(0.5 \leq m < 1)$ and *exp* is the *exponent*. Figure 3-4 illustrates how the information is stored in the binary form, a single bit for sign, 11 bits for exponent and 52 bits for mantissa. Since the mantissa is restricted to the range $0.5 \leq m < 1$, the bit for 2^{-1} is not used. The exponent is 1022 biased to ensure the stored exponent is always positive. For example the number -0.125 is stored as $1011111111000\cdots0$. Most left bit represents the sign $-$, next 11 bits 01111111100 is the biased exponent which is 1020-1022 = -2 and the rest of 52 bits which are all zero represents the mantissa 0.5. Hence $-0.5 \cdot 2^{-2} = -0.125$. If x and x' are consecutive positive double-precision numbers, they differ by an amount ϵ called *ulp* (one Unit in the Last Place), so that $\epsilon = 2^{-53} \cdot 2^{exp} = 2^{exp-53}$. Now it is possible to carry out the operation of interval arithmetic with rounding, so that the computed end points always contain the exact interval as follows

$$
\begin{aligned}
&[a,b] + [c,d] \equiv [a + c - \epsilon, b + d + \epsilon] \\
&[a,b] - [c,d] \equiv [a - d - \epsilon, b - c + \epsilon] \\
&[a,b] \cdot [c,d] \equiv [min(ac, ad, bc, bd) - \epsilon, max(ac, ad, bc, bd) + \epsilon] \\
&[a,b]/[c,d] \equiv [min(a/c, a/d, b/c, b/d) - \epsilon, max(a/c, a/d, b/c, b/d) + \epsilon]
\end{aligned} \tag{3.27}
$$

Each ϵ in the equations can be obtained by $\epsilon = 2^{exp-53}$ where *exp* is extracted from *each* computed lower or upper bound. We refer to the definitions given in equations (3.27) as *rounded interval arithmetic*.

3.5 Robust Formulation and Solution for Systems of Nonlinear Polynomial Equations

Floating point arithmetic (FPA) *usually* works robustly for well-conditioned problems, however in general, FPA does not guarantee not to miss any roots due to numerical errors. Yamaguchi et al [111] present a robust computational method using 4×4 determinant method by means of integer arithmetic of appropriate data length. To achieve a robust and accurate implementation of Bernstein subdivision methods the following methods may be used [59], [60]:

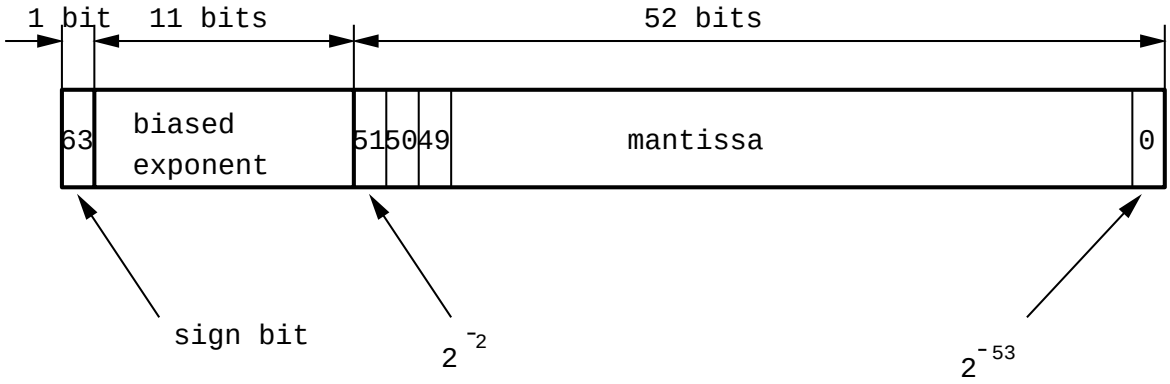

Figure 3-4: IEEE format for binary representation of double-precision floating-point number

- **Formulation** of the governing simultaneous nonlinear equations in multivariate Bernstein form starting from the given input Bézier curve or surface.
  - Use *rational arithmetic* (RA) or *rounded interval arithmetic* (RIA) [64] (see also section 3.4), if the control points of the given curve or surface are floating point numbers to maintain a pristine or guaranteed precision statement of the problem, respectively.
  - Use RIA if the control points of the given curve or surface are irrational numbers to avoid any numerical contamination by standard floating point arithmetic. This happens, for example, when the curve or surface is rotated, since the rotation matrix involves cosines and sines, which are generally irrational.
  - Convert the coefficients of the nonlinear equations in Bernstein form into intervals with floating point number boundaries if rational arithmetic is used in the formulation.
- **Solution** of nonlinear simultaneous equations
  - Use the subdivision method coupled with *rounded interval arithmetic* to solve the system of nonlinear equations which is discussed in section 3.6.

Rational and rounded interval arithmetic operations can be implemented effectively in object-oriented languages such as C++. In section 5.5.2, CPU time comparison of various combinations of arithmetic for formulation of the governing equations and solution of the equations is investigated.

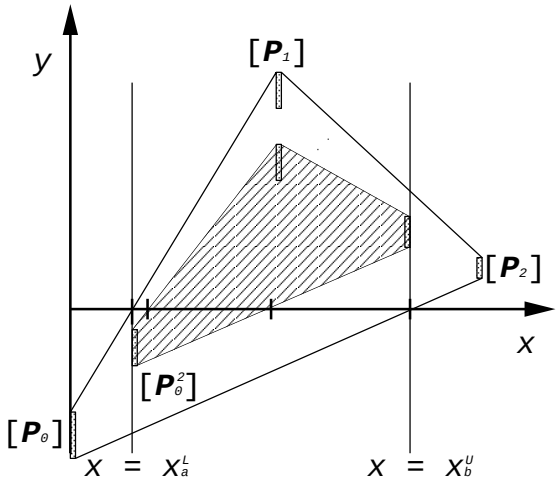

Figure 3-5: Interval de Casteljau algorithm applied to the quadratic Bézier curve

3.6 Bernstein Subdivision Method Coupled with Rounded Interval Arithmetic

In the sequel, we summarize an extension of the de Casteljau subdivision method to operate in rounded interval arithmetic in order to find all the roots of a polynomial system robustly, see also [59], [60]. We illustrate the concept for a single polynomial equation. Figure 3-5 shows the convex hull of a quadratic Bézier curve with control points given in interval boxes. If interval control points $[\mathbf{P}_0]$, $[\mathbf{P}_1]$ and $[\mathbf{P}_2]$ are given as

$$[\mathbf{P}_0] = ([x_0^L, x_0^U],\ [y_0^L, y_0^U]), \quad [\mathbf{P}_1] = ([x_1^L, x_1^U],\ [y_1^L, y_1^U]), \quad [\mathbf{P}_2] = ([x_2^L, x_2^U],\ [y_2^L, y_2^U])$$

we can compute the interval intersection $[x_a^L, x_a^U]$ of left most edge of the convex hull with the x axis using rounded interval arithmetic equations (3.27)

$$[x_a^L, x_a^U] = \frac{[y_0^U, y_0^U] \cdot [x_1^L, x_1^L] \ - \ [y_1^U, y_1^U] \cdot [x_0^L, x_0^L]}{[y_0^U, y_0^U] \ - \ [y_1^U, y_1^U]} \tag{3.28}$$

To be conservative we use the lower endpoint x_a^L for de Casteljau algorithm to discard region $x_0^L \leq x \leq x_a^L$. The interval box on the curve at parameter value $x = x_a^L$ can be obtained by

evaluating three steps of linear interpolation

$$
\begin{aligned}
\left[\mathbf{P}_0^1\right] &= \frac{\left[x_2^U, x_2^U\right] - \left[x_a^L, x_a^L\right]}{\left[x_2^U, x_2^U\right] - \left[x_0^L, x_0^L\right]} \cdot \left[\mathbf{P}_0\right] + \frac{\left[x_a^L, x_a^L\right] - \left[x_0^L, x_0^L\right]}{\left[x_2^U, x_2^U\right] - \left[x_0^L, x_0^L\right]} \cdot \left[\mathbf{P}_1\right] \\
\left[\mathbf{P}_1^1\right] &= \frac{\left[x_2^U, x_2^U\right] - \left[x_a^L, x_a^L\right]}{\left[x_2^U, x_2^U\right] - \left[x_0^L, x_0^L\right]} \cdot \left[\mathbf{P}_1\right] + \frac{\left[x_a^L, x_a^L\right] - \left[x_0^L, x_0^L\right]}{\left[x_2^U, x_2^U\right] - \left[x_0^L, x_0^L\right]} \cdot \left[\mathbf{P}_2\right] \\
\left[\mathbf{P}_0^2\right] &= \frac{\left[x_2^U, x_2^U\right] - \left[x_a^L, x_a^L\right]}{\left[x_2^U, x_2^U\right] - \left[x_0^L, x_0^L\right]} \cdot \left[\mathbf{P}_0^1\right] + \frac{\left[x_a^L, x_a^L\right] - \left[x_0^L, x_0^L\right]}{\left[x_2^U, x_2^U\right] - \left[x_0^L, x_0^L\right]} \cdot \left[\mathbf{P}_1^1\right]
\end{aligned} \tag{3.29}
$$

$\left[\mathbf{P}_0^2\right]$ is the interval box on the curve with parameter value $x = x_a^L$. If the lower bound in the x coordinate of $[\mathbf{P}_0^2]$, x_o^{2L} is smaller than x_a^L, then we set $x_o^{2L} = x_a^L$. We can repeat the same process to discard the region $x_b^U \leq x \leq x_2^U$ to obtain a new smaller convex hull which intersects the x axis with two points. Using projection of convex hulls, the method (referred to as Projected-Polyhedron method [93]) extends directly to arbitrary degree polynomial systems in n variables including the rounded interval arithmetic.

To illustrate the method, we list the following results which are the output of Bernstein subdivision algorithms with floating point arithmetic (FPA) and with rounded interval arithmetic (RIA) in Tables 3.1 and 3.2 respectively. This example finds the roots of a cubic polynomial equation $(x - 0.1)(x - 0.6)(x - 0.7) = 0$ by subdivision in $0 \leq x \leq 1$. This particular example was run at a tolerance of 10^{-4} and the binary subdivision was conducted when the box size did not reduce more than 5% from the previous step. If we compare the bounding box for each iteration, we can easily recognize that the bounding box of the RIA is always conservative with respect to the FPA. Also at iteration 9, FPA loses the root 0.7 due to floating point error, while the RIA never fails.

After the 11th iteration, the FPA method reports two intervals [0.59999999995706, 0.600001895444191], [0.1,0.100000002134303] which contain the roots 0.6 and 0.1, but the root 0.7 has been lost.

All the three intervals [0.699999999999993, 0.7], [0.599999999957053,0.600001895444203], [0.0999999999999988, 0.100000002134311] which contain the roots 0.7, 0.6 and 0.1 are found by the RIA method. The CPU time comparison between FPA and RIA is discussed in section 4.4 in the context of computing the singularities and intersections of offsets of planar polynomial curves.

Iter	*Bounding Box (FPA)*	*Message*
1	[0,1]	
2	[0.0763636363636364, 0.856]	
3	[0.098187732239346, 0.770083868323999]	
4	[0.0999880766853688, 0.72387404781026]	Binary Subdivision
5	[0.402239977003124, 0.704479954527487]	
6	[0.550441290533288, 0.700214508664293]	
7	[0.591018492648952, 0.700000534482207]	
8	[0.599458794784619, 0.700000000003332]	Binary Subdivision
9	[0.649998841568898, 0.699999999999999]	No Root in Box
10	[0.599997683137796, 0.649998841568898]	Root Found in Box
11	[0.099999999478761, 0.402239977003124]	Root Found in Box

Table 3.1: Bernstein subdivision method with FPA

Iter	*Bounding Box (RIA)*	*Message*
1	[0, 1]	
2	[0.076363636363635, 0.856000000000001]	
3	[0.0981877322393447, 0.770083868324001]	
4	[0.0999880766853675, 0.723874047810262]	Binary Subdivision
5	[0.402239977003124, 0.704479954527489]	
6	[0.550441290533286, 0.700214508664294]	
7	[0.591018492648947, 0.700000534482208]	
8	[0.599458794784611, 0.700000000003333]	Binary Subdivision
9	[0.649998841568894, 0.7]	Root Found in Box
10	[0.599997683137788, 0.649998841568895]	Root Found in Box
11	[0.0999999994787598, 0.402239977003124]	Root Found in Box

Table 3.2: Bernstein subdivision method with RIA

3.7 Comparison of Bernstein Subdivision Method Coupled with RIA and Interval Newton Method Coupled with RIA

3.7.1 Interval Newton Method

Interval Newton methods [1] have been the focus of significant attention in geometric computation. A thorough review of various types of interval Newton methods is presented in [7]. In the sequel we briefly review the interval Newton method. The interval Newton method solves a system of nonlinear equations in a numerically verifiable manner.

$$f_i(x_1, x_2, \cdots, x_n) = 0, \quad 1 \leq i \leq n \tag{3.30}$$

within boxes

$$a_i \leq x_i \leq b_i, \quad 1 \leq i \leq n \tag{3.31}$$

If we denote the n-vector whose ith component is x_i by $\mathbf{X}$ and the n-vector whose ith component is f_i by $\mathbf{F}(\mathbf{X})$, the interval Newton methods can be described as finding a box $\bar{\mathbf{X}}_k$ that contains all the solutions of the interval linear system

$$\mathbf{J}(\mathbf{X}_k)(\bar{\mathbf{X}}_k - \mathbf{X}_k^c) = -\mathbf{F}(\mathbf{X}_k^c) \tag{3.32}$$

where the subscript k denotes the kth iteration, $\mathbf{J}(\mathbf{X}_k)$ is the Jacobian matrix of $\mathbf{F}$ over the box $\mathbf{X}_k$, and $\mathbf{X}_k^c$ is some point in $\mathbf{X}_k$. There is a theorem about *unique solution* in the box given by [47]:

Theorem

If $\bar{\mathbf{X}}_k$ is *strictly* contained in $\mathbf{X}_k$, then the system of equations in (3.30) has a unique solution in $\mathbf{X}_k$, and Newton's method starting from any point in $\mathbf{X}_k$ will converge to that solution. Conversely, if $\mathbf{X}_k \cap \bar{\mathbf{X}}_k$ is empty, then there are no solutions of the system in (3.30).

[1] When we use the term "interval Newton method", we assume that bisection is included in the process when interval reduction is not substantial by the pure interval Newton step.

The next iteration $\mathbf{X}_{k+1}$ is evaluated by

$$\mathbf{X}_{k+1} = \mathbf{X}_k \cap \bar{\mathbf{X}}_k \tag{3.33}$$

According to the mean value theorem, the solutions in the $\mathbf{X}_k$ must be in $\mathbf{X}_{k+1}$. If the coordinate intervals of $\mathbf{X}_{k+1}$ are smaller than those of $\mathbf{X}_k$, equations (3.32) and (3.33) are iterated until the bounding boxes are smaller than a specified tolerance. If the coordinate intervals of $\mathbf{X}_{k+1}$ are not smaller than those of $\mathbf{X}_k$, then one of these intervals is bisected to form two new boxes. The boxes are pushed into a stack and iteration is continued until the stack becomes empty. The first interval Newton method introduced by Moore [64] involves computing the inverse of the interval matrix $\mathbf{J}(\mathbf{X}_k)$. Hansen [34] introduced the Gaussian elimination procedure to solve the linear equation system in a interval Newton method. Krawczyk [34] introduced a variation of the interval Newton method which avoids the Gaussian elimination of an interval matrix by not attempting to obtain a sharp solution of (3.32). He computes the new box $\bar{\mathbf{X}}_k$ as follows

$$\bar{\mathbf{X}}_k = \mathbf{K}(\mathbf{X}_k) = \mathbf{X}_k^c - \mathbf{Y}_k\mathbf{F}(\mathbf{X}_k^c) + (\mathbf{I} - \mathbf{Y}_k\mathbf{J}(\mathbf{X}_k))(\mathbf{X}_k - \mathbf{X}_k^c) \tag{3.34}$$

where $\mathbf{Y}_k$ is a preconditioned matrix of midpoints of the elements of the interval Jocobian matrix. Hansen and Sengupta [34] introduced a box which is generally smaller than $\mathbf{K}(\mathbf{X}_k)$. They simply solve the ith equation for the ith variable and replace the others by bounding intervals, which is the non-linear version of the Gauss-Seidel operator for linear systems. Let $\mathbf{x}_i^c$ be the ith component of $\mathbf{X}_k^c$ and $\mathbf{k}_i$ be the ith component of $\mathbf{Y}_k\mathbf{F}(\mathbf{X}_k)$ and $\mathbf{G}_{ij}$ be the entry in the ith row and jth column of $\mathbf{Y}_k\mathbf{J}(\mathbf{X}_k)$ then, the step for the ith row of the Hansen-Sengupta operator becomes

$$\begin{aligned} \bar{\mathbf{x}}_i &= \mathbf{x}_i - [\mathbf{G}_{ii}]^{-1}[\mathbf{k}_i + \sum_{j=1}^{i-1} \mathbf{G}_{ij}(\hat{\mathbf{x}}_j - \mathbf{x}_j^c) + \sum_{j=i+1}^{n} \mathbf{G}_{ij}(\mathbf{x}_j - \mathbf{x}_j^c)] \\ \hat{\mathbf{x}}_i &= \mathbf{x}_i \cap \bar{\mathbf{x}}_i \end{aligned} \tag{3.35}$$

for $i = 1, \cdots, n$.

3.7.2 INTBIS

We use the FORTRAN 77 package for interval Newton method called INTBIS [44], [43], [47], [45], which is available through netlib, for comparison with the Bernstein subdivision method coupled

with rounded interval arithmetic developed in this thesis. INTBIS is implemented for nonlinear polynomial equation systems only and employs the Hansen-Sengupta operator [34] with rounded interval arithmetic. There are two tolerances defined in INTBIS, tolerance for the box size ϵ_b and tolerance for the function range ϵ_f. Once it is guaranteed that there is a unique solution in the box, the classical Newton's method is used to obtain the point approximation to the root. The algorithm stops when $\max_{1\leq i\leq n}|b_i - a_i| < \epsilon_b$ or when $|f_i(x_1, x_2, \cdots, x_n)| < \epsilon_f$, for $1 \leq i \leq n$. For detailed algorithm description, see [44], [43], [47], [45].

3.7.3 Test Problems

The test problems are chosen to check the effect of *dimension* of the system of equations and *degree* of the polynomials with tolerance as a parameter. The first three tests are prepared to examine the effect of the d*imension* of systems of equations. Suppose we are given a point and a circle, two circles and two spheres. We want to compute the extrema of the squared distance of these three pairs which will form a set of simultaneous polynomial equations with dimensions 2, 4 and 6 if the circle and sphere are expressed implicitly.

The next three examples are selected to examine the effect of the *degree* of the polynomial. We define a degree m polynomial given by

$$P(x) = \prod_{i=1}^{m}(x - \frac{i}{m}) = 0 \tag{3.36}$$

We choose the degree as $m = 5,\ 8$ and 20. Especially when $m = 20$, the roots are extremely ill-conditioned which was shown by Wilkinson [107].

The final problem is chosen to illustrate the weakness of the interval Newton method when the Jacobian matrix becomes *singular* at its solution.

1. *Extrema of the squared distance of a point and a circle*

   Let (x_1, y_1) be the point on the circle

   $$(x_1 - \frac{1}{5})^2 + (y_1 - \frac{1}{5})^2 - \frac{1}{25} = 0 \tag{3.37}$$

   and the fixed point be $(0, 0)$, then the squared distance between this point and and a point on the circle becomes $D = x_1^2 + y_1^2$. We consider x_1 to be independent, while y_1 depends on x_1. Now, for (x_1, y_1) to be an extremum, the partial derivative of D with respect to x_1 must be

zero.

$$\frac{\partial D}{\partial x_1} = 2x_1 + 2y_1\frac{\partial y_1}{\partial x_1} = 0 \tag{3.38}$$

Implicit differentiation of (3.37) yields

$$\frac{\partial y_1}{\partial x_1} = -\frac{x_1 - \frac{1}{5}}{y_1 - \frac{1}{5}} \tag{3.39}$$

Substituting (3.39) into (3.38) and simplifying gives

$$x_1 - y_1 = 0 \tag{3.40}$$

Equation (3.40) together with equation (3.37) form a system of two simultaneous polynomial equations with two unknowns. This problem has two roots at $(\frac{1}{5} - \frac{1}{\sqrt{50}}, \frac{1}{5} - \frac{1}{\sqrt{50}})$ and $(\frac{1}{5} + \frac{1}{\sqrt{50}}, \frac{1}{5} + \frac{1}{\sqrt{50}})$.

2. *Extrema of squared distance between two circles*

Let (x_1, y_1) be a point on one circle and (x_2, y_2) be a point on the other circle, then

$$(x_1 - \frac{1}{5})^2 + (y_1 - \frac{1}{5})^2 - \frac{1}{25} = 0 \tag{3.41}$$

$$(x_2 - \frac{1}{5})^2 + (y_2 - \frac{4}{5})^2 - \frac{1}{25} = 0 \tag{3.42}$$

Then the squared distance between points on the two circles becomes $D = (x_1 - x_2)^2 + (y_1 - y_2)^2$. We consider x_1, x_2 to be independent variables while y_1 depends on x_1 and y_2 depends on x_2. For $(x_1,\ y_1,\ x_2,\ y_2)$ to be a extremum of D, the partial derivatives of D with respect to the independent parameters must be zero.

$$\frac{\partial D}{\partial x_1} = 2(x_1 - x_2) + 2(y_1 - y_2)\frac{\partial y_1}{\partial x_1} = 0 \tag{3.43}$$

$$\frac{\partial D}{\partial x_2} = -2(x_1 - x_2) - 2(y_1 - y_2)\frac{\partial y_2}{\partial x_2} = 0 \tag{3.44}$$

By implicit differentiation of (3.41) and (3.42), we obtain $\frac{\partial y_1}{\partial x_1}$ and $\frac{\partial y_2}{\partial x_2}$ which will be substituted

into (3.43) and (3.44) to yield

$$(x_1 - x_2)(y_1 - \frac{1}{5}) - (y_1 - y_2)(x_1 - \frac{1}{5}) = 0 \tag{3.45}$$

$$(x_1 - x_2)(y_2 - \frac{4}{5}) - (y_1 - y_2)(x_2 - \frac{1}{5}) = 0 \tag{3.46}$$

These two equations (3.45) and (3.46) with equations (3.41) and (3.42) form four equations with four unknowns. There are four roots given by $(x_1,\ y_1,\ x_2,\ y_2)$ = (0.2, 0, 0.2, 1), (0.2, 0.4, 0.2, 0.6), (0.2, 0, 0.2, 0.6), (0.2, 0.4, 0.2, 1).

3. *Extrema of squared distance between two spheres*

Let (x_1, y_1, z_1) be a point on one sphere and (x_2, y_2, z_2) be a point on the other sphere, then

$$(x_1 - \frac{1}{5})^2 + (y_1 - \frac{1}{5})^2 + (z_1 - \frac{1}{5})^2 - \frac{1}{25} = 0 \tag{3.47}$$

$$(x_2 - \frac{1}{5})^2 + (y_2 - \frac{1}{5})^2 + (z_2 - \frac{4}{5})^2 - \frac{1}{25} = 0 \tag{3.48}$$

Then the squared distance between points on the two spheres becomes $D = (x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2$. We consider x_1, y_1, x_2 and y_2 to be independent variables while z_1 depends on x_1 and y_1, and z_2 depends on x_2 and y_2. For $(x_1,\ y_1,\ z_1\ x_2,\ y_2,\ z_2)$ to be a extremum, the partial derivatives of D with respect to the independent parameters must be zero, hence

$$\frac{\partial D}{\partial x_1} = \frac{\partial D}{\partial y_1} = \frac{\partial D}{\partial x_2} = \frac{\partial D}{\partial y_2} = 0 \tag{3.49}$$

By implicit differentiation of (3.47) and (3.48), we obtain $\frac{\partial z_1}{\partial x_1}$, $\frac{\partial z_1}{\partial y_1}$, $\frac{\partial z_2}{\partial x_2}$, $\frac{\partial z_2}{\partial y_2}$, which can be substituted into (3.49) to yield

$$(x_1 - x_2)(z_1 - \frac{1}{5}) - (z_1 - z_2)(x_1 - \frac{1}{5}) = 0 \tag{3.50}$$

$$(y_1 - y_2)(z_1 - \frac{1}{5}) - (z_1 - z_2)(y_1 - \frac{1}{5}) = 0 \tag{3.51}$$

$$(x_1 - x_2)(z_2 - \frac{4}{5}) - (z_1 - z_2)(x_2 - \frac{1}{5}) = 0 \tag{3.52}$$

$$(x_1 - y_2)(z_2 - \frac{4}{5}) - (z_1 - z_2)(y_2 - \frac{1}{5}) = 0 \tag{3.53}$$

Together with equations (3.47) and (3.48), equations (3.50) to (3.53) form a system of six equations with six unknowns. There are four roots given by $(x_1,\ y_1,\ z_1,\ x_2,\ y_2,\ z_2)$ = (0.2, 0.2,

0, 0.2, 0.2, 1), (0.2, 0.2, 0.4, 0.2, 0.2, 1), (0.2, 0.2, 0, 0.2, 0.2, 0.6), (0.2, 0.2, 0.4, 0.2, 0.2, 0.6).

4. *Degree 5 polynomial equation* $P(x) = \prod_{i=1}^{5}(x - \frac{i}{5}) = 0$

5. *Degree 8 polynomial equation* $P(x) = \prod_{i=1}^{8}(x - \frac{i}{8}) = 0$

6. *Degree 20 polynomial equation* $P(x) = \prod_{i=1}^{20}(x - \frac{i}{20}) = 0$
   Rational arithmetic is used to expand this polynomial equation to obtain the coefficients of the monomial basis, and to convert the monomial basis to the Bernstein basis to maintain a guaranteed precision statement of the problem.

7. *Stationary points of an implicit surface*
   Let us consider the following implicit surface

$$f(x, y, z) = (x - \frac{1}{2})^4 + (y - \frac{1}{2})^4 + (z - \frac{1}{2})^4 - (\frac{1}{2})^4 = 0 \tag{3.54}$$

   The stationary points with respect to xy-plane can be obtained by $f = f_x = f_y = 0$. The partial derivatives of equation (3.54) are given by

$$f_x = 4(x - \frac{1}{2})^3 = 0 \tag{3.55}$$

$$f_y = 4(y - \frac{1}{2})^3 = 0 \tag{3.56}$$

   These three equations form three simultaneous polynomial equations with three unknows. The solutions are $(x, y, z) = (0.5, 0.5, 0),\ (0.5, 0.5, 1)$. Notice that at the solutions, the Jacobian matrix is singular.

3.7.4 Computational Results

The CPU time comparison for the solution (formulation time is not included) between the subdivision method we developed, ie the Bernstein subdivision method with Rounded Interval Arithmetic (PPRIA) implemented in C++ and the INTBIS are listed in Table 3.3. All the computations are conducted with double-precision numbers. The CPU time in seconds is measured on a graphics workstation running at 36MHz. Since our code is not optimized and INTBIS also sacrifices some speed for portability and ease of use [47], we can only compare their relative performance in a preliminary manner and try to find their merits and defects. Our method only requires tolerance for the box size, while INTBIS requires a box tolerance and a function range tolerance. To make the comparison

			CPU Time in Seconds							
No.	*Degree*	*Dimension*	*PPRIA*				*INTBIS*			
			10^{-6}	10^{-8}	10^{-10}	10^{-12}	10^{-6}	10^{-8}	10^{-10}	10^{-12}
1	2	2	0.3	0.4	0.4	0.4	*	*	*	*
2	2	4	2.7	3.3	4.0	4.8	2.8	2.9	3.2	3.4
3	2	6	16.7	17.7	19.5	21.6	254.7	255.0	255.5	257.3
4	5	1	0.1	0.2	0.2	0.2	0.2	0.2	0.3	0.3
5	8	1	0.4	0.6	0.6	0.8	9.3	9.3	9.3	9.6
6	20	1	5.6	6.5	15.3	c	b	b	b	b
7	4	3	47.8	b	b	b	b	b	b	b

Table 3.3: CPU Time Comparison between Bernstein Subdivision Method Coupled with RIA referred to as PPRIA and INTBIS (See Table 3.4 for symbolic entries)

Symbols	Meaning
b	Memory overflow due to too many binary subdivisions.
c	Width of the interval Bernstein coefficients is too large and results in performing an extensive amount of binary subdivision, and the roots are not resolved to the tolerance required. For such cases, extended precision interval arithmetic would be needed.
m	Some of the roots are *missed* due to floating point errors.
n/a	Measurement was not taken since formulation was not available due to excessive formulation cost.
*	Example executed too quickly to be measured.
∞	Example executed too slow to be measured (terminated when CPU time is more than 100 hours.)

Table 3.4: Definitions of symbolic entries

fair, first we compute the roots by our method using the box tolerances 10^{-6}, 10^{-8}, 10^{-10}, 10^{-12} and substitute them into the governing equations to find the function range. Since the range differs for each governing equation and each root, we choose the maximum range for the range tolerance used in INTBIS.

The following are some observations from these results.

1. Computational time grows nearly linearly as the box tolerance decreases for our method, while computational time remains the same for INTBIS. This is due to the fact the Bernstein subdivision method exhibits linear local convergence rates [93] for $n \geq 2$, while INTBIS exhibits the quadratic local convergence property of Newton's method.

2. Our method is slower than INTBIS when the dimension is low, but faster when the dimension is high. This can be explained by the fact that the amount of binary subdivision or bisection for INTBIS in high dimension problem is generally larger than for our method.

3. Our method performs slower than INTBIS for lower degree problems but faster for higher degree problems. Even for the extremely ill-conditioned Wilkinson's polynomial, our method can compute roots up to tolerance of 10^{-10}. On the other hand, INTBIS performs well for lower degree problems but becomes very slow for higher degree problems and generally can not solve with accuracy if the degree is more than ten. It is shown that the roots of a polynomial may be extracted with much less error by formulating the polynomial in the Bernstein basis [27], which is employed by our method, than in the monomial basis, which is employed by INTBIS. This is one of the reasons why INTBIS can not solve with accuracy the high degree polynomial problem. Also when the degree of the polynomial equation is high, the number of roots increases, and therefore the number of bisections needed by INTBIS increases.

4. When the Jacobian matrix is singular at the solution, INTBIS performs an excessive number of bisection leading to tremendous memory requirements which can quickly exhaust computer resources even if the requested accuracy is not very strict. On the contrary our method's performance degrades only moderately for such cases.

Chapter 4

Global Offsets of Planar Curves

4.1 Introduction and Literature Review

Offset curves, also called parallel curves, are defined as the locus of points traced by the unit normal vector of a planar generator curve multiplied by a signed offset distance d. Because of the square root involved in the expression of an unit normal vector, an offset curve is functionally more complex than its progenitor. Another difficulty arises when the progenitor has a tangent discontinuity. Then its exterior offset ($d > 0$) curve will become discontinuous and its interior offset ($d < 0$) curve will have a self-intersection, see Figure 4-1. Also when the radius of curvature is less than the magnitude of offset distance in a region where curvature and signed offset distance d have opposite sign, the offset curve may self-intersect, see Figure 2-4. In applications, discontinuity must be filled in and the loops must be trimmed off in the offset curve, which is equivalent to the locus of the center of a circle of radius d swept all over the curve. As described in section 2.2.2, computation of the singularities of the planar offset curve is very important to $2\frac{1}{2}$-D machining.

Offsets are also used in the definition of tolerance regions and access space representation in robotics [58], [88], [75]. Molds are slightly offset from the part so that the part can be extracted easily when the material cools down and so that machining can be applied to obtain an accurate shape. Offsets are also useful in feature recognition through construction of skeletons or medial axes of geometric models [76],[18], [108].

The books by Struik, doCarmo [98],[16] offer firm theoretical basis to the differential geometry aspects of shape description. A monograph by Banchoff et al. [3] gives detailed analysis of the

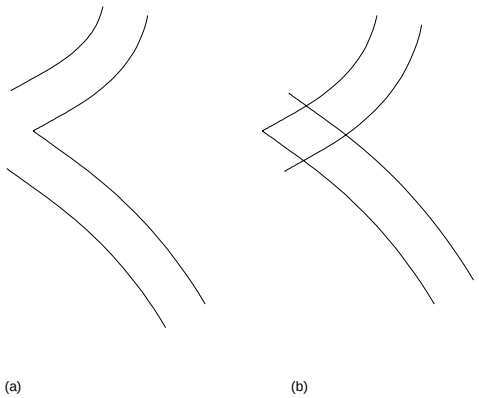

Figure 4-1: Exterior offsets (a) and interior offsets (b) to a tangent discontinuous curve

singularities of planar offset curves. Also the books by Walker [106], and Lawrence [55] study algebraic curves and their singularities. A recent survey by Pham [80] provides extensive bibliography on offset curves and surfaces.

Offset curves are inherently more complex than the progenitor curves. For example, offsets of polynomial or rational polynomial curves are not in general polynomial or rational polynomial curves, except for the special cases of straight line, circle, and the Pythagorean hodograph curves [24]. Because of the wide application of offset curves and surfaces and the difficulty in directly incorporating such entities in geometric modeling systems, due to their potential analytic and algebraic complexity, a number of researchers have developed approximation algorithms for these types of geometries in terms of piecewise polynomial or rational polynomial functions [48], [100], [41]. The related problems of approximation of geodesic offset curves on surfaces is studied in [74] and [109]. In more recent work, Sederberg and Farouki [92] represent the error in the approximation of a parametric curve by a Bézier curve using interval-valued control points. Along the same lines, Sederberg and Buehler [91] approximate an offset of planar Bézier curve as a special Bézier curve having one control point in a rectangular box. Detecting the location of singularities is vital for generating trimmed or global offset curves as discussed in section 2.2.2. Lasser [54] introduced a divide-and-conquer algorithm for finding all the self-intersection points of a planar parametric polynomial curve using the convex hull and approximation property of the Bézier polygon. Hoschek [40] developed an algorithm to find the intersection point of offsets of two curves or a self-intersection point of an offset curve based on Newton iteration which requires a good initial approximation for all such intersections. The analytic properties of planar offset curves have recently been studied by Farouki and Neff [26], where it is proved that the *trimmed offset* is the locus with the property that each point of it is at least dis-

tance $|d|$ from every point, and exactly $|d|$ from some point, of the progenitor. Farouki and Neff [25] have shown that the two-sided offsets of planar rational polynomial curves are high-degree implicit algebraic curves $f_o(x, y) = 0$ of potentially complex shape. These equations can not typically be separated into two equations describing interior and exterior offsets individually. The degree of the two-sided offset curve of a parabola $\mathbf{r}(t) = [t, t^2]^T$ is 6 and of a general polynomial cubic curve is 10. This method can be applied to detect the location of all singularities of offsets by computing the roots of $f_o = \partial f_o/\partial x = \partial f_o/\partial y = 0$. This set of roots also includes cusps, self-intersections and mutual intersections of the entire exterior and interior offsets, in effect providing much information that is not typically necessary. They also study an algebraic method to seek self-intersections only but this method leads to a high degree problem. For example to detect the self-intersections of the offsets of a cubic curve $\mathbf{r}(t) = [t, t^3]^T$, the degree of the resulting equation is 24 and for the quartic curve $\mathbf{r}(t) = [t, t^4]^T$ the degree is 66. Chiang et al. [11] studied a problem of global offsets without self-intersection. They addressed the problem by discretizing the ambient space and by posting a geometric datum in this common space. This approach poses a trade-off between the accuracy and the memory required. Algebraic and numerical techniques for offsets and blends are discussed by Hoffmann [38]. A theoretical investigation of the topological properties of the cut locus, medial axis and offsets is presented by Wolter [108].

This Chapter presents a method to detect the singularities of a normal offset of a planar integral polynomial curve and the intersection of two specific normal offsets of two planar integral polynomial curves. The Chapter is structured as follows. Section 4.2 reviews the differential geometry properties of planar curve. Section 4.3 derives the equations for the singularities of planar offset curves and the intersection of two specific normal offsets of two planar integral polynomial curves. Section 4.4 gives examples which can be applied for pocket machining, and compares the CPU time between FPA and RIA for solving the examples.

4.2 Differential Geometry of Planar Curves

We start by summarizing the relevant definitions. A planar parametric curve $\mathbf{r}$ is given by

$$\mathbf{r} = \mathbf{r}(t) = [x(t), y(t)]^T, \quad t \in [0, 1] \tag{4.1}$$

where x and y are differentiable with respect to t. The parameterization $\mathbf{r}(t) = [x(t), y(t)]^T$ is called *regular* [16] on $t \in [0, 1]$ if $|\dot{\mathbf{r}}| \neq 0$ for all $t \in [0, 1]$, where the dot denotes the derivative with respect

to t. *We assume that the progenitor curve* $\mathbf{r}(t)$ *is regular throughout this Chapter.* An unit tangent vector of a plane curve is defined as

$$\mathbf{t} = \frac{d\mathbf{r}}{ds} = \frac{d\mathbf{r}}{dt}\frac{dt}{ds} = \frac{\dot{\mathbf{r}}}{|\dot{\mathbf{r}}|} = \frac{[\dot{x}(t), \dot{y}(t)]^T}{\sqrt{\dot{x}^2(t) + \dot{y}^2(t)}} \tag{4.2}$$

where s denotes arc length. The unit normal vector of a plane curve, which is orthogonal to $\mathbf{t}$, is given by

$$\mathbf{n} = \mathbf{t} \times \mathbf{e}_z = \frac{[\dot{y}(t), -\dot{x}(t)]^T}{\sqrt{\dot{x}^2(t) + \dot{y}^2(t)}} \tag{4.3}$$

where $\mathbf{e}_z = (0, 0, 1)$ is an unit vector perpendicular to the plane of the curve, see Figure 4-2.

For a plane curve, the Frenet formulas reduce to

$$\frac{d\mathbf{t}}{ds} = -\kappa_c \mathbf{n}, \quad \frac{d\mathbf{n}}{ds} = \kappa_c \mathbf{t} \tag{4.4}$$

where κ_c is the signed curvature of the curve. The curvature κ_c of a curve at point P is positive when the direction of $\mathbf{n}$ and $\vec{PC}$ are opposite where C is the center of the curvature at P, see Figure 4-2. Using equation (4.2) the second of equations (4.4) can be rewritten as:

$$\dot{\mathbf{n}} = \kappa_c \dot{\mathbf{r}} \tag{4.5}$$

Curvature κ_c can be expressed in terms of parametric derivatives as follows:

$$\kappa_c = \frac{(\dot{\mathbf{r}} \times \ddot{\mathbf{r}}) \cdot \mathbf{e}_z}{|\dot{\mathbf{r}}|^3} \tag{4.6}$$

An offset curve $\mathbf{r}_o(t)$ with signed offset distance d to the progenitor $\mathbf{r}(t)$ is defined by

$$\mathbf{r}_o(t) = \mathbf{r}(t) + d\mathbf{n}(t) \tag{4.7}$$

where $d > 0$ corresponds to positive (exterior) and $d < 0$ corresponds to negative (interior) offsets. The unit tangent and normal vectors and curvature of the offset curve are given by

$$\mathbf{t}_o = \frac{\dot{\mathbf{r}}_o}{|\dot{\mathbf{r}}_o|} = \frac{1 + \kappa_c d}{|1 + \kappa_c d|}\mathbf{t} \tag{4.8}$$

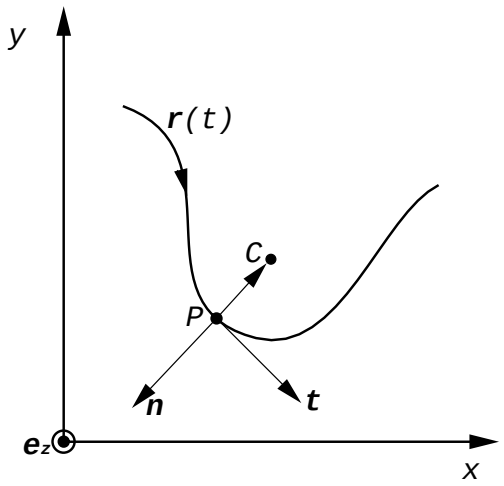

Figure 4-2: Definitions of unit tangent and normal vectors

$$\mathbf{n}_o = \mathbf{t}_o \times \mathbf{e}_z = \frac{1+\kappa_c d}{|1+\kappa_c d|}\mathbf{n} \tag{4.9}$$

$$\kappa_{co} = \frac{\kappa_c}{|1+\kappa_c d|} \tag{4.10}$$

where equation (4.5) is used for derivation.

4.3 Computation of Singularities and Intersections of Offsets of Planar Polynomial Curves

4.3.1 Formulation of Singularities and Intersections of Offsets of Planar Polynomial Curves

There are two kinds of singularities on the offset curves, *irregular points* and *self-intersections*. Irregular points include *isolated points* and *cusps*. A point P on a curve C is called isolated point of C if there is no other point of C in some neighborhood of P. This point occurs when the progenitor curve with radius R is a circle and the offset is $d = -R$. A cusp is an irregular point on the offset curve where the tangent vector vanishes. Cusp at $t = t_c$ can be further subdivided into ordinary cusps when $\dot{\kappa_c}(t_c) \neq 0$ and extraordinary points when $\dot{\kappa_c}(t_c) = 0$ and $\ddot{\kappa_c}(t_c) \neq 0$. An isolated point

and a cusp occur when $|\dot{\mathbf{r}}_o(t)| = 0$, which using equations (4.5), (4.7) reduces to

$$\kappa_c(t) = -\frac{1}{d} \tag{4.11}$$

Using equation (4.6) with $\mathbf{e}_z = [0,0,1]^T$, $\dot{\mathbf{r}}(t) = [\dot{x}(t), \dot{y}(t)]^T$ and $\ddot{\mathbf{r}}(t) = [\ddot{x}(t), \ddot{y}(t)]^T$, equation (4.11) reduces to

$$d\left[\ddot{x}(t)\dot{y}(t) - \dot{x}(t)\ddot{y}(t)\right] - \sqrt{\dot{x}^2(t) + \dot{y}^2(t)}\left[\dot{x}^2(t) + \dot{y}^2(t)\right] = 0 \tag{4.12}$$

Consequently, if $\mathbf{r}(t)$ is a polynomial curve, locations of irregular points can be obtained by solving an univariate irrational function involving polynomials and square roots of polynomials.

Self-intersections include *nodes* and *tacnodes*. A node P is a point of curve C where two arcs of C pass through P and the arcs have different tangents. A tacnode is a special case of a node whose two tangents coincide, see for example Figure 4-4. Self-intersections of an offset curve can be obtained by seeking pairs of distinct parameter values $s \neq t$ such that

$$\mathbf{r}(s) + d\mathbf{n}(s) = \mathbf{r}(t) + d\mathbf{n}(t) \tag{4.13}$$

Substitution of equation (4.3) in (4.13) yields the system

$$\begin{aligned} x(s) + \frac{\dot{y}(s)d}{\sqrt{\dot{x}^2(s) + \dot{y}^2(s)}} &= x(t) + \frac{\dot{y}(t)d}{\sqrt{\dot{x}^2(t) + \dot{y}^2(t)}} \\ y(s) - \frac{\dot{x}(s)d}{\sqrt{\dot{x}^2(s) + \dot{y}^2(s)}} &= y(t) - \frac{\dot{x}(t)d}{\sqrt{\dot{x}^2(t) + \dot{y}^2(t)}} \end{aligned} \tag{4.14}$$

If $\mathbf{r}(t)$ is a polynomial curve, equations (4.14) are two simultaneous bivariate irrational functions involving polynomials and square root of polynomials.

Similarly to equation (4.14), intersections of two specific normal offsets at distance d of two planar curves $\mathbf{r}^P(s)$ and $\mathbf{r}^Q(t)$ can be formulated as

$$\begin{aligned} x^P(s) + \frac{\dot{y}^P(s)d}{\sqrt{\dot{x}^{P2}(s) + \dot{y}^{P2}(s)}} &= x^Q(t) + \frac{\dot{y}^Q(t)d}{\sqrt{\dot{x}^{Q2}(t) + \dot{y}^{Q2}(t)}} \\ y^P(s) - \frac{\dot{x}^P(s)d}{\sqrt{\dot{x}^{P2}(s) + \dot{y}^{P2}(s)}} &= y^Q(t) - \frac{\dot{x}^Q(t)d}{\sqrt{\dot{x}^{Q2}(t) + \dot{y}^{Q2}(t)}} \end{aligned} \tag{4.15}$$

Note that for equation (4.14) we need to find a method to eliminate the trivial solution $s = t$, while

for equation (4.15) $s = t$ can be a solution.

4.3.2 Isolated Points and Cusps

For simplicity, the progenitor curve is assumed to be an integral Bézier curve

$$\mathbf{r}(t) = \sum_{i=0}^{m} \mathbf{P}_i B_{i,m}(t) \quad 0 \le t \le 1 \tag{4.16}$$

where m is the degree of the curve, $\mathbf{P}_i = [x_i, y_i]^T$ are the control points and $B_{i,m}(t)$ is the Bernstein basis function. Extension to rational B-spline (NURBS) curves, although tedious does not present conceptual difficulties. We also assume that the curve is regular [16], i.e. $|\dot{\mathbf{r}}(t)| \neq 0$.

The roots of equation (4.12) for *isolated points* and *cusps* can be obtained by using auxiliary variable method introduced in section 3.3. Let

$$\mu(t) = \ddot{x}(t)\dot{y}(t) - \dot{x}(t)\ddot{y}(t) = \sum_{i=0}^{2m-3} \mu_i B_{i,2m-3}(t) \quad 0 \le t \le 1 \tag{4.17}$$

$$\nu(t) = \dot{x}^2(t) + \dot{y}^2(t) = \sum_{i=0}^{2m-2} \nu_i B_{i,2m-2}(t) \quad 0 \le t \le 1 \tag{4.18}$$

then equation (4.12) can be rewritten as

$$d\mu(t) - \sqrt{\nu(t)}\nu(t) = 0 \tag{4.19}$$

Now we will introduce a new auxiliary variable $\check{t}$ by setting

$$\check{t}^2 = \nu(t) \qquad a \le \check{t} \le b \tag{4.20}$$

where $a = \sqrt{\min_i(\nu_i)}$, $b = \sqrt{\max_i(\nu_i)}$. When $\min_i(\nu_i)$ is negative, we just set $a = 0$. Then equation (4.19) can be rewritten as two equations with two variables.

$$d\mu(t) - \check{t}^3 = 0 \tag{4.21}$$

$$\check{t}^2 - \nu(t) = 0 \tag{4.22}$$

where $0 \le t \le 1$ and $a \le \check{t} \le b$. Similarly to the example in section 3.3, the irrational univariate

function involving polynomials and square roots of polynomials has been transformed into a system of two simultaneous bivariate nonlinear polynomial equations. For convenience, we scale $\check{t}$ such that $\tau \equiv \frac{\check{t}-a}{b-a}$, so that $0 \leq \tau \leq 1$, and therefore (4.21) and (4.22) become

$$\begin{aligned} \lambda(t,\tau) &= d\mu(t) - [(b-a)\tau + a]^3 = 0 \quad 0 \leq t, \tau \leq 1 \qquad (4.23) \\ \xi(t,\tau) &= [(b-a)\tau + a]^2 - \nu(t) = 0 \quad 0 \leq t, \tau \leq 1 \qquad (4.24) \end{aligned}$$

Since

$$\begin{aligned} (b-a)\tau + a &= \sum_{i=0}^{1} \alpha_i B_{i,1}(\tau) \;\; with \;\; \alpha_0 = a, \;\; \alpha_1 = b \\ [(b-a)\tau + a]^2 &= \sum_{i=0}^{2} \beta_i B_{i,2}(\tau) \;\; with \;\; \beta_0 = a^2, \quad \beta_1 = ab, \quad \beta_2 = b^2 \\ [(b-a)\tau + a]^3 &= \sum_{i=0}^{3} \gamma_i B_{i,3}(\tau) \;\; with \;\; \gamma_0 = a^3, \;\; \gamma_1 = a^2 b, \;\; \gamma_2 = ab^2, \;\; \gamma_3 = b^3 \qquad (4.25) \end{aligned}$$

equations (4.23) and (4.24) can be transformed to the following two simultaneous bivariate Bernstein polynomial equations

$$\begin{aligned} \lambda(t,\tau) &= \sum_{i=0}^{2m-3} \sum_{j=0}^{3} [d\mu_i - \gamma_j] \, B_{i,2m-3}(t) B_{j,3}(\tau) = 0 \quad 0 \leq t, \tau \leq 1 \qquad (4.26) \\ \xi(t,\tau) &= \sum_{i=0}^{2m-2} \sum_{j=0}^{2} [\beta_j - \nu_i] \, B_{i,2m-2}(t) B_{j,2}(\tau) = 0 \quad 0 \leq t, \tau \leq 1 \qquad (4.27) \end{aligned}$$

Using the linear precision property of the Bernstein basis equation (3.7), the system of Bernstein polynomial equations can be converted into two explicit Bézier patches in (t, τ, w) coordinate system, as follows

$$\begin{aligned} \mathbf{\Lambda}(t,\tau) &= \sum_{i=0}^{2m-3} \sum_{j=0}^{3} \begin{pmatrix} i/(2m-3) \\ j/3 \\ d\mu_i - \gamma_j \end{pmatrix} B_{i,2m-3}(t) B_{j,3}(\tau) = \mathbf{0} \quad 0 \leq t, \tau \leq 1 \qquad (4.28) \\ \mathbf{\Xi}(t,\tau) &= \sum_{i=0}^{2m-2} \sum_{j=0}^{2} \begin{pmatrix} i/(2m-2) \\ j/2 \\ \beta_i - \nu_j \end{pmatrix} B_{i,2m-2}(t) B_{j,2}(\tau) = \mathbf{0} \quad 0 \leq t, \tau \leq 1 \qquad (4.29) \end{aligned}$$

Bernstein subdivision method coupled with rounded interval arithmetic, which is discussed in section

3.6, is used to solve the equations.

4.3.3 Intersection of Two Offsets

In the case of intersections of *normal offsets of two polynomial curves*, we set

$$\omega^P(s) = \dot{x}^{P2}(s) + \dot{y}^{P2}(s) = \sum_{i=0}^{2m-2} \omega_i^P B_{i,2m-2}(s) \quad 0 \le s \le 1 \tag{4.30}$$

$$\omega^Q(t) = \dot{x}^{Q2}(t) + \dot{y}^{Q2}(t) = \sum_{i=0}^{2n-2} \omega_i^Q B_{i,2n-2}(t) \quad 0 \le t \le 1 \tag{4.31}$$

where m and n are degrees of two planar integral Bézier curves $\mathbf{r}^P(s)$ and $\mathbf{r}^Q(t)$. Substituting equations (4.30) and (4.31) in equations (4.15) , we obtain

$$\begin{aligned}
\sqrt{\omega^Q(t)}\left[x^P(s)\sqrt{\omega^P(s)} + \dot{y}^P(s)d\right] - \sqrt{\omega^P(s)}\left[x^Q(t)\sqrt{\omega^Q(t)} + \dot{y}^Q(t)d\right] = 0 \\
\sqrt{\omega^Q(t)}\left[y^P(s)\sqrt{\omega^P(s)} - \dot{x}^P(s)d\right] - \sqrt{\omega^P(s)}\left[y^Q(t)\sqrt{\omega^Q(t)} - \dot{x}^Q(t)d\right] = 0
\end{aligned} \tag{4.32}$$

where $0 \le s, t \le 1$. Now we will introduce two new auxiliary variables $\check{u}$ and $\check{v}$ by setting

$$\check{u}^2 = \omega^P(s) \quad a^u \le \check{u} \le b^u \tag{4.33}$$

$$\check{v}^2 = \omega^Q(t) \quad a^v \le \check{v} \le b^v \tag{4.34}$$

where $a^u = \sqrt{\min_i(\omega_i^P)}$, $b^u = \sqrt{\max_i(\omega_i^P)}$, $a^v = \sqrt{\min_i(\omega_i^Q)}$ and $b^v = \sqrt{\max_i(\omega_i^Q)}$. When $\min_i(\omega_i^P)$ or $\min_i(\omega_i^Q)$ is negative, we set $a^u = 0$ or $a^v = 0$. Therefore, equations (4.32) reduce to

$$\begin{aligned}
\check{v}\left[x^P(s)\check{u} + \dot{y}^P(s)d\right] - \check{u}\left[x^Q(t)\check{v} + \dot{y}^Q(t)d\right] = 0 \\
\check{v}\left[y^P(s)\check{u} - \dot{x}^P(s)d\right] - \check{u}\left[y^Q(t)\check{v} - \dot{x}^Q(t)d\right] = 0 \\
\check{u}^2 - \omega^P(s) = 0 \\
\check{v}^2 - \omega^Q(t) = 0
\end{aligned} \tag{4.35}$$

where $0 \le s, t \le 1$, $a^u \le \check{u} \le b^u$ and $a^v \le \check{v} \le b^v$. By rescaling $\check{u}$ and $\check{v}$ as above such that $u \equiv \frac{\check{u}-a^u}{b^u-a^u}$ and $v \equiv \frac{\check{v}-a^v}{b^v-a^v}$, so that $0 \le u, v \le 1$, equations (4.35) reduce to

$$f(s,t,u,v) = [(b^v - a^v)v + a^v]\left[x^P(s)[(b^u - a^u)u + a^u] + \dot{y}^P(s)d\right]$$

$$
\begin{aligned}
& - \quad [(b^u - a^u)u + a^u]\left[x^Q(t)[(b^v - a^v)v + a^v] + \dot{y}^Q(t)d\right] = 0 \\
g(s,t,u,v) &= [(b^v - a^v)v + a^v]\left[y^P(s)[(b^u - a^u)u + a^u] - \dot{x}^P(s)d\right] \\
& - \quad [(b^u - a^u)u + a^u]\left[y^Q(t)[(b^v - a^v)v + a^v] - \dot{x}^Q(t)d\right] = 0 \\
\zeta(s,t,u,v) &= [(b^u - a^u)u + a^u]^2 - \omega^P(s) = 0 \\
\eta(s,t,u,v) &= [(b^v - a^v)v + a^v]^2 - \omega^Q(t) = 0
\end{aligned} \tag{4.36}
$$

Using equations (4.25), (4.30) and (4.31), we have four simultaneous tetravariate Bernstein polynomial equations as follows:

$$f(s,t,u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n}\sum_{k=0}^{1}\sum_{l=0}^{1} f_{ijkl}B_{i,m}(s)B_{j,n}(t)B_{k,1}(u)B_{l,1}(v) = 0 \tag{4.37}$$

$$g(s,t,u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n}\sum_{k=0}^{1}\sum_{l=0}^{1} g_{ijkl}B_{i,m}(s)B_{j,n}(t)B_{k,1}(u)B_{l,1}(v) = 0 \tag{4.38}$$

$$\zeta(s,u) = \sum_{i=0}^{2m-2}\sum_{k=0}^{2} \zeta_{ik}B_{i,2m-2}(s)B_{k,2}(u) \tag{4.39}$$

$$\eta(t,v) = \sum_{j=0}^{2n-2}\sum_{l=0}^{2} \eta_{jl}B_{j,2n-2}(t)B_{l,2}(v) \tag{4.40}$$

where

$$f_{ijkl} = x_i^P\alpha_k^u\alpha_l^v + d\bar{y}_i^P\alpha_l^v - x_j^Q\alpha_k^u\alpha_l^v - d\bar{y}_j^Q\alpha_k^u \tag{4.41}$$

$$g_{ijkl} = y_i^P\alpha_k^u\alpha_l^v - d\bar{x}_i^P\alpha_l^v - y_j^Q\alpha_k^u\alpha_l^v + d\bar{x}_j^Q\alpha_k^u \tag{4.42}$$

$$\zeta_{ik} = \beta_k^u - \omega_i^P \tag{4.43}$$

$$\eta_{jl} = \beta_l^v - \omega_j^Q \tag{4.44}$$

where the *bar* denotes the degree elevation and α_k^u, α_l^v, β_k^u, β_l^v are Bézier coefficients of $(b^u-a^u)u+a^u$, $(b^v-a^v)v+a^v$, $[(b^u-a^u)u+a^u]^2$, $[(b^v-a^v)v+a^v]^2$ respectively, see equations (4.25). Each of these equations can be converted into a Bézier hyperpatch using the linear precision property of Bernstein

polynomial as follows

$$\mathbf{F}(s,t,u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n}\sum_{k=0}^{1}\sum_{l=0}^{1}\begin{pmatrix} i/m \\ j/n \\ k \\ l \\ f_{ijkl}\end{pmatrix} B_{i,m}(s)B_{j,n}(t)B_{k,1}(u)B_{l,1}(v) \tag{4.45}$$

$$\mathbf{G}(s,t,u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n}\sum_{k=0}^{1}\sum_{l=0}^{1}\begin{pmatrix} i/m \\ j/n \\ k \\ l \\ g_{ijkl}\end{pmatrix} B_{i,m}(s)B_{j,n}(t)B_{k,1}(u)B_{l,1}(v) \tag{4.46}$$

$$\mathbf{Z}(s,u) = \sum_{i=0}^{2m-2}\sum_{k=0}^{2}\begin{pmatrix} i/(2m-2) \\ k/2 \\ \zeta_{ik}\end{pmatrix} B_{i,2m-2}(s)B_{k,2}(u) \tag{4.47}$$

$$\mathbf{H}(t,v) = \sum_{j=0}^{2n-2}\sum_{l=0}^{2}\begin{pmatrix} j/(2n-2) \\ l/2 \\ \eta_{jl}\end{pmatrix} B_{j,2n-2}(t)B_{l,2}(v) \tag{4.48}$$

The above equations are simply an extension of equations (4.28) and (4.29) to *four* dimensions. We can use the same method that we used in two dimensional case to find the roots by projecting on each of the planes (s,w) (t,w) (u,w) (v,w) and following the same procedure as before using Bernstein subdivision method coupled with rounded interval arithmetic [59], [60] (see also section 3.6).

4.3.4 Self-Intersection

Self-intersections of a normal offset of a planar polynomial curve can be obtained by the similar method to the intersections of normal offsets of two planar polynomial curves, except that we need to factor out $s-t$ from the governing equations to avoid the trivial solution $s=t$ which delays a solution process based on subdivision as it attempts to resolve an infinite number of roots. By dropping the superscripts P and Q and rearranging equations (4.35) we obtain

$$\check{u}\check{v}\left[x(s)-x(t)\right]+d\left[\check{v}\dot{y}(s)-\check{u}\dot{y}(t)\right]=0$$
$$\check{u}\check{v}\left[y(s)-y(t)\right]+d\left[-\check{v}\dot{x}(s)+\check{u}\dot{x}(t)\right]=0$$

$$\begin{aligned} \check{u}^2 - \omega(s) &= 0 \\ \check{v}^2 - \omega(t) &= 0 \end{aligned} \tag{4.49}$$

Apparently there are no $s-t$ factors in the last two equations. In the first two equations, we recognize that the first terms of each equation (i.e. $x(s)-x(t)$ and $y(s)-y(t)$ where $x(\tau)$, $y(\tau)$ are polynomials) contain the factor $s-t$, but the second terms do not have this factor. Therefore we need to manipulate the first two equations so that the resulting two equations possess the factor $s-t$. First multiply the first equation by $\dot{x}(t)$ and the second by $\dot{y}(t)$ and add them together resulting in

$$\check{u}\left[\dot{x}(t)(x(s)-x(t)) + \dot{y}(t)(y(s)-y(t))\right] + d\left[\dot{x}(t)\dot{y}(s) - \dot{x}(s)\dot{y}(t)\right] = 0 \tag{4.50}$$

where $\check{v}$ is divided out, since we are assuming that the generator curve is regular. Similarly, multiplying the first equation by $\dot{x}(s)$ and the second by $\dot{y}(s)$ and adding them together and dividing out by $\check{u}$ yields

$$\check{v}\left[\dot{x}(s)(x(s)-x(t)) + \dot{y}(s)(y(s)-y(t))\right] + d\left[\dot{x}(t)\dot{y}(s) - \dot{x}(s)\dot{y}(t)\right] = 0 \tag{4.51}$$

The first brackets of equations (4.50) and (4.51) obviously contain the factor $s-t$. We will prove in the sequel that the second brackets of the two equations also have this factor. For the purposes of this proof, we express $x(\tau)$ and $y(\tau)$ in the monomial basis

$$x(\tau) = \sum_{i=0}^{m} c_i^x \tau^i \tag{4.52}$$

$$y(\tau) = \sum_{i=0}^{m} c_i^y \tau^i \tag{4.53}$$

then

$$\begin{aligned} \dot{x}(t)\dot{y}(s) - \dot{x}(s)\dot{y}(t) &= \sum_{i=1}^{m}\sum_{j=1}^{m} ij c_i^x c_j^y [s^{j-1}t^{i-1} - s^{i-1}t^{j-1}] \\ &= \sum_{i=1}^{m}\sum_{j=1}^{m} ij c_i^x c_j^y \begin{cases} s^{j-1}t^{j-1}(t^{i-j} - s^{i-j}) & i > j \\ s^{i-1}t^{i-1}(s^{j-i} - t^{j-i}) & i < j \end{cases} \end{aligned} \tag{4.54}$$

obviously the terms $t^{i-j} - s^{i-j}$ and $s^{j-i} - t^{j-i}$ possess the factor $s-t$. Therefore, equations (4.50) and (4.51) have the factor of $s-t$ so that we can divide out the factor. Since we have multiplied

the following matrix to a column vector containing the first two equations of system (4.49)

$$\begin{pmatrix} \dot{x}(t) & \dot{y}(t) \\ \dot{x}(s) & \dot{y}(s) \end{pmatrix} \tag{4.55}$$

we have induced an extra pair of roots where the tangent vectors $[\dot{x}(s), \dot{y}(s)]^T$, $[\dot{x}(t), \dot{y}(t)]^T$ are linearly dependent, or equivalently a pair of roots which satisfy the following equation

$$det \begin{pmatrix} \dot{x}(t) & \dot{y}(t) \\ \dot{x}(s) & \dot{y}(s) \end{pmatrix} = \dot{x}(t)\dot{y}(s) - \dot{y}(t)\dot{x}(s) = 0 \tag{4.56}$$

These roots can be removed from the solution set after completion of the algorithm without difficulty. Finally the governing equations without the trivial solution $s = t$ are given by

$$\begin{aligned} &\frac{\check{u}\left[\dot{x}(t)(x(s) - x(t)) + \dot{y}(t)(y(s) - y(t))\right] + d\left[\dot{x}(t)\dot{y}(s) - \dot{x}(s)\dot{y}(t)\right]}{s - t} = 0 \\ &\frac{\check{v}\left[\dot{x}(s)(x(s) - x(t)) + \dot{y}(s)(y(s) - y(t))\right] + d\left[\dot{x}(t)\dot{y}(s) - \dot{x}(s)\dot{y}(t)\right]}{s - t} = 0 \\ &\check{u}^2 - \omega(s) = 0 \\ &\check{v}^2 - \omega(t) = 0 \end{aligned} \tag{4.57}$$

where the degree of the equations from the first are $[s : m-1,\ t : 2m-2,\ \check{u} : 1]$, $[s : 2m-2,\ t : m-1,\ \check{v} : 1]$, $[s : 2m-2,\ \check{u} : 2]$, $[t : 2m-2,\ \check{v} : 2]$ respectively. Note that our method requires solutions of polynomials of degree 4 for a cubic polynomial curve and 6 for a quartic curve to compute the self-intersections. The first two equations of (4.57) are similar to the equations $79(a)$ and $79(b)$ of [25], except that their equations are all squared. The division operation of the Bernstein polynomials in two variables is given as follows. We consider two bivariate Bernstein polynomials expressed as

$$\phi(s,t) = \sum_{k=0}^{m_1} \sum_{l=0}^{n_1} \phi_{kl} B_{k,m_1}(s) B_{l,n_1}(t) \tag{4.58}$$

$$\psi(s,t) = \sum_{k=0}^{m_2} \sum_{l=0}^{n_2} \psi_{kl} B_{k,m_2}(s) B_{l,n_2}(t) \tag{4.59}$$

We assume $m_1 \geq m_2$ and $n_1 \geq n_2$. If $q(s,t)$ is the quotient with no remainder when $\phi(s,t)$ is divided by $\psi(s,t)$, then

$$\phi(s,t) = q(s,t)\psi(s,t) \tag{4.60}$$

The degrees of the quotient $q(s,t)$ are $m_1 - m_2$ and $n_1 - n_2$ in s and t respectively. If we write the Bernstein coefficients of the quotient as q_{kl}, which we want to determine, and equate the coefficients of each Bernstein basis on both sides of equation (4.60) we obtain a system of $(m_1+1)(n_1+1)$ linear equations with $(m_1 - m_2 + 1)(n_1 - n_2 + 1)$ unknowns:

$$\phi_{kl} = \sum_{i=max(0,k-m_2)}^{min(m_1-m_2,k)} \sum_{j=max(0,l-n_2)}^{min(n_1-n_2,l)} \frac{\binom{m_1-m_2}{i}\binom{m_2}{k-i}}{\binom{m_1}{k}} \frac{\binom{n_1-n_2}{j}\binom{n_2}{l-j}}{\binom{n_1}{l}} q_{ij}\psi_{k-i,l-j} \quad (4.61)$$

This linear system can be written as $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is a known $(m_1+1)(n_1+1) \times (m_1 - m_2 + 1)(n_1 - n_2 + 1)$ matrix of rank $(m_1 - m_2 + 1)(n_1 - n_2 + 1)$, $\mathbf{b}$ is a known $(m_1+1)(n_1+1)$-vector and $\mathbf{x}$ is a solution vector containing with $(m_1 - m_2 + 1)(n_1 - n_2 + 1)$ elements. We can solve this overdetermined linear system $\mathbf{Ax} = \mathbf{b}$ by minimizing the Euclidean length of $\mathbf{Ax} - \mathbf{b}$ using the singular value decomposition (SVD) method [56]. In this case, there is always an exact solution q_{ij}, since there is a unique solution to $\mathbf{Ax} = \mathbf{b}$.

4.4 Towards Fully Automated Pocket Machining

To illustrate the computation of the singularities of a normal offset of a planar integral polynomial curve and the intersections of the two specific normal offsets of two planar integral polynomial curves, we used four progenitor curves. A superbola ($y = x^4$) in Bézier form, a bottle-shaped degree six Bézier curve, two parabolas in Bézier form with the same shape but different orientations and a wing-shaped cubic B-spline curve.

The control points of the superbola are given by

$$(-1.5, 5.0625),\ (-0.75, -5.0625),\ (0, 5.0625),\ (0.75, -5.0625),\ (1.5, 5.0625) \quad (4.62)$$

The control points of the bottle-shaped Bézier curve are given by

$$(-0.1, 1),\ (0.3, 0.7),\ (-1, 0.6),\ (0, 0),\ (1, 0.6),\ (-0.3, 0.7),\ (0.1, 1)$$

The control points of the two parabolas are given by

$$(-1.5, 2.25),\ (0, -2.25),\ (1.5, 2.25)\ \textit{and}\ (-1.5, -2.16),\ (0, 2.34),\ (1.5, -2.16)$$

The control points and knot vector of the wing-shaped B-spline curve are given by

$$(9,1),\ (7.4,2.65),\ (4.5,4.3),\ (0,2),\ (4.1,0.95),\ (6.9,1.8),\ (9,1)$$

$$[0,\ 0,\ 0,\ 0,\ 0.25,\ 0.5,\ 0.75,\ 1,\ 1,\ 1,\ 1]$$

Figure 4-3 (a) shows the superbola curve and its interior offset with $d = -0.8$. Since the magnitude of the offset distance exceeds the smallest radius of curvature 0.4648 of the superbola, there are four cusps and two self-intersections. The locations of the cusps are at $t = 0.269, 0.391, 0.609, 0.731$ which corresponds to (-0.053, 0.711),(-0.216, 0.804),(0.216, 0.804),(0.053, 0.711) in the x-y coordinates. Two self-intersections occur at $(s,t) = (0.222, 0.466)$, (0.466,0.222) and (0.534, 0.778), (0.778,0.534) which are (-0.099,0.800), (0.099,0.800) in xy-coordinates. For this example there was no parallel tangents through out the progenitor curve therefore we did not obtain any unwanted roots which come from equation (4.56). Figure 4-3 (b) is the global offset of the superbola where the regions $0.222 \leq t \leq 0.466$ and $0.534 \leq t \leq 0.778$ have been trimmed off.

The next illustration is a degree six Bézier curve which is symmetric with respect to the y-axis and has a bottle-shape. Figure 4-4 (a) shows the progenitor curve and its internal offset with $d = -0.05$. There are two self-intersections due to the constriction of the curve compared with the offset distance. As depicted in Figure 4-4 (a) with circles, we have four pairs of unwanted roots due to the parallel tangents. The location of the self-intersections in parameter space are $(s,t) = (0.035, 0.965)$, $(0.965, 0.035)$ and $(0.102, 0.898)$, $(0.898, 0.102)$ which are $(0, 0.964)$ and (0, 0.815) in x-y coordinates. By trimming off the region $0.035 \leq t \leq 0.102$ and $0.898 \leq t \leq 0.965$ we obtain the global offset curve which is illustrated in Figure 4-5 (a). Figure 4-5 (b) shows the tool path along the global offset which leaves an undercut region. Figure 4-4 (b) is an example of a tacnode with the offset $d = -0.0314$. Our algorithm can also detect this point without failure which is $(s,t) = (0.065, 0.935)$, $(0.935, 0.065)$ and $(0, 0.893)$ in the x-y coordinates.

Figure 4-6 shows the intersection of normal offsets of the two parabolas with $d = -0.8$. The intersection point coincides with the self-intersection point of one of the parabolas. The locations of the intersection points are $(s,t) = (0.708, 0.5)$, $(0.292, 0.5)$ which are $(0, 0.89)$ in the x-y coordinates. When the floating point arithmetic is used with high tolerance, the subdivision method fails to find both roots, see Table 4.1.

Finally we will illustrate the intersections of normal offsets of two planar integral polynomial curves. The wing-shaped cubic B-spline curve can be split into four cubic Bézier curves by subdi-

vision at knots 0.25, 0.5, 0.75. Let us call the four curves C_1, C_2, C_3 and C_4 in increasing t order, see Figure 4-7 (a). Although we split the curve into four, parameter t refers to the original B-spline curve. When the interior offset $d = -0.8$ is taken, the smallest radius of curvature of curves C_2 and C_3 is smaller than 0.8. Therefore there are cusps for these offset curves, see Figure 4-7 (b). Normal offsets of these curves do not have self-intersections but the normal offsets of two curves intersect. Curve C_2 has a cusp at $t = 0.468$ and $(x, y) = (2.185, 1.977)$. Similarly curve C_3 has a cusp at $t = 0.559$ with $(x, y) = (2.124, 2.460)$. The intersections of the normal offsets of two curves C_2 and C_3 are $t = 0.389$ and $t = 0.634$ and $(x, y) = (2.608, 2.276)$. The starting point of curve C_1 and the ending point of curve C_4 coincides and they form a cusp. Consequently their interior offsets intersect. The intersection point is given by $t = 0.139$ and $t = 0.865$ with $(x, y) = (6.080, 2.278)$. Removing three regions $0 \leq t \leq 0.139$, $0.389 \leq t \leq 0.634$ and $0.865 \leq t \leq 1$, we have the global offset which are shown in Figure 4-7 (c).

The CPU time comparison for the solution (formulation time is not included, see section 3.5) between the subdivision method with floating point arithmetic (FPA) and with rounded interval arithmetic (RIA) implemented in C++ are listed in Table 4.1. The CPU time in seconds was measured on a graphics workstation running at 36MHz. These times are indicative from a non-optimized implementation. Tol refers to the tolerance. *We can observe from the table that RIA requires one order of magnitude higher CPU time than the FPA.* For computing the self-intersections of the normal offset of the superbola and the tacnode of the normal offset of bottle-shaped curve the CPU time is very large due to "tangency". The rate of convergence drops and an extensive amount of binary subdivision is performed. We can avoid this kind of problem by using a minimization technique after a coarse subdivision (for example 10^{-3}) and especially for the superbola case, we can split the curve into five curves at the cusps and trim off the tangency region and treat the problem as the intersections of the two normal offset curves.

As illustrated by examples above, a new robust method to compute the singularities of a normal offset of a planar integral polynomial curve and the intersections of two specific normal offsets of planar integral polynomial curves is introduced. This method can be applied to automatic tool path generation for $2\frac{1}{2}$-D milling NC machining, tolerance region construction and feature recognition through construction of skeletons of geometric models.

Item	Tol	CPU Time in Seconds	
		FPA	RIA
Cusps of Normal Offset of Superbola	10^{-8}	0.2	7.0
Cusps of Normal Offset of Superbola	10^{-10}	0.3	8.5
Cusps of Normal Offset of Superbola	10^{-12}	0.3	10.3
Self-Intersections of Normal Offset of Superbola	10^{-3}	54.7	768.8
Self-Intersections of Normal Offset of Superbola	10^{-4}	665.8	2813.4
Self-Intersections of Normal Offset of Bottle-Shaped Curve	10^{-8}	17.2	594.9
Self-Intersections of Normal Offset of Bottle-Shaped Curve	10^{-10}	21.1	617.7
Self-Intersections of Normal Offset of Bottle-Shaped Curve	10^{-12}	23.8	778.0
Tacnode of Normal Offset of Bottle-Shaped Curve	10^{-3}	6.1	209.7
Tacnode of Normal Offset of Bottle-Shaped Curve	10^{-4}	18.1	549.8
Intersections of Normal Offsets of Two Parabolas	10^{-8}	1.5	22.0
Intersections of Normal Offsets of Two Parabolas	10^{-11}	m	29.8
Intersections of Normal Offsets of Two Parabolas	10^{-12}	m	32.9
Cusp of Normal Offset of C_2	10^{-8}	*	0.4
Cusp of Normal Offset of C_3	10^{-8}	*	0.6
Intersections of Normal Offsets of C_1 and C_4	10^{-8}	0.6	10.9
Intersections of Normal Offsets of C_2 and C_3	10^{-8}	1.6	28.7

Table 4.1: CPU Time comparison between Bernstein subdivision method with FPA and with RIA (See Table 3.4 for symbolic entries)

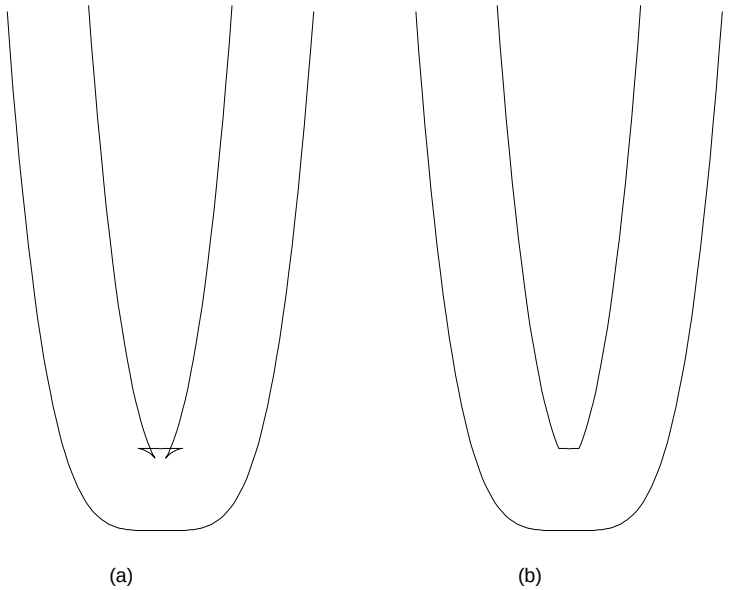

Figure 4-3: (a) The superbola and its interior offset with $d = -0.8$; (b) global offset.

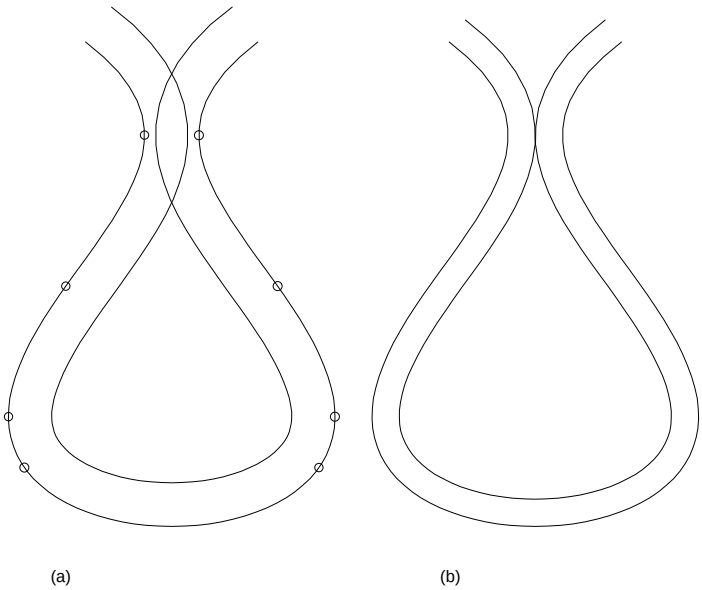

Figure 4-4: (a) Degree six Bézier curve and its internal offset with $d = -0.05$; (b) offset curve self-intersects forming a tacnode with $d \simeq -0.03141$.

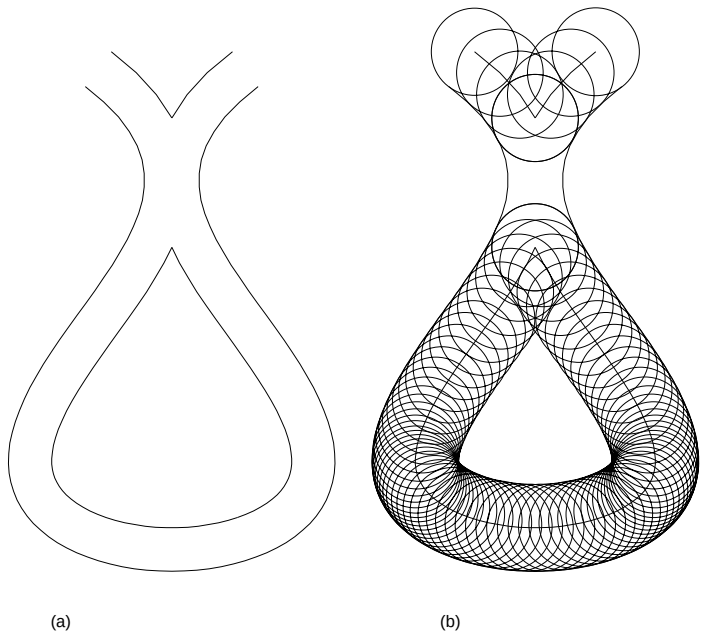

Figure 4-5: (a) Global offset of degree six Bézier curve with $d = -0.05$; (b) tool path along the trimmed offset.

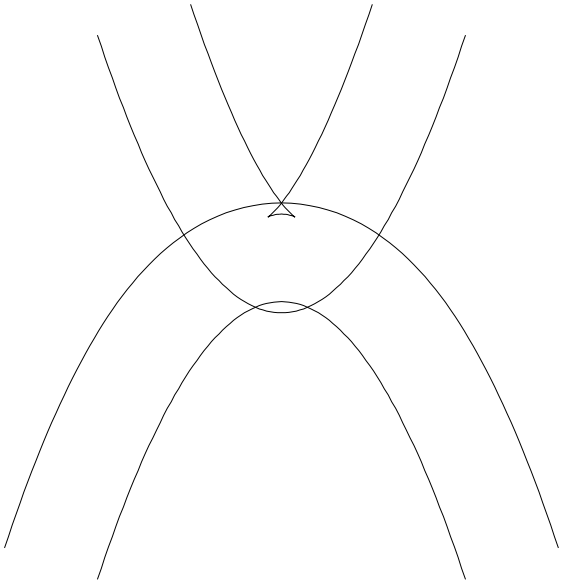

Figure 4-6: Two parabolas and their offsets with $d = -0.8$

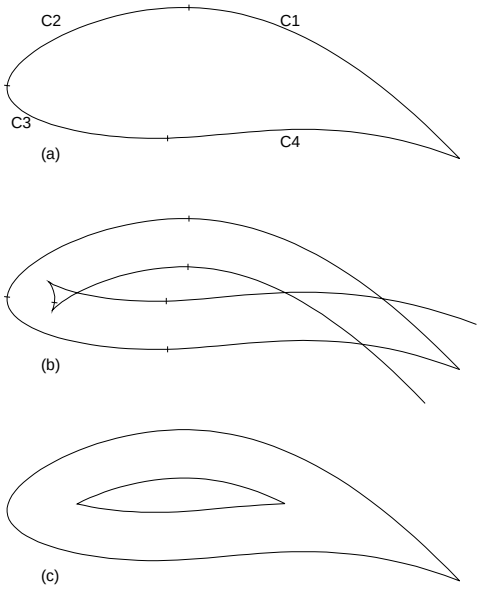

Figure 4-7: (a) Cubic B-spline curve subdivided into four cubic Bézier curves; (b) interior offsets of four cubic Bézier curves with $d = -0.8$; (c) global offsets of four cubic Bézier curves

Chapter 5

Continuous Decomposition of Surface Patches

5.1 Introduction and Literature Review

Free-form surfaces, also called sculptured surfaces, are widely used in scientific and engineering applications. Propeller and turbine blades are manufactured by numerically controlled (NC) milling machines. When a ball end-mill cutter is used, the cutter radius must be smaller than the smallest concave radius of curvature of the surface to be machined to avoid *local overcut (gouging)*, as explained in section 2.2.3. Gouging is the one of the most critical problems in NC machining of free-form surfaces. Therefore, we must determine the distribution of the principal curvatures of the surface, which are upper and lower bounds on the curvature at a given point, to select the cutter size [28].

Developable surfaces are the surfaces which can be unfolded or developed onto a plane without stretching or tearing. They are of considerable importance to plate-metal-based industries as shipbuilding. For a developable surface the Gaussian curvature is zero everywhere [16]. Thus the manufacturer would profit from prior knowledge of the distribution of the Gaussian curvature of the metal plate.

Fairing is the process of eliminating shape irregularities in order to produce a smoother shape, see section 2.3. The Gaussian, mean and principal curvatures are used for the detection of surface irregularities [67], [66], [102], [1], [2]. The set of curvature extrema of a fair surface should coincide

with the designer's intention. Therefore, computation of all extrema of curvatures is desirable.

The variation of curvature can be displayed using a color coded map. Color coded maps provide a rough idea of the differential properties of surfaces but are not sufficient to provide detailed machining information nor permit automation of the machining process or of fairing algorithms. Contour lines of constant curvature can also be used to display and visualize the variation of curvature. Munchmeyer [67] calculates the curvature on a lattice and linearly interpolates the contour points. Discrete color coded maps of curvature, such as those in Dill [15] and Beck et al [4] and lattice methods for curvature contouring do not guarantee to locate all the stationary points (local extrema and saddle points) of curvature, and hence may fail to provide the correct topological decomposition of the surface on the basis of curvature to the manufacturer or to a fairing process. At the time of writing the final draft of this thesis, Elber and Cohen [20] have presented a method to partition a NURBS surface into three disjoint trimmed surfaces (convex, concave and saddle) and to determine global bounds on surface curvatures using symbolic and numeric methods. However, we believe that our method discussed in this Chapter and in [77], [59] presents a far more general and robust algorithm for curvature analysis. In effect the paper by Elber and Cohen does not address the numerical robustness problem and does not directly process principal curvature functions involving radicals.

This Chapter presents a robust procedure for contouring curvature of a free-form surface and subdividing the surface into regions of specific range of curvature which can be used for design, fairing and manufacturing automation. This Chapter is structured as follows. Section 5.2 begins with a review of relevant differential geometry properties of parametric surfaces. Section 5.3 formulates the problem of computation of the stationary points of curvature of a Bézier surface. Section 5.4 describes how to contour constant curvature lines and outlines an algorithm to polygonize the area between the contour lines. Section 5.5 gives an example to illustrate our method, which may be useful for fairing and $3D$ and $5D$ machining. This Chapter refers to two appendices. Appendix A gives the formulas for evaluating the partial derivatives of various curvatures up to second order[1]. Appendix B reviews the classification of stationary points of functions.

[1] The first order partial derivatives are necessary for evaluating the stationary points of curvatures and the second order partial derivatives are necessary for classification of stationary points.

5.2 Differential Geometry of Surfaces

The differential geometry of curves and surfaces is fundamental in CAGD. The curves and surfaces treated in differential geometry are defined by functions which can be differentiated a certain number of times. A book by Hilbert and Cohn-Vossen [36] and a recent book by Koenderink [49] provide intuitive access to the extensive mathematical literature on three-dimensional shape analysis. The books by Struik, doCarmo and Banchoff [98], [16], [3] offer firm theoretical basis to the differential geometry aspects of three-dimensional shape description. In this section, we summarize the relevant definitions employed in this work.

A general parametric surface can be defined as a vector-valued mapping from two-dimensional parametric uv-space to a set of three-dimensional coordinates

$$\mathbf{r}(u,v) = [x(u,v),\ y(u,v),\ z(u,v)]^T \tag{5.1}$$

There are important geometric structures defined by the surface, the first and second fundamental forms. The shape of a surface is completely characterized by these two fundamental forms. The first fundamental form I provides us metrical properties of surfaces such as measurement of lengths, areas and angles between two curves on the surface. The first fundamental form I is defined as the dot product of infinitesimal displacement $d\mathbf{r}$ with itself.

$$\begin{aligned} I &= d\mathbf{r} \cdot d\mathbf{r} = (\mathbf{r}_u du + \mathbf{r}_v dv) \cdot (\mathbf{r}_u du + \mathbf{r}_v dv) \\ &= E du^2 + 2F du dv + G dv^2 \\ &= d\mathbf{q}[\Gamma] d\mathbf{q}^T \end{aligned} \tag{5.2}$$

where subscripts denote partial derivatives and

$$E = \mathbf{r}_u \cdot \mathbf{r}_u, \quad F = \mathbf{r}_u \cdot \mathbf{r}_v, \quad G = \mathbf{r}_v \cdot \mathbf{r}_v \tag{5.3}$$

$$d\mathbf{q} = [du \ \ dv] \tag{5.4}$$

$$[\Gamma] = \begin{pmatrix} E & F \\ F & G \end{pmatrix} \tag{5.5}$$

The second fundamental form II allows us to analyze the surface curvature at a given point and is defined as the dot product of infinitesimal displacement $d\mathbf{r}$ and infinitesimal variation $d\mathbf{N}$ of the

surface unit normal vector $\mathbf{N}$.

$$\begin{aligned} II &= -d\mathbf{r} \cdot d\mathbf{N} = -(\mathbf{r}_u du + \mathbf{r}_v dv) \cdot (\mathbf{N}_u du + \mathbf{N}_v dv) \\ &= Ldu^2 + 2Mdudv + Ndv^2 \\ &= d\mathbf{q}[\Delta]d\mathbf{q}^T \end{aligned} \tag{5.6}$$

where

$$\mathbf{N} = \frac{\mathbf{r}_u \times \mathbf{r}_v}{|\mathbf{r}_u \times \mathbf{r}_v|} \tag{5.7}$$

$$L = \mathbf{N} \cdot \mathbf{r}_{uu}, \quad M = \mathbf{N} \cdot \mathbf{r}_{uv}, \quad N = \mathbf{N} \cdot \mathbf{r}_{vv} \tag{5.8}$$

$$[\Delta] = \begin{pmatrix} L & M \\ M & N \end{pmatrix} \tag{5.9}$$

and $\frac{\partial}{\partial u}(\mathbf{r}_u \cdot \mathbf{N}) = 0$, $\frac{\partial}{\partial v}(\mathbf{r}_v \cdot \mathbf{N}) = 0$ are used in the derivation . In order to quantify the curvatures of a surface S, we consider a curve C on S which passes through point P as shown in Figure 5-1. $\mathbf{t}$ is the unit tangent vector and $\mathbf{n}$ is the unit normal vector of the curve C at point P. If $\mathbf{k}$ is the curvature vector of the curve C on the surface S at P, which can be obtained by $\mathbf{k} = \frac{d\mathbf{t}}{ds}$, we can represent $\mathbf{k}$ as sum of a normal and a tangential component $\mathbf{k}_n$ and $\mathbf{k}_g$. $\mathbf{k}_n$ is called the normal curvature vector and $\mathbf{k}_g$ is called the geodesic curvature vector. The normal curvature vector can be expressed as a multiple of the unit surface normal vector $\mathbf{N}$ namely

$$\mathbf{k}_n = -\kappa\mathbf{N} \tag{5.10}$$

where κ is the normal curvature and can be obtained by differentiating the equation $\mathbf{N} \cdot \mathbf{t} = 0$ along C with respect to the arc length.

$$\begin{aligned} \kappa &= -\frac{d\mathbf{t}}{ds} \cdot \mathbf{N} = \mathbf{t} \cdot \frac{d\mathbf{N}}{ds} = \frac{d\mathbf{r}}{ds} \cdot \frac{d\mathbf{N}}{ds} \\ &= -\frac{II}{I} = -\frac{Ldu^2 + 2Mdudv + Ndv^2}{Edu^2 + 2Fdudv + Gdv^2} = -\frac{L + 2M\lambda + N\lambda^2}{E + 2F\lambda + G\lambda^2} \end{aligned} \tag{5.11}$$

where $\lambda = \frac{dv}{du}$ specifies the direction of the curve. The sign convention used in equation (5.11) ensures that positive κ is on the side of the surface opposite to the direction of the normal. At a given point (u, v), κ varies with each direction λ. The extreme values of κ can be obtained by

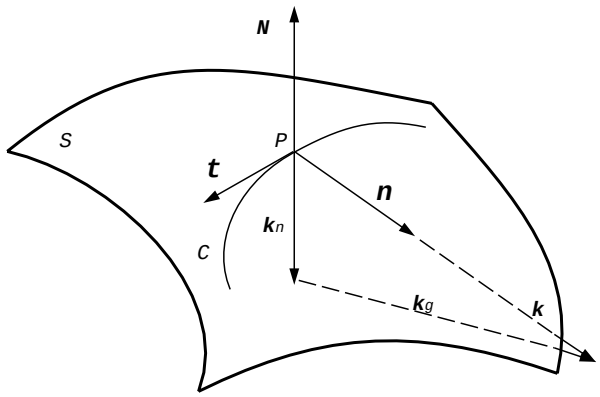

Figure 5-1: Definition of normal curvature

evaluating $\frac{d\kappa}{d\lambda} = 0$ which gives:

$$(E + 2F\lambda + G\lambda^2)(\lambda N + M) - (L + 2M\lambda + N\lambda^2)(\lambda G + F) = 0 \tag{5.12}$$

Since

$$\begin{aligned} E + 2F\lambda + G\lambda^2 &= (E + F\lambda) + \lambda(F + G\lambda), \\ L + 2M\lambda + N\lambda^2 &= (L + M\lambda) + \lambda(M + N\lambda) \end{aligned}$$

equation (5.12) can be reduced to

$$(E + F\lambda)(M + \lambda N) = (L + M\lambda)(F + \lambda G) \tag{5.13}$$

Using equation (5.13), equation (5.11) can be rewritten as:

$$\kappa = -\frac{L + 2M\lambda + N\lambda^2}{E + 2F\lambda + G\lambda^2} = -\frac{M + \lambda N}{F + \lambda G} = -\frac{L + M\lambda}{E + \lambda F} \tag{5.14}$$

Therefore κ satisfies the two simultaneous equations

$$\begin{aligned} (L + \kappa E)du + (M + \kappa F)dv = 0 \\ (M + \kappa F)du + (N + \kappa G)dv = 0 \end{aligned} \tag{5.15}$$

These equations can be simultaneously satisfied if and only if

$$\begin{vmatrix} L+\kappa E & M+\kappa F \\ M+\kappa F & N+\kappa G \end{vmatrix} = 0 \tag{5.16}$$

This quadratic equation in κ gives the upper and lower bounds of the normal curvature, which are the maximum principal curvature κ_{max} and the minimum principal curvature κ_{min}. The corresponding directions λ define directions in the uv-plane and the corresponding directions in the tangent plane are called *principal directions of curvature* and are always orthogonal. The two roots are given by

$$\kappa_{max} = H + \sqrt{H^2 - K} \tag{5.17}$$

$$\kappa_{min} = H - \sqrt{H^2 - K} \tag{5.18}$$

where K is the *Gaussian curvature* and H is the *mean curvature* defined by

$$K = \frac{LN - M^2}{EG - F^2} \tag{5.19}$$

$$H = \frac{2FM - EN - GL}{2(EG - F^2)} \tag{5.20}$$

From equations (5.17), (5.18), it is readily seen that

$$K = \kappa_{max}\kappa_{min} \tag{5.21}$$

$$H = \frac{\kappa_{max} + \kappa_{min}}{2} \tag{5.22}$$

If κ_{max} and κ_{min} have the same sign the Gaussian curvature is positive and the point is called *elliptic* point of the surface. Any patch on an ellipsoid is an elliptic region. If either of κ_{max} or κ_{min} is zero, the Gaussian curvature is zero and the point is called *parabolic*. *Developable surfaces* have zero Gaussian curvature at their regular points [16]. Finally, if κ_{max} and κ_{min} have different signs the Gaussian curvature is negative and the point is called *hyperbolic*. Any point on a hyperbolic paraboloid is a hyperbolic point. When κ_{max} and κ_{min} are identical, the point approximates a sphere and is called an *umbilical point*. In the special case, where the identical principal curvature vanish, the surface becomes locally *flat*. Note that at the *flat* point, $K = H = 0$. A spherical umbilic occurs at an elliptic point, while it never occurs at a hyperbolic point. From equation (5.11) it is apparent that at an umbilic I and II are proportional because $\kappa = constant$, and hence we have

the following relation at the umbilic

$$\frac{L}{E} = \frac{M}{F} = \frac{N}{G} \tag{5.23}$$

The net of lines, that have as tangents the principal curvature directions at all of their points, form two sets of curves intersecting at right angles and are called *lines of curvature*. The lines of curvature depend only on the shape of the surface, and not the parametrization. Lines of curvature provide a method to describe the variation of principal curvatures across a surface. At umbilical points only, the principal directions are indeterminate and the net of lines of curvature may have singular properties. Lines of curvature can be obtained by integrating equations (5.15), which will be discussed in section 7.4.

5.3 Stationary Points of Curvature of Free-Form Polynomial Surfaces

To subdivide the surface into regions of specific range of curvature, we need to determine the following.

1. Locations of all the stationary points of the curvature and the associated values of curvature to provide a correct topological decomposition of the surface on the basis of curvature.

2. Global maximum and minimum of the curvature to find the overall range of curvature.

For simplicity, the underlying surface is assumed to be an integral Bézier patch as follows

$$\mathbf{r}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} \mathbf{P}_{ij} B_{i,m}(u) B_{j,n}(v) \tag{5.24}$$

where m, n are the the degrees of the patch in u, v parametric directions, and $\mathbf{P}_{ij}$ are the control points, $0 \leq u, v \leq 1$ and $B_{i,m}(u)$, $B_{j,n}(v)$ are the Bernstein basis functions [110]. Extension to rational Bézier patch and rational B-spline patch, although tedious does not present conceptual difficulties. We also assume that the surface is regular, i.e. its Jacobian has full rank and therefore $\mathbf{r}_u \times \mathbf{r}_v \neq \mathbf{0}$. Points where $\mathbf{r}_u \times \mathbf{r}_v = \mathbf{0}$ correspond to either singularities of the parametrizations or intrinsic degeneracies of the surface like ridges and cusps.

Gaussian, mean and principal curvatures can be evaluated in terms of parametric derivatives of $\mathbf{r}(u,v)$ [16]. Let the curvature in question be denoted by $C(u,v)$, then the following need to be evaluated to locate all the stationary points of curvature and to find the global maximum and minimum values of the curvature to provide a correct topological decomposition of the surface.

1. The four values of curvature at the parameter domain corners

$$C(0,0),\ C(0,1),\ C(1,0),\ C(1,1) \tag{5.25}$$

2. Stationary points along parameter domain boundaries (roots of the 4 equations)

$$\begin{aligned} C_u(u,0) = 0,\ C_u(u,1) = 0,\ 0 \le u \le 1 \\ C_v(0,v) = 0,\ C_v(1,v) = 0,\ 0 \le v \le 1 \end{aligned} \tag{5.26}$$

3. Stationary points within the parameter domain (roots of the 2 simultaneous equations)

$$C_u(u,v) = 0,\ C_v(u,v) = 0,\ 0 \le u,\ v \le 1 \tag{5.27}$$

The curvature values at the parameter domain corners are readily computed. The computation of stationary points of the Gaussian, mean and principal curvatures along the boundary and within the parameter domain are discussed in sections 5.3.1, 5.3.2 and 5.3.3 respectively.

5.3.1 Gaussian Curvature K

To formulate the governing equation for computing the stationary points of Gaussian curvature along the boundary, we substitute the equation (5.19) into (5.26) and express the equation such that the denominator and the numerator only include polynomials.

$$K_u(u,0) = \frac{\hat{A}(u,0)}{S^6(u,0)} = 0, \quad K_u(u,1) = \frac{\hat{A}(u,1)}{S^6(u,1)} = 0, \quad 0 \le u \le 1 \tag{5.28}$$

$$K_v(0,v) = \frac{\bar{A}(0,v)}{S^6(0,v)} = 0, \quad K_v(1,v) = \frac{\bar{A}(1,v)}{S^6(1,v)} = 0, \quad 0 \le v \le 1 \tag{5.29}$$

where

$$S \quad = \quad |\mathbf{S}| = |\mathbf{r}_u \times \mathbf{r}_v| \tag{5.30}$$

$$\hat{A} = A_u S^2 - 4(\mathbf{S} \cdot \mathbf{S}_u)A \tag{5.31}$$

$$\bar{A} = A_v S^2 - 4(\mathbf{S} \cdot \mathbf{S}_v)A \tag{5.32}$$

$\hat{A}$, $\bar{A}$ are polynomials of degree $(10m-7, 10n-6)$, $(10m-6, 10n-7)$ in u and v. Polynomial A and its partial derivatives and partial derivatives of $\mathbf{S}$ are given in appendix A. Since we are assuming a regular surface, $S \neq 0$, we need only set the numerators of equations (5.28) and (5.29) to zero, resulting in

$$\hat{A}(u,0) = 0, \quad \hat{A}(u,1) = 0, \quad 0 \leq u \leq 1 \tag{5.33}$$

$$\bar{A}(0,v) = 0, \quad \bar{A}(1,v) = 0, \quad 0 \leq v \leq 1 \tag{5.34}$$

Therefore for stationary points of Gaussian curvature along the domain boundary we need to solve four univariate polynomial equations (5.33) of degree $10m-7$ in u and (5.34) of degree $10n-7$ in v.

For the stationary points within the domain, we substitute equation (5.19) into (5.27) which yields

$$K_u(u,v) = \frac{\hat{A}(u,v)}{S^6(u,v)} = 0, \quad K_v(u,v) = \frac{\bar{A}(u,v)}{S^6(u,v)} = 0, \quad 0 \leq u,v \leq 1 \tag{5.35}$$

As $S \neq 0$, equations (5.35) are satisfied if

$$\hat{A}(u,v) = 0, \quad \bar{A}(u,v) = 0, \quad 0 \leq u,v \leq 1 \tag{5.36}$$

which are two simultaneous bivariate polynomial equations of degree $(10m-7, 10n-6)$, $(10m-6, 10n-7)$ in u and v.

5.3.2 Mean Curvature H

Similarly to the Gaussian curvature, we have the following equations to evaluate the stationary points of mean curvature H along the boundary

$$H_u(u,0) = \frac{\hat{B}(u,0)}{2S^5(u,0)} = 0, \quad H_u(u,1) = \frac{\hat{B}(u,1)}{2S^5(u,1)} = 0, \quad 0 \leq u \leq 1 \tag{5.37}$$

$$H_v(0,v) = \frac{\bar{B}(0,v)}{2S^5(0,v)} = 0, \quad H_v(1,v) = \frac{\bar{B}(1,v)}{2S^5(1,v)} = 0, \quad 0 \leq v \leq 1 \tag{5.38}$$

where

$$\hat{B} = B_u S^2 - 3(\mathbf{S} \cdot \mathbf{S}_u)B \tag{5.39}$$

$$\bar{B} = B_v S^2 - 3(\mathbf{S} \cdot \mathbf{S}_v)B \tag{5.40}$$

$\hat{B}$, $\bar{B}$ are polynomials of degree $(9m-6, 9n-5)$, $(9m-5, 9n-6)$ in u and v. Polynomial B and its partial derivatives are given in appendix A. As $S \neq 0$, we need only set the numerators of equations (5.37) and (5.38) to zero, resulting in

$$\hat{B}(u,0) = 0, \quad \hat{B}(u,1) = 0, \quad 0 \leq u \leq 1 \tag{5.41}$$

$$\bar{B}(0,v) = 0, \quad \bar{B}(1,v) = 0, \quad 0 \leq v \leq 1 \tag{5.42}$$

Therefore, for the stationary points of mean curvature along the domain boundary we need to solve four univariate polynomial equations (5.41) of degree $9m-6$ in u and (5.42) of degree $9n-6$ in v.

For the stationary points within the domain, we have

$$H_u(u,v) = \frac{\hat{B}(u,v)}{2S^5(u,v)} = 0, \quad H_v(u,v) = \frac{\bar{B}(u,v)}{2S^5(u,v)} = 0, \quad 0 \leq u,v \leq 1 \tag{5.43}$$

Since $S \neq 0$, equations (5.43) reduce to two simultaneous bivariate polynomial equations

$$\hat{B}(u,v) = 0, \quad \bar{B}(u,v) = 0, \quad 0 \leq u,v \leq 1 \tag{5.44}$$

5.3.3 Principal Curvature κ

For obtaining the stationary points of principal curvature κ along the domain boundaries, we substitute equations (5.17) and (5.18) into (5.26) and express the equations such that the denominator and the numerator only include polynomials and square root of polynomials.

$$\begin{aligned}
\kappa_u(u,0) &= \frac{f_1(u,0) \pm f_2(u,0)\sqrt{f_3(u,0)}}{2S^5(u,0)\sqrt{f_3(u,0)}} = 0, \quad 0 \leq u \leq 1 \\
\kappa_u(u,1) &= \frac{f_1(u,1) \pm f_2(u,1)\sqrt{f_3(u,1)}}{2S^5(u,1)\sqrt{f_3(u,1)}} = 0, \quad 0 \leq u \leq 1 \\
\kappa_v(0,v) &= \frac{g_1(0,v) \pm g_2(0,v)\sqrt{f_3(0,v)}}{2S^5(0,v)\sqrt{f_3(0,v)}} = 0, \quad 0 \leq v \leq 1
\end{aligned} \tag{5.45}$$

$$\kappa_v(1,v) = \frac{g_1(1,v) \pm g_2(1,v)\sqrt{f_3(1,v)}}{2S^5(1,v)\sqrt{f_3(1,v)}} = 0, \quad 0 \le v \le 1 \tag{5.46}$$

The plus and minus signs correspond to the maximum and minimum principal curvatures, and $f_1(u,v)$, $f_2(u,v)$, $f_3(u,v)$, $g_1(u,v)$ and $g_2(u,v)$ are polynomials of degree $(14m-9, 14n-8)$, $(9m-6, 9n-5)$, $(10m-6, 10n-6)$, $(14m-8, 14n-9)$, $(9m-5, 9n-6)$ in u and v parameters and are given by

$$f_1(u,v) = (BB_u - 2A_uS^2)S^2 + (8AS^2 - 3B^2)(\mathbf{S}\cdot\mathbf{S}_u) \tag{5.47}$$

$$f_2(u,v) = B_uS^2 - 3(\mathbf{S}\cdot\mathbf{S}_u)B \tag{5.48}$$

$$f_3(u,v) = B^2 - 4AS^2 \tag{5.49}$$

$$g_1(u,v) = (BB_v - 2A_vS^2)S^2 + (8AS^2 - 3B^2)(\mathbf{S}\cdot\mathbf{S}_v) \tag{5.50}$$

$$g_2(u,v) = B_vS^2 - 3(\mathbf{S}\cdot\mathbf{S}_v)B \tag{5.51}$$

First we assume that $f_3 \neq 0$ and also $S \neq 0$, then equations (5.45), (5.46) become

$$f_1(u,0) \pm f_2(u,0)\sqrt{f_3(u,0)} = 0, \quad f_1(u,1) \pm f_2(u,1)\sqrt{f_3(u,1)} = 0, \quad 0 \le u \le 1 \tag{5.52}$$

$$g_1(0,v) \pm g_2(0,v)\sqrt{f_3(0,v)} = 0, \quad g_1(1,v) \pm g_2(1,v)\sqrt{f_3(1,v)} = 0, \quad 0 \le v \le 1 \tag{5.53}$$

Consequently for the stationary points of principal curvatures along the boundary we need to solve four univariate irrational equations involving polynomials and square roots of polynomials (which arise from the analytic expressions of principal curvatures).

When $f_3 = 0$ (or equivalently $H^2 - K = 0$ if $S \neq 0$), equations (5.45), (5.46) become singular. This condition is equivalent to the point where the two principal curvatures are identical, i.e. an umbilical point. If the umbilical point coincides with a local maximum or minimum of the curvature, we cannot use equations (5.52) and (5.53) to locate such a point. In such case we need to locate the umbilical point first. For umbilical points along the domain boundaries, we can do this by solving the following equations

$$H^2(u,0) - K(u,0) = \frac{f_3(u,0)}{4S^6(u,0)} = 0, \quad H^2(u,1) - K(u,1) = \frac{f_3(u,1)}{4S^6(u,1)} = 0, \quad 0 \le u \le 1 \tag{5.54}$$

$$H^2(0,v) - K(0,v) = \frac{f_3(0,v)}{4S^6(0,v)} = 0, \quad H^2(1,v) - K(1,v) = \frac{f_3(1,v)}{4S^6(1,v)} = 0, \quad 0 \le v \le 1 \tag{5.55}$$

Since $S \neq 0$, we need to solve $f_3(u,0) = 0$, $f_3(u,1) = 0$, $f_3(0,v) = 0$, $f_3(1,v) = 0$. Then we use the criterion (see section 7.5) at the umbilic to check if the point is a local extremum of the principal curvatures [61].

In the case of stationary points of principal curvature κ within the domain, the simultaneous

bivariate equations (5.27) become

$$\kappa_u(u,v) = \frac{f_1(u,v) \pm f_2(u,v)\sqrt{f_3(u,v)}}{2S^5(u,v)\sqrt{f_3(u,v)}} = 0, \quad 0 \le u,v \le 1$$
$$\kappa_v(u,v) = \frac{g_1(u,v) \pm g_2(u,v)\sqrt{f_3(u,v)}}{2S^5(u,v)\sqrt{f_3(u,v)}} = 0, \quad 0 \le u,v \le 1 \tag{5.56}$$

Assuming $f_3 \neq 0$ and $S \neq 0$, we obtain

$$f_1(u,v) \pm f_2(u,v)\sqrt{f_3(u,v)} = 0, \quad g_1(u,v) \pm g_2(u,v)\sqrt{f_3(u,v)} = 0, \quad 0 \le u,v \le 1 \tag{5.57}$$

These are two simultaneous bivariate irrational equations involving polynomials and square roots of polynomials (which arise from the analytical expressions of principal curvatures).

At the umbilics, equations (5.56) become singular and similarly to the univariate case for the domain boundaries, we need to locate the umbilical points first by finding the roots of the bivariate polynomial equation $f_3(u,v) = 0$. Let

$$W(u,v) = H^2(u,v) - K(u,v) \tag{5.58}$$

then $W(u,v) = \frac{f_3(u,v)}{4S^6(u,v)}$ is a non-negative function, therefore $W(u,v)$ has a global minimum at the umbilic , see section 7.5. The condition for global minimum at the umbilic implies that $\nabla W = \mathbf{0}$ or equivalently (given that $f_3(u,v) = 0$)

$$W_u = \frac{f_{3u}}{4S^6} = 0, \quad W_v = \frac{f_{3v}}{4S^6} = 0 \tag{5.59}$$

Therefore the locations of umbilics are the solutions of the following three equations, assuming $S \neq 0$

$$f_{3u}(u,v) = 0, \quad f_{3v}(u,v) = 0, \quad f_3(u,v) = 0 \quad 0 \le u,v \le 1 \tag{5.60}$$

These equations can be reduced to :

$$\begin{aligned} BB_u - 2A_uS^2 - 4A(\mathbf{S}\cdot\mathbf{S}_u) &= 0, \\ BB_v - 2A_vS^2 - 4A(\mathbf{S}\cdot\mathbf{S}_v) &= 0, \\ B^2 - 4AS^2 &= 0 \end{aligned} \tag{5.61}$$

with $0 \leq u, v \leq 1$. Since $f_3(u,v) = 0$ at the umbilics, equations (5.57) reduce to $f_1(u,v) = 0$, $g_1(u,v) = 0$. If we substitute the first equation of (5.61) into equation (5.47) and use the fact $f_3 = B^2 - 4AS^2 = 0$, we obtain $f_1(u,v) = 0$. Similarly by substituting the second equation of (5.61) into equation (5.50), we obtain $g_1(u,v) = 0$. Consequently the solutions of equation (5.57) include not only the locations of extrema of principal curvatures but also the locations of the umbilical points. Then we use the criterion in section 7.5 at the umbilical points to check if the umbilical point is a local extremum of principal curvatures.

Cusp is an isolated singular point on the surface where the surface tangent plane is undefined, i.e. $\mathbf{r}_u \times \mathbf{r}_v = \mathbf{0}$. Cusps of an offset surface correspond to points on the progenitor where both of the principal curvatures are equal to $-\frac{1}{d}$ (d is the offset distance) [23]. In this manner, cusps on an offset surface are associated with umbilics of the progenitor. Hence we can locate the cusps on the offset surface using equations (5.60).

The greatest magnitude of the negative principal curvature determines the largest offset distance without degeneration.

5.4 Contouring Method

The constant curvature lines divide the surface into regions of specific range of curvature. The contouring levels should be determined to faithfully represent the curvature distribution. To do this, the following should be used:

- Global maximum and minimum curvature values in the entire domain to find the range of curvature values.
- Locations of all the local maxima and minima of curvature inside the domain around which loops may be formed.
- Locations of all the saddle points of the curvature where the contour lines cross or exhibit more complex behavior.

Classification of stationary points of functions is briefly reviewed in Appendix B.

5.4.1 Finding Starting Points

Contour lines in the parameter space of a bivariate function can be separated into three categories:

- Local maxima and minima of the function are encircled by closed contour lines [50].
- At the precise level of a saddle point, contour line cross or exhibit more complex behavior (eg. monkey saddle $z = x^3 - 3xy^2$, dog saddle $z = 4x^3y - 4xy^3$ etc, [49]).
- Contour lines start from a domain boundary point and end at a domain boundary point.

If the surface is subdivided along the isoparametric line which contain the local maxima and minima of curvature inside the domain and the contouring levels of curvature are chosen such that the contour lines avoid saddle points, as shown in Figures 5-4, 5-5, 5-6, 5-7, 5-8, 5-9, 5-10, 5-11, each sub-patch will contain simple contour branches without loops or singularities. Therefore we can find all the starting points of the various levels of contour lines along the parameter domain boundary of each sub-patch by finding the roots of follow ing equations. Starting with Gaussian curvature

$$K(u,0) = \frac{A(u,0)}{S^4(u,0)} = C_K, \quad K(u,1) = \frac{A(u,1)}{S^4(u,1)} = C_K \quad 0 \le u \le 1 \tag{5.62}$$

$$K(0,v) = \frac{A(0,v)}{S^4(0,v)} = C_K, \quad K(1,v) = \frac{A(1,v)}{S^4(1,v)} = C_K \quad 0 \le v \le 1 \tag{5.63}$$

where C_K is the constant Gaussian curvature value. These equations can be rewritten as follows

$$C_K S^4(u,0) - A(u,0) = 0, \quad C_K S^4(u,1) - A(u,1) = 0 \quad 0 \le u \le 1 \tag{5.64}$$

$$C_K S^4(0,v) - A(0,v) = 0, \quad C_K S^4(1,v) - A(1,v) = 0 \quad 0 \le v \le 1 \tag{5.65}$$

Equations (5.64), (5.65) are univariate polynomials of degree $8m-4$ in u and $8n-4$ in v respectively.

Similarly for mean curvature

$$H(u,0) = \frac{B(u,0)}{2S^3(u,0)} = C_H, \quad H(u,1) = \frac{B(u,1)}{2S^3(u,1)} = C_H \quad 0 \le u \le 1 \tag{5.66}$$

$$H(0,v) = \frac{B(0,v)}{2S^3(0,v)} = C_H, \quad H(1,v) = \frac{B(1,v)}{2S^3(1,v)} = C_H \quad 0 \le v \le 1 \tag{5.67}$$

where C_H is the constant mean curvature value. These equations can be rewritten as follows

$$B(u,0) - 2C_H\sqrt{S^2(u,0)}S^2(u,0) = 0, \quad B(u,1) - 2C_H\sqrt{S^2(u,1)}S^2(u,1) = 0 \quad 0 \le u \le 1 \tag{5.68}$$

$$B(0,v) - 2C_H\sqrt{S^2(0,v)}S^2(0,v) = 0, \quad B(1,v) - 2C_H\sqrt{S^2(1,v)}S^2(1,v) = 0 \quad 0 \le v \le 1 \tag{5.69}$$

Equations (5.68), (5.69) are the univariate irrational functions involving polynomials and square roots of polynomials which come from the normalization of the normal vector of the surface, see

equation (5.7). $B(u,v)$ is a polynomial of degree $(5m-3, 5n-3)$ and $S^2(u,v)$ is a polynomial of degree $(4m-2, 4n-2)$.

Finally for the principal curvatures

$$\kappa(u,0) = \frac{B(u,0) \pm \sqrt{f_3(u,0)}}{2S^3(u,0)} = C_\kappa, \quad \kappa(u,1) = \frac{B(u,1) \pm \sqrt{f_3(u,1)}}{2S^3(u,1)} = C_\kappa, \quad 0 \le u \le 1 \tag{5.70}$$

$$\kappa(0,v) = \frac{B(0,v) \pm \sqrt{f_3(0,v)}}{2S^3(0,v)} = C_\kappa, \quad \kappa(1,v) = \frac{B(1,v) \pm \sqrt{f_3(1,v)}}{2S^3(1,v)} = C_\kappa, \quad 0 \le v \le 1 \tag{5.71}$$

where C_κ is the constant value of principal curvature and $f_3(u,v)$ is a polynomial function defined in equation (5.49). Equations (5.70) and (5.71) can be rewritten as follows

$$\begin{aligned} B(u,0) &\pm \sqrt{f_3(u,0)} - 2C_\kappa S^2(u,0)\sqrt{S^2(u,0)} = 0 \quad 0 \le u \le 1 \\ B(u,1) &\pm \sqrt{f_3(u,1)} - 2C_\kappa S^2(u,1)\sqrt{S^2(u,1)} = 0 \quad 0 \le u \le 1 \end{aligned} \tag{5.72}$$

$$\begin{aligned} B(0,v) &\pm \sqrt{f_3(0,v)} - 2C_\kappa S^2(0,v)\sqrt{S^2(0,v)} = 0 \quad 0 \le v \le 1 \\ B(1,v) &\pm \sqrt{f_3(1,v)} - 2C_\kappa S^2(1,v)\sqrt{S^2(1,v)} = 0 \quad 0 \le v \le 1 \end{aligned} \tag{5.73}$$

Equations (5.72), (5.73) are the univariate irrational functions involving polynomials and two square roots of polynomials which come from the analytical expression of the principal curvature and normalization of the normal vector of the surface.

The starting points for contour lines of curvature occur in pairs, since non-loop contour lines must start from a domain boundary and must end at a domain boundary point.

Ridges are surface curves where the surface tangent plane is undefined, i.e. $\mathbf{r}_u \times \mathbf{r}_v = \mathbf{0}$. Ridges of an offset surface correspond to points on the progenitor where one of the principal curvatures is equal to $-\frac{1}{d}$ (d is the offset distance) [23]. Therefore we can use equations (5.72), (5.73) to compute the starting points for tracing ridges on the offset, if the surface is subdivided along the isoparametric line which contains the local maxima and minima of principal curvatures such that each sub-patch will not contain loops nor singularities of ridges in its interior.

5.4.2 Mathematical Formulation of Contouring

Contour lines for constant curvature satisfy the following equation

$$C(u,v) = constant \tag{5.74}$$

where $C(u,v)$ is a curvature at the given point (u,v). We now consider a space curve which lies on the surface represented by the parametric form $\mathbf{r}(t) = \mathbf{r}[u(t), v(t)]$. Differentiating the equation (5.74) with respect to t yields

$$C_u \dot{u} + C_v \dot{v} = 0 \tag{5.75}$$

where $\dot{u}$, $\dot{v}$ are the first derivatives with respect to t. $(\dot{u}, \dot{v})$ gives the direction of the contour line in parameter space. The solutions to the equation (5.75) are

$$\dot{u} = \zeta C_v, \quad \dot{v} = -\zeta C_u \tag{5.76}$$

where ζ is an arbitrary non zero factor that can be chosen to provide arc-length parametrization as follows

$$\zeta = \pm \frac{1}{\sqrt{C_u^2 + C_v^2}} \tag{5.77}$$

C_u and C_v are given in equations (5.35), (5.43) and (5.56) for Gaussian, mean and principal curvatures respectively.

Contour lines of Gaussian curvature for K=0 separates the regions into elliptic (concave and convex) and hyperbolic (saddle) regions [67]. This information is useful for $3D$ and $5D$ machining as explained in section 2.2.3. Also the union of contour lines of maximum principal curvature for κ_{max}=0 and minimum principal curvature for κ_{min}=0 separate the region in a manner similar to the contour lines $K = 0$.

We used the Trip Algorithm introduced by Preusser [86] to polygonize the area between contour lines. The points of the contour lines are computed successively by integrating the initial value problem for a system of coupled nonlinear differential equations (5.76) using the variable stepsize and variable order Adams method [71]. Starting points were computed by the method described in section 5.4.1. Accuracy of the contour line depends on the number of points used to represent the contour line by straight line segments. Note that for principal curvatures, C_u and C_v become singular at umbilical point, therefore we avoid the contour level which is equivalent to the curvature value at the umbilics.

5.5 Towards Fully Automated 3D and 5D Machining and Fairing

All the governing equations derived in sections 5.3 and 5.4 are systems of polynomial equations or systems of irrational equations involving polynomials and square roots of polynomials. These equations can be solved by Bernstein subdivision method coupled with rounded interval arithmetic together with auxiliary variable method to handle the square roots involved.

5.5.1 Curvature Maps

To illustrate continuous surface decomposition, we used a saddle-like and a wave-like bicubic integral Bézier patches (see Figures 5-2 and 5-3). To display the curvature of the subdivided surface clearly, we assigned discrete color to each closed region based on curvature level. The level was determined by taking the average value of the curvature values of the contour lines excluding the boundary lines which form the closed region. We assigned R (red), G $(green)$ and B $(blue)$ to the minimum, zero and maximum curvature values of the whole domain. The color of the curvature values in between is linearly interpolated.

- **A Saddle-like Surface**
  The control points are given as follows

$$\begin{pmatrix} \mathbf{P}_{00}\ \mathbf{P}_{10}\ \mathbf{P}_{20}\ \mathbf{P}_{30} \\ \mathbf{P}_{01}\ \mathbf{P}_{11}\ \mathbf{P}_{21}\ \mathbf{P}_{31} \\ \mathbf{P}_{02}\ \mathbf{P}_{12}\ \mathbf{P}_{22}\ \mathbf{P}_{32} \\ \mathbf{P}_{03}\ \mathbf{P}_{13}\ \mathbf{P}_{23}\ \mathbf{P}_{33} \end{pmatrix} = \begin{pmatrix} (0,\ 0,\ 0.25) & (\frac{1}{3},\ 0,\ 0.1) & (\frac{2}{3},\ 0,\ -0.1) & (1,\ 0,\ -0.25) \\ (0,\ \frac{1}{3},\ 0.1) & (\frac{1}{3},\ \frac{1}{3},\ 0.05) & (\frac{2}{3},\ \frac{1}{3},-0.05) & (1,\ \frac{1}{3},\ -0.1) \\ (0,\ \frac{2}{3},\ -0.1) & (\frac{1}{3},\ \frac{2}{3},\ -0.05) & (\frac{2}{3},\ \frac{2}{3},0.05) & (1,\ \frac{2}{3},\ 0.1) \\ (0,\ 1,\ -0.25) & (\frac{1}{3},\ 1,\ -0.1) & (\frac{2}{3},\ 1,\ 0.1) & (1,\ 1,\ 0.25) \end{pmatrix}$$

  Therefore the surface is symmetric with respect to both diagonals. Since every point on the saddle-like surface is a hyperbolic point, there is no umbilic on the surface.

  - Gaussian Curvature

    Figure 5-4 shows a color map of the Gaussian curvature K. Since the shape of the surface is hyperbolic, the Gaussian curvature is negative everywhere, and has a global minimum value of K=-1.265 at (0.5, 0.5). The mid points of the domain boundaries have a local minimum of K reaching a value of K=-0.498. The four corners of the domain have the global maximum value for K=-0.365. Therefore the range of the Gaussian curvature is -1.265 $\leq K \leq$ -0.365. Since there is only one local extremum in the domain, we subdivide into two subdomains at isoparametric line u=0.5.

- Mean Curvature

  Figure 5-5 shows a color map of the mean curvature H. Because of symmetry at point (0.5, 0.5), we have a saddle point of H with value H=0.0. There are no stationary points along the domain boundaries. We have the global maximum 0.326 at two corners (0, 0), (1, 1) and the global minimum H=-0.326 at the other two corners. Therefore the mean curvature varies from -0.326 to 0.326.

- Maximum Principal Curvature

  Figure 5-6 shows a color coded map of the maximum principal curvature κ_{max}. Since the Gaussian curvature is negative everywhere, the maximum principal curvature κ_{max} is positive everywhere. There are three stationary points of κ_{max} within the domain. A global maximum at (0.5, 0.5) of the value κ_{max}=1.125 and two saddle points at (0.041, 0.041) and (0.959, 0.959) with value κ_{max}=1.010. There are no stationary points along the boundaries. We have the global minimum at the corners (1, 0) and (0, 1) of the value κ_{max}=0.360. Therefore the range of the maximum principal curvature is 0.360 to 1.125. Since there is only one local extremum in the domain, we subdivide into two subdomains at isoparametric line u=0.5.

- Minimum Principal Curvature

  Figure 5-7 shows a color map of the minimum principal curvature κ_{min}. As expected, the minimum principal curvature is negative everywhere. Similarly to the maximum principal curvature, there are three stationary points inside the domain. A global minimum at (0.5, 0.5) with value κ_{min}=-1.125 and two saddle points at (0.041, 0.959), (0.959, 0.041) with value κ_{min} =-1.010. There are no stationary points along the boundaries. The global maximum κ_{min} is located at (0, 0) and (1, 1) having the value κ_{min}=-0.360. Therefore the range of the minimum principal curvature is -1.125 to -0.360. Since there is only one local extremum in the domain, we subdivide into two subdomains at isoparametric line u=0.5.

- **A Wave-like surface** The boundary 12 control points are coplanar so that the boundary curves form a square. The remaining four interior control points are not on the same plane. The control points are given as follows

$$\begin{pmatrix} \mathbf{P}_{00} & \mathbf{P}_{10} & \mathbf{P}_{20} & \mathbf{P}_{30} \\ \mathbf{P}_{01} & \mathbf{P}_{11} & \mathbf{P}_{21} & \mathbf{P}_{31} \\ \mathbf{P}_{02} & \mathbf{P}_{12} & \mathbf{P}_{22} & \mathbf{P}_{32} \\ \mathbf{P}_{03} & \mathbf{P}_{13} & \mathbf{P}_{23} & \mathbf{P}_{33} \end{pmatrix} = \begin{pmatrix} (0,\ 0,\ 0) & (\frac{1}{3},\ 0,\ 0) & (\frac{2}{3},\ 0,\ 0) & (1,\ 0,\ 0) \\ (0,\ \frac{1}{3},\ 0) & (\frac{1}{3},\ \frac{1}{3},\ 1) & (\frac{2}{3},\ \frac{1}{3},-1) & (1,\ \frac{1}{3},\ 0) \\ (0,\ \frac{2}{3},\ 0) & (\frac{1}{3},\ \frac{2}{3},\ \frac{3}{5}) & (\frac{2}{3},\ \frac{2}{3},-\frac{3}{5}) & (1,\ \frac{2}{3},\ 0) \\ (0,\ 1,\ 0) & (\frac{1}{3},\ 1,\ 0) & (\frac{2}{3},\ 1,\ 0) & (1,\ 1,\ 0) \end{pmatrix}$$

Therefore the surface is anti-symmetric with respect to $u = 0.5$. Although the wireframe of the surface looks simple, the surface is rich in its variety of differential geometry properties. The wave-like surface has four umbilics at $(0.211, 0.052)$, $(0.211, 0.984)$, $(0.789, 0.052)$, $(0.789, 0.984)$ with principal curvature values 1.197, 0.267, -1.197, -0.267, and one flat umbilical point at $(0.5, 0.440)$. None of them are local extrema according to the criterion described in section 7.5.

- Gaussian Curvature

  Figure 5-8 shows a color map of the Gaussian curvature K. Since the surface is anti-symmetric with respect to $u = 0.5$, the Gaussian curvature which is the product of maximum and minimum principal curvatures is symmetric with respect to $u = 0.5$. The range of the curvature is $-81 \leq K \leq 10.297$. The global maximum Gaussian curvature $K = 10.297$ occurs at two stationary points within the domain $(0.195, 0.374)$, $(0.805, 0.374)$. The global minimum curvature $K = -81$ is located at two corners $(0, 0)$ and $(1, 0)$. There is also a saddle point inside the domain at $(0.5, 0.440)$, with value $K = 0$, which is a flat point of the surface. There are six local maxima and two local minima along the domain boundaries. Local maxima at $(0.211, 0)$, $(0.789, 0)$, $(0.211, 1)$, $(0.789, 1)$, $(0, 0.440)$, $(1, 0.440)$ with all values $K = 0$, and local minima at $(0.5, 0)$ with $K = -20.25$ and at $(0.5, 1)$ with $K = -7.29$. Since the two local maxima inside the domain have the same v coordinate, we subdivide the surface into two sub-domains along the isoparametric line $v = 0.374$. In this picture, we avoid the curvature level $K = 0$ so as not to deal with the self-intersecting contour at the saddle point.

- Mean Curvature

  Figure 5-9 shows a color map of the mean curvature H. Because of anti-symmetry with respect to $u = 0.5$, mean curvature has $H = 0$ contour line at $u = 0.5$. Mean curvature varies from -4.056 to 4.056. Both global maximum and minimum curvature are located inside the domain at $(0.190, 0.414)$ and $(0.810, 0.414)$ respectively. There are seven local maxima and seven local minima along the boundary. The local maxima are located at $(0.116, 0)$, $(0.319, 0)$, $(0.789, 0)$, $(0.211, 1)$, $(0, 0.089)$, $(0, 0.861)$, $(1, 0.440)$ with $H = 0.539$, 0.539, -0.524, 0.121, 1.155, 1.155, -0.607. The local minima are located at $(0.211, 0)$, $(0.681, 0)$, $(0.884, 0)$, $(0.789, 1)$, $(0, 0.440)$, $(1, 0.089)$, $(1, 0.861)$ with $H = 0.524$, -0.539, -0.539, -0.121, 0.607, -1.155, -1.155. Since the local maximum and minimum inside the domain are on the same isoparametric line $v = 0.414$, we subdivide

into two sub-domains at this line.

– Maximum Principal Curvature

Figure 5-10 shows a color map of the maximum principal curvature κ_{max}, which has a range of $-1.665 \leq \kappa_{max} \leq 9$. The global maximum is located at two corners $(0,0)$ and $(1,0)$, and global minimum is located at $(0.789, 0.303)$ which is a stationary point inside the domain. There is also a local maximum inside the domain at $(0.187, 0.440)$ with $\kappa_{max} = 6.607$ and four saddle points inside the domain at $(0.082, 0.802)$, $(0.114, 0.184)$, $(0.321, 0.157)$ and $(0.378, 0.851)$ with values $\kappa_{max} = 4.504$, 5.127, 3.276 and 2.470. There are two local maxima and six local minima along the boundary. The locations are $(0.478, 0)$, $(0.491, 1)$ with $\kappa_{max} = 4.569$ and $\kappa_{max} = 2.704$ for the local maxima and $(0.211, 0)$, $(0.789, 0)$, $(0.211, 1)$, $(0.789, 1)$, $(0, 0.440)$, $(1, 0.440)$ with $\kappa_{max} = 1.047$, 0, 0.242, 0, 1.213, 0 for the local minima. We subdivide the domain along the isoparametric line $u = 0.187$ and $u = 0.789$ which contain the local maximum and minimum. The reason we choose u isoparametric line is that the minimum size of each domain in the u direction is larger than in the v direction.

– Minimum Principal Curvature

Figure 5-11 shows a color map of the minimum principal curvature κ_{min} which has a range of -9 $\leq \kappa_{min} \leq 1.665$. The global maximum is located inside the domain at $(0.211, 0.303)$ with $\kappa_{min} = 1.665$. The global minimum is located at two corners (0,0), (1,0) with $\kappa_{min} = -9$. There are other stationary points within the domain, a local minimum at (0.813, 0.440) with the value κ_{min} =-6.607 and four saddle points inside the domain at (0.622, 0.851), (0.679, 0.157), (0.886, 0.184) and (0.918, 0.802) with values κ_{min} = -2.470, -3.276, -5.127 and -4.504. There are six local maxima and two local minima along the domain boundaries. The maxima are located at (0.211,0), (0.789,0), (0.211,1), (0.789,1), (0, 0.440), (1, 0.440) with $\kappa_{min} = 0, -1.047, 0, -0.242, 0, -1.213$. The local minima are located at (0.478,0), $(0.491, 1)$ with the value $\kappa_{min} = -4.299, -2.688$. For the same reason as in maximum principal curvature, the domain is subdivided at $u = 0.211$ and $u = 0.813$.

Formulation Time		
Arithmetic	*CPU Time in Seconds*	
	Auxiliary Variable Method	*Squaring Method*
FPA	10	121
RA	20457	∞
RIA	521	9935

Table 5.1: CPU Time Comparison for the Formulation between Auxiliary Variable Method and Squaring Method (See Table 3.4 for symbolic entries)

5.5.2 CPU Time Comparison between Auxiliary Variable Method and Squaring Method

The CPU time comparison for the *formulation* of the governing simultaneous polynomial equations for finding the stationary points of the maximum principal curvature inside the domain (see, section 5.3.3) between the auxiliary variable method and the squaring method (see equation (3.18)) are listed in Table 5.1 with double-precision floating point arithmetic (FPA), rational arithmetic (RA) and rounded interval arithmetic (RIA). The CPU time in seconds was measured on a graphics workstation running at 36MHz. These times are indicative from a non-optimized implementation. In this computation, we used a similar wave-like bicubic integral Bézier patch to that in section 5.5.1, but with the z-value scaled to 1/10 in order to better illustrate the effect of FPA in missing roots. For a bicubic patch, the degree of the three governing equations for finding stationary points of maximum principal curvature inside the domain by auxiliary variable method are $(33, 34, 1)$, $(34, 33, 1)$ and $(24, 24, 2)$ in (u, v, σ), while by squaring method the degree of the two governing equations are $(66, 68)$ and $(68, 66)$ in (u, v).

We can observe from the table that the auxiliary variable method is at least one order of magnitude faster than the squaring method for all three arithmetics. RIA is approximately 50 times slower than FPA for auxiliary variable method, while it is 80 times slower for the squaring method. With the auxiliary variable method RA is still *feasible*, even though it is 40 times slower than RIA and 2000 times slower than FPA, while with the squaring method it is not appropriate to compute in RA with this type of computer. The reason why the rational arithmetic is so slow for the higher degree formulation is that the pair of integers to represent a rational number precisely grows quite large in digits.

Next we compare the *solution* time for solving simultaneous polynomial equations formulated by

the auxiliary variable method and the squaring method. There are seven roots if it is formulated by the auxiliary variable method. The roots include two stationary points which are saddle points at $(0.319, 0.612)$ and $(0.427, 0.905)$ with values $\kappa_{max} = 0.371,\ 0.264$, four umblilics at $(0.211, 0.081)$, $(0.211, 0.9798)$, $(0.789, 0.081)$ and $(0.789, 0.979)$ with values $0.224,\ 0.039,\ -0.224,\ -0.039$ and one flat umbilical point at $(0.5, 0.440)$. Using the criterion in section 7.5, it is easily found that two of the umbilics $(0.211, 0.081)$ and $(0.789, 0.081)$ are local minima according to the criterion in section 7.5. Actually the latter one is the global minimum. On the other hand if the governing equations are formulated by the squaring method, there exist thirteen roots. The six additional roots come from three pairs of two simultaneous bivariate irrational equations, which are given by

$$\begin{aligned}
&f_1(u,v) - f_2(u,v)\sqrt{f_3(u,v)} = 0, \quad g_1(u,v) - g_2(u,v)\sqrt{f_3(u,v)} = 0, \quad 0 \le u, v \le 1 \\
&f_1(u,v) - f_2(u,v)\sqrt{f_3(u,v)} = 0, \quad g_1(u,v) + g_2(u,v)\sqrt{f_3(u,v)} = 0, \quad 0 \le u, v \le 1 \\
&f_1(u,v) + f_2(u,v)\sqrt{f_3(u,v)} = 0, \quad g_1(u,v) - g_2(u,v)\sqrt{f_3(u,v)} = 0, \quad 0 \le u, v \le 1
\end{aligned} \tag{5.78}$$

The first two simultaneous equations correspond to the governing equations for computing stationary points of minimum principal curvature inside the domain.

The results are listed in Table 5.2 with several tolerances. *formulation/solution* refers to the arithmetic that has been used for formulation and the arithmetic that has been used for solution. RIA/RIA is the only realistic way to find all the roots with low accuracy by the squaring method. The low accuracy is due to the error contamination of the coefficients of the governing equations. On the other hand, the auxiliary variable method finds all the roots for all the arithmetic combinations in the table for tolerance of 10^{-8}. With FPA/FPA some of the roots are missed for the tolerance of 10^{-9}, while RA/FPA finds all the roots in this example. This result stresses the importance of formulating the governing equations in a pristine way. Although RA/FPA finds all the roots with tolerance of 10^{-9}, there is no guarantee that this combination will work for other cases when the degree is high.

Consequently, we can conclude that the auxiliary variable method is the better way to handle multidimensional nonlinear problems involving polynomials and square roots of polynomials and RA/RIA is the most robust and accurate combination for problem formulation/solution.

A new accurate and robust method to decompose a Bézier surface patch into subpatches with specific ranges of curvature, including Gaussian, mean, maximum principal minimum principal, was introduced above. This method can be applied to automatic machine tool size selection and tool

Solution Time (formulation time is not included)								
Arithmetic formulation/solution	*CPU Time in Seconds*							
	Auxiliary Variable Method			*Squaring Method*				
	10^{-8}	10^{-9}	10^{-10}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}
FPA/FPA	792	m	m	m	m	m	m	m
RIA/RIA	46548	c	c	29924	39690	44290	76220	c
RA/FPA	817	894	m	n/a	n/a	n/a	n/a	n/a
RA/RIA	35543	49378	c	n/a	n/a	n/a	n/a	n/a

Table 5.2: CPU time comparison for the solution between auxiliary variable method and squaring method (See Table 3.4 for symbolic entries)

path generation for $3D$ and $5D$ NC machining, and surface interrogation for fairing.

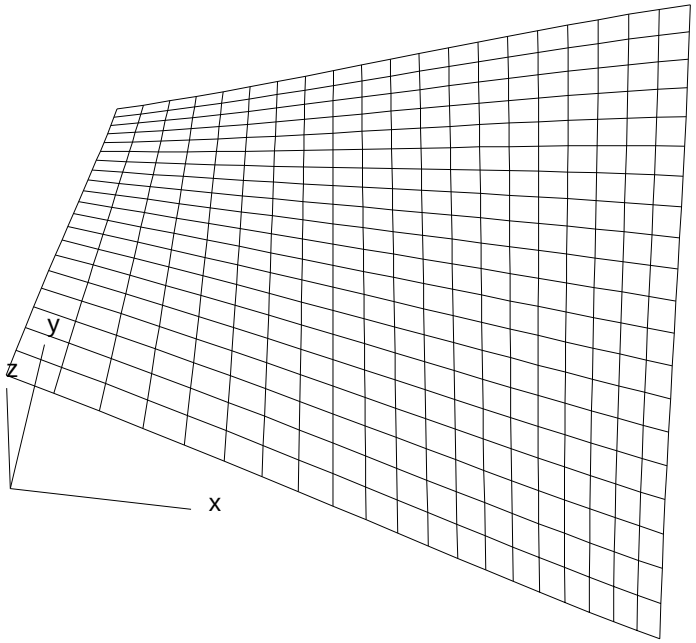

Figure 5-2: Saddle-like integral Bézier surface patch

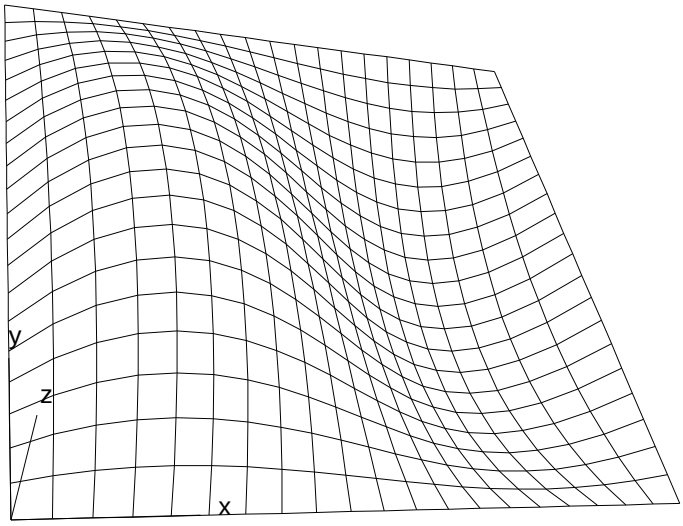

Figure 5-3: Wave-like integral Bézier surface patch

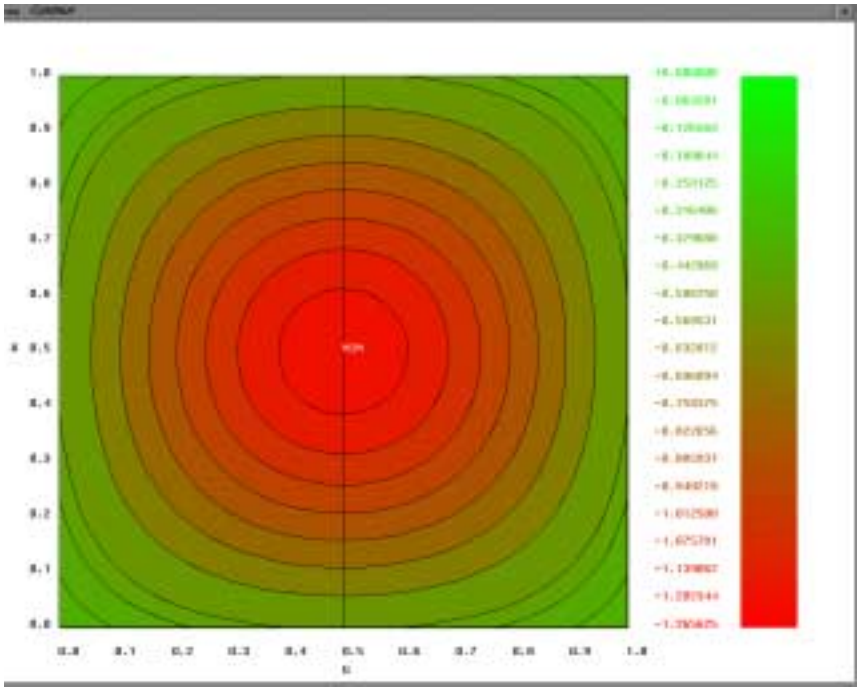

Figure 5-4: Gaussian curvature color map of saddle-like surface

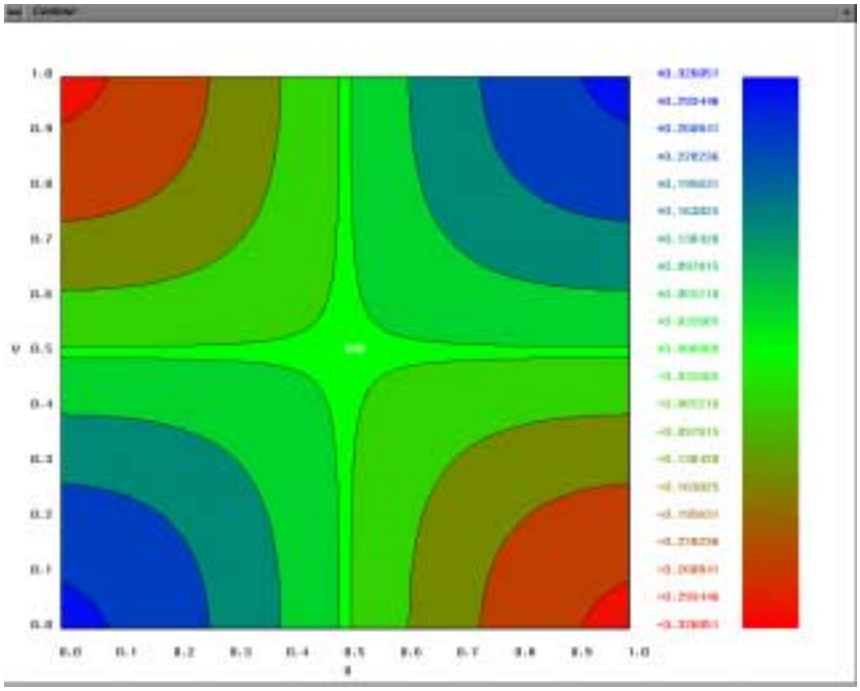

Figure 5-5: Mean curvature color map of saddle-like surface

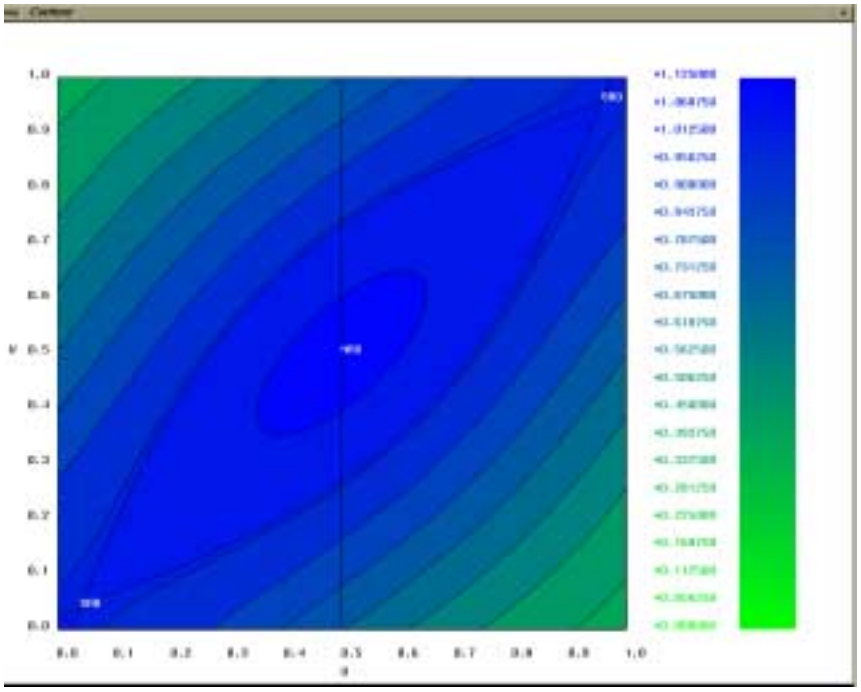

Figure 5-6: Maximun principal curvature color map of saddle-like surface

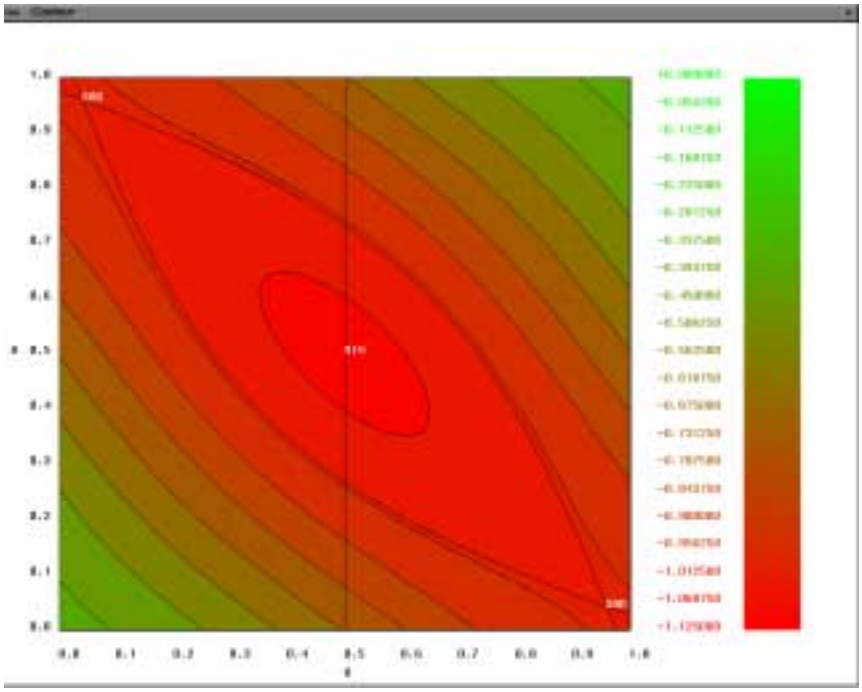

Figure 5-7: Minimum principal curvature color map of saddle-like surface

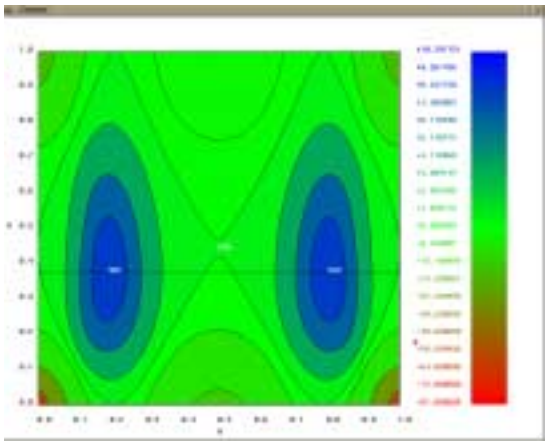

Figure 5-8: Gaussian curvature color map of wave-like surface

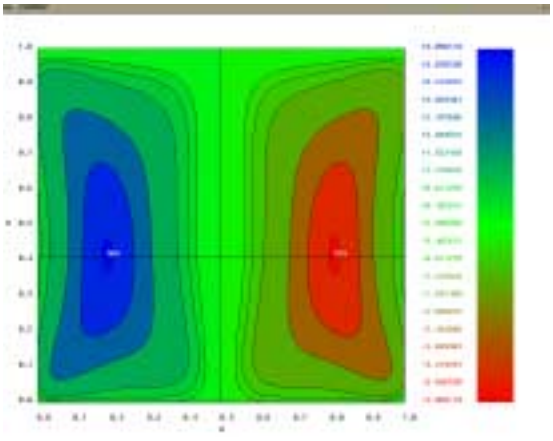

Figure 5-9: Mean curvature color map of wave-like surface

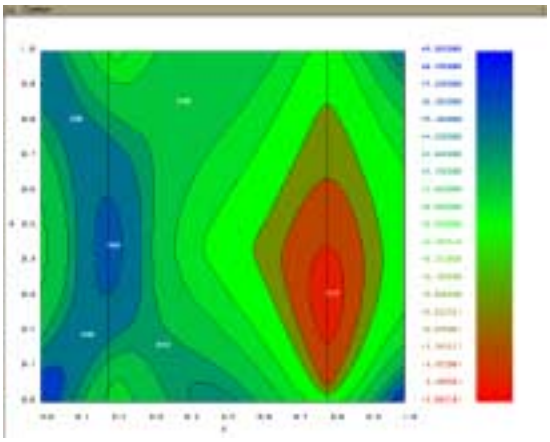

Figure 5-10: Maximum principal curvature color map of wave-like surface

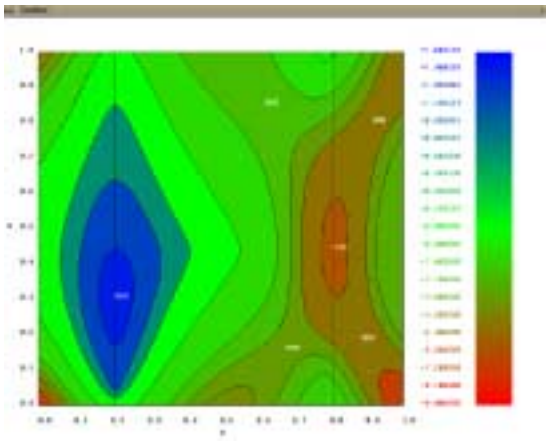

Figure 5-11: Minimum principal curvature color map of wave-like surface

Chapter 6

Surface Tessellation

6.1 Introduction and Literature Review

Up to now most of the algorithms for automatic mesh generation for finite element method have concentrated on 2D (planar) and 3D (solid) meshing [37], and very few algorithms for parametric surface meshing have been published. Gursoy [31] triangulates trimmed parametric surface patches based on the Medial Axis Transform (MAT). First a coarse mesh is created in the parameter plane of the trimmed patch based on MAT and mapped into 3D space. Mesh refinement is conducted if the distance from the centroid of an individual triangle to the corresponding point on the trimmed surface is large. Very recently Shimada and Gossard [94] have proposed an interesting computational method for a physically based triangular mesh generator. This computational method, which involves solving systems of ordinary differential equations, can be viewed as using equilibrium of floating soap bubbles placed on the surface computed via dynamical simulation. After all the bubble locations are decided, their center points are connected by Delaunay triangulation to provide a complete mesh topology. In computer graphics, Rockwood [87] developed an algorithm of real-time rendering of trimmed rational tensor product surfaces. The surface is uniformly tessellated into grid of rectangles connected by triangles to points evaluated along the boundary curves using so called coving and tiling technique. This algorithm does not reflect the differential geometry of the surface but discretizes the surface in real time. Our method is based on the curvature contouring technique which is introduced in Chapter 5. This Chapter is structured as follows. Section 6.2 starts with defining a new curvature measure, which is used for contouring. Section 6.3 presents a procedure for placing

nodes on contouring lines and on the boundaries. Finally section 6.4 illustrates two examples of root mean square curvature based Bézier surface meshing.

6.2 Root Mean Square Curvature

We want to assign small and numerous mesh elements to the region where the curvature is high, and large and few mesh elements to the region where the curvature is small. There are four different measures of surface curvature, maximum principal κ_{max}, minimum principal κ_{min}, Gaussian K and mean H curvatures, which have been discussed in Section 5.2. In addition to these curvatures, *absolute curvature* [22] is defined as

$$\kappa_{abs} = |\kappa_{max}| + |\kappa_{min}|$$

The absolute curvature may be among the most appropriate curvatures for meshing purposes. However, by definition absolute curvature is not a differentiable function, which makes its application difficult in practice. As will be explained in the next section, our meshing procedure requires a curvature function which is differentiable. Hence we define a new curvature measure, *root mean square curvature*:

$$\kappa_{rms} = \sqrt{\kappa_{max}^2 + \kappa_{min}^2} \tag{6.1}$$

It is apparent that the absolute curvature is always greater than the root mean square curvature. Note that all the derivatives of κ_{rms} have singularities at flat umbilical points. Using the similar methodology introduced in the Chapter 5, we can develop a meshing scheme based on root mean square curvature. The proposed finite element mesh generation scheme automatically discretizes a Bézier patch into a set of triangular meshes. Major steps of this scheme are presented in section 6.3.

6.3 Node Placement Procedure

6.3.1 Input

Our meshing scheme requires two sets of input, surface geometry and mesh density. Surface geometry for Bézier surface patches is determined by the degrees in the parameters and a set of control points.

Mesh density is determined by the number of contouring levels, contouring level density function and the bounds for the mesh size. The contouring level density function determines the spacing between the contouring lines, and the bounds for the mesh size determine the node spacing along the contour line.

6.3.2 Curvature Level

Range of the curvature is obtained by evaluating the global maximum κ_{rms}^{max} and minimum κ_{rms}^{min} of root mean square curvature from the four curvature values at the corners, local maxima and minima of the curvature inside the domain and local maxima and minima of the curvature along the four border lines as described in Chapter 5. Once we evaluate the the curvature range, each contouring level is determined by the number of contouring levels n_c and contouring level density function f_D. The curvature level density function is defined as follows:

$$f_D(\alpha) = c_1\alpha - c_2\alpha^2 \tag{6.2}$$

where $0 \leq \alpha \leq 1$ and c_1, c_2 are positive real numbers. The range of c_1 and c_2 are determined by the following constraints.

- $f_D(1) = 1$
- $0 \leq f_D'(1) < 1$

The first constraint says that the range of the density function is $0 \leq f_D(\alpha) \leq 1$ for $0 \leq \alpha \leq 1$. And the second constraint says that the slope of the function at $\alpha = 1$ is greater than zero and smaller than 45°. Consequently the ranges of c_1 and c_2 are

$$1 < c_1 \leq 2, \quad 0 < c_2 \leq 1, \quad c_1 - c_2 = 1$$

Curvature levels for contouring are determined by

$$\kappa_{rms} = \kappa_{rms}^{min} + (\kappa_{rms}^{max} - \kappa_{rms}^{min}) * f_D(\alpha) \tag{6.3}$$

where κ_{rms}^{min} and κ_{rms}^{max} are the global minimum and maximum, and $\alpha = \frac{i}{n_c+1}$ with $1 \leq i \leq n_c$.

6.3.3 Nodes on Contour Lines

Our algorithm is based on placing the nodes first and linking them afterwards. Nodes are placed on the curvature contour line and on the boundary. We employ the same procedure to contour constant κ_{rms} that we used in Chapter 5. Since the procedure is the same, we will not go into detail but rather just derive the first order derivatives of the κ_{rms}.

$$\kappa_{rms} = \sqrt{4H^2 - 2K} = \frac{\sqrt{B^2 - 2AS^2}}{S^3} \tag{6.4}$$

$$\frac{\partial \kappa_{rms}}{\partial u} = \frac{(BB_u - A_u S^2)S^2 + (4AS^2 - 3B^2)\mathbf{S} \cdot \mathbf{S}_u}{S^5\sqrt{B^2 - 2AS^2}} \tag{6.5}$$

$$\frac{\partial \kappa_{rms}}{\partial v} = \frac{(BB_v - A_v S^2)S^2 + (4AS^2 - 3B^2)\mathbf{S} \cdot \mathbf{S}_v}{S^5\sqrt{B^2 - 2AS^2}} \tag{6.6}$$

where K and H are defined by equations (5.19) and (5.20), and all the other symbols are defined in Appendix A. Note that all the equations are expressed so that the denominator and the numerator only include polynomials. The second derivatives which are necessary for classification of staionary points are listed in the Appendix A. First we define a spacing unit along the contour line which we denote $SIZE$. $SIZE$ is a function of the curvature level, and is determined similarly to the curvature level, see equation (6.3).

$$SIZE = SMALL + (LARGE - SMALL) * f_D(\alpha), \quad \alpha = \frac{\kappa_{rms}^{max} - \kappa_{rms}}{\kappa_{rms}^{max} - \kappa_{rms}^{min}} \tag{6.7}$$

where $SMALL$ and $LARGE$ are the constant bounds for the spacing given by the input. We integrate along each contour line and store the data (s, u, v) densely, where s is the arc length in the uv-plane from the initial points of the integration. The set of stored data points (s, u) and (s, v) are interpolated by the cubic-spline and evaluated at

$$s = \Delta s \times i, \quad 1 \leq i \leq nseg - 1 \tag{6.8}$$

where Δs is determined by the total length of the contour line s_l divided by the number of segments $nseg$. The number of segments $nseg$ is an integer defined as the integer part of the ratio $s_l/SIZE$. The points evaluated by equation (6.8) are placed as nodes and are linked later by the Delaunay triangulation to form the elements. Nodes are repeatedly placed for every contour line.

6.3.4 Nodes on Boundary

Since the boundary is in general not a contour line, placing nodes on the boundary is the most difficult task for the entire algorithm. The four corner points, all the starting points and the ending points of the integration are placed as nodes. There are mainly four special cases to avoid an ill-shaped mesh (see also Figure 6-1).

1. When the starting or ending points of the contour line are located within the distance of $SIZE/2$ from the corner points, the points are removed.

2. If the variation of the curvature along the boundary is small compared to the range of the global curvature, only a few nodes will be placed along the boundary. In such case, we do not integrate the contour line, instead nodes are placed along the boundary in the following way. The curvature value at two corner points, the local maxima and local minima are evaluated to find the range of the curvature along the boundary. If the local range of curvature along the boundary is small compared to the global range of curvature, the $SIZE$ is evaluated by equation (6.7), using the average curvature along the boundary. The nodes are placed along the boundary in the same manner as nodes on a contour.

3. If a local extremum exists along the boundary, the contour line whose curvature level is close to the extremum value forms a half loop. If the distance between the starting point and the ending point along the boundary d_{se} is larger than the $SIZE$, then $nseg$ is computed using $d_{se} = SIZE \times nseg$. Then $nseg - 1$ nodes are linearly interpolated along the boundary.

4. As a special case of 3, when the maximum distance between the contour line and the boundary edge is less than $SIZE$, the nodes on the contour line are removed.

6.3.5 Delaunay Triangulation

Once all the nodes are placed, Delaunay triangulation is computed to link the nodes to form the triangular mesh. Delaunay triangulation maximizes the smallest angles of the triangles. Let us consider a set of points $\mathbf{P}_1$, $\mathbf{P}_2, \cdots \mathbf{P}_n$ in the plane. We can construct N regions $V_1, V_2 \cdots V_n$ whose boundaries are the perpendicular bisectors of the lines joining the point $\mathbf{P}_i$ and $\mathbf{P}_j$ when V_i and V_j are next to each other. All regions V_i are open convex polygons called *Voronoi polygons* [85], see Figure 6-2. Each Voronoi polygon is associated with a single data point $\mathbf{P}_i$. The collection of Voronoi

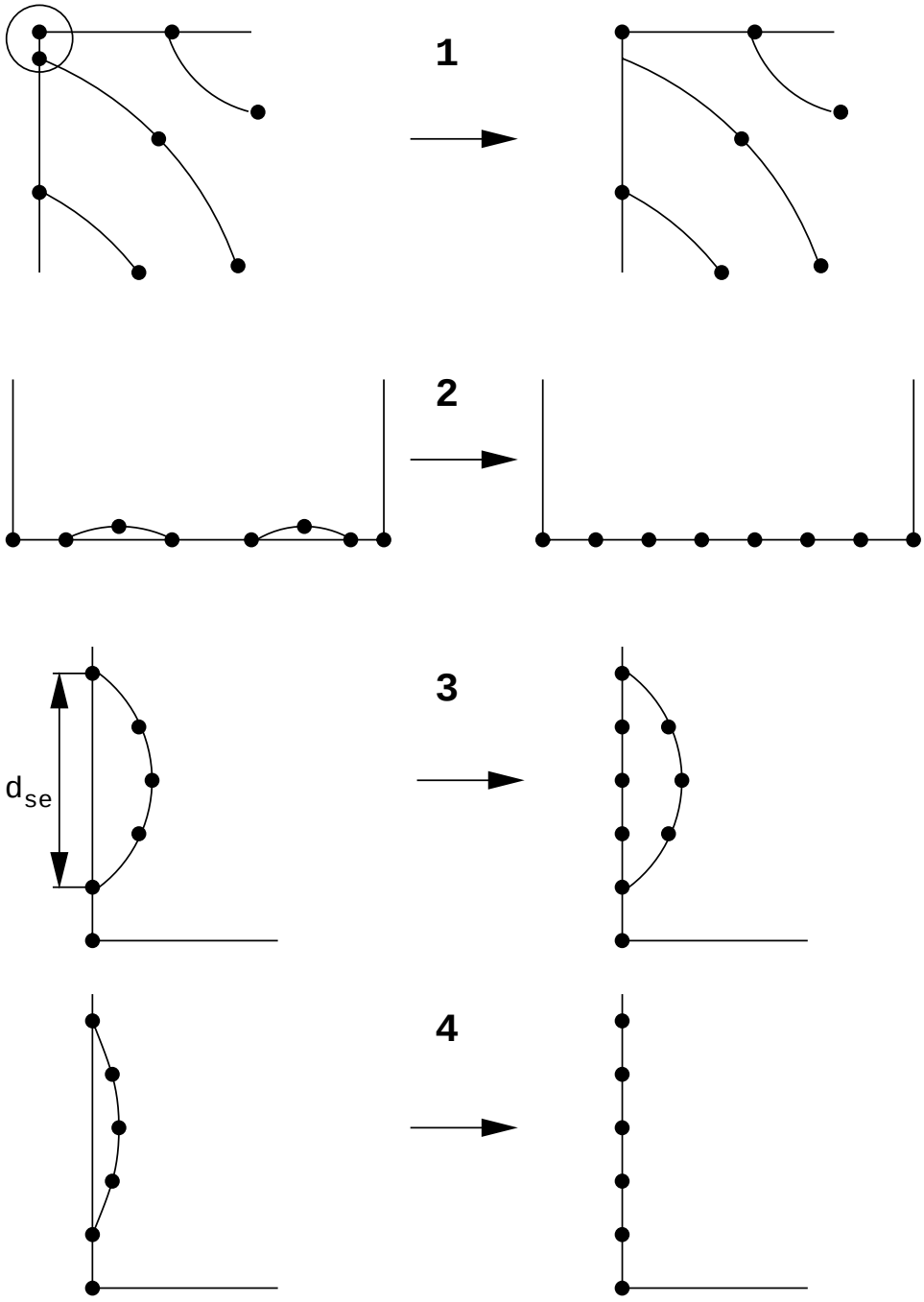

Figure 6-1: Special cases for nodes placement on boundary

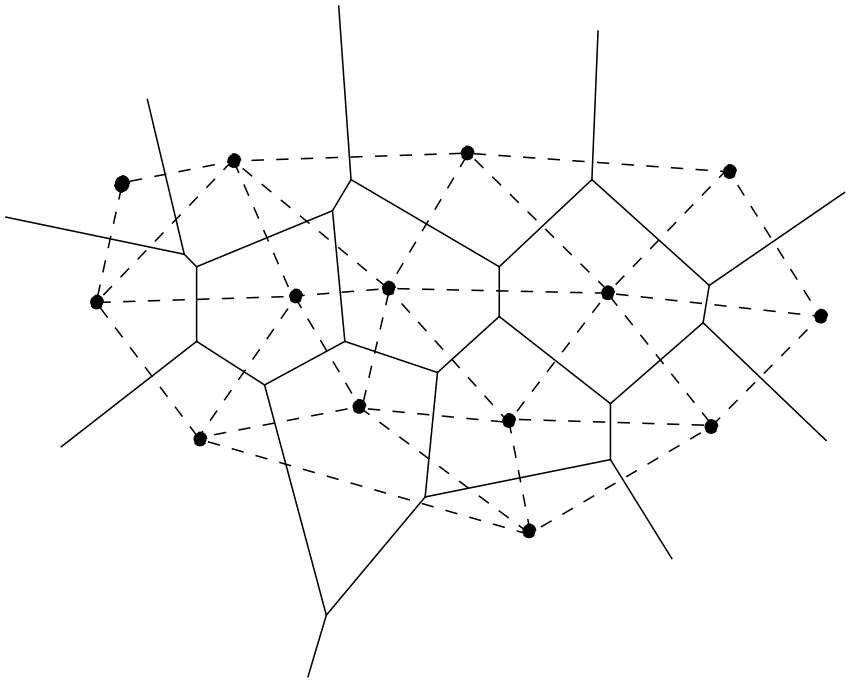

Figure 6-2: Voronoi polygons (solid) and Delaunay triangulations (dashed)

polygons is called *Dirichlet Tessellation*. By connecting the points associated with Voronoi polygons we can construct a set of triangles. These set of triangles are called the *Delaunay triangulation*.

6.4 Automated Triangular Mesh Generation

In this section, we give examples of meshing of two bicubic Bézier patches for illustration. Figures 6-3, 6-4 show a triangular mesh of the saddle-like bicubic integral Bézier patch which is also used in section 5.5. The input parameters are shown in Table 6.1. The maximum principal curvature κ_{max} is positive everywhere and has a global maximum at $(0.5, 0.5)$, while the minimum principal curvature κ_{min} is negative everywhere and has a global minimum at $(0.5, 0.5)$, hence κ_{rms} has a global maximum at $(0.5, 0.5)$ with value $\kappa_{rms} = 1.591$. Each boundary edge has two local maxima with $\kappa_{rms} = 1.076$ and one local minimum $\kappa_{rms} = 0.998$. All the local maxima are located by the distance of 0.038 from corner points, and all the local minima are located at the center of each boundary edge. These local minima are also global minima. Therefore κ_{rms} varies from 0.998 to 1.591.

Every point on the second surface is an elliptic point. The surface is also symmetric with respect

Surface	*Number of Contour Level* n_c	*Density Function*	*Bounds for Spacing*		*Number of Nodes*
			$SMALL$	$LARGE$	
Saddle-like Surface	8	$f_D = 2\alpha - \alpha^2$	0.04	0.09	224
Elliptic Surface	10	$f_D = 2\alpha - \alpha^2$	0.03	0.1	250

Table 6.1: Summary of data for triangulation examples

to $u = 0.5$ and $v = 0.5$. Figures 6-5, 6-6 show the triangular meshes of the elliptic surface. The input parameters are also shown in Table 6.1. Since the surface is elliptic and symmetric with respect to $u = 0.5$ and $v = 0.5$, κ_{rms} has a maximum at $(0.5, 0.5)$ with value $\kappa_{rms} = 2.372$. There are four local maxima and four local minima along the boundary. The local maxima are located at $(0.142, 0)$, $(0.858, 0)$, $(0.142, 1)$ and $(0.858, 1)$ with values $\kappa_{rms} = 1.590$. The local minima are located at mid points of all the four boundary edges with $\kappa_{rms} = 0.963$ Therefore κ_{rms} varies from 0.963 to 2.372.

We have described an algorithm for generating triangular meshes automatically for free-form surfaces based on root mean square curvature. The algorithm produces well-shaped elements and provides small and dense meshes in the region where the curvature is large, and large and sparse meshes in the region where the curvature is small without any mesh relaxation and smoothing as seen in the figures.

The algorithm requires surface geometry and density control parameters n_c, $SMALL$, $LARGE$, c_1 and c_2 as input. These parameters may be adjusted through a few iterations.

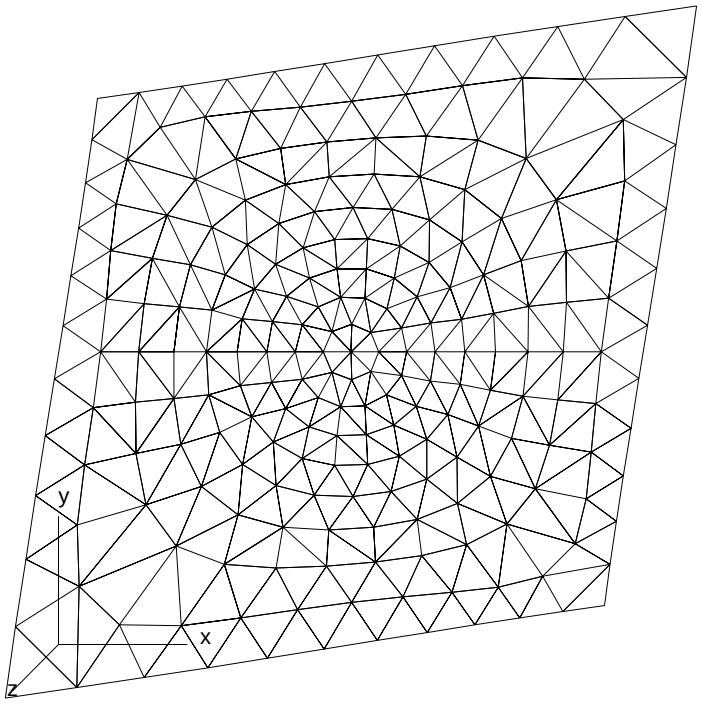

Figure 6-3: Meshing of saddle-like integral Bézier surface patch (Top view)

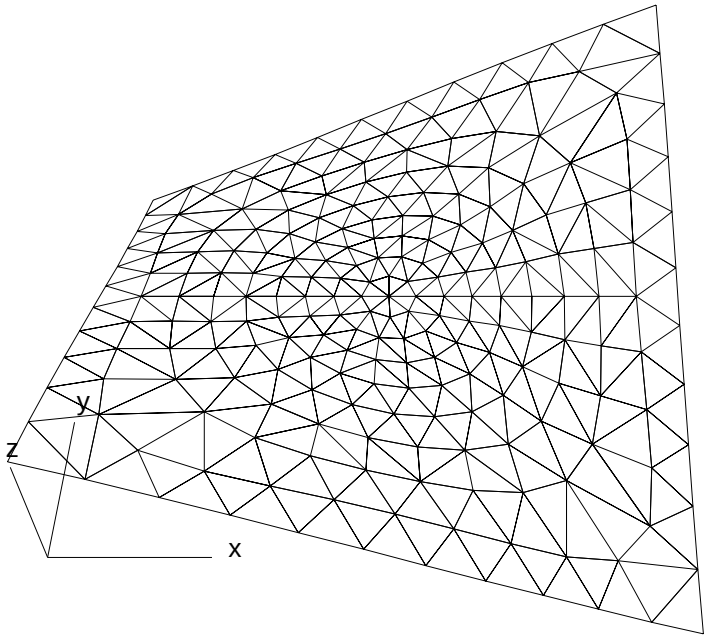

Figure 6-4: Meshing of saddle-like integral Bézier surface patch (Side view)

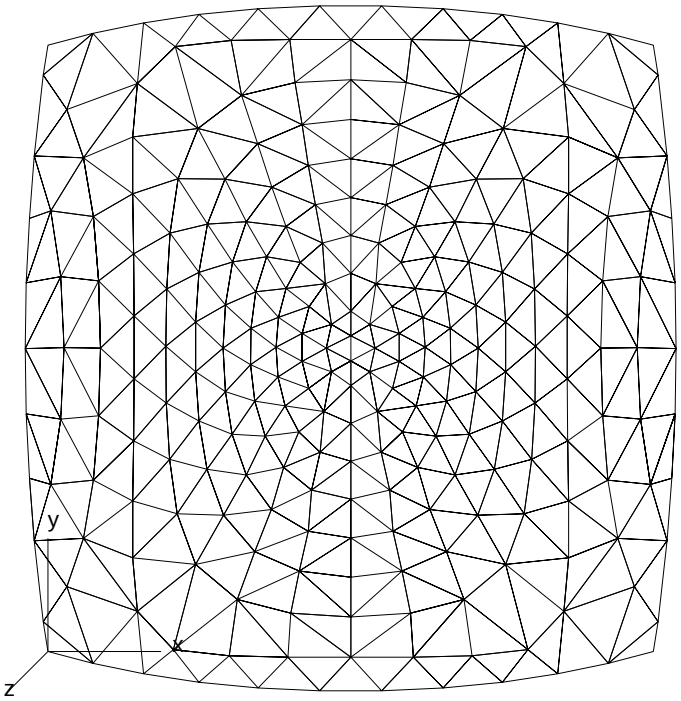

Figure 6-5: Meshing of elliptic integral Bézier surface patch (Top view)

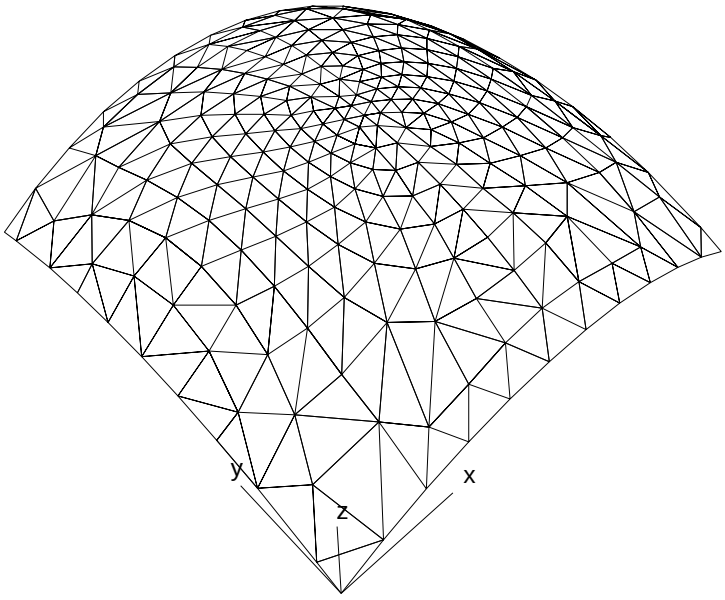

Figure 6-6: Meshing of elliptic integral Bézier surface patch (Side view)

Chapter 7

Surface Recognition

7.1 Introduction and Literature Review

An umbilic is a point on a surface where all *normal curvatures* are equal, thus the *principal directions* are indeterminate. Therefore the orthogonal net of *lines of curvature* become singular at an umbilic. An obvious example of a surface consisting entirely of umbilical points is the sphere. Actually, parts of spheres and planes are the only surfaces all of whose points are umbilics. The number of umbilics on a surface is often finite and they are isolated [36], [98]. At an umbilic, the *directions of principal curvature* can no longer be evaluated by second order derivatives and we need the aid of higher order derivatives to compute the lines of curvature near the umbilic. Monge $(1746-1818)$ who, with Gauss can be considered the founder of differential geometry of curves and surfaces, first computed the lines of curvature of the ellipsoid (1796) which has four umbilical points. He envisaged the net of lines of curvature as bounding tiles on the semi-ellipsoidal roof of a Napoleonic High Court chamber, with chandeliers hanging from the umbilics [81].

There exists an analogy between normal curvature and stress in elasticity theory [42]. For 2D problems for example, it is well known that whatever the state of stress at a point, there will always be two orthogonal directions through the point in each of which the shear stress is zero. These two directions are called the axes of principal stress. The curve which lies along one of the axes of principal stress at all its points is called line of principal stress. Such lines form an orthogonal net. The point where two principal stresses are equal is called isotropic point. The state of stress at such point is that of a radial compression or tension, uniform in all directions.

There are a number of papers which deal with lines of curvature. Martin [63] introduced so called the *principal patches* whose sides are lines of curvature for use in Computer Aided Geometric Design (CAGD). Principal patches can be created by imposing two conditions to the boundary curves known as position and frame matching. Among the principal patches, Dupin's cyclide patches whose lines of curvature are all circular arcs are used for blending surface [84], [19]. Alourdas [1] and Alourdas et al [2] provide a method to construct a net of lines of curvature on a B-spline surface. Lines of curvature are of considerable importance to plate-metal-based manufacturing [66]. When a sheet is to be shaped by rolling, then it is fed into the rolls according to a principal direction and the rolls are adjusted according to the principal curvature.

The generic features of the lines of curvature near an umbilic are fully discussed in classic work by Darboux (1896) [14], and more recently by Porteous [81], [82]. Berry and Hannay [5] calculate the average density of umbilics for a surface whose deviation from a plane is specified by a Gaussian random surface, and showed the rarity of the *monstar* pattern. In the CAGD area little attention has been paid to the umbilics for detailed shape analysis. Maekawa and Patrikalakis [59] describe a robust computational method to locate all isolated umbilics on a parametric polynomial surface, see also section 5.3.3. In computer vision, Brady et al [8] compute the lines of curvature and regions of umbilics from range images. Sander and Zucker [89] extracted the umbilics from an image by computing the index of the principal direction fields. Sinha and Besl [95] compute the lines of curvature from a range image and construct a quadrilateral mesh except at the umbilics.

In this Chapter a procedure to compute the lines of curvature near a generic umbilic on a parametric surface is discussed and the numerical implementation of generic feature extraction algorithms for application in automated shape recognition are provided. This Chapter is structured as follows. Section 7.2 shows a method to compute the behavior of lines of curvature near an umbilic provided the surface is in Monge form. Section 7.3 summarizes our method to locate all umbilics and generalizes the surface coordinate system into non-Monge form. Section 7.4 studies a method to trace the lines of curvature. Section 7.5 discusses the conditions for local extrema of principal curvature at umbilics which are used in section 5.3.3. Section 7.6 investigates the stability of generic umbilics due to perturbations and also examines the generic features of umbilics on a reconstructed surface from artificial noisy data. This Chapter refers to Appendix C which proves that if all the third derivatives of the height function of the surface in Monge form are zero then the gradient of the mean curvature to becomes zero.

7.2 Lines of Curvature Near Umbilics

Using the fact that $\kappa = H = \frac{2FM-EN-GL}{2(EG-F^2)}$ and equation (5.23), it is easily verified that $L + \kappa E$, $M + \kappa F$ *and* $N + \kappa G$ *simultaneously become zero if and only if they are evaluated at an umbilic.* Therefore for all du, dv equation (5.15) is satisfied, and hence we cannot determine the direction of the lines of curvature which pass through the umbilic. In this section we investigate the pattern of the lines of curvature near umbilics. Darboux [14] has described three generic features of lines of curvature in the vicinity of an umbilic. The three generic features are called *star*, *monstar* and *lemon* based on the pattern of the net of lines of curvature. Figures 7-1, 7-2 and 7-3 illustrate these three patterns of the net of lines of curvature at the umbilic. The solid line corresponds to the maximum principal curvature lines and the dotted line corresponds to minimum principal curvature lines. Three straight lines of curvature pass through the umbilic for monstar and star, while only one passes for the lemon. The criterion distinguishing monstar from star is that all three directions of lines of curvature through an umbilic are contained in a right angle, whereas in the star case they are not contained in a right angle. There are no other patterns except for non-generic cases. An example of a non-generic umbilic can be offered by the two poles of a convex closed surface of revolution [36]. Figure 7-4 shows the non-generic umbilic of a paraboloid of revolution $z = -\frac{1}{4}(x^2+y^2)$ which has an umbilic that infinite number of lines of curvature pass through. Notice that the paraboloid of revolution is in Monge form and all the coefficients of equation (7.6) below are all zero. *Generic umbilics are stable with respect to small perturbations of the function representing the surface, while non-generic umbilics are unstable* [5], [89], [95]. If we perturb a coefficient in the function representing the surface slightly to $z = -(\frac{x^2}{25/6} + \frac{y^2}{4})$ corresponding to an elliptic paraboloid then the non-generic umbilic splits into two lemon-type generic umbilics as shown in Figure 7-5.

Consider a surface of the form:

$$\mathbf{r} = [x, y, h(x,y)]^T \tag{7.1}$$

We can Taylor expand the z component of the surface as follows

$$\begin{aligned} h(x,y) &= h(0,0) + [xh_x(0,0) + yh_y(0,0)] + \frac{1}{2!}[x^2h_{xx}(0,0) + 2xyh_{xy}(0,0) + y^2h_{yy}(0,0)] \\ &+ \frac{1}{3!}[x^3h_{xxx}(0,0) + 3x^2yh_{xxy}(0,0) + 3xy^2h_{xyy}(0,0) + y^3h_{yyy}(0,0)] \\ &+ O(x^4,\ y^4) \end{aligned} \tag{7.2}$$

Suppose the surface $\mathbf{r}$ has an umbilic at the origin and its tangent plane coincides with the xy-plane, then it is apparent that $h(0,0) = h_x(0,0) = h_y(0,0) = 0$. Also we can say that $h_{xy}(0,0) = 0$, since the vicinity of an umbilic approximates a sphere. We refer to this form as the *Monge form*.

In the vicinity of the origin we can write the z component $h(x,y)$ as follows

$$h(x,y) = \pm\sqrt{a^2 - x^2 - y^2} \mp a \tag{7.3}$$

where a is the radius of curvature at the umbilic. The upper plus sign corresponds to the sphere with its center on the positive z-axis, and the minus sign corresponds to the negative z-axis. By evaluating the second partial derivatives, we obtain

$$h_{xx}(0,0) = h_{yy}(0,0) = -\frac{1}{a} \tag{7.4}$$

Consequently we can rewrite equation (7.2) into a simpler form:

$$\begin{aligned} h(x,y) &= -\frac{1}{2a}(x^2+y^2) \\ &+ \frac{1}{6}[x^3 h_{xxx}(0,0) + 3x^2 y h_{xxy}(0,0) + 3xy^2 h_{xyy}(0,0) + y^3 h_{yyy}(0,0)] \\ &+ O(x^4, y^4) \end{aligned} \tag{7.5}$$

From equation (7.5), we can observe that the equation of the surface near the umbilic is governed by the cubic form $h_c(x,y)$.

$$h_c(x,y) = \frac{1}{6}(\alpha x^3 + 3\beta x^2 y + 3\gamma x y^2 + \delta y^3) \tag{7.6}$$

where

$$\alpha = h_{xxx}(0,0),\ \beta = h_{xxy}(0,0),\ \gamma = h_{xyy}(0,0),\ \delta = h_{yyy}(0,0) \tag{7.7}$$

To study the behavior of the umbilics we can express equation (7.6) in polar coordinates $x = r\cos\theta$ and $y = r\sin\theta$ [5]:

$$h_c(\theta) = \frac{r^3}{6}(\alpha\cos^3\theta + 3\beta\cos^2\theta\sin\theta + 3\gamma\cos\theta\sin^2\theta + \delta\sin^3\theta) \tag{7.8}$$

It can be easily verified that $h_c(\theta+\pi) = -h_c(\theta)$. Therefore the cubic function is an anti-symmetric

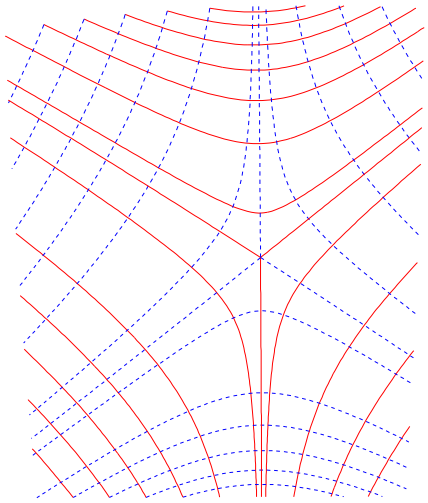

Figure 7-1: Star Pattern (Extracted from lower left umbilic of Figure 7-12)

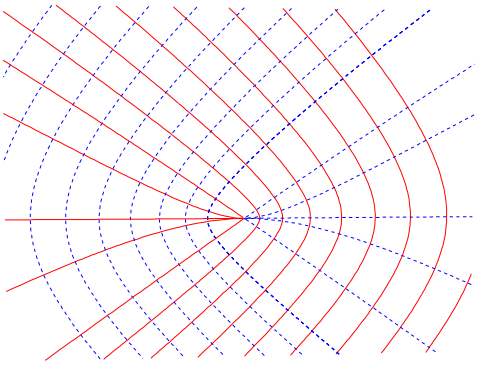

Figure 7-2: Monstar Pattern (Extracted from center umbilic of Figure 7-12)

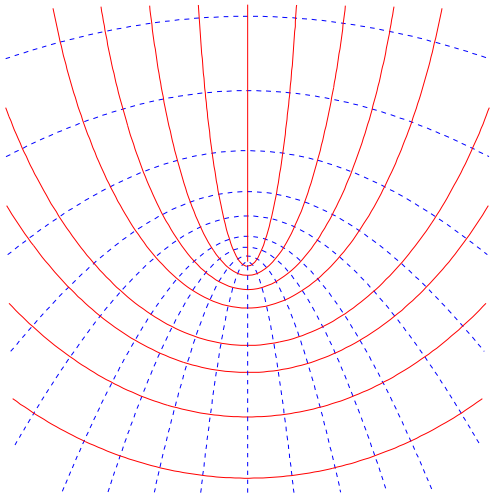

Figure 7-3: Lemon Pattern (Extracted from lower umbilic of Figure 7-5)

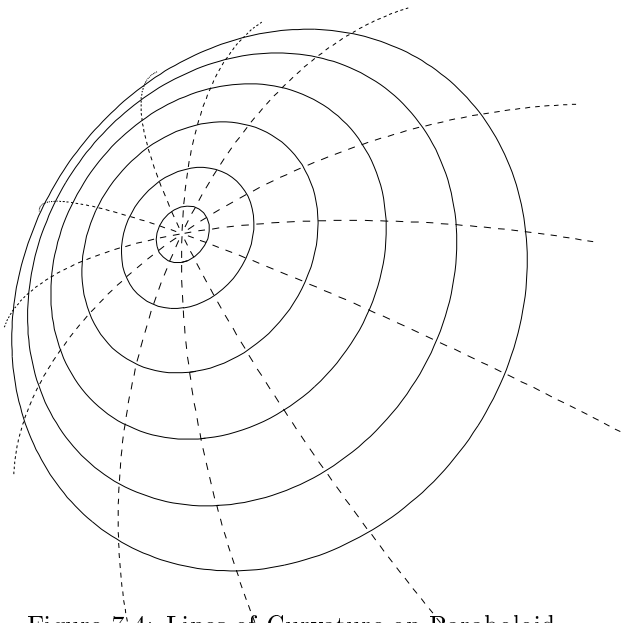

Figure 7-4: Lines of Curvature on Paraboloid

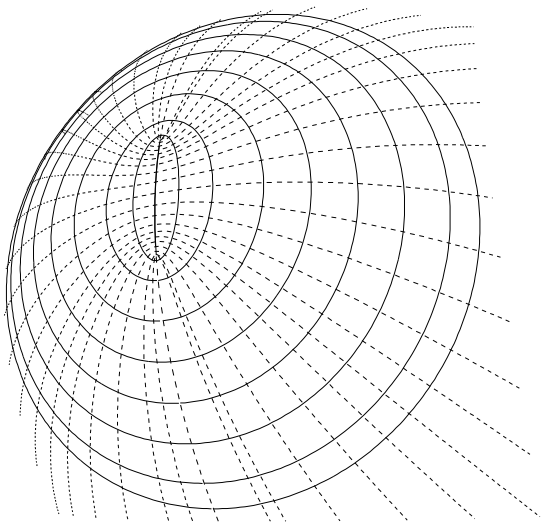

Figure 7-5: Lines of Curvature on Perturbed Paraboloid

function of θ. The roots of $dh_c/d\theta = 0$ will give the angles where local maxima and minima of $h_c(\theta)$ occur around the umbilic. Since it is an antisymmetric function, maxima and minima of h occur on the same straight line which passes through the umbilic. Differentiating (7.8) with respect to θ and setting the equation equal to zero yields

$$\frac{dh_c(\theta)}{d\theta} = \frac{r^3}{2}(\beta\cos^3\theta - (\alpha - 2\gamma)\sin\theta\cos^2\theta + (\delta - 2\beta)\sin^2\theta\cos\theta - \gamma\sin^3\theta) = 0 \tag{7.9}$$

When one of the roots of equation (7.9) is $\theta = 0$ or π then β must be zero, and when $\theta = \frac{\pi}{2}$ or $\frac{3}{2}\pi$ then γ must be zero. Conversely we can say that when $\beta = 0$ one of the roots is $\theta = 0$ or π, and when $\gamma = 0$ one of the roots is $\theta = \frac{\pi}{2}$ or $\frac{3}{2}\pi$. Consequently when $\beta \neq 0$, we can divide equation (7.9) by $\beta\sin^3\theta$ resulting in

$$t^3 - \frac{\alpha - 2\gamma}{\beta}t^2 + \frac{\delta - 2\beta}{\beta}t - \frac{\gamma}{\beta} = 0 \tag{7.10}$$

where $t = \cot\theta$. Similarly when $\gamma \neq 0$, we can divide equation (7.9) by $\gamma\cos^3\theta$ resulting in

$$\hat{t}^3 - \frac{\delta - 2\beta}{\gamma}\hat{t}^2 + \frac{\alpha - 2\gamma}{\gamma}\hat{t} - \frac{\beta}{\gamma} = 0 \tag{7.11}$$

where $\hat{t} = \tan\theta$. These cubic equations may be reduced by the substitution

$$t = s + \frac{\alpha - 2\gamma}{3\beta} \qquad \hat{t} = s + \frac{\delta - 2\beta}{3\gamma} \tag{7.12}$$

to the normal form [97]

$$s^3 + 3ps + 2q = 0 \tag{7.13}$$

where

$$when \quad \beta \neq 0 \quad p = \frac{3\beta(\delta - 2\beta) - (\alpha - 2\gamma)^2}{9\beta} \tag{7.14}$$

$$q = \frac{(2\gamma - \alpha)[2(\alpha - 2\gamma)^2 - 9(\delta - 2\beta)\beta] - 27\beta^2\gamma}{54\beta^3} \tag{7.15}$$

$$when \quad \gamma \neq 0 \quad p = \frac{3\gamma(\alpha - 2\gamma) - (\delta - 2\beta)^2}{9\gamma} \tag{7.16}$$

$$q = \frac{(2\beta - \delta)[2(\delta - 2\beta)^2 - 9(\alpha - 2\gamma)\gamma] - 27\beta\gamma^2}{54\gamma^3} \tag{7.17}$$

The solutions to the cubic equation are given by

- if $q^2 + p^3 > 0$; there are one real root and two conjugate complex roots and the real root is given by

$$s = \sqrt[3]{-q + \sqrt{q^2 + p^3}} + \sqrt[3]{-q - \sqrt{q^2 + p^3}} \tag{7.18}$$

- if $q^2 + p^3 = 0$; there are three real roots at least two of which are equal and are given by

$$s = \mp 2\sqrt{-p}, \;\; \pm\sqrt{-p}, \;\; \pm\sqrt{-p} \tag{7.19}$$

  where the upper sign is to be used if q is positive and the lower sign if q is negative. This is a non-generic case, since small perturbation will yield either case above or case below.

- if $q^2 + p^3 < 0$; there are three unequal roots given by

$$s = 2\sqrt{-p}\,\cos(\frac{\tau}{3}), \;\; 2\sqrt{-p}\,\cos(\frac{\tau}{3} + 120^\circ), \;\; 2\sqrt{-p}\,\cos(\frac{\tau}{3} + 240^\circ) \tag{7.20}$$

where $\cos\tau = \mp\sqrt{-\frac{q^2}{p^3}}$ and the upper sign is to be used if q is positive and the lower if q is negative.

Consequently there is either one single angle or three different angles corresponding to one maximum opposite one minimum or three maxima opposite three minima for generic cases. Corresponding to these angles there are straight lines either one or three passing through the umbilics are the tangent lines to the lines of curvature passing through the umbilics.

Another way of classifying the umbilic is to compute the *index* around the umbilic [5], [89]. The lemon and monstar have the same index $+\frac{1}{2}$, while the star has the index $-\frac{1}{2}$. The index is defined as an amount of rotation that a straight line tangent to lines of curvature experiences when rotating in the counterclockwise direction around a small closed path around the umbilic. To compute the index of the umbilic, we can evaluate the angle ψ_i at n points along the boundary curve which surrounds the umbilic. The angle ψ_i is obtained by using the first of equation (5.15).

$$\tan\psi_i = \frac{dv}{du} = -\frac{L+\kappa E}{M+\kappa F} \quad or \quad \psi_i = arctan(-\frac{L+\kappa E}{M+\kappa F}) \tag{7.21}$$

where $-\frac{\pi}{2} \leq \psi_i \leq \frac{\pi}{2}$. Since ψ_i can also be obtained from the second of equation (5.15) we also get

$$\tan\psi_i = \frac{dv}{du} = -\frac{M+\kappa F}{N+\kappa G} \quad or \quad \psi_i = arctan(-\frac{M+\kappa F}{N+\kappa G}) \tag{7.22}$$

If $N+\kappa G = 0$ or small in absolute value, we use (7.21) else if $M+\kappa F = 0$ or small in absolute value we use (7.22). Consequently the index can be computed by:

$$Index = \frac{1}{2\pi}\sum_{i=0}^{n}\Delta\psi_i \tag{7.23}$$

where

$$\Delta\psi_i = \psi_{(i+1)\ mod\ n} - \psi_i \ \ and \ \ -\frac{\pi}{2} \leq \Delta\psi_i \leq \frac{\pi}{2} \tag{7.24}$$

where *mod* is the modulo operator and is used to account for the first point which is also the last point at which the direction field is evaluated. For most of the examples in this thesis 20 points per boundary curve were adequate for estimation of the index. Figures 7-6, 7-7 and 7-8 illustrate the direction field of maximum principal curvature around the star, monstar and lemon type umbilics.

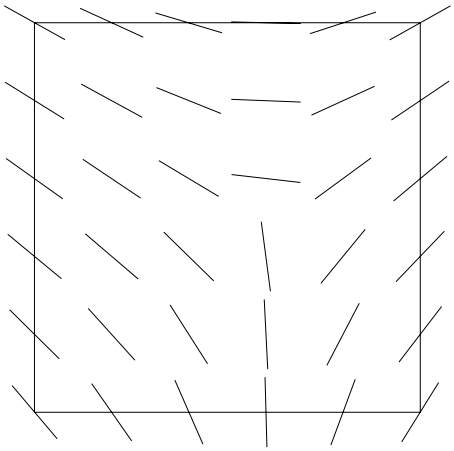

Figure 7-6: Direction Field Near Star-Type Umbilic

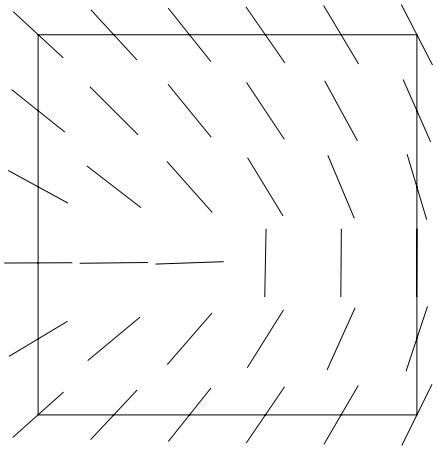

Figure 7-7: Direction Field Near Monstar-Type Umbilic

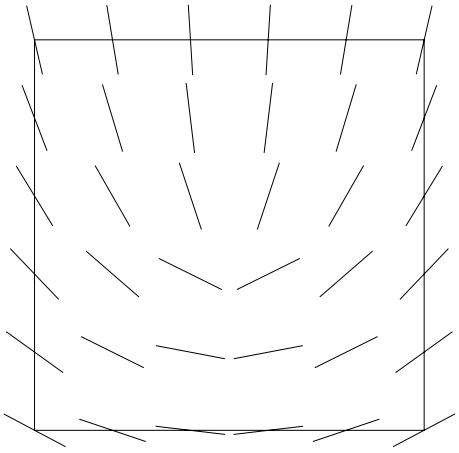

Figure 7-8: Direction Field Near Lemon-Type Umbilic

7.3 Conversion to Monge Form

To compute the angles that the tangents to the lines of curvature at the umbilics make with the axes, the surface has to be set in Monge form for each umbilic separately. Therefore for each umbilic on the surface, a coordinate transformation is needed. But before we conduct the transformation, we need to locate all umbilical points. The principal curvature functions κ are defined in equations (5.17), (5.18) as

$$\kappa(u,v) = H(u,v) \pm \sqrt{H^2(u,v) - K(u,v)} \tag{7.25}$$

We used the symbol W to represent the function inside the square root in equation (5.58). An umbilic occurs precisely at a point where the function $W(u,v)$ is zero. Since κ is a real valued function, it follows that $W(u,v) \geq 0$. Consequently, an umbilic occurs where the function $W(u,v)$ has a global minimum. If the surface representation is non-degenerate and C^k smooth [1] then $H(u,v)$, $K(u,v)$ and hence $W(u,v)$ are C^{k-2} smooth. Although we are particularly interested in Bézier surfaces which are C^∞, we relax the continuity assumption for $W(u,v)$ by assuming that $W(u,v)$ is at least C^2 smooth. Already the assumption of differentiability for $W(u,v)$, which is weaker than C^2, and the condition that $W(u,v)$ for global minimum at the umbilic implies that $\nabla W = \mathbf{0}$ at an umbilic.

[1] A function is called C^k smooth if it has continuous derivatives up to order k.

Therefore the governing equation for locating the umbilics are given by

$$W_u(u,v) = 0, \quad W_v(u,v) = 0, \quad W(u,v) = 0 \tag{7.26}$$

If the surface $\mathbf{r}(u,v)$ is a polynomial parametric surface patch (eg. a Bézier patch) then, these three equations reduce to polynomial equations as we discussed in section 5.3.3. This system of equations is overdetermined, therefore we first solve the first two equations using the Bernstein subdivision method coupled rounded interval arithmetic and select the solutions which satisfy the third.

Consider a frame O-XYZ and a surface $\mathbf{R} = [X(u,v), Y(u,v), Z(u,v)]^T$ with an umbilical point o as illustrated in Figure 7-9. The umbilical point is represented by a position vector $\mathbf{R}_o$ given by:

$$\mathbf{R}_o = (X_o, Y_o, Z_o)^T = [X(u_o,v_o), Y(u_o,v_o), Z(u_o,v_o)]^T \tag{7.27}$$

To represent the surface in the Monge form at the umbilic, we need to attach an orthogonal Cartesian reference frame to it, say o-xyz, and we represent a surface point $\mathbf{r}(u,v)$ in the frame to o-xyz. We choose unit vectors $\frac{\mathbf{R}_u}{|\mathbf{R}_u|}$, $\mathbf{N} \times \frac{\mathbf{R}_u}{|\mathbf{R}_u|}$, $\mathbf{N}$ as directions of x, y and z axes as shown in Figure 7-9, where $\mathbf{R}_u$ is the tangential vector in u direction and $\mathbf{N}$ is the unit normal vector of the surface at the umbilic, which is given by:

$$\mathbf{N} = (N_X, N_Y, N_Z)^T = \frac{\mathbf{R}_u \times \mathbf{R}_v}{|\mathbf{R}_u \times \mathbf{R}_v|} \tag{7.28}$$

If we concatenate these three unit vectors $\frac{\mathbf{R}_u}{|\mathbf{R}_u|}$, $\mathbf{N} \times \frac{\mathbf{R}_u}{|\mathbf{R}_u|}$, $\mathbf{N}$ in a single matrix, we obtain a description of the orientation of the Monge form with respect to the frame O-XYZ which is called a *rotation matrix* $\mathbf{\Omega}$.

$$\mathbf{\Omega} = \begin{pmatrix} \frac{X_u}{|\mathbf{R}_u|} & \frac{N_Y Z_u - N_Z Y_u}{|\mathbf{R}_u|} & N_X \\ \frac{Y_u}{|\mathbf{R}_u|} & \frac{N_Z X_u - N_X Z_u}{|\mathbf{R}_u|} & N_Y \\ \frac{Z_u}{|\mathbf{R}_u|} & \frac{N_X Y_u - N_Y X_u}{|\mathbf{R}_u|} & N_Z \end{pmatrix}_{(u_o, v_o)} \tag{7.29}$$

Then the relation between $\mathbf{R}(u,v)$ and $\mathbf{r}(u,v)$ is:

$$\mathbf{R}(u,v) = \mathbf{R}_o + \mathbf{\Omega}\mathbf{r}(u,v) \tag{7.30}$$

Using equation (7.30), we can solve for $\mathbf{r}(u,v)$ as a function of $\mathbf{R}(u,v)$ that is the coordinate of P

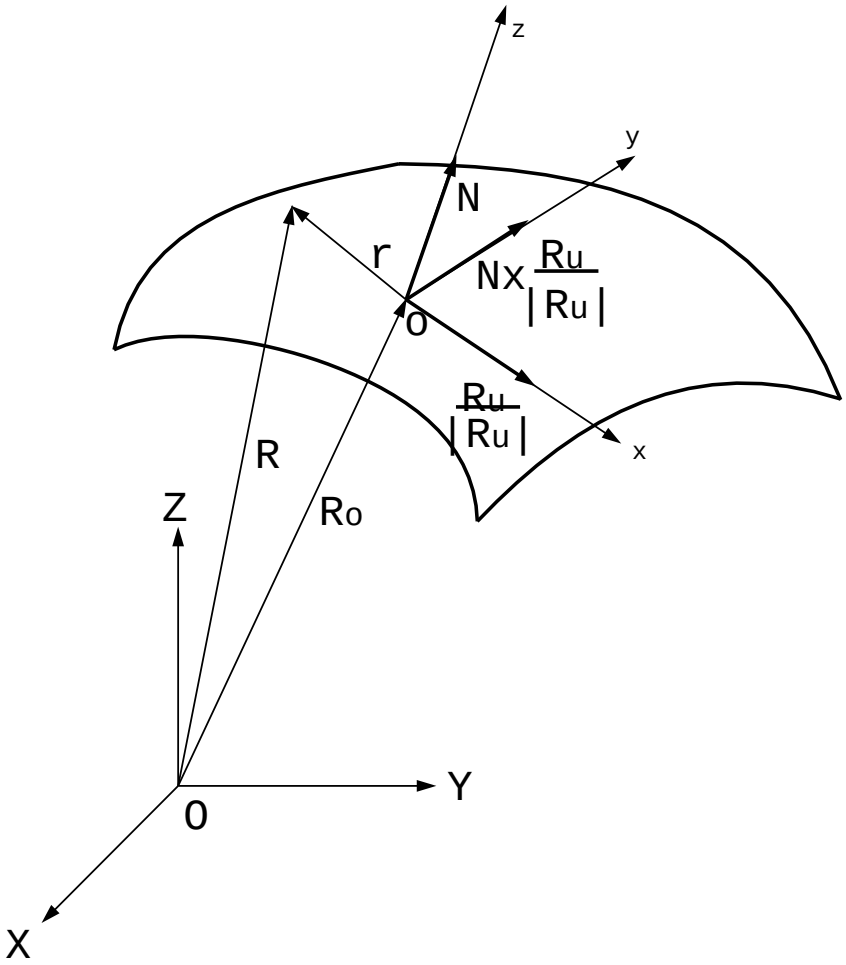

Figure 7-9: Definition of Coordinate System

expressed in frame o-xyz as a function of the coordinate of point P expressed in O-XYZ frame.

$$\mathbf{r}(u,v) = \mathbf{\Omega}^{-1}(\mathbf{R}(u,v) - \mathbf{R}_o) \tag{7.31}$$

where $\mathbf{\Omega}^{-1}$ is the inverse matrix of $\mathbf{\Omega}$. Since $\mathbf{\Omega}$ is an orthonormal matrix, $\mathbf{\Omega}^{-1}$ can be replaced by the transpose matrix $\mathbf{\Omega}^T$, therefore

$$\mathbf{r}(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix} = \mathbf{\Omega}^T[\mathbf{R}(u,v) - \mathbf{R}_o] \tag{7.32}$$

or equivalently

$$x = f(u,v) = \frac{\mathbf{R}_u}{|\mathbf{R}_u|} \cdot [\mathbf{R}(u,v) - \mathbf{R}_o] \tag{7.33}$$

$$y = g(u,v) = \frac{(\mathbf{R}_u \times \mathbf{R}_v) \times \mathbf{R}_u}{|\mathbf{R}_u \times \mathbf{R}_v||\mathbf{R}_u|} \cdot [\mathbf{R}(u,v) - \mathbf{R}_o] \tag{7.34}$$

$$z = h(u,v) = \frac{\mathbf{R}_u \times \mathbf{R}_v}{|\mathbf{R}_u \times \mathbf{R}_v|} \cdot [\mathbf{R}(u,v) - \mathbf{R}_o] \tag{7.35}$$

In equations (7.33)-(7.35) $\mathbf{R}(u,v)$ is the only term that is the function of u and v, whereas all terms involving $\mathbf{R}_u$ and $\mathbf{R}_v$ are evaluated at $(u_o,\ v_o)$. Considering $u = u(x,y)$ and $v = v(x,y)$, using equations (7.33) and (7.34), $\mathbf{r}$ can be written as:

$$\mathbf{r} = (x,y,z)^T = [x, y, h(u(x,y), v(x,y))]^T \tag{7.36}$$

which has the same form as equation (7.1). To evaluate $\alpha,\ \beta,\ \gamma,\ \delta$ we need to compute $h_{xxx},\ h_{xxy},\ h_{xyy},\ h_{yyy}$ which are given by using chain rule as follows:

$$\begin{aligned}
h_x &= h_u u_x + h_v v_x \\
h_y &= h_u u_y + h_v v_y \\
h_{xx} &= h_{uu}u_x^2 + 2h_{uv}u_x v_x + h_{vv}v_x^2 + h_u u_{xx} + h_v v_{xx} \\
h_{xy} &= h_{uu}u_x u_y + h_{uv}(u_x v_y + u_y v_x) + h_{vv}v_x v_y + h_u u_{xy} + h_v v_{xy} \\
h_{yy} &= h_{uu}u_y^2 + 2h_{uv}u_y v_y + h_{vv}v_y^2 + h_u u_{yy} + h_v v_{yy} \\
h_{xxx} &= h_{uuu}u_x^3 + 3h_{uuv}u_x^2 v_x + 3h_{uvv}u_x v_x^2 + h_{vvv}v_x^3
\end{aligned}$$

$$
\begin{aligned}
&+ \quad 3(h_{uu}u_x u_{xx} + h_{uv}u_x v_{xx} + h_{uv}u_{xx}v_x + h_{vv}v_x v_{xx}) + h_u u_{xxx} + h_v v_{xxx} \\
h_{xxy} &= h_{uuu}u_x^2 u_y + h_{uuv}u_x(2u_y v_x + u_x v_y) + h_{uvv}v_x(2u_x v_y + u_y v_x) + h_{vvv}v_x^2 v_y \\
&+ \quad h_{uu}(2u_x u_{xy} + u_{xx}u_y) + h_{uv}(2u_x v_{xy} + u_{xx}v_y + u_y v_{xx} + 2u_{xy}v_x) \\
&+ \quad h_{vv}(2v_x v_{xy} + v_{xx}v_y) + h_u u_{xxy} + h_v v_{xxy} \\
h_{xyy} &= h_{uuu}u_x u_y^2 + h_{uuv}u_y(2u_x v_y + u_y v_x) + h_{uvv}v_y(2u_y v_x + u_x v_y) + h_{vvv}v_x v_y^2 \\
&+ \quad h_{uu}(2u_{xy}u_y + u_x u_{yy}) + h_{uv}(2u_{xy}v_y + u_x v_{yy} + u_{yy}v_x + 2u_y v_{xy}) \\
&+ \quad h_{vv}(2v_{xy}v_y + v_x v_{yy}) + h_u u_{xyy} + h_v v_{xyy} \\
h_{yyy} &= h_{uuu}u_y^3 + 3h_{uuv}u_y^2 v_y + 3h_{uvv}u_y v_y^2 + h_{vvv}v_y^3 \\
&+ \quad 3(h_{uu}u_y u_{yy} + h_{uv}u_y v_{yy} + h_{uv}u_{yy}v_y + h_{vv}v_y v_{yy}) + h_u u_{yyy} + h_v v_{yyy}
\end{aligned}
\tag{7.37}
$$

Partial derivatives of h with respect to u and v can be obtained easily from equation (7.35). By rewriting equations (7.33) and (7.34) as

$$F(x, y, u, v) = x - f(u, v) = 0 \tag{7.38}$$

$$G(x, y, u, v) = y - g(u, v) = 0 \tag{7.39}$$

and using the implicit function theorem we can determine u_x, u_y, v_x and v_y by differentiating the system with respect to x and y, considering that u and v are functions of both x and y, then F and G can be considered as depending on x and y directly and also intermediately through u and v, so obtain the four relations.

$$\frac{\partial F}{\partial x} + \frac{\partial F}{\partial u}\frac{\partial u}{\partial x} + \frac{\partial F}{\partial v}\frac{\partial v}{\partial x} = (\frac{\partial}{\partial x} + \frac{\partial u}{\partial x}\frac{\partial}{\partial u} + \frac{\partial v}{\partial x}\frac{\partial}{\partial v})F = 0 \tag{7.40}$$

$$\frac{\partial G}{\partial x} + \frac{\partial G}{\partial u}\frac{\partial u}{\partial x} + \frac{\partial G}{\partial v}\frac{\partial v}{\partial x} = (\frac{\partial}{\partial x} + \frac{\partial u}{\partial x}\frac{\partial}{\partial u} + \frac{\partial v}{\partial x}\frac{\partial}{\partial v})G = 0 \tag{7.41}$$

$$\frac{\partial F}{\partial y} + \frac{\partial F}{\partial u}\frac{\partial u}{\partial y} + \frac{\partial F}{\partial v}\frac{\partial v}{\partial y} = (\frac{\partial}{\partial y} + \frac{\partial u}{\partial y}\frac{\partial}{\partial u} + \frac{\partial v}{\partial y}\frac{\partial}{\partial v})F = 0 \tag{7.42}$$

$$\frac{\partial G}{\partial y} + \frac{\partial G}{\partial u}\frac{\partial u}{\partial y} + \frac{\partial G}{\partial v}\frac{\partial v}{\partial y} = (\frac{\partial}{\partial y} + \frac{\partial u}{\partial y}\frac{\partial}{\partial u} + \frac{\partial v}{\partial y}\frac{\partial}{\partial v})G = 0 \tag{7.43}$$

where operational notation is also used. The partial derivatives of F and G with respect to u and v can be easily obtained by using equations (7.33), (7.34). For the purpose of obtaining analytical formulae for higher order partial derivatives, it is often convenient to use the operational notation.

Applying Cramer's rule to equations (7.40) and (7.41), we can solve for u_x and v_x in terms of determinants as follows.

$$u_x = -\frac{\begin{vmatrix} F_x & F_v \\ G_x & G_v \end{vmatrix}}{\begin{vmatrix} F_u & F_v \\ G_u & G_v \end{vmatrix}} \qquad v_x = -\frac{\begin{vmatrix} F_u & F_x \\ G_u & G_x \end{vmatrix}}{\begin{vmatrix} F_u & F_v \\ G_u & G_v \end{vmatrix}} \tag{7.44}$$

Similarly equations (7.42) and (7.43) are solved for u_y and v_y.

$$u_y = -\frac{\begin{vmatrix} F_y & F_v \\ G_y & G_v \end{vmatrix}}{\begin{vmatrix} F_u & F_v \\ G_u & G_v \end{vmatrix}} \qquad v_y = -\frac{\begin{vmatrix} F_u & F_y \\ G_u & G_y \end{vmatrix}}{\begin{vmatrix} F_u & F_v \\ G_u & G_v \end{vmatrix}} \tag{7.45}$$

We assumed, however, that the common denominator in equations (7.44) and (7.45) does not vanish. Using the same procedure, we can also obtain for the higher-order derivatives such as u_{xx}, v_{xx}, u_{xy}, v_{xx}, u_{yy}, v_{yy}, u_{xxx}, v_{xxx}, u_{xxy}, v_{xxy}, u_{xyy}, v_{xyy}, u_{yyy}, v_{yyy}. For example, to evaluate u_{xxy}, v_{xxy}, we need to compute

$$(\frac{\partial}{\partial y} + \frac{\partial u}{\partial y}\frac{\partial}{\partial u} + \frac{\partial v}{\partial y}\frac{\partial}{\partial v})(\frac{\partial}{\partial x} + \frac{\partial u}{\partial x}\frac{\partial}{\partial u} + \frac{\partial v}{\partial x}\frac{\partial}{\partial v})^2 F = 0 \tag{7.46}$$

$$(\frac{\partial}{\partial y} + \frac{\partial u}{\partial y}\frac{\partial}{\partial u} + \frac{\partial v}{\partial y}\frac{\partial}{\partial v})(\frac{\partial}{\partial x} + \frac{\partial u}{\partial x}\frac{\partial}{\partial u} + \frac{\partial v}{\partial x}\frac{\partial}{\partial v})^2 G = 0 \tag{7.47}$$

Once $h_{xxx},\ h_{xxy},\ h_{xyy},\ h_{yyy}$ are obtained, we can compute the angles of tangent lines to the lines of curvature passing through the umbilic using equations (7.10) to (7.20). Since the angles are evaluated in the xy-plane we need to map back to the parametric uv-space for integration. Consider a point on the tangent line which passes through the origin and lies on the xy-plane, say $(r\cos\theta,\ r\sin\theta)$. Then the point can be expressed in terms of u, v using the vectors along the x and y axes.

$$\begin{aligned} & r\cos\theta\frac{\mathbf{R}_u}{|\mathbf{R}_u|} \ + \ r\sin\theta\frac{\mathbf{R}_u \times \mathbf{R}_v}{|\mathbf{R}_u \times \mathbf{R}_v|} \times \frac{\mathbf{R}_u}{|\mathbf{R}_u|} \\ = \ & r\left[\frac{\cos\theta}{|\mathbf{R}_u|} - \frac{\sin\theta(\mathbf{R}_u \cdot \mathbf{R}_v)}{|\mathbf{R}_u \times \mathbf{R}_v||\mathbf{R}_u|}\right]\mathbf{R}_u + r\left[\frac{\sin\theta|\mathbf{R}_u|}{|\mathbf{R}_u \times \mathbf{R}_v|}\right]\mathbf{R}_v \\ = \ & \lambda\mathbf{R}_u + \xi\mathbf{R}_v \end{aligned} \tag{7.48}$$

$\phi = arctan(\frac{\xi}{\lambda})$ gives the angle in the uv parametric space.

7.4 Integration of Lines of Curvature

A line of curvature is a curve on a surface that has tangents which are principal directions at all of its points. The principal directions at a given point are those directions for which the normal curvature takes on minimum and maximum values. If the point is not an umbilic the principal directions are orthogonal. A line of curvature indicates a directional flow for the maximum or the minimum curvature across the surface. It is advantageous to express the curvature line with an arc length parametrization as $u = u(s) \quad v = v(s)$. Every principal curvature direction vector must fulfill equations (5.15). Hence from the first equation of (5.15) we get

$$\begin{aligned} \dot{u} &= \frac{du}{ds} = \eta(M + \kappa F) \\ \dot{v} &= \frac{dv}{ds} = -\eta(L + \kappa E) \end{aligned} \tag{7.49}$$

where η is an arbitrary nonzero factor. Since a principal curvature direction vector must also fulfill the second equation of (5.15) we also get

$$\begin{aligned} \dot{u} &= \frac{du}{ds} = \mu(N + \kappa G) \\ \dot{v} &= \frac{dv}{ds} = -\mu(M + \kappa F) \end{aligned} \tag{7.50}$$

The solutions $\dot{u}, \dot{v}$ of the first and the second equations (5.15) are linearly dependent, because the system of linear equations given by equations (5.15) has a rank smaller than 2. It is possible:

A. That the coefficients in one of the equations can both be zero while they are not both zero in the other equation.

B. That both coefficients in one equation are small in absolute value while the other equation contains one coefficient which is large in absolute value.

Case **B** is encountered more often than case **A**. In case **A**, using the equation with zero coefficients yields an incorrect result, because this equation does not contain enough information to find the principal curvature direction. In case **B**, using the equation with the small coefficients may yield numerical inaccuracies which could be avoided by using the other equation. Such an algorithm which makes the choice of the equation dependent on the size of the coefficients was implemented by [1], [2]. Since $M + \kappa F$ is a common coefficient, if $|L + \kappa E| \geq |N + \kappa G|$ we solve (7.49) otherwise we solve (7.50).

We want to point out that also case **A** may easily occur. Therefore one needs provisions in the computational algorithm and in its implementation which take this into account. The possibility of case **A** appears to have been overlooked in the literature on lines of curvature computation. We give now a simple example illustrating case **A** using a parabolic cylinder $\mathbf{r}(u,v) = (u, v, v^2)$. Clearly the minimum principal curvature on the parabolic cylinder is zero everywhere. Also it is apparent that $\mathbf{r}_{uu} = \mathbf{0}$ and $\mathbf{r}_{uv} = \mathbf{0}$, hence $L = M = 0$. Therefore $L + \kappa_{min}E$ and $M + \kappa_{min}F$ become zero, while $N + \kappa_{min}G \neq 0$, which can be seen by an easy computation or using the fact that parabolic cylinder has no umbilics.

It remains to determine factors η and μ. The curvature line needs to be arc length parametrized. From the first fundamental form, arc length parametrization implies

$$E(\frac{du}{ds})^2 + 2F\frac{du}{ds}\frac{dv}{ds} + G(\frac{dv}{ds})^2 = 1 \tag{7.51}$$

Substituting (7.49) into (7.51), η is determined to be

$$\eta = \frac{\pm 1}{\sqrt{E(M+\kappa F)^2 - 2F(M+\kappa F)(L+\kappa E) + G(L+\kappa E)^2}} \tag{7.52}$$

Substituting (7.50) into (7.51), μ is determined to be

$$\mu = \frac{\pm 1}{\sqrt{E(N+\kappa G)^2 - 2F(N+\kappa G)(M+\kappa F) + G(M+\kappa F)^2}} \tag{7.53}$$

The sign of η or μ determines the direction in which the solution proceeds. Choosing a fixed sign for η or μ does not guarantee that the vector $(\dot{u}, \dot{v})$ would not change direction. The need to adjust the sign of η or μ becomes even more obvious if one determines the principal curvature vector always by the numerically preferable equation in the system (5.15). The vectors obtained from equations (7.49) and (7.50) are linearly dependent but they need not to have the same orientation.

The criterion which is employed in order to determine the sign of η or μ is given by the following inequality

$$|-(\dot{u}^p\mathbf{r}_u^p + \dot{v}^p\mathbf{r}_v^p) - (\dot{u}\mathbf{r}_u + \dot{v}\mathbf{r}_v)| < |(\dot{u}^p\mathbf{r}_u^p + \dot{v}^p\mathbf{r}_v^p) + (\dot{u}\mathbf{r}_u + \dot{v}\mathbf{r}_v)| \tag{7.54}$$

where $\mathbf{r}$ is a curvature line represented by the parametric form $\mathbf{r}(s) = \mathbf{r}[u(s), v(s)]$ and the superscript p means evaluation at the previous time step during the integration of the curvature line. It is obvious

that inequality (7.54) is true if and only if the tangent vector $(\dot{u}\mathbf{r}_u + \dot{v}\mathbf{r}_v)$ changes sign because (7.54) says that the negative tangent vector of the preceding time step is closer to the new tangent vector than the positive tangent vector of the preceding time step. When inequality (7.54) is true, the sign of η or μ should be changed to assure that the solution path does not reverse direction.

We can trace the lines of curvature by integrating the initial value problem for a system of coupled nonlinear differential equations using a variable stepsize and variable order Adams method. Starting points for lines of curvature passing through the umbilics are obtained by slightly shifting outwards in the directions given by equation (7.48) from the umbilic. Accuracy of the lines of curvature depends on the number of integrated points used to represent the contour line by straight line segments.

7.5 Local Extrema of Principal Curvatures at Umbilics

In this section we discuss a criterion which assures the existence of local extrema of the principal curvature functions κ_{max} and κ_{min} at umbilical points of the surface. *The problem to detect local extrema of principal curvature functions is motivated by engineering applications. When a ball end-mill cutter is used for NC milling machines, the cutter radius must be smaller than the smallest concave radius of curvature of the surface to be machined to avoid local overcut (gouging). Gouging is the one of the most critical problems in NC machining of free-form surfaces*, see section 2.2. Therefore, we must determine the distribution of the principle curvatures of the surface, which are upper and lower bounds on the curvature at a given point, to select the cutter size. A natural approach to locate local extrema of the functions κ_{max} and κ_{min} would in principle be to search for zeros of the gradient vector field $\nabla\kappa_{max}$ and $\nabla\kappa_{min}$ and then use tools from differential calculus to decide if at those zeros the principal curvature functions attain extrema. The problem with this approach however is that the curvature functions κ_{max} and κ_{min} are generally not differentiable at the umbilics although those points may also be candidates for local principal curvature extrema, see section 5.3.3. We will present a sufficient criterion which guarantees the existence of a local extremum of the principal curvature functions κ_{max} and κ_{min} at an umbilic. This criterion is practical because it is almost always applicable and easily evaluated.

We discuss the local behavior of the functions κ_{max} and κ_{min} in the neighborhood of an umbilic. First let us consider a Taylor expansion around an umbilic (u_o, v_o) for the function defined in (5.58). We obtain

$$W(u,v) \quad = \quad W(u_o,v_o) + [(u-u_o)W_u(u_o,v_o) + (v-v_o)W_v(u_o,v_o)]$$

$$+ \quad \frac{1}{2!}[(u-u_o)^2 W_{uu}(u_o,v_o) + 2(u-u_o)(v-v_o)W_{uv}(u_o,v_o) + (v-v_o)^2 W_{vv}(u_o,v_o)]$$
$$+ \quad O(u-u_o, v-v_o)|(u-u_o, v-v_o)|^2 \tag{7.55}$$

with

$$\lim_{u \to u_o, v \to v_o} O(u-u_o, v-v_o) = 0 \tag{7.56}$$

Note that (7.55) describes the remainder term in case of a second order Taylor approximation of a C^2 smooth function. In the special case where all the second partial derivatives of W vanish, the condition $W(u,v) \geq 0$ implies that the third order partial derivatives must also vanish. This special case rarely happens; therefore, we focus our attention now on the generic case where at least one of the second order partial derivatives of W does not vanish. Using equation (7.26), we obtain $W(u_o, v_o) = 0$ and $\nabla W(u_o, v_o) = \mathbf{0}$ at the umbilic, therefore equation (7.55) reduces to

$$W(u,v) \quad = \quad W_Q(u,v) + O(u-u_o, v-v_o)|(u-u_o, v-v_o)|^2 \tag{7.57}$$

where

$$\begin{aligned} W_Q(u,v) &= \frac{1}{2}(u-u_o, v-v_o) \begin{pmatrix} W_{uu} & W_{uv} \\ W_{uv} & W_{vv} \end{pmatrix} (u-u_o, v-v_o)^T \\ &= \frac{1}{2} \frac{(u-u_o, v-v_o)}{|(u-u_o, v-v_o)|} \begin{pmatrix} W_{uu} & W_{uv} \\ W_{uv} & W_{vv} \end{pmatrix} \frac{(u-u_o, v-v_o)^T}{|(u-u_o, v-v_o)|} |(u-u_o, v-v_o)|^2 \end{aligned} \tag{7.58}$$

Now we can Taylor expand $\sqrt{W(u,v)}$ up to first order [2]

$$\sqrt{W(u,v)} \quad = \quad \sqrt{W_Q} + \frac{O(u-u_o, v-v_o)}{2\sqrt{W_Q}} |(u-u_o, v-v_o)|^2 \tag{7.59}$$

$$= \quad [C(u,v) + \frac{O(u-u_o, v-v_o)}{2C(u,v)}] |(u-u_o, v-v_o)| \tag{7.60}$$

where

$$C(u,v) = \sqrt{\frac{1}{2} \frac{(u-u_o, v-v_o)}{|(u-u_o, v-v_o)|} \begin{pmatrix} W_{uu} & W_{uv} \\ W_{uv} & W_{vv} \end{pmatrix} \frac{(u-u_o, v-v_o)^T}{|(u-u_o, v-v_o)|}} \tag{7.61}$$

[2] Note that here the Taylor expansion of the square root first yields an approximation instead of the equal sign in (7.59). However absorbing here the error term of this square root Taylor expansion in the remainder of (7.59) justifies the equality sign.

Next we Taylor expand the mean curvature $H(u,v)$ as follows:

$$H(u,v) = H(u_o,v_o) + (H_L(u,v) + O(u-u_o, v-v_o))|(u-u_o, v-v_o)| \tag{7.62}$$

where

$$H_L(u,v) = [H_u(u_o,v_o), H_v(u_o,v_o)]\frac{(u-u_o, v-v_o)^T}{|(u-u_o, v-v_o)|} \tag{7.63}$$

Although the function $O(u-u_o, v-v_o)$ in the remainder terms are different in equations (7.57), (7.60), (7.62), we nonetheless use the same notation for simplicity, since we are essentially interested in the common property described in equation (7.56). Consequently $\kappa(u,v)$ in equation (7.25) can be expanded in the vicinity of an umbilic (u_o,v_o) as follows

$$\begin{aligned} \kappa(u,v) &= H(u_o,v_o) + (H_L(u,v) \pm C(u,v) + O(u-u_o, v-v_o))|(u-u_o, v-v_o)| \\ &= H(u_o,v_o) + \bar{H}_L(u,v) \pm \bar{C}(u,v) + \bar{O}(u-u_o, v-v_o) \end{aligned} \tag{7.64}$$

where

$$\bar{H}_L(u,v) = H_L(u,v)|(u-u_o, v-v_o)| \tag{7.65}$$

$$\bar{C}(u,v) = C(u,v)|(u-u_o, v-v_o)| \tag{7.66}$$

$$\bar{O}(u-u_o, v-v_o) = O(u-u_o, v-v_o)|(u-u_o, v-v_o)| \tag{7.67}$$

Therefore $\kappa(u,v)$ can be considered as sum of the constant term $H(u_o,v_o)$, the plane $\bar{H}_L(u,v)$ which is the tangent plane of $H(u,v)$ at (u_o,v_o) and the elliptic cone $\bar{C}(u,v)$ whose axis of symmetry is perpendicular to uv-plane, since $W(u_o,v_o) = 0$, $\nabla W(u_o,v_o) = \mathbf{0}$. First we assume that $\bar{H}_L(u,v) = 0$, in other words the tangent plane of $H(u,v)$ coincides with the uv-plane. In this case $\kappa(u,v) - H(u_o,v_o)$ reduces to $\pm\bar{C}(u,v)$. Figure 7-10 shows a positive elliptic cone $+\bar{C}(u,v)$ (maximum principal curvature) having a minimum at (u_o,v_o). When the elliptic cone is negative, minimum principal curvature has a maximum at (u_o,v_o). The condition $\bar{H}_L(u,v) = 0$ occurs when all the third order partial derivatives of the height function in the Monge form are zero, see Appendix C. Note that in case $\bar{H}_L = 0$ the term $\bar{O}(u-u_o, v-v_o)$ is negligible for local extremum properties of the function $\kappa(u,v) - H(u_o,v_o)$, see also the footnote 2. Consequently when $\bar{H}_L(u,v) = 0$, or alternatively when $\nabla H(u_o,v_o) = \mathbf{0}$, the function $\kappa(u,v) - H(u_o,v_o)$, hence $\kappa(u,v)$ has a local ex-

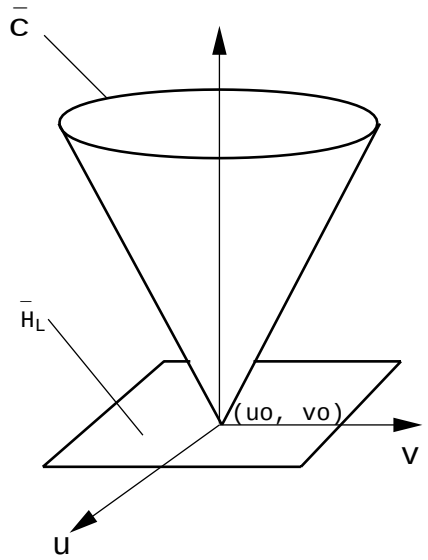

Figure 7-10: Cone $\bar{C}(u,v)$ is perpendicular to the plane $\bar{H}_L$

tremum at (u_o, v_o), or more precisely, κ_{max} has a local minimum and κ_{min} has a local maximum at an umbilical point (u_o, v_o).

It is also possible that $\kappa(u,v)$ may have a local extremum at the umbilic when $\bar{H}_L(u,v) \neq 0$. This is the situation when the plane $\bar{H}_L(u,v)$ is tilted against the uv-plane.[3] Figure 7-11 (a) is the case when the plane intersects the cone in two straight lines. In this case the sum $\bar{H}_L(u,v) \pm \bar{C}(u,v)$ does not have an extremum at (u_o, v_o), while in case (b) the plane intersects the cone only at (u_o, v_o), and therefore $\bar{H}_L(u,v) \pm \bar{C}(u,v)$ has a local extremum at (u_o, v_o). Consequently we need to examine the equation $\pm\bar{C}(u,v) = -\bar{H}_L(u,v)$ which upon squaring and using equations (7.63) and (7.61) can be reduced to

$$(W_{uu} - 2H_u^2)(u-u_o)^2 + 2(W_{uv} - 2H_uH_v)(u-u_o)(v-v_o) + (W_{vv} - 2H_v^2)(v-v_o)^2 = 0 \qquad (7.68)$$

We can rewrite equation (7.68) as

$$A(u-u_o)^2 + 2B(u-u_o)(v-v_o) + C(v-v_o)^2 = 0 \qquad (7.69)$$

[3] Note that we use the following observation illustrated by Figure 7-11 (b). The term $\bar{O}(u-u_o, v-v_o)$ is negligible for investigating the local extrema properties of the function $\kappa(u,v)$ at the umbilic (u_o, v_o), provided the cone $\bar{C}(u,v)$ and the plane $\bar{H}_L = 0$ meet only at the point (u_o, v_o). Namely in that case we have a positive number α such that $|\bar{C}(u,v) - \bar{H}_L(u,v)| \geq \alpha|(u-u_o, v-v_o)|$. α is related to the smallest possible slope between plane and cone. Hence clearly $O(u-u_o, v-v_o)|(u-u_o, v-v_o)|$ is negligible to $\alpha|(u-u_o, v-v_o)|$.

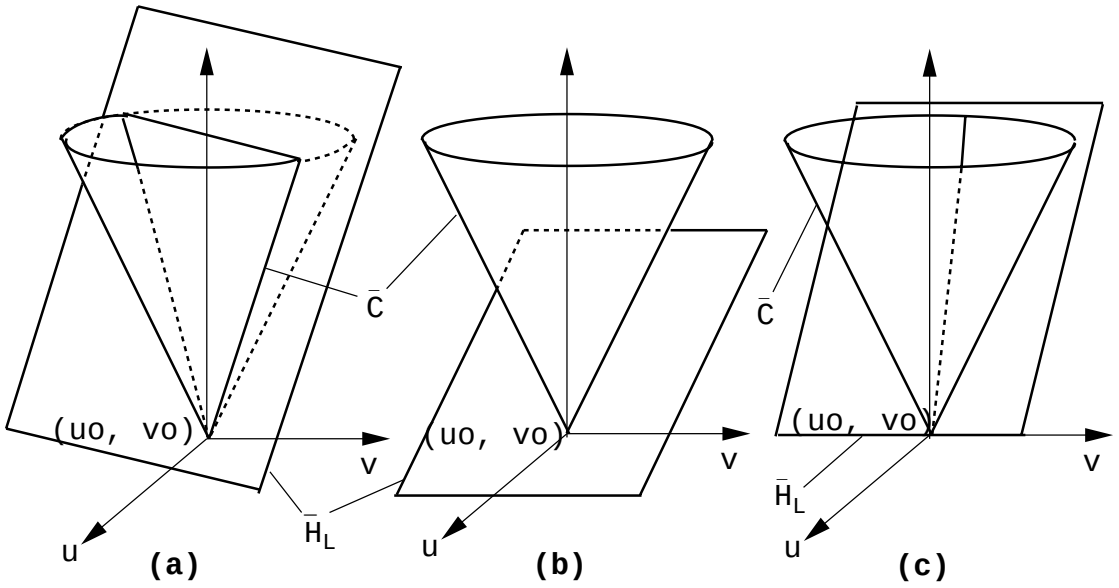

Figure 7-11: Cone $\bar{C}(u,v)$ is not perpendicular to the plane $\bar{H}_L$

so that we can view equation (7.68) as a quadratic equation with unknown $u-u_o$ or $v-v_o$. If $B^2-AC>0$ there exist two distinct real roots, and thus there will be a real intersection between the plane and the cone made up of two straight lines. If $B^2-AC=0$ there exist two identical real roots, and thus the cone and the plane are tangent to each other, and additional evaluation of higher order terms in the Taylor expansion is necessary to decide if we have an extremum at the umbilic. If $B^2-AC<0$ there will be no real root, and thus there is no intersection between the cone and the plane. Consequently the criterion to have a local extremum of principal curvatures, when $\bar{H}_L(u,v)\neq 0$ or $\nabla H(u_o,v_o)\neq \mathbf{0}$, is equivalent to the condition $B^2-AC<0$. Hence the condition is

$$(W_{uv}-2H_uH_v)^2-(W_{uu}-2H_u^2)(W_{vv}-2H_v^2)<0 \tag{7.70}$$

or equivalently upon using $W(u,v)=H^2(u,v)-K(u,v)$

$$(2HH_{uv}-K_{uv})^2-(2HH_{uu}-K_{uu})(2HH_{vv}-K_{vv})<0 \tag{7.71}$$

Finally we can state the criterion as follows:

Theorem (Criterion for extrema of principal curvature functions at umbilics):

If we denote $W(u,v) = H^2(u,v) - K(u,v)$ and assume that $W(u,v)$ is at least C^2 smooth and at least one of the second order partial derivatives of $W(u,v)$ does not vanish then:

1. If $\nabla H = \mathbf{0}$ at the umbilic, then κ_{max} has a local minimum and κ_{min} has a local maximum.

2. If $\nabla H \neq \mathbf{0}$ at the umbilic, then κ_{max} has a local minimum and κ_{min} has a local maximum if and only if $D = (2HH_{uv} - K_{uv})^2 - (2HH_{uu} - K_{uu})(2HH_{vv} - K_{vv}) < 0$ provided $D \neq 0$. In case $D = 0$, additional evaluation of higher order terms in the Taylor expansion is necessary.

7.6 Surface Recognition

In this section, we give a few numerical examples to demonstrate how generic umbilics are stable with respect to perturbations. The example surface is a wave-like bicubic integral Bézier patch which is also used in section 5.5.1, see Figure 5-3. There are four spherical umbilics and one flat umbilic point on the surface. We gradually perturb the control points of the surface and observe the behavior of the lines of curvature which pass through umbilics. The control points are perturbed in the following manner. Since the example is a bicubic patch, it has 16 control points. Each control point consists of three Cartesian coordinates $x,\ y,\ z$, hence there are 48 components to be perturbed. A random number which varies from -1 to 1 is used to determine the 48 components. Let us denote the randomly chosen numbers for each control point as $[e_{ij}^x,\ e_{ij}^y,\ e_{ij}^z]^T,\ 0 \leq i \leq 3,\ 0 \leq j \leq 3$. We normalize the vector and add to each control point as follows:

$$\tilde{\mathbf{P}}_{ij} = \mathbf{P}_{ij} + \zeta \frac{[e_{ij}^x,\ e_{ij}^y,\ e_{ij}^z]^T}{\sqrt{{e_{ij}^x}^2 + {e_{ij}^y}^2 + {e_{ij}^z}^2}} \tag{7.72}$$

where $\zeta = 0.02$. We increase the amount of perturbations gradually by increasing ζ from 0.02 by 0.02 up to 0.1. The curvature value κ, the four coefficients of the cubic terms $\alpha,\ \beta,\ \gamma,\ \delta$, angles of the tangent lines to the lines of curvature which pass through the umbilic in the xy-plane $\theta_1,\ \theta_2,\ \theta_3$ and in the uv-space $\phi_1,\ \phi_2,\ \phi_3$ all in radians, index and the type are listed for original surface and five perturbed surfaces in Tables 7.1 to 7.6. The angles $\theta_i,\ \phi_i\ (1 \leq i \leq 3)$ are restricted in the range $-\frac{\pi}{2} \leq \theta_i,\ \phi_i \leq \frac{\pi}{2}$. Figure 7-12 to 7-17 illustrate how the lines of curvature which only pass through the umbilic behave when the control points are perturbed. The thick solid line represents the lines of curvature for maximum principal curvature, thick dotted line represents the lines of curvature for minimum principal curvature and the thin solid lines are iso-parametric lines.

(u, v)	(0.211, 0.052)	(0.211, 0.984)	(0.789, 0.052)	(0.789, 0.984)	(0.5, 0.440)
κ	1.197	0.267	-1.197	-0.267	0.
α	4.147	0.926	4.147	0.926	6.514
β	-18.306	14.670	18.306	-14.670	0.
γ	0.	0.	0.	0.	4.2763
δ	-2.337	1.411	2.337	-1.411	0.
θ_1	0.671	0.562	0.592	0.638	0.
θ_2	-0.592	-0.638	-0.671	-0.616	-0.604
θ_3	1.571	-1.571	-1.571	1.571	0.604
ϕ_1	0.567	0.562	0.495	0.583	0.
ϕ_2	-0.495	-0.583	-0.567	-0.562	-0.752
ϕ_3	1.571	1.571	-1.571	-1.571	0.752
index	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
type	star	star	star	star	monstar

Table 7.1: Umbilics of original surface

From the figures and tables we can observe that the umbilic on the upper right jumps off from the domain but the other four umbilics remain inside the domain. *All the umbilics which stay in the domain do not change their index nor their type.* From these observations we can conclude that the umbilics are quite stable to the perturbation. Also the locations and the angles $\theta_i,\ \phi_i$ $(1 \leq i \leq 3)$ of the umbilics do not move nor rotate too much.

In computer vision the geometric information of an object are obtained by range imaging sensors. Generally the data include noise and are processed using image processing techniques to exclude the noise, then the derivatives are directly computed from the digital data to evaluate the curvatures. What we do in the sequel is to fit a surface directly from artificial noisy data and observe the behavior of the umbilics on the fitted surface. The noisy data are produced in the following way. Evenly spaced 10×10 grid points (x, y) on $0 \leq x \leq 1$, $0 \leq y \leq 1$ domain are chosen to evaluate the z-value of the wave-like bicubic Bézier patch. We add randomly perturbed vectors with $\zeta = 0.05$, as introduced in equation (7.72), to the (x, y, z) points on the surface as noise. Then the data points (x, y, z) are fit by a bicubic Bézier patch $s(x, y)$ such that the sum of squares of residuals $s(x, y) - z$ becomes minimum. Figure 7-18 and Table 7.7 illustrate the results. We observe that all the umbilics stay in the domain with index and types unchanged. Also the locations and the angles $\theta_i,\ \phi_i$ $(1 \leq i \leq 3)$ do not move nor rotate too much. These results provide us confidence for using the umbilics for shape recognition problems.

(u, v)	(0.201, 0.053)	(0.213, 0.981)	(0.792, 0.066)	n/a	(0.497, 0.432)
κ	1.244	0.197	-1.321	n/a	0.034
α	3.266	0.708	4.538	n/a	6.429
β	-17.220	13.837	18.621	n/a	0.082
γ	0.231	-0.830	-0.343	n/a	4.462
δ	-2.755	1.544	2.776	n/a	0.232
θ_1	0.666	0.605	0.587	n/a	0.657
θ_2	-0.606	-0.653	-0.681	n/a	-0.033
θ_3	1.564	1.539	-1.561	n/a	-0.626
ϕ_1	0.563	0.556	0.509	n/a	0.810
ϕ_2	-0.513	-0.586	-0.581	n/a	-0.045
ϕ_3	1.552	1.562	-1.552	n/a	-0.784
index	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	n/a	$\frac{1}{2}$
type	star	star	star	n/a	monstar

Table 7.2: Umbilics on perturbed surface ($\zeta = 0.02$)

(u, v)	(0.190, 0.055)	(0.214, 0.978)	(0.794, 0.081)	n/a	(0.492, 0.424)
κ	1.293	0.136	-1.458	n/a	0.083
α	2.390	0.551	5.014	n/a	6.351
β	-16.119	13.046	18.926	n/a	0.163
γ	0.563	-1.524	-0.360	n/a	4.666
δ	-3.182	1.711	3.234	n/a	0.510
θ_1	0.658	0.593	0.586	n/a	0.701
θ_2	-0.623	-0.667	-0.689	n/a	-0.055
θ_3	1.551	1.509	1.560	n/a	-0.644
ϕ_1	0.559	0.549	0.528	n/a	0.857
ϕ_2	-0.537	-0.589	-0.596	n/a	-0.076
ϕ_3	1.529	1.552	-1.532	n/a	-0.811
index	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	n/a	$\frac{1}{2}$
type	star	star	star	n/a	monstar

Table 7.3: Umbilics on perturbed surface ($\zeta = 0.04$)

(u, v)	(0.179, 0.059)	(0.216, 0.974)	(0.795, 0.097)	n/a	(0.485, 0.417)
κ	1.351	0.084	-1.610	n/a	0.154
α	1.534	0.443	5.606	n/a	6.307
β	-15.048	12.301	19.200	n/a	0.239
γ	1.019	-2.086	-0.016	n/a	4.893
δ	-3.642	1.889	3.676	n/a	0.850
θ_1	0.647	0.583	0.588	n/a	0.740
θ_2	-0.645	-0.681	-0.692	n/a	-0.070
θ_3	1.532	1.480	1.570	n/a	-0.655
ϕ_1	0.558	0.541	0.551	n/a	0.900
ϕ_2	-0.570	-0.592	-0.613	n/a	-0.098
ϕ_3	1.500	1.542	-1.510	n/a	-0.832
index	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	n/a	$\frac{1}{2}$
type	star	star	star	n/a	monstar

Table 7.4: Umbilics on perturbed surface ($\zeta = 0.06$)

(u, v)	(0.167, 0.065)	(0.217, 0.970)	(0.795, 0.113)	n/a	(0.474, 0.411)
κ	1.426	0.042	-1.779	n/a	0.261
α	0.701	0.374	6.355	n/a	6.356
β	-14.070	11.604	19.405	n/a	0.307
γ	1.621	-2.520	0.727	n/a	5.155
δ	-4.184	2.057	4.072	n/a	1.273
θ_1	0.632	0.573	0.594	n/a	0.773
θ_2	-0.674	-0.694	-0.692	n/a	-0.079
θ_3	1.504	1.455	-1.550	n/a	-0.655
ϕ_1	0.557	0.534	0.577	n/a	0.928
ϕ_2	-0.614	-0.594	-0.631	n/a	-0.112
ϕ_3	1.466	1.532	-1.485	n/a	-0.841
index	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	n/a	$\frac{1}{2}$
type	star	star	star	n/a	monstar

Table 7.5: Umbilics on perturbed surface ($\zeta = 0.08$)

(u, v)	(0.156, 0.075)	(0.218, 0.964)	(0.793, 0.131)	n/a	(0.457, 0.406)
κ	1.535	0.007	-1.961	n/a	0.419
α	-0.128	0.334	7.322	n/a	6.623
β	-13.270	10.959	19.475	n/a	0.366
γ	2.378	-2.833	1.907	n/a	5.477
δ	-4.898	2.201	4.381	n/a	1.806
θ_1	0.612	0.564	0.604	n/a	0.799
θ_2	-0.710	-0.706	-0.689	n/a	-0.087
θ_3	1.463	1.432	-1.515	n/a	-0.640
ϕ_1	0.559	0.526	0.607	n/a	0.950
ϕ_2	-0.670	-0.596	-0.650	n/a	-0.124
ϕ_3	1.427	1.521	-1.455	n/a	-0.831
index	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	n/a	$\frac{1}{2}$
type	star	star	star	n/a	monstar

Table 7.6: Umbilics on perturbed surface ($\zeta = 0.1$)

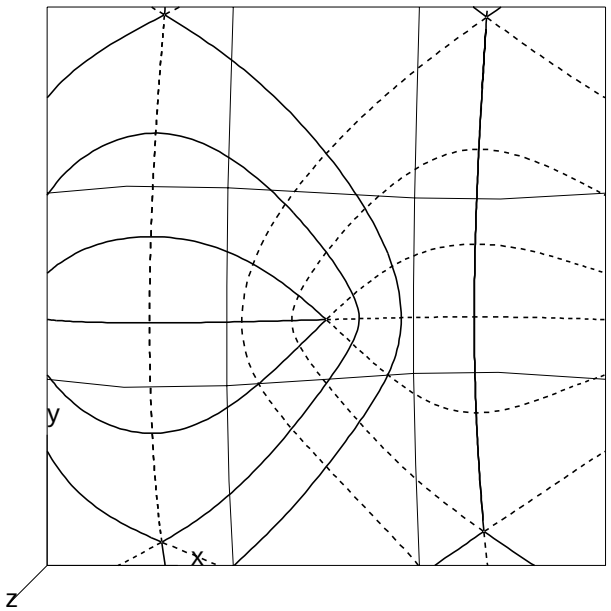

Figure 7-12: Lines of curvature passing through the umbilics ($\zeta = 0$)

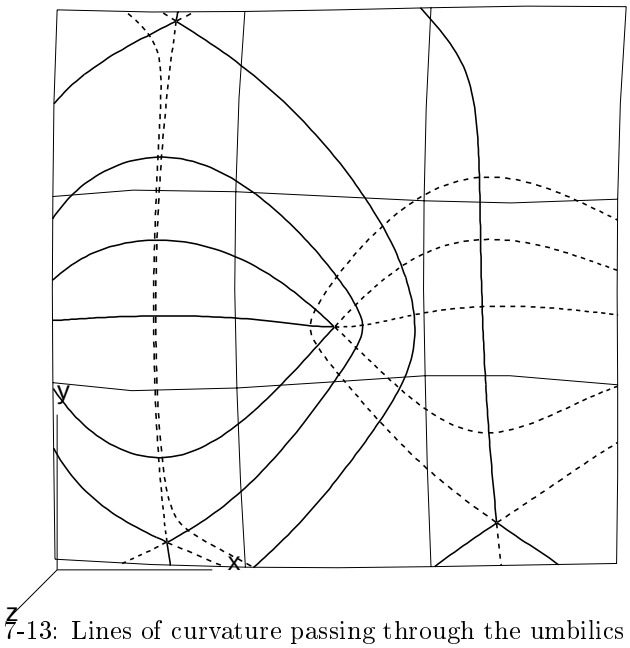

Figure 7-13: Lines of curvature passing through the umbilics ($\zeta = 0.02$)

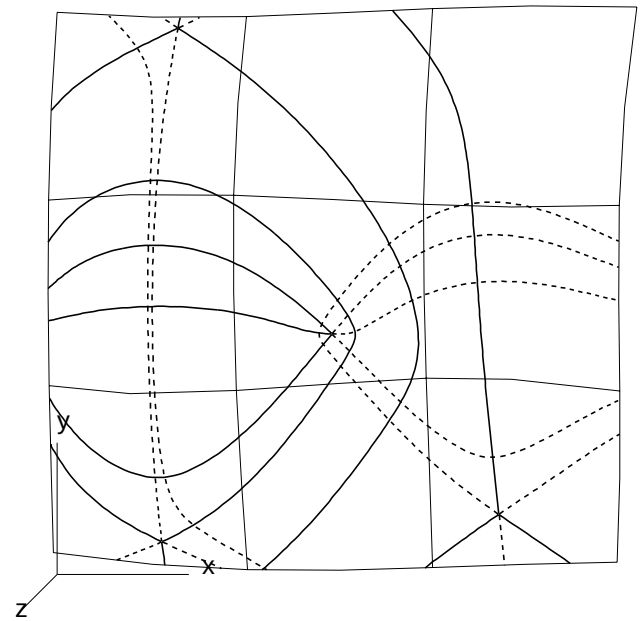

Figure 7-14: Lines of curvature passing through the umbilics ($\zeta = 0.04$)

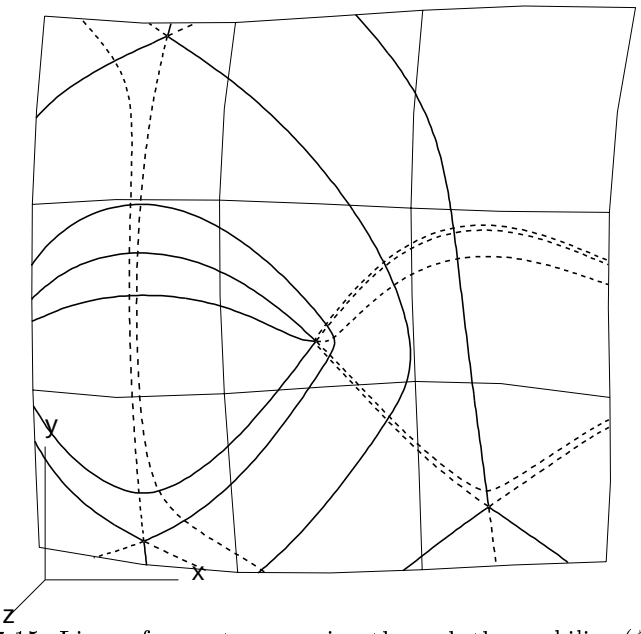

Figure 7-15: Lines of curvature passing through the umbilics ($\zeta = 0.06$)

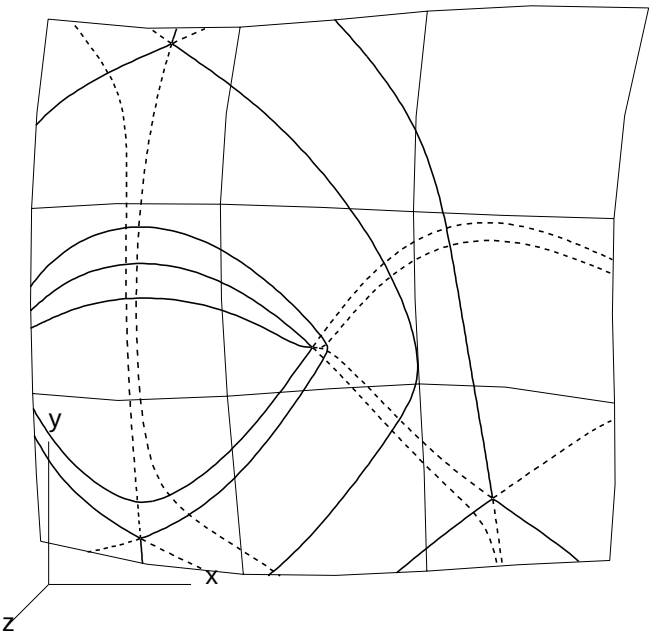

Figure 7-16: Lines of curvature passing through the umbilics ($\zeta = 0.08$)

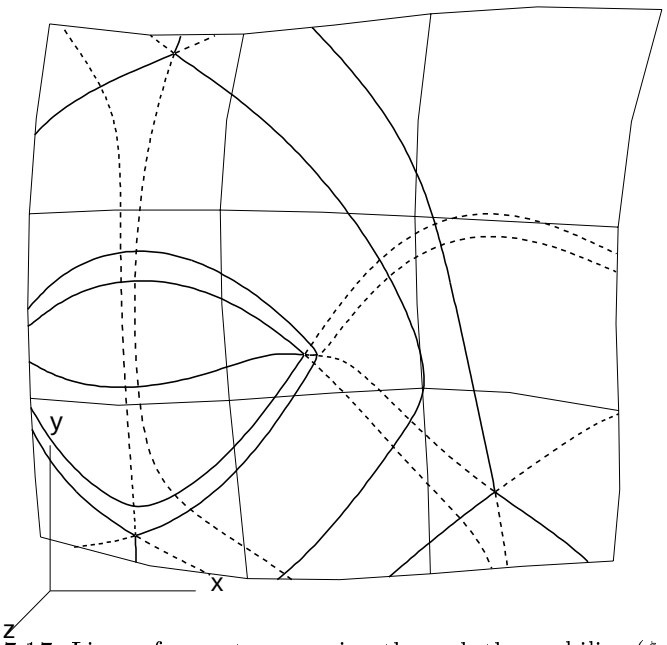

Figure 7-17: Lines of curvature passing through the umbilics ($\zeta = 0.1$)

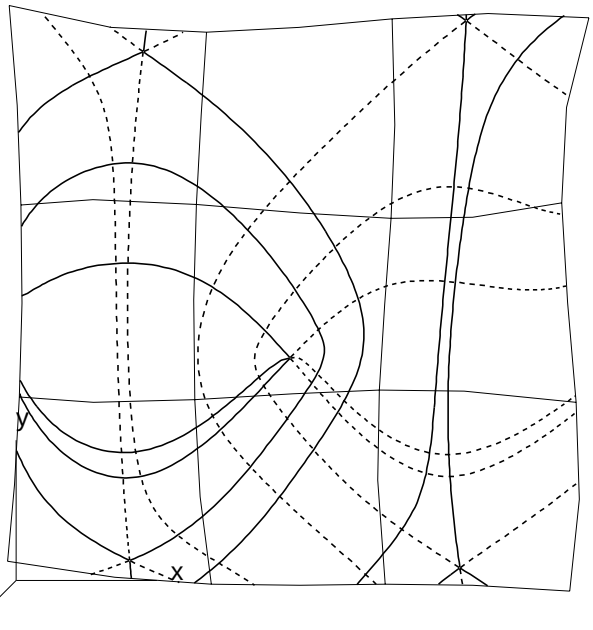

Figure 7-18: Lines of curvature passing through the umbilics on fitted surface ($\zeta = 0.05$)

(u, v)	(0.198, 0.043)	(0.227, 0.954)	(0.813, 0.025)	(0.796, 0.991)	(0.493, 0.399)
κ	1.278	0.147	-1.005	-0.124	0.090
α	1.748	0.999	0.678	0.357	6.572
β	-16.370	16.161	19.451	-11.981	-0.293
γ	-0.054	0.061	4.716	0.536	4.849
δ	-2.705	2.176	3.082	-1.233	-0.410
θ_1	0.656	0.622	0.686	0.622	0.094
θ_2	-0.616	-0.642	-0.575	-0.634	-0.692
θ_3	-1.569	-1.569	-1.443	1.547	0.657
ϕ_1	0.532	0.615	0.528	0.543	0.132
ϕ_2	-0.535	-0.630	-0.482	-0.554	-0.856
ϕ_3	1.497	-1.562	-1.513	1.544	0.827
index	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
type	star	star	star	star	monstar

Table 7.7: Umbilics on reconstructed surface ($\zeta = 0.05$)

Chapter 8

Conclusions and Recommendations

In this Chapter, we summarize the major results and contributions of this thesis. We also identify other potential applications and related future research topics on robust shape interrogation.

8.1 Summary and Contributions

- A robust method based on Bernstein subdivision coupled with rounded interval arithmetic algorithm is developed for solving a system of n nonlinear polynomial equations with a finite number of roots. This can be applied generally to a variety of geometry processing problems.
- The auxiliary variable method, which is an efficient and accurate method to handle multidimensional nonlinear problems involving polynomials and square root of polynomials, is developed. This method allows processing of a variety of problems of computer aided geometric design which involve the analytical expressions of principal curvatures and of the unit normal vectors of curves and surfaces.
- Comparison of root finding schemes using interval Newton method and Bernstein subdivision method (both implemented in rounded interval arithmetic) is conducted. The results show that interval Newton method is suitable for low degree low dimension problems, while Bernstein subdivision method performs better for high degree high dimension problems.
- Rounded interval arithmetic is only one order of magnitude slower than floating point arithmetic for the Bernstein subdivision method, while rational arithmetic is several order of mag-

nitude slower, and therefore ineffective for the solution process.

- Formulating the governing simultaneous nonlinear equations in multivariate Bernstein form with rational arithmetic and solving the equations with rounded interval arithmetic is the most robust and accurate combination for the computation.

- A new robust method to compute the singularities of a normal offset of a planar integral polynomial parametric curve and the intersections of two specific normal offsets of planar integral polynomial parametric curves is introduced. This method can be applied to automatic tool path generation for $2\frac{1}{2}$-D milling NC machining, tolerance region construction and feature recognition through construction of skeletons of geometric models.

- A new accurate and robust method to decompose a Bézier surface patch into subpatches with specific ranges of curvature, including Gaussian, mean, maximum principal and minimum principal curvature, is introduced. This method can be applied to automatic machine tool size selection and tool path generation for $3D$ and $5D$ NC machining, and surface interrogation for fairing.

- An algorithm for automatically generating a triangular mesh approximation for polynomial parametric surfaces based on root mean square curvature is presented. The algorithm produces well-shaped elements and provides small and dense mesh elements in the region where the curvature is large, and large and sparse mesh elements in the region where the curvature is small without any mesh relaxation and smoothing.

- A theoretical development and numerical implementation of generic feature extraction algorithms for application in shape recognition of polynomial parametric surfaces using umbilics is presented. Algorithms include the detection of all the locations of umbilics and classification of their types on the Bézier surface. Numerical experiments show the stability of the generic umbilics under perturbation.

- A new criterion which guarantees the existence of a local extremum of principal curvature functions at an umbilic is introduced.

8.2 Future Research

- Speedup of rounded interval arithmetic with lower level machine functions is needed.

- Development of hybrid methods combining Bernstein subdivision in early stages of the process with interval Newton method coupled with rounded interval arithmetic for root refining in order to accelerate convergence is an interesting topic for future work.
- Parallelization of the solution method.
- Our solver for a system of n nonlinear polynomial equations is restricted to a finite number of roots. Further research is necessary for the general case.
- The robust numerical solution of nonlinear equations via interval arithmetic concepts described in this thesis suggests the application of *interval splines, such as the interval non-uniform rational B-splines (INURBS)*, in the representation and interrogation of functions with uncertainty [104], [103] and in geometric and solid modeling, eg. in surface intersections and distance computations.
- When a number of solutions exist within a domain, methods other than binary subdivision to enhance efficiency are also needed to speed up the computation.
- Extension of the offset curve interrogation method to offset surface interrogation method is an interesting topic for future work.
- Research towards fully automated tool path generation and tool size selection for $2\frac{1}{2}$-D pocket machining and 3D, 5D machining is an interesting topic for future work.
- Extension from integral polynomial parametric curves and surfaces to the non-uniform rational B-splines (NURBS) curves and surfaces is needed.
- A further area worthy of investigation involves the development of automated methods for the creation of quadrilateral orthogonal meshes using the lines of curvature combined with our technique to handle the umbilics.
- Analytical investigation of the behavior of the umbilics under perturbation is desirable for understanding the nature of umbilics.
- Research towards fully automated solid free-form fabrication techniques exploiting the methods of this thesis, in direct robust and efficient surface contouring or through adaptive surface tessellation, is a future research topic.

Appendix A

Formulas for Curvature Partial Derivatives

In this appendix all the derivatives that are needed to evaluate the partial derivatives of Gaussian, mean, principal and root mean square curvatures up to second order are derived. All the denominators and numerators of the first derivatives are expressed in polynomials, square roots of polynomials or combination of both. The denominators and numerators of the second derivatives of Gaussian and mean curvatures are expressed in polynomials or square roots of polynomials, while the second derivatives of principal and root mean square curvatures include fractional expressions, since the second derivatives are only needed for point evaluation.

Surface normal

$$\mathbf{S} = \mathbf{r}_u \times \mathbf{r}_v \tag{A.1}$$

$$\mathbf{S}_u = \mathbf{r}_{uu} \times \mathbf{r}_v \ + \ \mathbf{r}_u \times \mathbf{r}_{uv}, \quad \mathbf{S}_v = \mathbf{r}_{uv} \times \mathbf{r}_v \ + \ \mathbf{r}_u \times \mathbf{r}_{vv} \tag{A.2}$$

$$\mathbf{S}_{uu} = \mathbf{r}_{uuu} \times \mathbf{r}_v \ + \ 2\mathbf{r}_{uu} \times \mathbf{r}_{uv} \ + \ \mathbf{r}_u \times \mathbf{r}_{uuv} \tag{A.3}$$

$$\mathbf{S}_{uv} = \mathbf{r}_{uuv} \times \mathbf{r}_v \ + \ \mathbf{r}_{uu} \times \mathbf{r}_{vv} \ + \ \mathbf{r}_u \times \mathbf{r}_{uvv} \tag{A.4}$$

$$\mathbf{S}_{vv} = \mathbf{r}_{uvv} \times \mathbf{r}_v \ + \ 2\mathbf{r}_{uv} \times \mathbf{r}_{vv} \ + \ \mathbf{r}_u \times \mathbf{r}_{vvv} \tag{A.5}$$

Scalar magnitude of surface normal

$$S = |\mathbf{r}_u \times \mathbf{r}_v| = \sqrt{det[\Gamma]} = \sqrt{EG - F^2} \tag{A.6}$$

$$S_u = \frac{\mathbf{S} \cdot \mathbf{S}_u}{S}, \quad S_v = \frac{\mathbf{S} \cdot \mathbf{S}_v}{S} \tag{A.7}$$

$$S_{uu} = \frac{\mathbf{S} \cdot \mathbf{S}_{uu} \ + \ \mathbf{S}_u \cdot \mathbf{S}_u \ - \ S_u^2}{S} \tag{A.8}$$

$$S_{uv} = \frac{\mathbf{S} \cdot \mathbf{S}_{uv} \ + \ \mathbf{S}_u \cdot \mathbf{S}_v \ - \ S_u S_v}{S} \tag{A.9}$$

$$S_{vv} = \frac{\mathbf{S} \cdot \mathbf{S}_{vv} \ + \ \mathbf{S}_v \cdot \mathbf{S}_v \ - \ S_v^2}{S} \tag{A.10}$$

The coefficients of first fundamental form

$$E = \mathbf{r}_u \cdot \mathbf{r}_u \tag{A.11}$$

$$E_u = 2\mathbf{r}_u \cdot \mathbf{r}_{uu}, \quad E_v = 2\mathbf{r}_u \cdot \mathbf{r}_{uv} \tag{A.12}$$

$$E_{uu} = 2(\mathbf{r}_{uu} \cdot \mathbf{r}_{uu} \ + \ \mathbf{r}_u \cdot \mathbf{r}_{uuu}) \tag{A.13}$$

$$E_{uv} = 2(\mathbf{r}_{uv} \cdot \mathbf{r}_{uu} \ + \ \mathbf{r}_u \cdot \mathbf{r}_{uuv}) \tag{A.14}$$

$$E_{vv} = 2(\mathbf{r}_{uv} \cdot \mathbf{r}_{uv} \ + \ \mathbf{r}_u \cdot \mathbf{r}_{uvv}) \tag{A.15}$$

$$F = \mathbf{r}_u \cdot \mathbf{r}_v \tag{A.16}$$

$$F_u = \mathbf{r}_{uu} \cdot \mathbf{r}_v \ + \ \mathbf{r}_u \cdot \mathbf{r}_{uv}, \quad F_v = \mathbf{r}_{uv} \cdot \mathbf{r}_v \ + \ \mathbf{r}_u \cdot \mathbf{r}_{vv} \tag{A.17}$$

$$F_{uu} = \mathbf{r}_{uuu} \cdot \mathbf{r}_v \ + \ 2\mathbf{r}_{uu} \cdot \mathbf{r}_{uv} \ + \ \mathbf{r}_u \cdot \mathbf{r}_{uuv} \tag{A.18}$$

$$F_{uv} = \mathbf{r}_{uuv} \cdot \mathbf{r}_v \ + \ \mathbf{r}_{uu} \cdot \mathbf{r}_{vv} \ + \ \mathbf{r}_{uv} \cdot \mathbf{r}_{uv} \ + \ \mathbf{r}_u \cdot \mathbf{r}_{uvv} \tag{A.19}$$

$$F_{vv} = \mathbf{r}_{uvv} \cdot \mathbf{r}_v \ + \ 2\mathbf{r}_{uv} \cdot \mathbf{r}_{vv} \ + \ \mathbf{r}_u \cdot \mathbf{r}_{vvv} \tag{A.20}$$

$$G = \mathbf{r}_v \cdot \mathbf{r}_v \tag{A.21}$$

$$G_u = 2\mathbf{r}_v \cdot \mathbf{r}_{uv}, \quad G_v = 2\mathbf{r}_v \cdot \mathbf{r}_{vv} \tag{A.22}$$

$$G_{uu} = 2(\mathbf{r}_{uv} \cdot \mathbf{r}_{uv} \ + \ \mathbf{r}_v \cdot \mathbf{r}_{uuv}) \tag{A.23}$$

$$G_{uv} = 2(\mathbf{r}_{vv} \cdot \mathbf{r}_{uv} \ + \ \mathbf{r}_v \cdot \mathbf{r}_{uvv}) \tag{A.24}$$

$$G_{vv} = 2(\mathbf{r}_{vv} \cdot \mathbf{r}_{vv} \ + \ \mathbf{r}_v \cdot \mathbf{r}_{vvv}) \tag{A.25}$$

The coefficients of second fundamental form multiplied by S

$$\tilde{L} = SL = \mathbf{S} \cdot \mathbf{r}_{uu} \tag{A.26}$$

$$\tilde{L}_u = \mathbf{S}_u \cdot \mathbf{r}_{uu} + \mathbf{S} \cdot \mathbf{r}_{uuu}, \quad \tilde{L}_v = \mathbf{S}_v \cdot \mathbf{r}_{uu} + \mathbf{S} \cdot \mathbf{r}_{uuv} \tag{A.27}$$

$$\tilde{L}_{uu} = \mathbf{S}_{uu} \cdot \mathbf{r}_{uu} + 2\mathbf{S}_u \cdot \mathbf{r}_{uuu} + \mathbf{S} \cdot \mathbf{r}_{uuuu} \tag{A.28}$$

$$\tilde{L}_{uv} = \mathbf{S}_{uv} \cdot \mathbf{r}_{uu} + \mathbf{S}_u \cdot \mathbf{r}_{uuv} + \mathbf{S}_v \cdot \mathbf{r}_{uuu} + \mathbf{S} \cdot \mathbf{r}_{uuuv} \tag{A.29}$$

$$\tilde{L}_{vv} = \mathbf{S}_{vv} \cdot \mathbf{r}_{uu} + 2\mathbf{S}_v \cdot \mathbf{r}_{uuv} + \mathbf{S} \cdot \mathbf{r}_{uuvv} \tag{A.30}$$

$$\tilde{M} = SM = \mathbf{S} \cdot \mathbf{r}_{uv} \tag{A.31}$$

$$\tilde{M}_u = \mathbf{S}_u \cdot \mathbf{r}_{uv} + \mathbf{S} \cdot \mathbf{r}_{uuv}, \quad \tilde{M}_v = \mathbf{S}_v \cdot \mathbf{r}_{uv} + \mathbf{S} \cdot \mathbf{r}_{uvv} \tag{A.32}$$

$$\tilde{M}_{uu} = \mathbf{S}_{uu} \cdot \mathbf{r}_{uv} + 2\mathbf{S}_u \cdot \mathbf{r}_{uuv} + \mathbf{S} \cdot \mathbf{r}_{uuuv} \tag{A.33}$$

$$\tilde{M}_{uv} = \mathbf{S}_{uv} \cdot \mathbf{r}_{uv} + \mathbf{S}_u \cdot \mathbf{r}_{uvv} + \mathbf{S}_v \cdot \mathbf{r}_{uuv} + \mathbf{S} \cdot \mathbf{r}_{uuvv} \tag{A.34}$$

$$\tilde{M}_{vv} = \mathbf{S}_{vv} \cdot \mathbf{r}_{uv} + 2\mathbf{S}_v \cdot \mathbf{r}_{uvv} + \mathbf{S} \cdot \mathbf{r}_{uvvv} \tag{A.35}$$

$$\tilde{N} = SN = \mathbf{S} \cdot \mathbf{r}_{vv} \tag{A.36}$$

$$\tilde{N}_u = \mathbf{S}_u \cdot \mathbf{r}_{vv} + \mathbf{S} \cdot \mathbf{r}_{uvv}, \quad \tilde{N}_v = \mathbf{S}_v \cdot \mathbf{r}_{vv} + \mathbf{S} \cdot \mathbf{r}_{vvv} \tag{A.37}$$

$$\tilde{N}_{uu} = \mathbf{S}_{uu} \cdot \mathbf{r}_{vv} + 2\mathbf{S}_u \cdot \mathbf{r}_{uvv} + \mathbf{S} \cdot \mathbf{r}_{uuvv} \tag{A.38}$$

$$\tilde{N}_{uv} = \mathbf{S}_{uv} \cdot \mathbf{r}_{vv} + \mathbf{S}_u \cdot \mathbf{r}_{vvv} + \mathbf{S}_v \cdot \mathbf{r}_{uvv} + \mathbf{S} \cdot \mathbf{r}_{uvvv} \tag{A.39}$$

$$\tilde{N}_{vv} = \mathbf{S}_{vv} \cdot \mathbf{r}_{vv} + 2\mathbf{S}_v \cdot \mathbf{r}_{vvv} + \mathbf{S} \cdot \mathbf{r}_{vvvv} \tag{A.40}$$

The determinant of second fundamental matrix multiplied by S^2

$$A = \tilde{L}\tilde{N} - \tilde{M}^2 \tag{A.41}$$

$$A_u = \tilde{L}_u\tilde{N} + \tilde{L}\tilde{N}_u - 2\tilde{M}\tilde{M}_u, \quad A_v = \tilde{L}_v\tilde{N} + \tilde{L}\tilde{N}_v - 2\tilde{M}\tilde{M}_v \tag{A.42}$$

$$A_{uu} = \tilde{L}_{uu}\tilde{N} + \tilde{L}\tilde{N}_{uu} + 2(\tilde{L}_u\tilde{N}_u - \tilde{M}_u^2 - \tilde{M}\tilde{M}_{uu}) \tag{A.43}$$

$$A_{uv} = \tilde{L}_{uv}\tilde{N} + \tilde{L}_u\tilde{N}_v + \tilde{L}_v\tilde{N}_u + \tilde{L}\tilde{N}_{uv} - 2(\tilde{M}_v\tilde{M}_u + \tilde{M}\tilde{M}_{uv}) \tag{A.44}$$

$$A_{vv} = \tilde{L}_{vv}\tilde{N} + \tilde{L}\tilde{N}_{vv} + 2(\tilde{L}_v\tilde{N}_v - \tilde{M}_v^2 - \tilde{M}\tilde{M}_{vv}) \tag{A.45}$$

The numerator of the mean curvature, equation (5.20) multiplied by S

$$B = 2F\tilde{M} - E\tilde{N} - G\tilde{L} \tag{A.46}$$

$$B_u = 2(F\tilde{M}_u + F_u\tilde{M}) \; - \; (E_u\tilde{N} + E\tilde{N}_u) \; - \; (G_u\tilde{L} + G\tilde{L}_u) \tag{A.47}$$

$$B_v = 2(F\tilde{M}_v + F_v\tilde{M}) \; - \; (E_v\tilde{N} + E\tilde{N}_v) \; - \; (G_v\tilde{L} + G\tilde{L}_v) \tag{A.48}$$

$$\begin{aligned} B_{uu} = {} & 2(F\tilde{M}_{uu} + 2F_u\tilde{M}_u + F_{uu}\tilde{M}) \; - \; (E_{uu}\tilde{N} + 2E_u\tilde{N}_u + E\tilde{N}_{uu}) \\ & - \; (G_{uu}\tilde{L} + 2G_u\tilde{L}_u + G\tilde{L}_{uu}) \end{aligned} \tag{A.49}$$

$$\begin{aligned} B_{uv} = {} & 2(F\tilde{M}_{uv} + F_v\tilde{M}_u + F_u\tilde{M}_v + F_{uv}\tilde{M}) \; - \; (E_{uv}\tilde{N} + E_u\tilde{N}_v + E_v\tilde{N}_u + E\tilde{N}_{uv}) \\ & -(G_{uv}\tilde{L} + G_u\tilde{L}_v + G_v\tilde{L}_u + G\tilde{L}_{uv}) \end{aligned} \tag{A.50}$$

$$\begin{aligned} B_{vv} = {} & 2(F\tilde{M}_{vv} + 2F_v\tilde{M}_v + F_{vv}\tilde{M}) \; - \; (E_{vv}\tilde{N} + 2E_v\tilde{N}_v + E\tilde{N}_{vv}) \\ & -(G_{vv}\tilde{L} + 2G_v\tilde{L}_v + G\tilde{L}_{vv}) \end{aligned} \tag{A.51}$$

The Gaussian curvature

$$K = \frac{det[\Delta]}{det[\Gamma]} = \frac{LN - M^2}{EG - F^2} = \frac{\tilde{L}\tilde{N} - \tilde{M}^2}{S^4} = \frac{A}{S^4} \tag{A.52}$$

$$K_u = \frac{A_u S^2 - 4(\mathbf{S} \cdot \mathbf{S}_u)A}{S^6} = \frac{\hat{A}}{S^6} \tag{A.53}$$

$$K_v = \frac{A_v S^2 - 4(\mathbf{S} \cdot \mathbf{S}_v)A}{S^6} = \frac{\bar{A}}{S^6} \tag{A.54}$$

$$K_{uu} = \frac{\hat{A}_u S^2 - 6(\mathbf{S} \cdot \mathbf{S}_u)\hat{A}}{S^8} \tag{A.55}$$

$$K_{uv} = \frac{\hat{A}_v S^2 - 6(\mathbf{S} \cdot \mathbf{S}_v)\hat{A}}{S^8} \tag{A.56}$$

$$K_{vv} = \frac{\bar{A}_v S^2 - 6(\mathbf{S} \cdot \mathbf{S}_v)\bar{A}}{S^8} \tag{A.57}$$

where

$$\hat{A} = A_u S^2 - 4(\mathbf{S} \cdot \mathbf{S}_u)A \tag{A.58}$$

$$\hat{A}_u = A_{uu} S^2 - 2A_u(\mathbf{S} \cdot \mathbf{S}_u) - 4(\mathbf{S}_u \cdot \mathbf{S}_u + \mathbf{S} \cdot \mathbf{S}_{uu})A \tag{A.59}$$

$$\hat{A}_v = A_{uv} S^2 + 2A_u(\mathbf{S} \cdot \mathbf{S}_v) - 4(\mathbf{S}_u \cdot \mathbf{S}_v + \mathbf{S} \cdot \mathbf{S}_{uv})A - 4(\mathbf{S} \cdot \mathbf{S}_u)A_v \tag{A.60}$$

$$\bar{A} = A_v S^2 - 4(\mathbf{S} \cdot \mathbf{S}_v)A \tag{A.61}$$

$$\bar{A}_v = A_{vv}S^2 - 2A_v(\mathbf{S}\cdot\mathbf{S}_v) - 4(\mathbf{S}_v\cdot\mathbf{S}_v + \mathbf{S}\cdot\mathbf{S}_{vv})A \tag{A.62}$$

$$\tag{A.63}$$

The mean curvature

$$H = \frac{2FM - EN - GL}{2(EG - F^2)} = \frac{2F\tilde{M} - E\tilde{N} - G\tilde{L}}{2S^3} = \frac{B}{2S^3} \tag{A.64}$$

$$H_u = \frac{B_uS^2 - 3(\mathbf{S}\cdot\mathbf{S}_u)B}{2S^5} = \frac{\hat{B}}{2S^5} \tag{A.65}$$

$$H_v = \frac{B_vS^2 - 3(\mathbf{S}\cdot\mathbf{S}_v)B}{2S^5} = \frac{\bar{B}}{2S^5} \tag{A.66}$$

$$H_{uu} = \frac{\hat{B}_uS^2 - 5(\mathbf{S}\cdot\mathbf{S}_u)\hat{B}}{2S^7} \tag{A.67}$$

$$H_{uv} = \frac{\hat{B}_vS^2 - 5(\mathbf{S}\cdot\mathbf{S}_v)\hat{B}}{2S^7} \tag{A.68}$$

$$H_{vv} = \frac{\bar{B}_vS^2 - 5(\mathbf{S}\cdot\mathbf{S}_v)\bar{B}}{2S^7} \tag{A.69}$$

where

$$\hat{B} = B_uS^2 - 3(\mathbf{S}\cdot\mathbf{S}_u)B \tag{A.70}$$

$$\hat{B}_u = B_{uu}S^2 - B_u(\mathbf{S}\cdot\mathbf{S}_u) - 3(\mathbf{S}_u\cdot\mathbf{S}_u + \mathbf{S}\cdot\mathbf{S}_{uu})B \tag{A.71}$$

$$\hat{B}_v = B_{uv}S^2 + 2.0B_u(\mathbf{S}\cdot\mathbf{S}_v) - 3(\mathbf{S}_u\cdot\mathbf{S}_v + \mathbf{S}\cdot\mathbf{S}_{uv})B - 3(\mathbf{S}\cdot\mathbf{S}_u)B_v \tag{A.72}$$

$$\bar{B} = B_vS^2 - 3(\mathbf{S}\cdot\mathbf{S}_v)B \tag{A.73}$$

$$\bar{B}_v = B_{vv}S^2 - B_v(\mathbf{S}\cdot\mathbf{S}_v) - 3(\mathbf{S}_v\cdot\mathbf{S}_v + \mathbf{S}\cdot\mathbf{S}_{vv})B \tag{A.74}$$

The principal curvatures

$$\kappa = H \pm \sqrt{H^2 - K} = \frac{B \pm \sqrt{B^2 - 4AS^2}}{2S^3} \tag{A.75}$$

$$\kappa_u = \frac{2H_u\kappa - K_u}{2(\kappa - H)} \tag{A.76}$$

$$= \frac{(BB_u - 2A_uS^2)S^2 + (8AS^2 - 3B^2)(\mathbf{S}\cdot\mathbf{S}_u) \pm (B_uS^2 - 3(\mathbf{S}\cdot\mathbf{S}_u)B)\sqrt{B^2 - 4AS^2}}{2S^5\sqrt{B^2 - 4AS^2}} \tag{A.77}$$

$$\kappa_v = \frac{2H_v\kappa - K_v}{2(\kappa - H)} \tag{A.78}$$

$$= \frac{(BB_v - 2A_vS^2)S^2 + (8AS^2 - 3B^2)(\mathbf{S}\cdot\mathbf{S}_v) \pm (B_vS^2 - 3(\mathbf{S}\cdot\mathbf{S}_v)B)\sqrt{B^2 - 4AS^2}}{2S^5\sqrt{B^2 - 4AS^2}} \tag{A.79}$$

$$\kappa_{uu} = \frac{2H_{uu}\kappa + 4H_u\kappa_u - 2\kappa_u^2 - K_{uu}}{2(\kappa - H)} \tag{A.80}$$

$$\kappa_{uv} = \frac{2H_{uv}\kappa + 2(H_u\kappa_v + H_v\kappa_u) - 2\kappa_u\kappa_v - K_{uv}}{2(\kappa - H)} \tag{A.81}$$

$$\kappa_{vv} = \frac{2H_{vv}\kappa + 4H_v\kappa_v - 2\kappa_v^2 - K_{vv}}{2(\kappa - H)} \tag{A.82}$$

The root mean curvature

$$\kappa_{rms} = \sqrt{\kappa_{max}^2 + \kappa_{min}^2} = \sqrt{4H^2 - 2K} = \frac{\sqrt{B^2 - 2AS^2}}{S^3} \tag{A.83}$$

$$\frac{\partial \kappa_{rms}}{\partial u} = \frac{4HH_u - K_u}{\sqrt{4H^2 - 2K}} = \frac{(BB_u - A_uS^2)S^2 + (4AS^2 - 3B^2)\mathbf{S}\cdot\mathbf{S}_u}{S^5\sqrt{B^2 - 2AS^2}} \tag{A.84}$$

$$\frac{\partial \kappa_{rms}}{\partial v} = \frac{4HH_v - K_v}{\sqrt{4H^2 - 2K}} = \frac{(BB_v - A_vS^2)S^2 + (4AS^2 - 3B^2)\mathbf{S}\cdot\mathbf{S}_v}{S^5\sqrt{B^2 - 2AS^2}} \tag{A.85}$$

$$\frac{\partial^2 \kappa_{rms}}{\partial u^2} = \frac{8(H_u^2 + HH_{uu} - K_{uu})(2H^2 - K) - (4HH_u - K_u)^2}{(4H^2 - 2K)^{\frac{3}{2}}} \tag{A.86}$$

$$\frac{\partial^2 \kappa_{rms}}{\partial u \partial v} = \frac{8(H_uH_v + HH_{uv} - K_{uv})(2H^2 - K) - (4HH_u - K_u)(4HH_v - K_v)}{(4H^2 - 2K)^{\frac{3}{2}}} \tag{A.87}$$

$$\frac{\partial^2 \kappa_{rms}}{\partial v^2} = \frac{8(H_v^2 + HH_{vv} - K_{vv})(2H^2 - K) - (4HH_v - K_v)^2}{(4H^2 - 2K)^{\frac{3}{2}}} \tag{A.88}$$

Appendix B

Classification of Stationary Points of Functions

In this appendix we review some relevant material from the extrema theory of functions necessary in the classification of stationary points of curvature [33].

Single Variable: Let $f(x)$ be a continuous, sufficiently differentiable, function of one variable x, then a necessary condition that f is a maximum or a minimum at $x = a$ is

$$f'(x) = 0 \;\; at \;\; x = a \tag{B.1}$$

The function $f(x)$ has a maximum, a minimum or neither maximum nor minimum according to

- if $f''(a) < 0$; maximum
- if $f''(a) > 0$; minimum
- if $f''(a) = 0$;
  - if $f'''(a) \neq 0$; neither maximum nor minimum
  - if $f'''(a) = 0$ and $f^{(iv)}(a) < 0$; maximum
  - if $f'''(a) = 0$ and $f^{(iv)}(a) > 0$; minimum

In general, if $f^{(n)}(a)$ is the first derivative function that does not vanish then

- if n is odd; neither maximum nor minimum;

- if n is even;
  - if $f^{(n)}(a) < 0$; maximum
  - if $f^{(n)}(a) > 0$; minimum

Two Variables: Let $f(x, y)$ be a continuous function of two variables x and y. A necessary condition that f has an extremum at (x_0, y_0) is

$$f_x = f_y = 0 \quad at \quad (x_0, y_0) \tag{B.2}$$

Let H^* denote the Hessian matrix $\begin{pmatrix} f_{xx} & f_{xy} \\ f_{xy} & f_{yy} \end{pmatrix}$, then $f(x, y)$ has a maximum, a minimum or a saddle point according to

- if $f_{xx} < 0$ and $det[H^*] > 0$ at (x_0, y_0) : maximum
- if $f_{xx} > 0$ and $det[H^*] > 0$ at (x_0, y_0) : minimum
- if $det[H^*] < 0$ at (x_0, y_0) : saddle point
- if $det[H^*] = 0$: higher-order partial derivatives must be considered

For degenerate case *i.e.* $det[H^*] = 0$, there is a theorem by Scheeffer to classify the extrema, see [33].

Appendix C

Stationary Condition for Mean Curvature

Assertion

If the parametric surface is represented in Monge form, $\mathbf{r} = (x, y, z)^T = [x,\, y,\, h(u(x,y),\, v(x,y))]^T$, as we discussed in section 7.3, and its third order partial derivatives h_{xxx}, h_{xxy}, h_{xyy}, h_{yyy} are all zero, then $\nabla H = (H_u, H_v) = \mathbf{0}$, where H is the mean curvature.

Proof : First we start with evaluating the coefficients of first and second fundamental forms of the surface in Monge form. Since the point on the surface in Monge form is given by equation (7.36), it is straightforward to evaluate.

$$E = 1 + h_x^2, \quad F = h_x h_y, \quad G = 1 + h_y^2 \tag{C.1}$$

$$L = \frac{h_{xx}}{\sqrt{1 + h_x^2 + h_y^2}}, \quad M = \frac{h_{xy}}{\sqrt{1 + h_x^2 + h_y^2}}, \quad N = \frac{h_{yy}}{\sqrt{1 + h_x^2 + h_y^2}} \tag{C.2}$$

and their first order partial derivatives with respect to x.

$$E_x = 2h_x h_{xx}, \quad F_x = h_{xx} h_y + h_x h_{xy}, \quad G_x = 2h_y h_{xy} \tag{C.3}$$

$$L_x = \frac{h_{xxx}\sqrt{(1 + h_x^2 + h_y^2)} + h_{xx}(1 + h_x^2 + h_y^2)^{-\frac{3}{2}}(h_x h_{xx} + h_y h_{xy})}{1 + h_x^2 + h_y^2}$$

$$M_x = \frac{h_{xxy}\sqrt{(1 + h_x^2 + h_y^2)} + h_{xx}(1 + h_x^2 + h_y^2)^{-\frac{3}{2}}(h_x h_{xx} + h_y h_{xy})}{1 + h_x^2 + h_y^2}$$

$$N_x = \frac{h_{xyy}\sqrt{(1+h_x^2+h_y^2)} + h_{xx}(1+h_x^2+h_y^2)^{-\frac{3}{2}}(h_x h_{xx} + h_y h_{xy})}{1+h_x^2+h_y^2} \tag{C.4}$$

Now we will differentiate equation (5.20) with respect to x

$$\begin{aligned} H_x &= \frac{(2F_x M + 2FM_x - E_x N - EN_x - G_x L - GL_x)}{2(EG-F^2)} \\ &- \frac{(2FM - EN - GL)(E_x G + EG_x - 2FF_x)}{2(EG-F^2)^2} \end{aligned} \tag{C.5}$$

Since the surface is in Monge form, $h_x = h_y = 0$, which leads to $E_x = F_x = G_x = 0$. Consequently if $h_{xxx} = h_{xxy} = h_{xyy} = 0$, then $L_x = M_x = N_x = 0$ and hence $H_x = 0$. Similarly we can say that if $h_{xxy} = h_{xyy} = h_{yyy} = 0$, then $H_y = 0$. Since H_u and H_v can be written as

$$\begin{aligned} H_u &= H_x x_u + H_y y_u \\ H_v &= H_x x_v + H_y y_v \end{aligned} \tag{C.6}$$

We can conclude that if $h_{xxx} = h_{xxy} = h_{xyy} = h_{yyy} = 0$ then $H_u = H_v = 0$.

Bibliography

[1] P. G. Alourdas. Shape creation, interrogation and fairing using B-splines. Engineer's thesis, Massachusetts Institute of Technology, Department of Ocean Engineering, Cambridge, Massachusetts, 1989.

[2] P. G. Alourdas, G. R. Hottel, and S. T. Tuohy. A design and interrogation system for modeling with rational splines. In S. K. Chakrabarti et al., editors, *Proceedings of the Ninth International Symposium on OMAE*, volume I, Part B, pages 555–565, Houston, Texas, February 1990. New York: ASME, 1990.

[3] T. Banchoff, T. Gaffney, and C. McCrory. *Cusps of Gauss Mappings.* Pittman, Boston, 1982.

[4] J. M. Beck, R. T. Farouki, and J. K. Hinds. Surface analysis methods. *IEEE Computer Graphics and Applications*, 6(12):18–36, December 1986.

[5] M. V. Berry and J. H. Hannay. Umbilic points on Gaussian random surfaces. *Journal of Physics A.*, 10(11):1809–1821, 1977.

[6] P. J. Besl. *Surfaces in Range Image Understanding.* Springer-Verlag, New York, 1988.

[7] C. Bliek. *Computer Methods for Design Automation.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, July 1992.

[8] M. Brady, J. Ponce, A. Yuille, and H. Asada. Describing surfaces. *Computer Vision, Graphics and Image Processing*, 32(1):1–28, October 1985.

[9] B. Buchberger. Gröbner bases: An algorithmic method in polynomial ideal theory. In N. K. Bose, editor, *Multidimensional Systems Theory: Progress, Directions and Open Problems in Multidimensional Systems*, pages 184–232, 1985. Dordrecht, Holland: D. Reidel Publishing Company.

[10] J. Canny. Generalized characteristic polynomials. *Journal of Symbolic Computation*, 9:241–250, 1990.

[11] C. S. Chiang, C. M. Hoffmann, and R. E. Lynch. How to compute offsets without self-intersection. In M. J. Silbermann and D. Tagare, editors, *Proceedings of The International Society for Optical Engineering Volume 1610, Curves and Surfaces in Computer Vision and Graphics II*, pages 76–87. SPIE, Boston, Massachusetts, 1991.

[12] B. K. Choi and C. S. Jun. Ball-end cutter interference avoidance in NC machining of sculptured surfaces. *Computer Aided Design*, 21(6):371–378, 1989.

[13] G. Dahlquist and A. Björck. *Numerical Methods*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1974.

[14] G. Darboux. *Leçons Sur la Théorie Générale des Surfaces Vol.4*. Gauthier-Villars, Paris, 1896.

[15] J. C. Dill. An application of color graphics to the display of surface curvature. *ACM Computer Graphics*, 15(3):153–161, August 1981.

[16] P. M. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1976.

[17] T. Duff. Interval arithmetic and recursive subdivision for implicit functions and constructive solid geometry. *ACM Computer Graphics*, 26(2):131–138, July 1992.

[18] D. Dutta and C. M. Hoffmann. A geometric investigation of the skeleton of CSG objects. In B. Ravani, editor, *Proceedings of the 16th ASME Design Automation Conference: Advances in Design Automation, Computer Aided and Computational Design*, volume I, pages 67–75, Chicago, IL, September 1990. New York: ASME, 1990.

[19] D. Dutta, R. R. Martin, and M. J. Pratt. Cyclides in surface and soild modeling. *IEEE Computer Graphics and Applications*, 13(1):53–59, January 1993.

[20] G. Elber and E. Cohen. Second-order surface analysis using hybrid symbolic and numeric operators. *ACM Transactions on Graphics*, 12(2):160–178, April 1993.

[21] W. Enger. Interval ray tracing - a divide and conquer strategy for realistic computer graphics. *The Visual Computer*, 9(2):91–104, November 1992.

[22] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide.* Academic Press, San Diego, CA, 3rd edition, 1993.

[23] R. T. Farouki. The approximation of non-degenerate offset surfaces. *Computer Aided Geometric Design*, 3(1):15–43, 1986.

[24] R. T. Farouki. Pythagorean-hodograph curves in practical use. In R. E. Barnhill, editor, *Geometry Processing for Design and Manufacturing*, pages 3–33. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992. Chapter 1.

[25] R. T. Farouki and C. A. Neff. Algebraic properties of plane offset curves. *Computer Aided Geometric Design*, 7(1 - 4):101–127, 1990.

[26] R. T. Farouki and C. A. Neff. Analytic properties of plane offset curves. *Computer Aided Geometric Design*, 7(1 - 4):83–99, 1990.

[27] R. T. Farouki and V. T. Rajan. On the numerical condition of polynomials in Bernstein form. *Computer Aided Geometric Design*, 4:191–216, March 1987.

[28] I. D. Faux and M. J. Pratt. *Computational Geometry for Design and Manufacture.* Ellis Horwood, Chichester, England, 1981.

[29] C. B. Garcia and W. I. Zangwill. Global continuation methods for finding all solutions to polynomial systems of equations in n variables. In A. V. Fiacco and K. O. Kortanek, editors, *Extremal Methods and Systems Analysis*, pages 481–497. Springer-Verlag, New York, NY, 1980.

[30] A. Geisow. *Surface Interrogations.* PhD thesis, School of Computing Studies and Accountancy, University of East Anglia, Norwich NR47TJ, U. K., July 1983.

[31] H. N. Gursoy. *Shape Interrogation by Medial Axis Transform for Automated Analysis.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, November 1989.

[32] G. D. Hager. Constraint solving methods and sensor-based decision making. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, pages 1662–1667. IEEE, 1992.

[33] H. Hancock. *Theory of Maxima and Minima.* Dover, New York, 1960.

[34] E. Hansen and S. Sengupta. Bounding solutions of systems of equations using interval analysis. *BIT*, 21:201–211, 1981.

[35] M. Held. *On the Computational Geometry of Pocket Machining*. Springer-Verlag, Berlin, Germany, 1991.

[36] D. Hilbert and S. Cohn-Vossen. *Geometry and the Imagination*. Chelsea, New York, 1952.

[37] K. Ho-Le. Finite element mesh generation methods: A review and classification. *Computer Aided Design*, 20(1):27–38, 1988.

[38] C. M. Hoffmann. Algebraic and numerical techniques for offsets and blends. In W. Dahmen et al., editors, *Computation of Curves and Surfaces*, pages 499–528, Netherlands, 1990. Kluwer Academic Publishers.

[39] C. M. Hoffmann. A dimensionality paradigm for surface interrogation. *Computer Aided Geometric Design*, 7:517–532, 1990.

[40] J. Hoschek. Offset curves in the plane. *Computer Aided Design*, 17(2):77–82, 1985.

[41] J. Hoschek. Spline approximation of offset curves. *Computer Aided Geometric Design*, 5(1):33–40, June 1988.

[42] H. T. Jessop and F. C. Harris. *Photoelasticity, Principles and Methods*. New York: Dover Publications, 1950.

[43] R. B. Kearfott. Abstract generalized bisection and a cost bound. *Mathematics of Computation*, 49(179):187–202, July 1987.

[44] R. B. Kearfott. Some tests of generalized bisection. *ACM Transaction on Mathematical Software*, 13(3):197–220, September 1987.

[45] R. B. Kearfott. Interval arithmetic techiques in the computational solution of nonlinear systems of equations: Introduction, examples, and comparisons. *Lectures in Applied Mathematics*, 26:337–357, 1990.

[46] R. B. Kearfott. Decomposition of arithmetic expressions to improve the behavior of interval iteration for nonlinear systems. *Computing*, 47:169–191, 1991.

[47] R. B. Kearfott and M. Novoa. INTBIS, a portable interval Newton/bisection package (algorithm 681). *ACM Transactions on Mathematical Software*, 16(2):152–157, June 1990.

[48] R. Klass. An offset spline approximation for plane cubic splines. *Computer Aided Design*, 15(5):297–299, September 1983.

[49] J. J. Koenderink. *Solid Shape*. MIT Press, Cambridge, MA, 1990.

[50] G. A. Kriezis, N. M. Patrikalakis, and F.-E. Wolter. Topological and differential equation methods for surface intersections. *Computer Aided Design*, 24(1):41–55, January 1992.

[51] G. A. Kriezis, P. V. Prakash, and N. M. Patrikalakis. A method for intersecting algebraic surfaces with rational polynomial patches. *Computer Aided Design*, 22(10):645–654, December 1990.

[52] T. Kuragano, N. Sasaki, and A. Kikuchi. The FRESDAM system for designing and manufacturing freeform objects. In R. Martin, editor, *USA-Japan Cross Bridge. Flexible Automation Volume 2*, pages 931–938, 1988.

[53] J. M. Lane and R. F. Riesenfeld. Bounds on a polynomial. *BIT: Nordisk Tidskrift for Informations-Behandling*, 21(1):112–117, 1981.

[54] D. Lasser. Calculating the self-intersections of Bézier curves. *Computers in Industry*, 12:259–268, 1989.

[55] J. D. Lawrence. *A Catalogue of Special Plane Curves*. Dover Publications, Inc., New York, 1972.

[56] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems*. Prentice Hall, Inc., Englewood Cliffs, NJ, 1974.

[57] Y. Lee and T. Chang. CASCAM - an automated system for sculptured surface cavity machining. *Computers in Industry*, 16:321–342, 1991.

[58] T. Lozano-Perez and M. A. Wesley. An algorithm for planning collision-free paths amongst polyhedral obstacles. *Communications of the ACM*, 25(9):560–570, October 1979.

[59] T. Maekawa and N. M. Patrikalakis. Interrogation of differential geometry properties for design and manufacture. *The Visual Computer*, May 1992. Revised April 1993. To appear.

[60] T. Maekawa and N. M. Patrikalakis. Computation of singularities and intersections of offsets of planar curves. *Computer Aided Geometric Design*, September, 1992, Revised: February 1993. To appear.

[61] T. Maekawa, F.-E. Wolter, and N. M. Patrikalakis. Umbilics and lines of curvature for shape interrogation. Design Laboratory Memorandum 93-4, MIT Department of Ocean Engineering, Cambridge, MA, April 1993.

[62] D. Manocha. Solving polynomial systems for curve, surface and solid modeling. In J. Rossignac, J. Turner, and G. Allen, editors, *Proceedings of 2nd ACM/IEEE Symposium on Solid Modeling and Applications*, pages 169–178, Montreal, May 1993. New York: ACM Press, 1993.

[63] R. R. Martin. Principal patches - a new class of surface patch based on differential geometry. In P. J. W. Ten Hagen, editor, *Eurographics '83, Proceedings of the 4th Annual European Association for Computer Graphics Conference and Exhibition, Zagreb, Yugoslavia*, pages 47–55. Amsterdam: North-Holland, 31 August - 2 September 1983.

[64] R. E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, NJ, 1966.

[65] S. P. Mudur and P.A. Koparkar. Interval methods for processing geometric objects. *IEEE Computer Graphics and Applications*, 4(2):7–17, February 1984.

[66] F. C. Munchmeyer. On surface imperfections. In R. Martin, editor, *Mathematics of Surfaces II*, pages 459–474. Oxford University Press, 1987.

[67] F. C. Munchmeyer. Shape interrogation: A case study. In G. Farin, editor, *Geometric Modeling*, pages 291–301. SIAM, Philadelphia, PA, 1987.

[68] A. Neumaier. *Interval Methods for Systems of Equations*. Cambridge University Press, Cambridge, 1990.

[69] T. Nishita, T. W. Sederberg, and M. Kakimoto. Ray tracing trimmed rational surface patches. *ACM Computer Graphics*, 24(4):337–345, August 1990.

[70] M. F. Nittel. Numerically controlled machining of propeller blades. *Marine Technology*, 26(3):202–209, July 1989.

[71] Numerical Algorithms Group, Oxford, England. *NAG Fortran Library Manual, Volumes 1-8*, mark 15 edition, 1991.

[72] N. O. Olesten. *Numerical Control*. Wiley-Interscience, 1970.

[73] N. M. Patrikalakis. Surface-to-surface intersections. *IEEE Computer Graphics and Applications*, 13(1):89–95, January 1993.

[74] N. M. Patrikalakis and L. Bardis. Offsets of curves on rational B-spline surfaces. *Engineering with Computers*, 5:39–46, 1989.

[75] N. M. Patrikalakis and L. Bardis. Localization of rational B-spline surfaces. *Engineering with Computers*, 7(4):237–252, 1991.

[76] N. M. Patrikalakis and H. N. Gursoy. Shape interrogation by medial axis transform. In B. Ravani, editor, *Proceedings of the 16th ASME Design Automation Conference: Advances in Design Automation, Computer Aided and Computational Design, Vol. I*, pages 77–88, Chicago, IL, September 1990. New York: ASME.

[77] N. M. Patrikalakis, T. Maekawa, E. C. Sherbrooke, and J. Zhou. Computation of singularities for engineering design. In T. L. Kunii and Y. Shinagawa, editors, *Modern Geometric Computing for Visualization*, pages 167–191. Tokyo: Springer-Verlag, June 1992.

[78] N. M. Patrikalakis and P. V. Prakash. Surface intersections for geometric modeling. *Journal of Mechanical Design, ASME Transactions*, 112(1):100–107, March 1990.

[79] H. Persson. NC machining of arbitrarily shaped pockets. *Computer-Aided Design*, 10(3):169–174, May 1978.

[80] B. Pham. Offset curves and surfaces: a brief survey. *Computer Aided Design*, 24(4):223–229, April 1992.

[81] I. R. Porteous. Ridges and umbilics of surfaces. In R. Martin, editor, *The Mathematics of Surfaces II*, pages 447–458. Oxford University Press, 1987.

[82] I. R. Porteous. The circles of a surface. In R. Martin, editor, *The Mathematics of Surfaces III*, pages 135–143. Oxford University Press, 1988.

[83] P. V. Prakash and N. M. Patrikalakis. Surface-to-surface intersections for geometric modeling. Technical Report MITSG 88-8, Cambridge, MA: MIT Sea Grant College Program, 1988.

[84] M. J. Pratt. Cyclides in computer aided geometric design. *Computer Aided Geometric Design*, 7(1 - 4):221–242, 1990.

[85] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, 1985.

[86] A. Preusser. Computing area filling contours for surface defined by piecewise polynomials. *Computer Aided Geometric Design*, 3:267–279, 1986.

[87] A. Rockwood, K. Heaton, and T. Davis. Real-time rendering of trimmed surfaces. *ACM Computer Graphics*, 23(3):107–116, July 1989.

[88] J. R. Rossignac and A. G. Requicha. Offsetting operations in solid modelling. *Computer Aided Geometric Design*, 3(2):129–148, 1986.

[89] P. T. Sander and S. W. Zucker. Singularities of principal direction fields from 3-D images. In *IEEE Second International Conference on Computer Vision, Tampa Florida*, pages 666–670, 1988.

[90] T. W. Sederberg. Algorithms for algebraic curve intersection. *Computer Aided Design*, 21(9):547–554, November 1989.

[91] T. W. Sederberg and D. B. Buehler. Offsets of polynomial Bézier curves: Hermite approximation with error bounds. In T. Lyche and L. L. Schumaker, editors, *Mathematical Methods in Computer Aided Geometric Design*, volume II, pages 549–558. Academic Press, 1992.

[92] T. W. Sederberg and R. T. Farouki. Approximation by interval Bézier curves. *IEEE Computer Graphics and Applications*, 12(5):87–95, September 1992.

[93] E. C. Sherbrooke and N. M. Patrikalakis. Computation of the solutions of nonlinear polynomial systems. *Computer Aided Geometric Design*, October 1991, Revised: February, 1993. To appear.

[94] K. Shimada and D. C. Gossard. Computational methods for physically-based FE mesh generation. In *Proceedings of the IFIP TC5/WG5.3 8th International Conference on PROLAMAT '92*, Tokyo, June 22-24, 1992.

[95] S. S. Sinha and P. J. Besl. Principal patches a viewpoint-invariant surface description. In *IEEE International Robotics and Automation, Cincinnati, Ohio*, pages 226–231, May 1990.

[96] J. M. Snyder. Interval analysis for computer graphics. *ACM Computer Graphics*, 26(2):121–130, July 1992.

[97] M. Spivak. *Calculus*. New York: W. A. Benjamin, Inc., 1967.

[98] D. J. Struik. *Lectures on Classical Differential Geometry.* Addison-Wesley, Cambridge Mass., 1950.

[99] Symbolics, Inc. *MACSYMA Reference Manual*, version 13 edition, November 1988.

[100] W. Tiller and E. G. Hanson. Offsets of two-dimensional profiles. *IEEE Computer Graphics and Applications*, 4(9):36–46, September 1984.

[101] D. Toth. On ray tracing parametric surfaces. *ACM Computer Graphics*, 19(3):171–179, July 1985.

[102] S. T. Tuohy. A comparison of ship surface design methods. Master's thesis, University of New Orleans, Department of Naval Architecture and Marine Engineering, May 1988.

[103] S. T. Tuohy, T. Maekawa, and N. M. Patrikalakis. Interrogation of geophysical maps with uncertainty for AUV micro-navigation. In *Engineering in Harmony with the Ocean, Proceedings of Oceans '93, Victoria, Canada.* IEEE Oceanic Engineering Society, October 1993. (To appear).

[104] S. T. Tuohy and N. M. Patrikalakis. Representation of geophysical maps with uncertainty. In N. M. Thalmann and D. Thalmann, editors, *Proceedings of the 11th Computer Graphics International Conference, CG International '93*, Lausanne, Switzerland, June 1993. Springer, Tokyo. (To appear).

[105] M. E. Vafiadou and N. M. Patrikalakis. Interrogation of offsets of polynomial surface patches. In F. H. Post and W. Barth, editors, *Eurographics '91, Proceedings of the 12th Annual European Association for Computer Graphics Conference and Exhibition*, pages 247–259 and 538, Vienna, Austria, September 1991. Amsterdam: North-Holland.

[106] R. J. Walker. *Algebraic Curves.* Princeton University Press, Princeton, New Jersey, 1950.

[107] J. H. Wilkinson. *Rounding Errors in Algebraic Processes.* H. M. Stationery Office, London, England, 1963.

[108] F.-E. Wolter. Cut locus and medial axis in global shape interrogation and representation. Memorandum 92-2, Cambridge MA: MIT Ocean Engineering Design Laboratory, January 1992.

[109] F. E. Wolter and S. T. Tuohy. Approximation of high degree and procedural curves. *Engineering with Computers*, 8(2):61–80, 1992.

[110] F. Yamaguchi. *Curves and Surfaces in Computer Aided Geometric Design*. Springer-Verlag, NY, 1988.

[111] F. Yamaguchi, K. Toshimitsu, H. Sato, and J. Nakagawa. An adaptive error-free computation based on the 4x4 determinant method. *The Visual Computer*, 9:173–181, January 1993.

[112] W. I. Zangwill and C. B. Garcia. *Pathways to solutions, fixed points, and equilibria*. Prentice-Hall, Englewood Cliffs, NJ, 1981.

[113] J. Zhou, E. C. Sherbrooke, and N. M. Patrikalakis. Computation of stationary points of distance functions. *Engineering with Computers*, June 1992. Revised March 1993. To appear.