ALGORITHMS FOR RATIONAL SPLINE SURFACE INTERSECTIONS

by

George Anthony Krieziş

B.S. in Naval Architecture and Marine Engineering, M.I.T., June 1986
M.S. in Naval Architecture and Marine Engineering, M.I.T., June 1986
M.S. in Ocean Systems Management, M.I.T., February 1990

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

March 1990

Copyright (c) 1990 Massachusetts Institute of Technology

Signature of Author ____________________________

Department of Ocean Engineering
March 1, 1990

Certified by ____________________________

Nicholas M. Patrikalakis
Thesis Supervisor

Accepted by ____________________________

A. Douglas Carmichael
Chairman, Departmental Committee on Graduate Students

MASSACHUSETTS INSTITUTE
OF TECHNOLOGY

JUL 18 1990
ALGORITHMS FOR RATIONAL SPLINE SURFACE INTERSECTIONS

by

George Anthony Kriezis

Submitted to the Department of Ocean Engineering on March 1, 1990 in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Abstract

The computation of intersections of surfaces is a fundamental problem in computational geometry and geometric modeling. This work addresses the development of highly reliable, accurate, efficient and automatic intersection algorithms. The application of interest is the computation of intersections of rational B-spline patches. Methods are developed for the automatic computation of features of the intersection set. Automatic computation of initial approximations to significant points (border, termination, turning, singular and collinear normal points) of an intersection are provided using a coarse approximation to the intersection and topological properties of plane vector fields defined using the gradient of the oriented distance function between two surfaces. Accurate computation of these points is subsequently performed using direct numerical techniques. The significant points of an intersection are used to partition the intersection domain into subdomains which can be processed independently.

A method is developed to verify the detection of all features of an intersection set. This uses a necessary and sufficient criterion based on convexity properties of the derivatives of the spline surfaces and subdivision. Tight bounds for the parametric derivatives and the normal vectors of the parametric surfaces are developed using rectangular pyramids to assist in the efficient implementation of loop detection.

Two methods are developed for tracing transversal intersection curves with a finite number of singular points. The first is based on a combination of adaptive subdivision / faceting and Newton techniques, while the second is based on tensorial differential equations which describe the intersection curve. The second method relies on a new characterization procedure for significant points which allows computation of the tangent directions to the intersection curve at such points. A class of non-transversal intersections with infinite singular points is also addressed using tensorial differential equations. Three algorithms based on these methods were developed, tested and compared in a large number of complex intersections with diverse features.

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

Αφερώνεται στους γονείς μου, Αντώνη και Φωφή, για την αγάπη τους και τη βοήθειά τους.

This thesis is dedicated to my parents Anthony and Fofi, who always sought the best for me and provided me with a stimulating environment to grow and learn.

I would like to express my gratitude to Professor N. M. Patrikalakis, my thesis supervisor and long time friend, for his support, encouragement and above all expertise. I would like also to thank the other members of my doctoral thesis committee Professor C. Chryssostomidis, Professor D. C. Gossard, Dr. P. V. Prakash and Dr. F.-E. Wolter for their advice during the development of this work.

I would also like to thank Mr M. S. Drooker, manager of the Design Laboratory and all my co-workers and friends at the Design Laboratory in the Department of Ocean Engineering for their support, help and patience. Special thanks should go to UROP students, E. C. Sherbrooke and T. N. Xanthopoulos for their assistance in the implementation of some of this work.

Last but not least, thanks must go to my friends near or distant and to my brothers and sisters, especially my twin brother Constantine with whom I had the pleasure of sharing accommodations during my academic life.

This work was supported in part by the MIT Sea Grant College Program, the National Science Foundation and the Office of Naval Research under grant numbers NA86AA-D-SG0089, DMC-8720720 and N00014-87-K-0462, respectively.
# Table of Contents

Abstract .................................................................................................................. 2  
Acknowledgements ................................................................................................. 3  
Table of Contents .................................................................................................... 4  
List of Figures ......................................................................................................... 7  
List of Tables .......................................................................................................... 11

1. INTRODUCTION ................................................................................................. 12

2. LITERATURE REVIEW ....................................................................................... 17  
   2.1 Analytic Methods .......................................................................................... 17  
   2.2 Lattice Evaluation ......................................................................................... 20  
   2.3 Marching Methods ......................................................................................... 21  
   2.4 Subdivision Methods .................................................................................... 24  
   2.5 Loop Detection Techniques ......................................................................... 27  
   2.6 Summary of Intersection Algorithm Developed at the M.I.T. Design Laboratory ............................................................... 29

3. FORMULATION OF SURFACE INTERSECTION PROBLEMS ....................... 31  
   3.1 Introduction .................................................................................................. 31  
      3.1.1 Parametric - Implicit .............................................................................. 31  
      3.1.2 Implicit - Implicit .................................................................................. 32  
      3.1.3 Parametric - Parametric ...................................................................... 32  
   3.2 Intersection Problem Formulation Using Distance Functions ..................... 32  
   3.3 Algorithmic Issues ....................................................................................... 38  
   3.4 Intersection Problems Addressed .................................................................. 40  
      3.4.1 Algebraic - Rational Polynomial Parametric Surface Intersection ......... 41  
          3.4.1.1 Governing Equations ..................................................................... 41  
          3.4.1.2 Direct Computation of Intersection Curve Equation .................... 43  
          3.4.1.3 Geometric Interpretation of Algebraic Curves in a Rectangular Domain ............................................................... 44  
      3.4.2 Rational Polynomial Surface Intersections ............................................ 45  
          3.4.2.1 Problem Formulation .................................................................. 45

4. CONVEXITY PROPERTY BOUNDS FOR B-SPLINES ......................................... 47  
   4.1 Tight Rectangular Bounding Box ................................................................. 47  
   4.2 Tight Bounds For First Parametric Partial Derivatives and Normal Vectors ... 47  
      4.2.1 Bounding Cones .................................................................................... 51  
      4.2.2 Bounding Pyramids .............................................................................. 54

5. SIGNIFICANT POINTS IN INTERSECTION PROBLEMS ................................ 65  
   5.1 Definitions .................................................................................................... 65  
      5.1.1 Algebraic Surface to Rational Parametric Surface .................................. 65  
      5.1.2 Rational Polynomial Surfaces ............................................................... 67  
   5.2 Initial Approximations to Significant Points in Intersections of NURBS Patches ........................................................................................................... 72  
      5.2.1 Turning and Singular Points .................................................................. 72
5.2.2 Border and Termination Points
5.2.3 Collinear Normal Points
5.3 Direct Numerical Solution Methods
  5.3.1 Constrained Minimization Techniques
  5.3.2 Newton Iteration
5.4 Examples
5.5 Algorithmic Issues

6. FEATURE DETECTION USING TOPOLOGY PROPERTIES OF THE VECTOR FIELD OF THE DISTANCE FUNCTION
6.1 Motivation
6.2 Results from the Theory of Plane Vector Fields
6.3 Verification Method
6.4 Applications

7. FEATURE DETECTION USING CONVEXITY PROPERTIES
  7.1 Introduction
  7.2 Loop Detection Using Convexity Properties of Rational Spline Surfaces
  7.3 Application of Feature Detection Techniques to Rational Spline Surface Intersections
  7.4 Examples
  7.5 Summary

8. TRACING USING SUBDIVISION AND NEWTON TECHNIQUES
  8.1 Partitioning Scheme for Intersection Domain
  8.2 Tracing Scheme
    8.2.1 Motivation
    8.2.2 Polyhedral Intersections
    8.2.3 Connection Phase
    8.2.4 High Accuracy Computation of Intersection Points
    8.2.5 Final Connection Phase
  8.3 Implementation Issues

9. TRACING USING TENSORIAL DIFFERENTIAL EQUATIONS
  9.1 Motivation
  9.2 Tensorial Differential Equations
  9.3 Marching from Singularities
  9.4 Surfaces Tangent Along General Curves
  9.5 Tracing Scheme
    9.5.1 Numerical Integration Scheme
    9.5.2 Algorithm 1
    9.5.3 Algorithm 2
  9.6 Comparison of Efficiency of Tracing Methods

10. INTERSECTION ALGORITHMS AND APPLICATIONS
  10.1 Summary of Intersection Algorithms
    10.1.1 Rational Polynomial Parametric - Implicit Algebraic Surface Intersection
    10.1.2 Rational Polynomial Parametric Surface Intersection
  10.2 Numerical Experiments
    10.2.1 Intersections of Algebraic Surfaces with Rational Biquadratic and Bicubic Patches
10.2.2 Algebraic Curves
10.3 Intersections of Rational Polynomial Patches

11. CONCLUSIONS AND RECOMMENDATIONS

Appendix A. DERIVATIVES FOR THE MINIMIZATION OBJECTIVE FUNCTIONS
   A.1 Partial Derivatives for Rational Surfaces
   A.2 Partial Derivatives of the Surface Normal
   A.3 Partial Derivatives of \( r_a/\|r_a\| \)
   A.4 Partial Derivatives of \( \|r - q\|^2 \)
   A.5 Partial Derivatives of Turning and Singular Point Objective Function
   A.6 Partial Derivatives of Collinear Normal Point Objective Function

Appendix B. DERIVATIVES OF THE DISTANCE FUNCTION AND TENSORIAL DIFFERENTIAL EQUATIONS
   B.1 Tensorial Differential Equations
   B.2 Partial Derivatives of the Oriented Distance Function

Appendix C. DISTANCE OF A POINT FROM A PARAMETRIC SURFACE

Appendix D. COMPUTATION OF SURFACE AREA ENCLOSED BY BOUNDING CONES AND RECTANGULAR PYRAMIDS ON THE UNIT SPHERE
List of Figures

Figure 3-1: Projection of a Point to A Surface 34
Figure 3-2: Examples with Ill-Defined Distance Function, Adapted from [Markot 89] 36
Figure 3-3: Common Types of Intersection Problems 41
Figure 4-1: Rectangular Box Overlap Test 51
Figure 4-2: Cone Bounding Curves of Constant v, Adapted from [Sederberg 88a] 52
Figure 4-3: Conservative Bounds from Incremental Creation of Bounding Circles 53
Figure 4-4: Conservative Bounds from Use of Bounding Circles 54
Figure 4-5: Bounding Pyramid Construction, Angle Determination 56
Figure 4-6: Bounding Pyramid Construction, Pyramid Definition 57
Figure 4-7: Bounding Pyramids and Cones for a Bezier Patch - Example 1 61
Figure 4-8: Bounding Pyramids and Cones for a Bezier Patch - Example 2 62
Figure 4-9: Bounding Pyramids and Cones for a Bezier Patch - Example 3 63
Figure 4-10: Bounding Pyramids and Cones for a Piece of a Torus - Example 4 64
Figure 5-1: Distance Function of Example 3 of Figure 3-2 Defined on the Swept Surface 70
Figure 5-2: Significant Points of an Intersection Curve 71
Figure 5-3: Starting Points For Significant Point Computation of Intersection of
\[ w = -6u^4 + 21u^3 - 19u^2 - 6u^2v^2 + 11uv^2 + 3v^2 - 4v^4 = 0 \] with the Plane w = 0. Two Levels of Initial Subdivision. 76
Figure 5-4: Starting Points For Significant Point Computation of Intersection Between Two Patches of Degree Six in Each Variable. One Level of Initial Subdivision. 77
Figure 5-5: Starting Points For Significant Point Computation for a Bicubic Bezier to Quadratic - Cubic Bezier Patch Intersection. One Level of Initial Subdivision. 78
Figure 5-6: Starting Points For Significant Point Computation for a Rational Biquadratic B-Spline to Bicubic Bezier Patch Intersection. One Level of Initial Subdivision. 79
Figure 5-7: Starting Points For Significant Point Computation of Intersection Between Two Biquartic Bezier Patches. Two Levels of Initial Subdivision. 80
Figure 5-8: Quadrilateral with Possibility of Collinear Normal Point in its Interior 84
Figure 5-9: Comparison of Approximate and Accurate Vector Fields of Distance Function Between Surfaces in Figure 10-12 87
Figure 5-10: Comparison of Approximate and Accurate Vector Fields of Distance Function Between Surfaces in Figure 10-14 88
Figure 5-11: Initial Approximations and Numerically Computed Collinear Normal Points for Surfaces in Figure 10-12 92
Figure 5-12: Initial Approximations and Numerically Computed Collinear Normal Points for Surfaces in Figure 10-14 93
Figure 5-13: Initial Approximations and Numerically Computed Collinear Normal Points for Surfaces in Figure 10-19 94
Figure 5-14: Initial Approximations and Numerically Computed Collinear Normal Points for Surfaces in Figure 10-11 95
Figure 7-11: Domain Subdivision for Feature Detection Condition Satisfaction for Surfaces in Figure 10-11. Use of Significant Points to Initially Split Domain.

Figure 7-12: Domain Subdivision for Feature Detection Condition Satisfaction for Surfaces in Figure 10-11. Binary Subdivision.

Figure 7-13: Domain Subdivision for Feature Detection Condition Satisfaction for Surfaces in Figure 10-14. Use of Significant Points to Initially Split Domain.

Figure 7-14: Domain Subdivision for Feature Detection Condition Satisfaction for Surfaces in Figure 10-14. Binary Subdivision.

Figure 7-15: Domain Subdivision for Feature Detection Condition Satisfaction for Surfaces in Figure 10-18. Binary Subdivision.

Figure 8-1: Partitioning Scheme and Tree Structure

Figure 8-2: Quadrilateral Intersections

Figure 8-3: Data Structures for Representation of Intersection Information

Figure 8-4: Starting Point Examples

Figure 8-5: Corner Resolution Scheme for Quadrilaterals on the Control Polyhedron

Figure 8-6: Types of Intersection Segments

Figure 8-7: Marching Directions for Newton Iteration

Figure 8-8: Procedure to Ensure Adequate Intersection Points

Figure 8-9: Final Connection Phase - One Step

Figure 9-1: Characterization of Significant Points of an Intersection Curve

Figure 10-1: Torus-Torus Intersection, Equal Radii.

Figure 10-2: Torus-Torus Intersection, Smaller Radii 50.0 and 49.99, Equal Larger Radii.

Figure 10-3: Torus-Bicubic Patch Intersection, Two Loops.

Figure 10-4: Cylindrical Section of a Propeller Blade Represented as a Bicubic B-Spline Patch.

Figure 10-5: Two-Sided Offset of Parabola \( y = x^2 = 0 \), by a Distance of 1.0.

Figure 10-6: Three Lines with a Triple Point.

Figure 10-7: Three Lines with a Triple Point.

Figure 10-8: Multiple Parametric Lines, One With Multiplicity 2.

Figure 10-9: Perturbation of Multiple Parametric Lines Equation With Line of Multiplicity 2 by 10\(^{-5}\).

Figure 10-10: Bicubic Patch with Plane Intersection.

Figure 10-11: Biquartic Patch with Plane Intersection.

Figure 10-12: Bicubic - Bicubic Patch Intersection - One Small Loop.

Figure 10-13: Bicubic - Biquadratic Patch Intersection.

Figure 10-14: Bicubic - Cubic / Quadratic Patch Intersection.

Figure 10-15: Quartic / Quadratic Bezier Patch - Plane Intersection.

Figure 10-16: Cylinder - Elliptic Paraboloid Intersection.

Figure 10-17: Intersection of Two Patches of Degree Six in Each Parameter - Three Loops.

Figure 10-18: Intersection of Two Biquartic Bezier Patches - Four Small Loops.

Figure 10-19: Intersection of Two Nearly Coincident Biquadratic Patches.

Figure 10-20: Intersection of a Bicubic Patch and a Torus.

Figure 10-21: Intersection of a Propeller Blade Represented as a Bicubic B-Spline Patch and a Propeller Hub Represented as a Rational Biquadratic Patch.

Figure 10-22: Torus-Torus Intersection, One Torus Translated by 0.01.

Figure 10-23: Cylinder - Cylinder Intersection, Equal Radii (2.5).
Figure 10-24: Cylinder-Cylinder Intersection, One Cylinder Translated by 0.001.

Figure 10-25: Biquartic - Biquartic Patch Intersection Involving Tangency Along a Curve.

Figure 10-26: Biquartic - Biquartic Patch Intersection Involving Tangency Along a Closed Loop.

Figure C-1: Distance of a Point From a Surface

Figure D-1: Definition of a Bounding Cone on a Unit Sphere

Figure D-2: Definition of a Spherical Triangle on a Unit Sphere
List of Tables

**Table 4-I:** Comparison of Surface Area on the Unit Sphere Enclosed by Rectangular Pyramids and Cones, Expressed as a Percentage of the Surface Area of the Sphere  
60

**Table 6-I:** Algorithm for Verification of Computation of Critical Points of the Oriented Distance Function Between Two Surfaces  
127

**Table 7-I:** Computational Times for the Feature Detection Test Using Binary Subdivision  
148

**Table 8-I:** Algorithm for Tracing of Intersection Curves Using Subdivision and Newton Methods  
176

**Table 9-I:** Algorithm 1 for Tracing the Intersection Curve Using Tensorial Differential Equations  
192

**Table 9-II:** Algorithm 2 for Tracing the Intersection Curve Using Tensorial Differential Equations  
194

**Table 9-III:** Computational Times for the Various Tracing Methods in the Parametric-Parametric Surface Intersection  
196

**Table 10-I:** Algorithm for NURBS / Algebraic Surface Intersection  
199

**Table 10-II:** Algorithm for NURBS / NURBS Intersection  
200

**Table 10-III:** Total Computational Times for Four Intersection Algorithms  
203
Chapter 1

INTRODUCTION

Interrogation of intersections of surfaces is a fundamental problem in computational geometry and geometric modeling of complex shapes in a computer environment. For anything but the simplest artifacts, objects must be combined to increase complexity. Such a combination of objects involves the use of set operations, where the bounding surfaces of the primitive objects are intersected with each other to determine the surface geometry of the new objects. This process, called boundary evaluation, is also used in the conversion of representations of solids using a Constructive Solid Geometry scheme to a Boundary Representation scheme [Requicha 82]. The fundamental step involved in such a process is to determine if pairs of bounding surfaces from the objects being combined intersect and if so to compute the intersection curve between them. These intersection curves, when properly trimmed by other bounding surfaces, form the boundaries of new trimmed surface patches, which, in turn, bound the solid being created. An explicit specification of the geometry of the new solid, therefore, requires complete knowledge of the intersection of the surfaces involved. Computation of intersections between surfaces is also needed in analysis such as in automatic finite element discretizations of three-dimensional solids, in feature recognition and in simulation of manufacturing processes such as tool path generation. Contouring of complex surfaces and sectional views of objects are some other general intersection applications in the visualization and interpretation of scientific data and the design of complex artifacts.

There is a significant body of literature addressing the problem of surface to surface intersections. The intersection of two surfaces can be very complicated, in general, with a number of closed loops, segments extending from boundary to boundary, common surface
segments, self-intersections, isolated points and other singularities. The ability to detect and describe all such features of the intersection set and to trace them correctly and in an efficient manner forms an important problem in this field as seen by the large volume of literature addressing the problem [Geisow 83, Sarraga 84, Solomon 85, Farouki 86a, Bajaj 87, Prakash 88a, Cheng 88, Patrikalakis 90a, Hoffmann 89, Markot 89].

A good surface-intersection technique has to balance three conflicting goals: efficiency, robustness and accuracy. Robustness is a key property of automated solution procedures in a computer environment. If such a property is not guaranteed, then the designer needs to manually ensure that the resulting approximations do not result in global inconsistencies of the geometric database. Global inconsistencies are very undesirable in a true unified computer environment supporting design, analysis and fabrication. The solutions resulting from present state-of-the-art techniques are further complicated by imprecisions introduced by numerical errors present in all finite precision computations [Farouki 87]. These imprecisions, may by themselves, also introduce global inconsistencies of geometric data and further exacerbate the quality of available techniques [Pratt 86]. The accuracy a numerical method can deliver varies with the local surface geometry at the intersection, the angle at which the surfaces intersect etc.

Efficiency is another important property of a surface to surface intersection algorithm. Intersections are frequently invoked, basic operations and as a result should be efficient to avoid delaying the design process significantly. There is a trade-off between robustness and efficiency in an intersection algorithm, since the larger the number of robustness enhancement computations the slower the execution of the algorithm even for simple examples.

This thesis attempts to address these three important factors. The objective was to develop automated algorithms for the computation of rational B-spline surface to rational B-spline surface intersections, which are robust in treating complex intersections and lead to intersection curve solutions of high accuracy at small computational expense.
The thesis is structured as follows.

Chapter 2 presents a comprehensive review of the large literature dealing with the computation of surface intersections. Some of the latest and most sophisticated algorithms are also described briefly in this Chapter. A detailed review of techniques developed for loop detection is also presented.

Chapter 3 classifies the various types of intersection problems, describes the important issues in the design of an algorithm for computing intersections and formulates the intersection problems addressed in this work. The oriented distance function between two parametric surfaces is defined and some of its properties are also described.

Chapter 4 presents new algorithms for the computation of tight bounds for the position, first parametric partial derivatives and the normal vectors of rational B-spline surface patches. These bounds are useful in subdivision methods for rapidly verifying absence of intersection between two surfaces before performing any detailed intersection computations, thus improving the efficiency of these computations. A new efficient algorithm for the detection of intersection of rectangular bounding boxes of the surfaces is also described.

Chapter 5 introduces the concept of significant points of an intersection set, describes novel methods for the automatic computation of initial approximations to these points and presents direct numerical techniques for the accurate computation of these points. One of the major contributions of this thesis is the development of techniques to automatically compute approximations to significant points of an intersection set, i.e. border, termination, turning, singular and collinear normal points, and then use numerical techniques to compute these points very accurately. These techniques are based on convexity properties of B-splines and the theory of plane vector fields, and have as objective the computation of starting points in all intersection branches, in order to improve the robustness of marching methods. In addition, these techniques identify the singular points which slow down
subdivision methods and, as a result, improve substantially their efficiency by reducing the level of subdivision required to accurately trace the various intersection branches.

Chapter 6 presents a new necessary condition for the identification of collinear normal points of two surfaces using topological properties of the vector field of the gradient of the oriented distance function defined on one of the surfaces and describes an algorithm for the efficient use of this condition.

Chapter 7 presents two similar necessary and sufficient conditions for the detection of all features of an intersection curve using convexity based bounds of the parametric partial derivatives of the two surfaces in terms of a set of rectangular pyramids and describes some results from an experimental implementation of these conditions. These conditions could assist in determination of initial approximations to singular points of the intersection set and in the verification of the result of the intersection computation to enhance the confidence in the results of an intersection algorithm.

Chapter 8 presents a new algorithm for tracing transversal intersections of two surfaces involving a finite number of significant points using a coarse subdivision and faceting technique in combination with a local Newton-type refinement of the intersection. The adaptive partitioning scheme for the intersection domain is also described and some implementation issues are addressed.

Chapter 9 presents a new method for tracing transversal intersections of two surfaces involving a finite number of significant points using tensorial differential equations. This method is also extended to trace non-transversal intersections between surfaces involving infinite singularities. Two algorithms are developed to implement this method. The primary step in these algorithms is the characterization of all computed significant points of the intersection to determine the marching directions from these points. A comparison of the efficiency of the marching and subdivision algorithms developed in this work is also performed.
Chapter 10 summarizes the various intersection algorithms developed, compares their efficiency and illustrates their properties in a large number of intersection examples chosen for their complexity and diversity.

Chapter 11 summarizes the contributions of this work and provides recommendations for further research in this area.

The thesis also includes the following appendices.

Appendix A presents the parametric partial derivatives for rational surfaces, the partial derivatives of the surface normal, and the partial derivatives of the objective functions used in minimization for the computation of significant points.

Appendix B derives and presents the derivatives of the oriented distance function and the tensorial differential equations which describe the intersection curve between two surfaces.

Appendix C presents a methodology for the computation of the minimum distance from a point to a parametric surface. This methodology uses coarse subdivision to determine initial approximations to minimum distance points and uses a minimization technique to compute these points.

Appendix D presents the mensuration formulas used to determine the surface area on the unit sphere, which is enclosed by a cone or rectangular pyramid bounding the first parametric partial derivatives and the normal vectors of surfaces.
Chapter 2

LITERATURE REVIEW

The computation of the intersection set between two surfaces is considered to be a difficult problem. Except for some simple surfaces such as quadrics, for which analytical methods may be used to simplify the numerical work, intersection problems require extensive numerical computations to solve the non-linear equations describing the intersection. Surface intersection methods can be classified in four main categories: analytic, lattice evaluation, marching and subdivision. Most of the methods have been developed in the context of polynomial surfaces. Many recent techniques also combine elements from more than one of the above categories leading to hybrid methods. Some of the characteristics of the various intersection methods will be presented next. Our discussion is essentially an expansion and update of a related summary compiled by [Patrikalakis 90b].

2.1 Analytic Methods

These methods rely on the derivation of a governing equation describing the intersection of two surfaces. For polynomial surfaces, the resulting equation is an algebraic curve which is an implicit polynomial in two variables. This equation can, in principle, be obtained by elimination of one Cartesian coordinate for the case of two implicit surfaces [Sederberg 84a] or by elimination of three Cartesian coordinates for the case of an implicit surface intersecting a parametric rational polynomial surface [Farouki 86a]. In the first case, an equation for the projection of the intersection curve on one coordinate plane is obtained and the inversion algorithm of [Sederberg 84b] is needed to compute the third coordinate of the three dimensional intersection curve. In the second case, the resulting
intersection equation is an implicit polynomial in the parameters of the patch. When the implicit surfaces of interest are actually bounded, (i.e. they are algebraic patches [Sederberg 85, Patrikalakis 89] or they resulted from implicitization of parametric patches [Hoffmann 89]), points which are found to satisfy the intersection equation within the space of one patch should be tested to verify if they lie on the appropriate portion of the other surface.

The intersection between two algebraic surfaces can also be found by converting the problem to the simpler problem of algebraic-rational polynomial parametric surface intersection when a rational polynomial parametrization of one algebraic surface is possible [Abhyankar 86a]. In this approach, either one of the algebraic surfaces is converted into its rational polynomial parametric (RPP) form or another parametric polynomial surface which contains the curve of intersection is found. This approach has been used in the intersection of two quadric surfaces [Levin 79, Sarraga 83, Chiang 89a] where a third parametric polynomial ruled surface, the parametrization surface, is found which also contains the intersection between the quadric surfaces. One of the quadric surfaces is then intersected with the parametrization surface to get the required intersection.

Similarly, the intersection between two parametric surfaces can be attempted by first obtaining a representation for their curve of intersection. Since all rational polynomial parametric surfaces have an algebraic surface representation [Sederberg 84b], we could, in theory, convert one of the two RPP surfaces to its algebraic representation and treat the problem as an algebraic surface/RPP surface intersection problem. The major problem involved here is that the algebraic representation of a patch with degree m, n in its two parameters is of degree 2mn which can be, relatively, high. The subsequent substitution of the second parametric surface (say of degrees p, q in its parameters u and v) in the algebraic surface will lead to an algebraic curve of very high degree (2mnp in u and 2mnq in v). For the case of intersections between rational biquadratic patches, we have a curve of degree 16 in each variable. Similarly, for rational bicubic patches, the resulting curve is of degree 54
in each variable. The high degree of the representation of the intersection curve discourages the use of methods relying on explicit representation for the computation of general RPP patch intersections.

In practical situations, once the equation describing the intersection curve between low degree algebraics and RPP patches is obtained as above, it must be traced. For special cases, the resulting implicit equations (algebraic curves) can be solved in terms of explicit expressions involving radicals [Levin 79] or in terms of rational parametric polynomials for algebraic curves with known singularities and whose genus is zero [Abhyankar 86b]. Once the range of the independent variable in such cases is determined, the above explicit equations can be used to trace the intersection curve. The local extrema and singular points of the curve can also be used to advantage as explained in [Sarraga 83]. However, for general cases explicit representation of algebraic curves in terms of elementary functions is impossible [Sederberg 84b].

Algebraic curves with integer coefficients can be analyzed using the cylindrical algebraic decomposition algorithm [Arnon 83], to provide a structure graph of the curve that defines pieces of the curve between significant points on the curve. The method separates the curve into a set of simple segments by creating parallel bands bounded by lines passing through singular points and curve extrema in one principal direction. Numerical methods (with or without rational arithmetic) are then used to step along each piece of the curve. This method, as implemented in rational arithmetic, although providing a guarantee that the solution is topologically reliable, is impractical, because of its very large memory requirements and poor efficiency. In addition, algebraic curves with integer coefficients are not general enough for geometric modeling, where simple rotation of intersecting primitives creates curves with possibly irrational or transcendental coefficients. The above considerations suggest the need to revert to other methods that trace general algebraic curves and in a more efficient manner. [Farouki 86a] uses the representation of
the algebraic curve in order to compute significant points of the curve and to initiate a marching method of tracing the curve as discussed later.

As can be seen in this thesis, availability of a representation facilitates computation of the significant points of the curve, as compared to cases where a representation is not explicitly available. Further, for trimmed patch processing it provides a complete and compact representation of the intersection in comparison to a linear approximation to the curve.

An alternative to the analytic techniques for solving quadric intersection problems are the geometric methods suggested by [Miller 87] and [Piegl 89]. These methods use geometric principles to aid in the computation of the intersections of quadric surfaces and are very accurate and efficient. However, they require different algorithms for every pair of surfaces, thereby increasing the effort of the implementation, and, unfortunately, are not generalizable in an obvious manner to treat more general intersections.

2.2 Lattice Evaluation

Lattice methods reduce the dimensionality of surface-to-surface intersection problems by computing intersections of a number of parametric curves of one surface with the other surface followed by connection of the resulting discrete intersection points to form different solution branches [Varady 83, Mortenson 85]. For intersections of parametric patches, the method reduces to the solution of a large number of independent systems of three nonlinear equations in three unknowns. For intersections of parametric polynomial patches with algebraic surfaces it reduces to the computation of the real roots of a large number of independent univariate polynomials within an interval. For univariate polynomials, no initial estimate of the solutions are required while for systems of nonlinear equations, available numerical techniques (such as Newton and minimization methods) require good initial approximations for convergence, an important disadvantage.
By definition of lattice methods, the reduction of the dimensionality of the problem involves an initial choice of grid resolution, which, in turn, may lead the method to miss important features of the solution, such as small loops and isolated points which reflect near tangency or tangency of intersecting surfaces. Finally, the second element of the method involves connection of discrete solution points to form solution branches. This, typically, requires determining adjacency on the basis of minimum mutual distance which may lead to incorrect connectivity particularly at singular points or for near singular situations. In the lattice method of [Rossignac 87], this idea has been used in the approximation of curves of intersection by piecewise circular arcs. There an adaptive subdivision strategy has been used to prevent failure near closely spaced curve features, except the limiting case of singularities.

2.3 Marching Methods

Marching methods involve generation of sequences of points of an intersection curve branch by stepping from a given point on the required curve in a direction prescribed by the local differential geometry. Marching methods are relatively simple to implement and very efficient as they do not require solution of large numbers of nonlinear equations. However, such methods require starting points for every branch and a stepping size which is case dependent and difficult to determine. Incorrect step size may lead to erroneous connectivity of solution branches or even to endless looping in the presence of closely spaced features [Geisow 83]. Simple marching methods, not employing appropriate expansions of implicit polynomials near singular points, may also fail near such points. A desingularization method based on birational transformations outlined in [Bajaj 87] provides an elegant rectification of this type of failure in marching methods for algebraic surface to parametric surface intersection problems and a way to connect the proper branches by marching across singular points. Marching methods significantly
benefit from the use of curvature analysis or power series expansions about each point of
the solution to control the step size [Faux 81], [Chen 88], [Bajaj 87], [Hoffmann 87], [Bajaj
88], [deMontaudouin 86], [Chuang 88]. In [Faux 81], [Chen 88] the intersection curve is
approximated locally by its osculating circle and the local curvature of this circle is used to
determine the step length for the marching method. In [Bajaj 87] a plane algebraic curve is
traced using a numerical method coupled with algebraic geometry techniques for automatic
resolution of the curve at its singular points (desingularization) as mentioned above. In
[Hoffmann 87] and [Bajaj 88] a numerical procedure is implemented for tracing the
intersection of implicitly defined algebraic surfaces and parametric surfaces. A third order
Taylor approximant is constructed for taking marching steps of variable length. Newton
iteration is then used to improve the accuracy of the intersection points obtained. In
[deMontaudouin 86] an estimate of the radius of convergence of local explicit power series
expansions about non-singular points is derived and employed to specify the variable step
size. This reduces the chances of incorrect connectivity and endless looping arising from
simple marching methods. In [Chuang 88] a local implicit approximation to the
intersection curve is employed to assist in stepping for curve tracing or for constructing a
piecewise algebraic curve approximation to the intersection curve. This method allows a
low-degree algebraic curve approximation of the intersection curve.

The determination of a set of significant points of the curve within the finite domain
enhances the reliability of lattice evaluation and marching methods used to trace the curve
within such a domain [Sarraga 83, Farouki 86a]. This set of points includes border, turning
and singular points of the curve on the border of or within the domain of interest. The
spacing of these points should be beneficial in the selection of, possibly, non-uniform grid
size for lattice methods and stepping size and initial points for every branch in marching
methods. Knowledge of significant points and the multiplicity of singular points also
provides an independent count of the number of monotonic branches between significant
points. This count can confirm the number of branches obtained using lattice and marching methods and provides added confidence in the solution. Encouraging results using such a method were reported in plane sectioning of low order parametric patches [Farouki 86a]. As noted in this reference, however, the above method does not establish the proper number of branches in the presence of tacnodes, i.e., cusps with a tangent direction which is a multiple tangent of the curve; and it requires rotation of the curve to handle branches between singular points without intervening turning points, as it does not provide a method to start tracing at singular points. Starting at singular points requires additional information which can be found from the desingularization procedure of [Bajaj 87] for algebraic curves. For general implicit intersection curves, a method to initiate marching from singular points is described in Chapter 9 of this thesis. The method to estimate the number of monotonic branches between significant points also requires adjustment when turning and singular points are also border points.

Concluding our review for lattice evaluation and marching methods, we should also point out the extreme importance of accurate computations of all singular points and at least one turning point per intersection loop in the domain of interest. Unfortunately, the a priori computation of turning and singular points in the domain of interest, although very useful, is complex. Using elimination [Farouki 86a] the computation of turning points, in theory, involves the solution of two univariate characteristic polynomial equations of degree $2ij-i$ and $2ij-j$, where $i, j$ are the degrees of the algebraic curve in each parameter. Computation of singular points, once the solution of the above equations is available, is relatively direct. The derivation and solution of these polynomials for high degree cases involves such large scale computation which makes this direct approach unattractive due to error accumulation and poor efficiency.

For intersections of more general classes of parametric surfaces, beyond the polynomial class, [Barnhill 87] has developed a general marching method, which only
requires a procedure to evaluate the surface position and its first partial derivatives at any point. The method employs a combination of lattice evaluation, subdivision and Newton methods to determine starting points for different branches of the intersection which is then traced by marching. [Barnhill 89] extended this method to handle triangular patches and the intersection of tangent surfaces or surfaces leading to singularities. They included a method for approximating the step length and for relaxing intersection points onto surface boundaries. An efficient implementation of the subdivision process was also introduced in this method using the divide-and-conquer paradigm [Houghton 85]. The basic advantage of this technique is its generality, allowing intersection of arbitrary parametric surfaces such as offsets, sweeps and blends. The technique also allows computation of self-intersections of a parametric surface patch, a useful feature in applications.

Recently [Markot 89] used Newton techniques to compute singular points of the intersection curve between two general parametric surfaces. [Markot 89] uses the vector field defined as the gradient of the distance function between the two surfaces to compute such points. These points are critical points of the vector field. This methodology does not provide an automated way to determine initial approximations for all these singular points.

2.4 Subdivision Methods

The main idea of subdivision methods involves recursive decomposition of the original intersection problem into simpler similar problems until a level of simplicity is reached, which allows simple direct solution, (e.g. plane / plane intersection). This is followed by a connection phase of the individual solutions to form the complete solution. By definition, the application of the method requires that the problem be subdividable, and, at each stage, there must be some recognizable reduction in the complexity of the problem. Parametric polynomial surfaces used in geometric design applications are amenable to subdivision [Boehm 80, Lane 80, Lyche 85] allowing computation of their intersections.
Subdivision methods are convergent in the limit and do not suffer from many of the degeneracies of simple marching methods.

Initially conceived in the context of intersections of polynomial parametric surfaces [Lane 80, Thomas 84, Mudur 84, Peng 84, Dokken 85, Houghton 85, Lasser 86, Chen 86, Chen 87], they can be extended to the computation of algebraic/polynomial parametric and algebraic/algebraic surface intersections. This extension is possible by reformulating the algebraic curves arising from such intersections in terms of the Bernstein basis within a rectangular window [Geisow 83] and interpreting such curves as intersections of Bezier surfaces and a plane [Patrikalakis 87]. The use of bounding boxes of subdivided surface elements is essential in subdivision algorithms to eliminate pairs of surface elements with no intersection. Min / max bounding boxes, rectangular bounding boxes or interval type algebraic bounds on equations are commonly used for such elimination of non-intersecting elements.

Subdivision techniques do not require starting points as marching methods, an important advantage from the reliability point of view. Many elements of subdivision techniques are also parallelizable, which is an important advantage for future large-scale real-time applications. General non-uniform subdivision allows easy selective refinement of the solution providing the basis for an adaptive intersection technique. However, when subdivision is used to obtain highly accurate solutions, it leads to data proliferation and is consequently slow. A major disadvantage of subdivision techniques is that, in actual implementations with finite subdivision steps, correct connectivity of solution branches near singular points is difficult to guarantee, small loops may be missed or extraneous loops may be present in the approximation of the solution. In general, features of the solution smaller than the final subdivision size are not resolved. For example, in the method proposed in [Geisow 83], the implicit intersection curve, expressed in the Bernstein form within a square, is subdivided using a quadtree approach until each cell contains one
monotonic piece of the curve. The presence of such a simple piece of the curve within a cell can be detected by interrogation of the Bernstein coefficients of the curve for that cell. Such simple curve segments can then be traced using marching. However, this simplicity criterion fails, for example, at turning and singular points and the algorithm recurses until the cell size is the limiting precision of the machine, an inefficient feature given that the entire computation is based on subdivision. An important variant of the subdivision-polyhedron faceting methods involves degree reduction and subdivision to reach the simplicity of plane/plane intersections [Petersen 84, Solomon 85, Arner 87] or parametric approximations to intersection curves [Waggenspack 89, Sederberg 89a, Sederberg 89b]. [Solomon 85] generates a linear approximation to the rational parametric surfaces within a specified tolerance using a combination of degree reduction and subdivision and uses a divide and conquer approach to interrogate the resulting intersection curve. [Arner 87] worked along similar lines using a combination of degree reduction and subdivision to generate a linear or quadratic approximation to the piecewise rational Bezier surfaces. [Waggenspack 89] presents a methodology for creating a \( C^0 \) piecewise rational parametric approximation for general bivariate algebraic curves. This algorithm couples subdivision methods with techniques from the classical algebraic and projective geometry to generate the necessary piecewise approximation. [Sederberg 89b] presents a methodology to approximate an algebraic curve of any degree using rational Bezier curves resulting in a \( G^2 \) piecewise approximation with fast convergence and tight error bounds. [Sederberg 89a] extended this work to approximate the intersection curve of two parametric surface patches using a \( G^k \) piecewise approximation computed directly from two points on the intersection curve. This approximation is performed after an adequate level of subdivision is used to identify all closed loops in the intersection curve.
2.5 Loop Detection Techniques

The problem with most of the techniques presented previously is one of robustness. Marching methods require starting points in all intersection segments to ensure tracing of all features of the intersection and subdivision techniques require large amounts of subdivision to resolve closely spaced features and singularities. As a result, simultaneously with the development of new ideas for the computation of the intersection curve between two surfaces, there is a strong recent interest in developing techniques to detect loops in intersection curves [Sinha 85, Prakash 88b, Sederberg 88a, Sederberg 89c, Cheng 88, Patrikalakis 90a, deMontaudouin 89].

[Sinha 85, deMontaudouin 89] proved that if two at least $C^1$ surfaces intersect in a closed loop, then there exists a normal vector on one surface which is parallel to a normal vector in the other surface (parallel normal vectors). [Sinha 85] suggested use of this condition to assist in selective subdivision of two surfaces that are intersected to identify all intersection segments. If bounds to the normal vector directions of two subpatches do not intersect, these subpatches do not intersect in an interior loop and further subdivision is unnecessary.

[Sederberg 89c] extended the above work and proved that if two at least $C^1$ surfaces intersect in a closed loop (in both parametric spaces), then there exists a line which is perpendicular to both surfaces (collinear normal vectors), provided the inner product between any normal vector on one surface and any other normal vector on the other surface is never zero. The advantage of this method over the previous method is that the non-existence of a collinear normal line and, consequently, absence of intersection loops, can often be deduced even though parallel normal vectors may exist. [Sederberg 89a] suggested an interval type method to bound a set of functions specifying collinearity to determine the existence or not of a collinear normal. For more details see also Section 6.1.
[Cheng 88] recently outlined a method based on vector fields which promises to provide global information about the topology of the intersection. According to this method the vector field of the gradient of the distance function between two surfaces is defined. Starting from a set of points inside the u-v domain of one patch, the connected components of the intersection set are determined by marching along the so-called "connecting curves", which are certain gradient lines of the distance function. As we discuss also in the following Chapter the distance function between two surfaces may include ill-defined regions arising from intersection of one surface and the cut locus with respect to the other surface point set. Marching along gradient lines may lead to such regions, and it is unclear how such degeneracies would be handled by the above algorithm.

[Prakash 88b, Patrikalakis 90a] capitalized on precomputation of most significant points of the intersection curve between an algebraic surface and a rational B-spline surface patch to identify the features of the intersection curve. This computation was performed using efficient minimization and Newton methods initiated from approximate solutions obtained by coarse subdivision. The control surface is partitioned using all available border, turning and singular points. Next, each subpatch is examined for possibility of an intersection using the convexity properties of the algebraic curve of intersection in the Bernstein basis. Next, for the subpatches with a possibility of an intersection the convexity properties of the partial derivatives of the representation of the algebraic curve of intersection are used and, if necessary, subdivision is employed to ensure computation of all significant points and, consequently, the detection of all loops of the intersection.

Finally, [Sederberg 88a] showed that if two at least C¹ surfaces intersect in a closed loop, then there exists a normal vector on one surface which is perpendicular to a tangent vector on the other surface. [Sederberg 88a] defined cones that bound the tangent directions of all curves of constant u or v in a rational B-spline patch, u, v bounding cones, and also cones that bound all of the normal vectors on a rational B-spline patch, normal cone. Using
the normal cone, the tangent plane cone was also defined by the following property. If its vertex is translated to any point on the surface, the tangent plane at that point will not cut the tangent plane cone. The criterion used to evaluate the existence of closed loops in the intersection curve is as follows: If the v or v cone from one of the surfaces lies completely within the tangent plane cone of a second surface then all v or u isoparameter curves of the first surface intersect the second surface at most once. This guarantees single valued intersection curves in v or u. The u and v cones from each surface are tested with the tangent plane cone of the other surface with respect to this condition and subdivision (splitting) is invoked until all the intersection loops are identified. [Sederberg 88a] observed that, in actual practice, the above loop detection test required a very large amount of computation even for simple intersections. More details on this promising technique and some implementation issues of such a loop detection method is presented in Chapter 7.

2.6 Summary of Intersection Algorithm Developed at the M.I.T. Design Laboratory

A recent algorithm based on analytic and subdivision methods allowing computation of intersections of algebraic and piecewise rational polynomial surface patches (in the B-spline form) has been developed at the M.I.T. Ocean Engineering Laboratory [Patrikalakis 87] [Prakash 88a] [Prakash 88b] [Patrikalakis 90a] and provided the basis from which the present work begun. This earlier algorithm represents the intersection set as an algebraic curve expressed in the Bernstein basis in the parameter space of the polynomial patch. The interrogation of a piecewise rational polynomial patch can be handled by first splitting the underlying B-spline surface to its rational polynomial elements [Lyche 85]. The robustness of the method is based on the a priori computation of the significant points of the intersection curve, i.e. boundary, turning and singular points. The significant points of each algebraic curve are used to split the intersection problem into independent sub-domains containing (possibly more than one) monotonic intersection segments which are processed
using adaptive subdivision and faceting techniques [Lyche 85]. The computation of turning and singular points of the curve exploiting the properties of the Bernstein representation has been studied using elimination and univariate polynomial solution techniques as well as direct numerical techniques based on minimization and Newton methods. Direct numerical techniques hold the promise of more accurate and efficient computation of turning and singular points, because they avoid error-prone large-scale elimination computations and the solution of high degree polynomials used in the early computational techniques. The direct numerical methods for the computation of these points, such as minimization and Newton techniques initiated from approximations created by coarse subdivision and faceting methods are followed by verification of the topological consistency of the results. This verification process uses the partial derivatives of the curve and convexity properties of their representation in the Bernstein basis. In case of failure subdivision is employed. After each subdivision step and prior to the partial derivative test, convexity properties of the curve itself in the Bernstein basis are used to eliminate subpatches with no intersection.
Chapter 3

FORMULATION OF SURFACE INTERSECTION PROBLEMS

3.1 Introduction

The computation of intersections of surfaces is a fundamental problem in computer aided design. The two types of surfaces commonly used in geometric modeling systems are parametric surfaces defined by \( r = r(u,v) \) depending on two parameters which are defined on a finite rectangular domain and the implicit surfaces \( f(r) = 0 \) which are usually defined in an infinite domain. These two types of surfaces lead to three types of intersection problems, parametric-implicit, implicit-implicit and parametric-parametric.

3.1.1 Parametric - Implicit

If a surface is defined by the implicit equation

\[
f(x,y,z) = 0
\]  
(3.1)

and another surface is defined by the parametric equation

\[
r = r(u,v)
\]  
(3.2)

by substituting the three components of the second equation into the first equation we obtain

\[
f(x(u,v), y(u,v), z(u,v)) = F(u,v) = 0
\]  
(3.3)

which is the equation of the intersection curve, represented as an implicit curve in the parameter domain of the parametric patch. Equation (3.3) represents one scalar equation to be solved for two unknowns.
3.1.2 Implicit - Implicit

If two surfaces are defined by implicit equations

\[ f_1(x,y,z) = 0 \]
\[ f_2(x,y,z) = 0 \]  \hspace{1cm} (3.4)

their intersection is defined by simultaneous solution of these two equations for the three Cartesian coordinates.

3.1.3 Parametric - Parametric

If two surfaces are defined parametrically by

\[ r = r(u,v) \]
\[ q = q(s,t) \]  \hspace{1cm} (3.5)

their intersection is defined by

\[ F(u,v,s,t) = r(u,v) - q(s,t) = 0 \]  \hspace{1cm} (3.6)

Equation (3.6) represents a system of three simultaneous equations one for each of the Cartesian coordinates, to solve for the four unknown parameters.

In each of the above cases, there is one more variable than there are equations, since the points of an intersection curve have one degree of freedom (except for degenerate cases involving isolated intersection points and points in a surface overlap region).

3.2 Intersection Problem Formulation Using Distance Functions

The computation of the intersection set of two parametric surfaces \( S_1 \) and \( S_2 \) may be regarded either as a problem involving the solution of simultaneous (usually non-linear) equations, or as a minimization problem in which the distance between two variable points on the two surfaces is minimized by adjusting the two points, until the distance becomes
zero. Alternatively the intersection set can be expressed in terms of the distance function between two surfaces, i.e. the sequence of points in the two surfaces with zero distance. It is useful to further expand on this generalization for the case of parametric surfaces.

We define the oriented distance function \( \phi \) between a surface \( q(s,t) \) and a point moving on another surface \( r(u,v) \). This function is a real valued function \( \phi \), which is defined on the \((u, v)\) parameter space of surface \( r \) as follows:

\[
\phi(u,v) = n_2[Q(r(u,v))] \cdot [r(u,v) - Q(r(u,v))] 
\]

where \( Q(r(u,v)) \) is a point on surface \( q(s,t) \) which is nearest to the point \( r(u,v) \), and \( n_2 \) is the unit normal vector on surface \( q \) at the point \( Q(r(u,v)) \). The value of \(|\phi(u,v)|\) is the actual Euclidean distance of point \( r(u,v) \) from point \( Q(r(u,v)) \) on surface \( q(s,t) \), provided the vectors \( n_2(Q(r(u,v))) \) and \( (r(u,v) - Q(r(u,v))) \) are collinear (well defined distance function).

In this case, point \( Q(r(u,v)) \) is the orthogonal projection of point \( r(u,v) \) on surface \( q(s,t) \) as shown in Figure 3-1 and the first and second order derivatives of \( \phi \), equation (3.7), may be computed as shown in Appendix B. In this work, the above minimum distance point pair is considered a "proper" pair. Assuming that \( \phi \) is a well defined distance function, the intersection set between surfaces \( r(u,v) \) and \( q(s,t) \) is equivalent to the zero set of the function \( \phi \), i.e. the set of points satisfying the implicit equation \( \phi(u,v)=0 \). Note, that if \( q(s,t) \) is a closed orientable surface, the sign of \( \phi \) determines if the point \( r(u,v) \) is inside or outside of the first surface.

There are special cases, where the function \( \phi \) is ill-defined. For example, there may be infinite or more than one but finite number of points \( Q(r(u,v)) \) on the surface \( q(s,t) \) nearest to point \( r(u,v) \). This occurs when point \( r(u,v) \) belongs to the cut locus of \( q(s,t) \). Given a point set \( A \), the cut locus with respect to \( A \) is defined as the set of points with two or more minimal joins to the set \( A \) together with their limit points [Wolter 85]. The cut locus is a generalization of the medial axis concept introduced by Blum [Blum 73]. As is well known, the medial axis of a closed curve or surface in two- or three-dimensional space
Figure 3-1: Projection of a Point to A Surface

consists of the centers of all maximal disks or spheres which fit into the domain bounded by
the curve or surface. Furthermore, another special case leading to ill-defined $\phi$ may arise
when the surface $q(s,t)$ has a finite boundary, which is the image of the boundary of the
parameter space $(s,t)$. In such a case, the nearest point $Q(r(u,v))$ may lie on the border and
may frequently fail to satisfy the condition that the vectors $n_2(Q(r(u,v)))$ and $(r(u,v) -
Q(r(u,v)))$ are collinear.

Figure 3-2 adapted from [Markot 89] presents four examples of such situations. In
this Figure, surface $S_1$ refers to $r(u,v)$ and surface $S_2$ refers to $q(s,t)$. The top example
represents a plane $S_1$ and a torus $S_2$ that is sliced in half by the plane. Point $p$ in $S_1$ is in the
center of the inner intersection circle of the plane. Function $\phi$ is ill-defined at this point,
since every point on this inner circle is at the same shortest distance to $p$. Point $p$ is on the
cut locus of the torus. The second example shows a plane $S_1$ and a swept surface $S_2$. Here
$p$ is on the plane and points $q_1$ and $q_2$ on surface $S_2$ are both the same shortest distance
from $p$. Once again $p$ is on the cut locus of the swept surface and the function $\phi$ is multiply
defined. The third example shows a plane $S_1$ and a swept surface $S_2$. Here points of the
plane are matched with the corresponding minimum distance points of surface $S_2$. There is
a point \( p \) on surface \( S_1 \), where the distance function is multiply defined. Away from this point the function is well behaved. It should be noted that for \( C^1 \) surfaces with Lipschitz continuous derivatives (as is the case for regular \( C^1 \) rational spline surfaces of at least order 3), the cut locus of a surface stays away from the surface [Wolter 85] and, therefore, these points where the distance function is ill-defined occur away from the actual intersection. The fourth example shows the effect of the definition of the distance function in a bounded parametric domain. Here, the roles of the plane \( S_2 \) and the swept surface \( S_1 \) are reversed. Point \( p \) is as shown on surface \( S_1 \). The point on \( S_2 \) closest to \( p \) is \( q \), which is on the border of the parametric domain of the plane. If the plane had infinite extent, then \( r \) would have been the "proper" minimum distance point. In this case point \( q \) is not a "proper" minimum distance point. In practice the presence of these points does not affect the robustness of our algorithms that are based on the vector field of the gradient of the distance function as described in Chapters 5 and 6.

A crucial element of the intersection problem is the identification of all connected intersection components. The definition of the function \( \phi \) can assist in the identification of all these intersection components as will be seen in the following chapters of this thesis. A closed loop in the intersection set of the two surfaces, corresponds to a closed level curve of \( \phi \) in the parameter domain, i.e. \( \phi(u,v) = c = 0 \). If the domain enclosed by such a curve is simply connected, then this domain contains an extremum of function \( \phi(u,v) \), and consequently a critical point of \( \phi \) (provided that \( \phi \) is well-defined at all points in the connected domain, see Figure 3-2 example 1 for a counter-example). At a critical point the gradient \( \nabla \phi = 0 \). As can be seen, the structure of the critical set of \( \phi \) is intimately related to the structure of the zero set of \( \phi \); i.e. the intersection set. If we could describe the structure of the critical set of \( \phi \), including ill-defined regions, then we could possibly identify all the features of the intersection set. Topological tools based on the rotation number of the vector field \( \nabla \phi \) along a closed curve on the \((u,v)\) parameter space are key to the solution of
Figure 3-2: Examples with Ill-Defined Distance Function, Adapted from [Markot 89]
the intersection problem formulated as above will be described in Chapters 5 and 6. A good treatment of the relevant theoretical background may be found in [Krasnoselskiy 66]. The plane vector field resulting from the gradient vector of the oriented distance function is defined by

\[ \nabla \phi(u,v) = [n_2(Q(r(u,v))) \cdot r_u(u,v), n_2(Q(r(u,v))) \cdot r_v(u,v)]^T \]  \hspace{1cm} (3.8)

where subscripts u, v denote partial derivatives and superscript T denotes transpose. As can be seen in Appendix B, equation (3.8) assumes that the vectors \( n_2(Q(r(u,v))) \) and \( (r(u,v) - Q(r(u,v))) \) are collinear. This field is also the vector field suggested by [Cheng 88] to compute the intersection of two parametric surfaces and the vector field used by [Markot 89] to compute singular points of the intersection using Newton iteration. [Cheng 88] introduces the above vector field in a procedural manner by considering equal height curves of \( \phi \), i.e. curves \( \phi(u,v) = c \) where \( c \) is a constant, without introducing the formalized oriented distance function concept given above.

For an algebraic to rational polynomial parametric surface intersection problem, we have seen in section 3.1.1 that we can substitute the parametric surface equations to the implicit equation to obtain an algebraic curve in the parameter domain of the parametric patch, which completely represents the intersection set. A similar procedure can in theory be used in the case of two rational polynomial parametric surfaces, by converting one of the rational polynomial surfaces to its implicit algebraic representation and then performing the substitution. The major problem involved here is that the algebraic representation of a rational polynomial patch is of very high degree making this substitution impractical [Sederberg 84b]. The distance function zero set \( \phi = 0 \) presented above, provides an alternate implicit representation of the intersection set and is directly analogous to the algebraic curve of intersection in the parameter domain of one of the patches which would have been obtained from such a substitution. Explicit analytic algorithms for the computation of the distance function \( \phi \) have been developed and tensorial differential
equations based on orthogonal projections are used to trace equal height curves of the
distance function $\phi$, see Chapter 9.

The critical set of $\phi$ has an additional interpretation from the field of optimization.
We define a height function which is a scalar function of two parameters:

$$h = \phi(u,v) \quad (3.9)$$

This function can be the objective function of a nonlinear constrained optimization
problem. A fundamental problem in nonlinear optimization is the identification of all local
extrema of an objective function and consequently its global extremum [Gill 81].
Function $h$, may be viewed as the distance between a particular point on the graph of
function $h$ and the $u$-$v$ plane with $h=0$. The critical points of this distance function are the
local minima and maxima of the height function. In this case the distance function is well-
defined at all points in the domain of function $h$, since the cut locus of plane $h=0$ does not
exist and we have an unbounded domain for hyperplane $h=0$ and none of the examples
presented in Figure 3-2 can occur. As can be seen, determining all the features of an
intersection set is equivalent with finding all critical points of a height-distance function. In
this work we develop topological and numerical techniques to compute critical points of a
plane vector field. These techniques can be used to determine the complete extrema set of a
height function of the form (3.9) of two variables. For height functions of more than two
variables, complexity increases and a more general theory of vector fields is required.

3.3 Algorithmic Issues

The intersection of two surfaces can be very complicated, in general, with a number
of closed loops, segments extending from boundary to boundary, isolated points, common
surface segments, tangencies along a curve, self-intersections and other singularities. The
ability to detect and describe all such features of the intersection curve and to trace these
correctly and efficiently is a formidable computational problem.
A good surface-intersection algorithm has to balance several conflicting goals such as robustness, accuracy and efficiency. Robustness is a key property of automated solution procedures in a computer environment. If such a property is not guaranteed, then the designer needs to manually ensure that the resulting approximations do not result in global inconsistencies of the geometric database. For example, missing a connected component of the intersection set (like a loop) is unacceptable for geometric modeling applications. The overall automation of the intersection process is an additional factor to consider. A good algorithm should be able to identify the various intricacies of an intersection set in an automated manner, i.e. with minimal user interaction. In addition a verification methodology should exist that would increase the confidence about the computed result. The solutions resulting from any intersection algorithms are further complicated by imprecisions introduced by numerical errors present in all finite precision computations. The accuracy a numerical method can deliver varies with the local surface geometry at the intersection, the angle at which the surfaces intersect etc. In a good numerical intersection algorithm it is essential that a consistent tolerancing scheme is implemented to determine the accuracy of the computed solution. This scheme should not vary in the different parts of the intersection algorithm. Efficiency is another important property of a surface to surface intersection algorithm. Intersections are frequently invoked, basic operations and as a result should be efficient to avoid delaying the design process. There is a trade-off between robustness and efficiency in an intersection algorithm, since the larger the number of robustness enhancement computations (verification) the slower the execution of the algorithm even for simple examples.

In this work we study all the issues mentioned above. For this we identify a number of critical points in the intersection between two surfaces. These points are called significant points and their definition and computation is described in Chapter 5. These points have both a geometric and algebraic interpretation. Most of these points are on the
intersection set and serve to identify all the features of an intersection curve such as simple segments, loops and singular points. In addition, there are a number of points on the two surfaces related to the distance function defined in the previous section which are not necessarily on the intersection curve and may also serve to identify all features of an intersection set. These points are computed in the first step of an intersection algorithm and are then used to assist in tracing all the features of an intersection. An adaptive verification technique is then employed to enhance the confidence of the intersection results and, in case of failure of the criterion under consideration, to provide a method to detect any missing features of the solution.

3.4 Intersection Problems Addressed

Figure 3-3 presents a further subdivision of common intersection problems in more narrow categories by distinguishing between piecewise rational polynomial parametric surfaces such as B-spline, Bezier, Hermite and Coons surfaces and general parametric surfaces such as generalized cylinders, sweep surfaces, blend surfaces, offset surfaces etc., and between polynomial implicit (algebraic) surfaces such as quadrics, cubics and general implicit surfaces such as surfaces involving transcendental functions etc. The present thesis addresses primarily two types of intersection problems. The first is the intersection of a piecewise rational polynomial parametric surface patch with an implicit polynomial (algebraic) surface and the second is the intersection of two piecewise rational polynomial parametric surfaces. The representation selected for the piecewise rational polynomial parametric surface is the non-uniform rational B-spline (NURBS) formulation which is the most widely used and flexible representation, and has been adopted by many industries and exchange standards as a canonical representation [Smith 88]. Many of the ideas in this thesis are also applicable to the more general problem of intersecting two general parametric surfaces and this will be noted in appropriate sections.
3.4.1 Algebraic - Rational Polynomial Parametric Surface Intersection

3.4.1.1 Governing Equations

Given a rational parametric polynomial patch \( r(u,v) \) expressed in the Bernstein basis with degree \( q \) in \( u \) and \( r \) in \( v \), we are interested in its intersection with an algebraic surface of the form

\[
f(x,y,z,w) = \sum_{i=0}^{p} \sum_{j=0}^{p-i} \sum_{k=0}^{p-i-j} C_{ijk} x^i y^j z^k w^{p-i-j-k} = 0
\]  

(3.10)

where \( p \) is the degree of the surface and \( x/w, y/w, \) and \( z/w \) give the Cartesian coordinates of points on the algebraic surface. To obtain the representation of the intersection in the parameter space of the patch, we substitute the coordinates of the parametric patch in the algebraic surface equation.
\[
f(x,y,z,w) = f(r(u,v)) = F(u,v) = 0
\] (3.11)

Such an equation, in general, represents an algebraic curve of degree \( pq \) in \( u \) and \( pr \) in \( v \) where \( u, v \) are restricted to the parametric space of patch \( r(u,v) \). For the intersection problem at hand, where we are interested only in a finite portion of the algebraic curve \( F(u,v) \) within a rectangular domain, it is expedient to represent the algebraic curve in the Bernstein basis

\[
F(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} w_{ij} B_{i,m}(u) B_{j,n}(v) = 0,
\] (3.12)

where without loss of generality \( u \in [0,1], v \in [0,1] \) and

\[
B_{i,k}(t) = \binom{k}{i} t^i (1-t)^{k-i}
\] (3.13)

In the exceptional case, where \( w_{ij} = 0 \) for all \( i = 0,1...m, j = 0,1...n \), equation (3.12) does not define a curve as the intersection of two surfaces. It, rather, points out that the rational polynomial patch coincides with a portion of the algebraic surface. When the coefficients of the algebraic surface are normalized, and floating point arithmetic is employed, the condition \( |w_{ij}| \leq \epsilon \), where \( \epsilon \) is a small positive number, may be used to test for such an exceptional occurrence. This provides a method to identify the possibility of surface overlap in the intersection set. The non-dimensional number \( \epsilon << 1 \) is related to a distance tolerance, \( \delta \), for coincidence of the two surfaces by an approximate equation \( \epsilon \approx \delta |\nabla f| \), where \( \nabla \) denotes gradient and \( \nabla f \) is evaluated at a point on the parametric surface assumed to be close to the algebraic surface.

When the \( w_{ij} \) are either all positive or all negative, there is no intersection since the \( B_{i,m}(u) \) and \( B_{j,n}(v) \) are always non-negative and as a result the pair of surfaces under consideration can be discarded. This test should be applied prior to any other computation to efficiently reject non-intersecting surfaces. It is convenient to normalize the coefficients \( w_{ij} \) using the maximum \( |w_{ij}| \neq 0 \), so that \( |w_{ij}| \leq 1 \) for all \( i \) and \( j \).
3.4.1.2 Direct Computation of Intersection Curve Equation

In the algorithm presented in [Prakash 88a], the computation of the intersection equation (3.12) was performed symbolically leading to expressions for \( w_{ij} \) in terms of original data. This algorithm was developed to treat intersections of planes and quadrics with up to rational bicubic patches. The degree restriction stemmed from derivation of the representation of the algebraic curve of intersection using symbolic computations. For higher degree intersections this symbolic computation becomes prohibitively expensive in terms of storage and processing requirements.

The present algorithm, also briefly reported in [Krietzis 89], uses direct numerical evaluation of the algebraic curve representation using the Bernstein basis throughout the calculation, saving some unnecessary operations and avoiding intermediate use of the monomial basis [Farouki 87]. The only requirement is an ability to multiply Bernstein polynomials numerically [Solomon 85]

\[
B_{i,n}(u) \cdot B_{j,m}(u) = \binom{n}{i} \binom{m}{j} \binom{n+m}{i+j}^{-1} B_{i+j,m+n}(u)
\] (3.14)

Using this property, a method was developed to multiply two arbitrary Bernstein polynomials. For example, suppose we are intersecting a rational bicubic patch with the sphere \( x^2 + y^2 + z^2 - w^2 = 0 \). After substitution of the parametric patch each of the terms \( x^2, y^2, z^2 \) and \( w^2 \) can be computed separately using the multiplication property. The results can then be added together, to obtain the representation of the intersection curve in the Bernstein basis. For higher degree cases the multiplication property needs to be applied repeatedly, until the degree of each of the terms of the surface is obtained and addition can be performed. Since all the terms of the algebraic surface have the same total degree in \( x, y, z \) and \( w \), this procedure can always be used to obtain the intersection curve of any algebraic surface without involving a degree elevation process. This procedure was used to obtain the representation of the algebraic curve of intersection of various torii (fourth
degree algebraics) and other algebraic surfaces with rational biquadratic and bicubic patches.

3.4.1.3 Geometric Interpretation of Algebraic Curves in a Rectangular Domain

It is convenient to visualize (3.12) as the intersection of the explicit surface \( w = F(u,v) \) with the control plane \( w = 0 \). The above explicit surface can be recast as a tensor product Bezier patch

\[
T(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} T_{ij} B_{i,m}(u) B_{j,n}(v) \quad \text{where} \quad T = [u \ v \ w]
\]

\[
T_{ij} = [u_i \ v_j \ w_{ij}]
\]

\[
u_i = \frac{i}{m} \quad ; \quad v_j = \frac{j}{n} \quad i = 0,1,...m \quad j = 0,1,...n
\]

(3.15)

The coefficients \( w_{ij} \) introduced in (3.12) are the \( w \) coordinates of the control polyhedron vertices \( T_{ij} \) of the parametric surface defined by the above equation while the \( u \) and \( v \) coordinates of control polyhedron vertices are uniformly spaced in the range of the parametric variables. The above reformulation has been first used by [Geisow 83] in a study of surface intersections. [Sederberg 84a] employed a similar method to represent algebraic curve portions within triangles. [Patrikalakis 88, Patrikalakis 89] introduced an extension of (3.15) to B-splines as a method of sculptured shape creation in terms of low order piecewise algebraic curves and surfaces within rectangular boxes. Since the control polyhedron provides an approximation of the geometry of the surface (3.15), we can get an approximation of the intersection curve (3.12) by intersecting a faceting of the control polyhedron with the plane \( w = 0 \). The accuracy of the approximate intersection curve depends on the accuracy to which the control polyhedron approximates the surface. As is well known, the control polyhedron becomes an increasingly better approximation of the surface by the use of subdivision. This geometric interpretation and subdivision are used during the tracing part of our algorithm to compute approximate intersection points which are later refined by more efficient methods.
3.4.2 Rational Polynomial Surface Intersections

3.4.2.1 Problem Formulation

A non-uniform rational B-spline (NURBS) surface patch \( r(u, v) \) of degree \( M-1 \) and \( N-1 \) (order \( M \) and \( N \)) in the parametric variables \( u \) and \( v \) is given by [Tiller 83]

\[
r(u, v) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} P_{ij} N_{i,M}(u) N_{j,N}(v)
\]  
(3.16)

where without loss of generality \( u \in [0, 1], v \in [0, 1], \) \( P_{ij} \) are homogeneous coordinates of the \( m \times n \) control points and \( m \geq M, n \geq N \). The three-dimensional Cartesian coordinates of a point \( (u, v) \) of the patch are obtained by dividing each of the first three homogeneous coordinates of \( r(u, v) \) by its fourth homogeneous coordinate which is assumed to be positive. To ensure positivity of the fourth homogeneous coordinate of \( r(u, v) \), it is sufficient to require that the fourth homogeneous coordinate of \( P_{ij} \) is positive. For the algorithms developed in this work, the first partial derivatives of (3.16) are not allowed to vanish at any point in the intersection domain and also \( r_u x r_v \neq 0 \). \( N_{i,M}(u) \) and \( N_{j,N}(v) \) are the B-spline basis functions defined over non-uniform knot vectors in the \( u \) and \( v \) directions given by \( \{u_0, \ldots, u_{M+m}\} \) and \( \{v_0, \ldots, v_{N+n}\} \), respectively. The B-spline basis functions may be computed by de Boor's recursion [De Boor 72].

\[
N_{i,M}(t) = \frac{t-t_i}{t_{i+M-1}-t_i} N_{i,M-1}(t) + \frac{t_{i+M}-t}{t_{i+M}-t_{i+1}} N_{i+1,M-1}(t) \text{ if } M > 1
\]  
(3.17)

\[
N_{i,1}(t) = \begin{cases} 
1 & \text{if } t_i \leq t < t_{i+1} \\
0 & \text{otherwise}
\end{cases}
\]  
(3.18)

where the \( 0/0 = 0 \) convention is used, whenever such a ratio arises. The node vectors \( n_u \) and \( n_v \) of (3.16) in the \( u \) or \( v \) direction are given as a function of the knot vectors by:

\[
n_u,i = \frac{1}{M-1} (u_{i+1} + u_{i+2} + \cdots + u_{i+M-1}) \text{ with } i = 0, \ldots, m-1, \text{ and }
\]  
(3.19)

\[
n_v,i = \frac{1}{N-1} (v_{i+1} + v_{i+2} + \cdots + v_{i+N-1}) \text{ with } i = 0, \ldots, n-1
\]  
(3.20)
Given two non-uniform rational B-spline surface patches \( r(u,v) \) and \( q(s,t) \), we are interested in computing their intersection, i.e. solutions to equations (3.6) or \( \phi(u,v) = 0 \). Before performing any intersection computations, the two parametric patches in equation (3.6) or (3.7) are translated and scaled to fit in a cube \([-1,1] \times [-1,1] \times [-1,1]\) in order to improve the convergence properties and accuracy of subsequent numerical computations [Gill 81]. In this thesis we first address the case of transversal intersections, namely cases where the intersection involves only finitely many singular points, at which the two surfaces are tangent. Our method is also extended to identify and describe a class of non-transversal intersections involving infinite singular points arising from intersection of two surfaces along a curve on which the two surfaces have the same normal direction.

An important property of B-spline surface patches is that the control polyhedron used in the definition of the patch provides an approximation of the geometry of the surface. Thus, we can get an approximation of the intersection curve between two surfaces by intersecting a faceting of their control polyhedra. The accuracy of the approximate intersection curve depends on the accuracy to which the control polyhedra approximate the surfaces. The control polyhedron becomes an increasingly better approximation of the surface by the use of subdivision [Lyche 85]. This property is used during the tracing part of one of our algorithms to compute approximate intersection points which are later refined by more efficient methods.
Chapter 4

CONVEXITY PROPERTY BOUNDS FOR B-SPLINES

This Chapter presents new algorithms for the computation of tight bounds for the position, first partial parametric derivatives and the normal vectors of non-uniform rational B-spline surface patches. These bounds are useful in subdivision methods for identifying non-intersecting surfaces in an efficient manner, as well as for verifying the completeness of an intersection computation (identification of all intersection segments). An application of the above bounds in intersection problems is presented in Chapter 7.

4.1 Tight Rectangular Bounding Box

It is standard practice in subdivision type intersection methods, before actual computation of the intersection between two subpatches, to check for intersection of boxes that bound these subpatches. This eliminates unnecessary computations and improves the efficiency of the computation. The idea is to reduce the complexity of the search problem by restricting the area which can possibly contain components of the intersection set. The boxes, possibly containing intersection components, can then serve in constructing approximations of the intersection set. The convex hull property of rational B-spline surface patches and their control polyhedra assist in easily computing bounds for these subpatches. Many intersection algorithms use min/max rectangular boxes for this check [Lane 80, Sederberg 88a]. For the computation of the min/max box of a B-spline patch the minima and the maxima of the \( x \), \( y \) and \( z \) coordinates of the control points of the patch are determined and the box is defined by halfspaces \( x=x_{\text{min}}, x=x_{\text{max}}, y=y_{\text{min}}, y=y_{\text{max}}, z=z_{\text{min}} \) and \( z=z_{\text{max}} \). The min/max boxes are simple to compute and check for intersection, but require many subdivision steps to separate surfaces not similarly oriented to the global
coordinate axes. A more efficient alternative is to use rectangular boxes oriented parallel to a local coordinate system naturally associated with the orientation of each subpatch. For example, [Houghton 85] proposed using three subpatch corner points to find more natural local x, y and z coordinate directions. Constructing bounding boxes with non-linear boundary for spline entities has recently attracted attention. In a recent paper, [Sederberg 89d] proposes circular arcs with finite thickness (fat arcs) as bounds of planar Bezier curves. Experiments suggest that fat arcs are interesting because they exhibit cubic convergence. By contrast, convex hulls and bounding strips, oriented according to the line joining the curve end points, show quadratic convergence, and min/max boxes linear convergence rates. Such arcs are useful in speeding intersection problems. [Sederberg 89d] suggests studying fat spherical and toroidal sectors for surface patches, in order to facilitate surface intersection problems.

In our algorithms, we devise a variation of the rectangular bounding boxes proposed by [Houghton 85]. In order to allow for a more natural orientation of the rectangular box, than the orientation determined using the corner control points of a subpatch, we use information from the first parametric partial derivatives of the patch and their bounds defined in terms of rectangular pyramids described in the following section. The center vectors of these bounding rectangular pyramids for the first parametric partial derivatives of the patch are used to define the local coordinate system naturally associated with the patch. This local coordinate system is defined by \( R_{u,cen} \), \( R_{u,cen} \times R_{v,cen} \) and \( (R_{u,cen} \times R_{v,cen}) \times R_{u,cen} \) directions, where \( R_{u,cen} \) and \( R_{v,cen} \) are the center vectors for the pyramids which bound the first parametric partial derivatives of the patch in the u and v directions, respectively. The rectangular box to the surface patch is then defined as follows. The oriented distance of each control point of the surface patch is computed from each of the local coordinate planes by evaluating

\[
d_{ij,a} = R_{ij} \cdot x_a
\]  

(4.1)
where, $d_{ij,a}$ is the coordinate of control point $R_{ij}$ in the coordinate direction $a$ (or the oriented distance of $R_{ij}$ from the coordinate plane orthogonal to $x_a$) and $x_a$ is the unit vector axis for coordinate direction $a$. The maximum and minimum coordinates $d_{ij,a}$ of the control points from the three local coordinate planes are then determined to complete the definition of the rectangular box. This definition of a derivative based coordinate system for the rectangular bounding box to a patch results in a box which is normally smaller than rectangular boxes defined in an arbitrary coordinate system. The computation of the bounding box for a patch requires $3n$ inner products and comparison operations, where $n$ is the number of control points in the patch. The computation of the bounding pyramids for the patches is not considered as part of the cost of this computation, since these bounds are required in other parts of one of our intersection algorithms (see Chapter 7).

To determine if two rectangular boxes intersect, a standard algorithm is to first transform one of the boxes into the local coordinate system of the other box. Each edge of the first box is checked for intersection with each face of the second box. The role of the boxes is then reversed and the check is performed again. Finally, a test of containment of one box within the other box is easily performed. This method is efficient in determining when two boxes intersect, and is costly for verifying non-intersecting boxes since all the box edges need to be tested before a conclusion is reached.

We developed a new algorithm to perform the rectangular box intersection test which is very efficient in deciding absence of intersection between two boxes. This is explained next for the two-dimensional case. Figure 4-1 illustrates our test, which does not require transformation to the local system of any box. Each of the coordinate directions of a box are treated separately. For direction $n_2$ of the first box in Figure 4-1 we draw perpendicular lines to direction $n_2$ at diagonal corner points $A$ and $B$ of the second box. These lines provide a bound to the second box in the direction $n_2$. The choice of diagonal points $A$ and $B$ is easily made by verifying that $n_2$ lies in the first quadrant of the coordinate system of
the second box. This is done as follows. We compute the dot product of \( n_2 \) with the coordinate directions of the other box and depending on the value of the dot products the relevant quadrant is selected and points A and B are specified. The oriented distance of points A and B from the line through O and direction \( n_1 \) is then computed:

\[
d_A = \overrightarrow{OA} \cdot n_2 \quad \text{and} \quad d_B = \overrightarrow{OB} \cdot n_2
\]

where \( n_2 \) is a unit vector. If interval \([d_A, d_B]\) does not intersect with interval \([0, d_2]\), then the two rectangular boxes also do not intersect and the process terminates. If they intersect, this condition is tested for the remaining coordinate directions of both boxes. Testing the coordinate directions of one of the boxes is not sufficient to determine intersection. If in at least one coordinate direction the corresponding intervals do not intersect, then the boxes do not intersect and the process terminates. This method discards non-intersecting boxes very efficiently, while it requires testing of all coordinate directions for intersecting boxes. For the three-dimensional case, testing of non-intersecting boxes requires testing of three directions on the average, while for intersecting boxes it requires testing of six coordinate directions. Since non-intersecting subboxes are the majority in a subdivision scheme, our algorithm is more efficient than the alternative method to compare for intersection of rectangular boxes.

### 4.2 Tight Bounds For First Parametric Partial Derivatives and Normal Vectors

There are many applications for which the determination of tight bounds for the first parametric partial derivatives of a patch is an important tool. One example of such an application is the method described in Chapter 7 for the detection of loops in surface intersections and for verification of complete computation of the intersection curve. Another application where these bounds are useful is interval analysis [Mudur 84]. In
Figure 4-1: Rectangular Box Overlap Test

interval analysis tight interval bounds need to be determined for certain equations, which contain parametric derivative directions and normal vector directions. If 0 does not belong in these interval bounds, then we are guaranteed that the equation has no solution in that range. [Sederberg 89c] for example uses interval analysis for equations (5.33) to (5.36) to find regions where these equations cannot be satisfied simultaneously, i.e. to find regions where the two surfaces have no collinear normal points. Equations (5.33) to (5.36) require tight bounds for the first parametric derivatives of both patches as well as for the normal vector of one of the two surfaces.

4.2.1 Bounding Cones

[Sederberg 88a] defines cones that bound the tangent directions of all curves of constant u or v in a Bezier patch (u, v bounding cones) and also cones that bound all of the normal vectors on a Bezier patch. The computation of these cones is done by finding the partial derivative of the surface with respect to u or v, which results in a tensor product hodograph. For an integral Bezier surface of degree m in the u and n in the v direction, the
u-hodograph is a Bezier surface of degree m-1 in the u and degree n in the v direction. The cone which bounds the m(n+1) control points of this hodograph from the convexity property of Bezier surfaces will bound the direction of all the v iso-parameter curves on the surface. As shown in Figure 4-2 from [Sederberg 88a], if this cone is translated so that its vertex lies on any point on the surface, the cone will bound the v isoparameter curve which passes through that point. [Sederberg 88a] also defined a cone which bounds all of the normal vectors on a surface patch (normal cone). He selected the efficient but conservative solution of computing the normal cone which bounds the cross-product between any vector in the u cone and any vector in the v cone for the patch. The normal cone is not defined if the u or v cones are too large and subdivision is required to reduce the half-angle of the u or v cones to enable definition of the normal cone.

Figure 4-2: Cone Bounding Curves of Constant v, Adapted from [Sederberg 88a]

For the application of these bounds in loop detection (see Chapter 7), it is very important for these bounds to be computed efficiently and to be tight. One problem with [Sederberg 88a]'s definition of the bounding cones is the incremental construction of the u
or $v$ bounding cones. Each of the control points of the hodograph of the parametric patches is used in sequence to update the bounding cone. Each of the control points is examined if it belongs inside the current cone. If it is inside nothing is done, while if it is outside, a new cone is defined which encloses the previous cone and the last control point. In two dimensions this process is equivalent to incremental creation of the circle that encloses a number of points. Figure 4-3 illustrates this process for three points. As can be seen the incremental creation results in a larger circle (bounding cone) and not a very tight bound to the control points. Since this incremental process is repeated for more points, this effect is accentuated and the resulting bounding circle may be exaggerated significantly. An additional issue of concern is, that in most practical applications, the derivatives in the $u$ or $v$ direction of a surface patch tend to vary predominantly in a plane, and as a result, a cone enclosing the control points of such a patch does not represent a tight bound to the derivative directions. Figure 4-4 illustrates this problem, for the two-dimensional case of fitting a circle through a number of points which happen to be collinear. The circular bound does not represent a tight bound to the position of the points. The size problem arising from the use of the bounding cone is documented in Table 4-I as will be described in the following section.

**Figure 4-3:** Conservative Bounds from Incremental Creation of Bounding Circles

![Incremental Creation](image1)

![Tight Bound](image2)
4.2.2 Bounding Pyramids

In order to overcome the size problem arising from the conservative bounding cones, we identified a less conservative and comparably efficient in computation alternative to the cone to bound the appropriate vectors. We introduce rectangular pyramids to bound the parametric derivative direction vectors and the normal directions of a parametric patch. As with the method of [Sederberg 88a], we bound the control points of the first parametric partial derivative patches (hodograph).

The first step to define bounding pyramids to the first parametric partial derivatives of a parametric patch is to select a set of orthogonal planes which will be the planes defining the initial axes \((C_0, n_1, n_2)\) of the pyramid. To select the first orthogonal plane, we examine the angles between the position vectors of the four corner control points of the first derivative parametric patch. The position vectors with the largest angle are used to define the first plane. The second plane is defined orthogonal to the first plane along the bisector \(C_0\) (initial central axis) of the angle between the two position vectors selected. Such selection of orthogonal planes for the pyramid was found to allow construction of tight rectangular pyramids to bound the tangent directions of the patch.
Figure 4-5 illustrates the process of determining the half angles of the bounding pyramid, once the orthogonal planes and the initial axes of the pyramid are selected. In Figure 4-5 vectors $n_1$ and $n_2$ are unit vectors perpendicular to the above orthogonal planes and act as initial axes of the pyramid. Each of the control points $P_i$ of the derivative patch is projected on the two orthogonal planes with unit normal vectors $n_1$ and $n_2$ and the angle of each projected vector with the bisector $C_0$ is determined (angles $a$ and $b$ in Figure 4-5). These angles are defined as follows:

$$a = \text{sign}(P_a \cdot n_1) \arccos \left( \frac{C_0 \cdot P_a}{\|C_0\| \|P_a\|} \right) \quad \text{where}$$

(4.2)

$$P_a = P_i - (P_i \cdot n_2) n_2$$

(4.3)

$\text{sign}( )$ is the sign function, and the arc cosine function assumes values in the range $[0, \pi]$. For angle $b$, the roles of vectors $n_1$ and $n_2$ are reversed in the above equation. Angles $a$ and $b$ take values in interval $[-\pi, \pi]$. The minimum and maximum angles in each orthogonal plane are thus obtained and the faces of the rectangular pyramid can be constructed as planes forming these angles with the two initial orthogonal planes. This construction is as follows, see also Figure 4-6. First, the central axis of the rectangular pyramid is defined from:

$$C_p = C_0 + D + F = C_0 + \tan(c_1) n_1 + \tan(c_2) n_2$$

(4.4)

where $c_1 = (a_{\max} + a_{\min})/2$ and $c_2 = (b_{\max} + b_{\min})/2$, $C_0$, $n_1$, $n_2$ in the above equation are unit vectors. The pyramid’s half angles are $a_p = (a_{\max} - a_{\min})/2$ and $b_p = (b_{\max} - b_{\min})/2$. If the resulting half angle in any plane is greater than $\pi/2$, the bounding pyramid is not defined and further subdivision is needed to obtain subpatches with smaller parametric derivative variation. The orthogonal symmetry planes which intersect at the center vector and define the sides of the rectangular pyramid have the following normal vectors:

$$n_{p1} = n_1 - (C_p \cdot n_1) C_p$$

(4.5)

$$n_{p2} = n_{p1} \times C_p$$

(4.6)
where \( \mathbf{C}_p, \mathbf{n}_1 \) and \( \mathbf{n}_2 \) are unit vectors. The edge vectors \( \mathbf{C}_i \) of the rectangular pyramid can then be easily defined similarly to equation (4.4) by:

\[
\mathbf{C}_i = \mathbf{C}_p + tan(\pm a_p) \mathbf{n}_{p1} + tan(\pm b_p) \mathbf{n}_{p2}
\]  \hspace{1cm} (4.7)

where \( \mathbf{C}_p, \mathbf{n}_{p1} \) and \( \mathbf{n}_{p2} \) are unit vectors and \( i = 0,...,3 \), for the various \( \pm \) combinations (combinations ++,+-,-+,+-+ respectively). Figure 4-6 displays the various steps in the definition of the rectangular pyramid, once the ranges of angles a and b are determined. The algorithm for the computation of the bounding pyramid is near-optimal, i.e. \( \mathcal{O}(n) \) in the number of control points in the derivative patch.

For rational B-spline surfaces with positive weights, the bounding pyramids can be derived similarly by computing the hodographs of the integral B-spline surface patch derived from setting all weights to one in the original rational B-spline surface patch and bounding their control points [Sederberg 88b].

![Figure 4-5: Bounding Pyramid Construction, Angle Determination](image)
Figure 4-6: Bounding Pyramid Construction, Pyramid Definition

To evaluate the pyramid bounding the normal vectors of a patch, we employ the efficient but non-optimal solution of computing the pyramid which bounds the cross-product between any vector in the u pyramid and any vector in the v pyramid. This is still a conservative normal pyramid. For its computation, the cross-product of the vectors of the four edges of each of the pyramids with the vectors of the four edges of the other pyramid is computed and the rectangular pyramid which bounds these vectors is determined in a way similar to the definition of the u, v rectangular pyramids discussed above. As can be seen below, the convexity property of B-spline / Bezier surfaces ensures that the normal pyramid defined using the edges of the u and v rectangular pyramids encloses all normal vectors of the surface:

Theorem 1: For any vector \( \mathbf{R} \) in the u-pyramid of a patch, and for any vector \( \mathbf{Q} \) in the v-pyramid of a patch, vector \( \mathbf{R} \times \mathbf{Q} \) belongs in the normal rectangular pyramid of the patch which is defined by the cross-products of the edge vectors of the u and v pyramids.

Proof: Since a direction vector within the u or the v pyramid can be expressed as a convex combination of the four edge directions of the pyramid, we
have $R = \sum_{i=0}^{3} a_i C_{u,i}$ with $\sum_{i=0}^{3} a_i = 1$ and $Q = \sum_{i=0}^{3} b_j C_{v,j}$ with $\sum_{i=0}^{3} b_i = 1$ and $a_i$, $b_i$ are positive. The cross-product vector $L = R \times Q$ can be expressed as

$$L = \sum_{i=0}^{3} a_i C_{u,i} \times \sum_{j=0}^{3} b_j C_{v,j} = \sum_{i=0}^{3} \sum_{j=0}^{3} a_i b_j (C_{u,i} \times C_{v,j})$$

Since $\sum_{i=0}^{3} \sum_{j=0}^{3} a_i b_j = 1$, $L$ can be written as a convex combination of the cross-product of the edge vectors of the $u$ and $v$ pyramids and therefore $L$ belongs in the pyramid which contains these cross-products.

Sixteen cross-products need to be evaluated for the computation of the normal pyramid once the $u$ and $v$ pyramids are computed. The normal bounding pyramid discussed, can be computed fast and it provides a tighter bound to the normal vectors as compared to the normal cone defined by [Sederberg 88a]. Still, a tighter normal pyramid is the one which bounds the cross-product of the $u$ and $v$ hodographs of the surface. This normal direction surface is of degree $(m + n - 1) \times (m + n - 1)$ and has $(m + n)^2$ control points [Sederberg 88a]. The pyramid bounding these control points would provide a tighter bound on the normal directions than does the pyramid discussed above. The difference is not large in most cases [Sederberg 88a] and since the computation of this "tighter" pyramid is more costly, it is also not used in this work.

Figures 4-7 to 4-10 present several examples, where the bounding pyramids and the bounding cones for several surfaces are determined to provide a comparison of the two methodologies. These figures present the parametric patch and the $u$, $v$ and normal bounding pyramids and cones from both formulations. Table 4-I presents the surface area of the unit sphere, that is enclosed by the cones and the pyramids (both sides) for the same examples to compare the tightness of the different formulations. These areas are presented as a percentage of the whole surface area of the sphere. Spherical trigonometry is used to compute these areas as shown in Appendix D [Beyer 82]. The first patch in Figure 4-7 has non-zero mixed partial derivatives $R_{uv} \neq 0$. As can be seen there and in Table 4-I, the bounding pyramids are tighter than the bounding cones. In addition, the normal rectangular
pyramid can be defined, as opposed to the cone formulation, where the normal cone is very large with half angle larger than \( \pi/2 \) and the normal cone is not defined. The second patch in Figure 4-8 has again non-zero mixed partial derivatives \( R_{uv} \neq 0 \). Again the rectangular pyramids provide a tighter bound to the derivative directions of the surface than the bounding cones. The normal cone is very large and is not defined, while the normal pyramid is defined. The third patch in Figure 4-9 has zero mixed partial derivatives \( R_{uv} = 0 \) and planar isoparameter curves for both parametric directions, which also lie on parallel planes. In this case the \( u \) and \( v \) tangent directions lie in a plane. The rectangular pyramid in this case degenerates to a two-dimensional wedge and is a tight bound to the derivative directions, unlike the bounding cones. The fourth patch in Figure 4-10 is a piece of a torus and has planar isoparameter curves in one of the parametric directions also lying on parallel planes. In this case also the bounding pyramids are tighter than the bounding cones. As can be seen in Table 4-I the surface area enclosed by the rectangular pyramids is usually reduced by a factor of two as compared to the surface area enclosed by the cones for most of the examples shown.
Table 4-I: Comparison of Surface Area on the Unit Sphere Enclosed by Rectangular Pyramids and Cones, Expressed as a Percentage of the Surface Area of the Sphere

<table>
<thead>
<tr>
<th>Figure Number</th>
<th>Type of Bound</th>
<th>Rectangular Pyramid % of Sphere Area</th>
<th>Cone % of Sphere Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-7</td>
<td>U Deriv.</td>
<td>11</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>V Deriv.</td>
<td>13</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>Normal Vector</td>
<td>56</td>
<td>N/D</td>
</tr>
<tr>
<td>4-8</td>
<td>U Deriv.</td>
<td>16</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>V Deriv.</td>
<td>5</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>Normal Vector</td>
<td>53</td>
<td>N/D</td>
</tr>
<tr>
<td>4-9</td>
<td>U Deriv.</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>V Deriv.</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Normal Vector</td>
<td>13</td>
<td>15</td>
</tr>
<tr>
<td>4-10</td>
<td>U Deriv.</td>
<td>11</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>V Deriv.</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Normal Vector</td>
<td>13</td>
<td>25</td>
</tr>
</tbody>
</table>

N/D = Not defined.
Figure 4-7: Bounding Pyramids and Cones for a Bezier Patch - Example 1
Figure 4-8: Bounding Pyramids and Cones for a Bezier Patch - Example 2
Figure 4-9: Bounding Pyramids and Cones for a Bezier Patch - Example 3
Figure 4-10: Bounding Pyramids and Cones for a Piece of a Torus - Example 4
Chapter 5

SIGNIFICANT POINTS IN INTERSECTION PROBLEMS

5.1 Definitions

In order to enable efficient non-exhaustive detection of all features of an intersection set, we introduce and employ the concept of significant points of the intersection.

5.1.1 Algebraic Surface to Rational Parametric Surface

For the algebraic curve $F(u,v) = 0$ representing the intersection of an algebraic surface with a rational polynomial patch, the following significant points of the algebraic curve are defined [Farouki 86a, Prakash 88a]:

**Border** points at which at least one parametric variable takes values equal to the borders of the parametric domain, e.g. $F(u,0) = 0$ for the $v=0$ border.

**Turning** points at which the normal vector to the curve is parallel to the parametric $u$ or $v$ axes. These may be defined by the following two sets of simultaneous equations

\[
F(u,v) = 0, \text{ and } F_u(u,v) = 0, \text{ v-turning points} \tag{5.1}
\]

and

\[
F(u,v) = 0, \text{ and } F_v(u,v) = 0, \text{ u-turning points} \tag{5.2}
\]

**Singular** points at which the first partial derivatives vanish,

\[
F(u,v) = F_u(u,v) = F_v(u,v) = 0 \tag{5.3}
\]

Once all real solutions of (5.1) and (5.2) in the rectangular domain of interest are found, the singular points can be readily identified as the common solutions of (5.1) and (5.2).

For the computation of the border points of the algebraic curve of intersection, using
the properties of the Bernstein basis we obtain a univariate polynomial in the \( u \) or \( v \) direction. For \( v=0 \) border points, for example,

\[
F(u,0) = \sum_{i=0}^{n} w_{i,0} B_{i,m}(u) = 0 \tag{5.4}
\]

The real roots of the above type of polynomials of degree \( m \) or \( n \) in the parameter space of the control surface may be found using recursive binary subdivision and the variation diminishing property that directly exploit the Bernstein basis [Lane 81]. To enhance efficiency this method may be used just for root isolation in an interval, whereas accurate root computation may be performed with efficient direct numerical techniques [NAG 89].

The computation of turning and singular points can be reduced to the solution of two bivariate implicit polynomials as in (5.1) or (5.2). Two such equations can, in theory, be solved using elimination techniques of algebraic geometry. [Prakash 88a] found that the high degree of the resulting polynomials and the characteristics of the elimination process, involving large scale computation, contribute to substantial loss of accuracy making the method unattractive for practical applications.

For these reasons [Prakash 88a, Prakash 88b] employed direct numerical techniques (minimization followed by Newton methods) to compute these points. These methods depend only on evaluations of \( F(u,v) \) and its derivatives, which are of relatively low degree compared to the polynomials arising from elimination and, hence, suffer less inaccuracy in their computation. The above numerical methods require initial approximations. To obtain such approximations, a coarse subdivision of the control polyhedron of the curve of intersection is intersected with the control plane to obtain a coarse approximation of the intersection curve. This coarse approximation does not reflect the true connectivity of the curve, but was found to provide good initial approximations for most turning and singular points [Prakash 88a]. More details on the computation of turning and singular points for the algebraic surface to rational polynomial surface problem and a reliable way to verify computation of all such points can be found in [Prakash 88b].
The problem of computing turning points as expressed by simultaneous equations (5.1) or (5.2) can be also thought of as a special case of the problem of computing intersections of two general planar algebraic curves. This problem has been recently addressed by [Sederberg 88c] and [Johnstone 89]. [Sederberg 88c] applies the classical theory of polar curves to compute all points at which two algebraic curves intersect, to compute double points (singular points) of an algebraic curve, as well as to compute all points on an algebraic curve which have a specified tangent direction (e.g. turning points). [Sederberg 88c]'s algorithm is an interval type algorithm and utilizes the simple form of the polar curves of an algebraic curve, when expressed in the Bernstein form. The user defines a triangle within which the algorithm searches for all intersection points of two algebraic curves. If there is a single intersection point, the algorithm behaves as follows. With each iteration, the algorithm determines how far it can safely shrink the triangle and still enclose the intersection point. If there is more than one intersection point, the triangle is split in four subtriangles and each of the four subtriangles is processed in a similar fashion. Singular points are determined by computing the intersections of three polar curves, while turning points can be determined by computing the intersections of two algebraic curves. [Johnstone 89] presented a local method for the computation of the intersections of two plane algebraic curve segments. This method is based on marching (crawling) along the curves. Intersection points are found by crawling along the two curve segments in a coordinated fashion. This method is particularly applicable, when the intersection points between specific segments of two algebraic curves are required instead of all the intersection points of the two algebraic curves.

5.1.2 Rational Polynomial Surfaces

For the rational polynomial surface intersection case, we define the significant points of the intersection set as manifested in the parameter space of one of the two patches. The significant points in the \( u-v \) parameter space of patch \( r(u,v) \) are defined as follows:
Border points. These are points of the intersection at which at least one of the parametric variables \( u \) and \( v \) takes value equal to the border of the \( u-v \) parametric domain, e.g. \( r(u,0) - q(s,t) = 0 \) for the border points with \( v=0 \).

Termination points. These are points of the intersection at which at least one of the parametric variables \( s \) and \( t \) of the second rational polynomial patch takes value equal to the border of the \( s-t \) parametric domain, e.g. \( r(u,v) - q(0,t) = 0 \) for the termination points with \( s=0 \). These points indicate abrupt ending of intersection branches in the parameter space of the first patch.

Turning points. These are points at which the tangent vector to the intersection curve, as projected in the \( u-v \) parameter space, is parallel to the \( u \) or \( v \) parametric axes of patch \( r(u,v) \). The tangent vector of the intersection curve is given by

\[
t = (r_u x r_v)x(q_s x q_t)
\]

(5.5)

and is assumed to be non-zero. Using this definition of the tangent vector \( t \), the turning points may be defined by the following conditions

\[
r_u \text{ parallel to } t, \text{ i.e. } r_u x t = 0, \text{ for } v\text{-turning points}
\]

(5.6)

and

\[
r_v \text{ parallel to } t, \text{ i.e. } r_v x t = 0, \text{ for } u\text{-turning points}
\]

(5.7)

Singular points. These are points of the intersection curve at which the two surface patch normals are collinear, i.e.

\[
t = 0
\]

(5.8)

Collinear Normal points. These are points on the two parametric surfaces at which the normal vectors are collinear. These points do not necessarily lie on the intersection curve of the two surfaces but, as will see below, provide essential information about the topology of the intersection curve. If these points lie on the intersection curve, they are also singular points.
The singular points can be readily identified if common solutions of (5.6) and (5.7) are found (allowing \( t = 0 \) in such solutions) or if collinear normal points are found, which also lie on the intersection curve.

An alternative definition of turning, singular and collinear normal points can be obtained using the oriented distance function defined in equation (3.7). According to this definition, turning points are points on the intersection curve, \( \phi(u,v) = 0 \), at which the gradient vector of the oriented distance function, \( \nabla \phi \), is parallel to the parametric \( u \) or \( v \) axes. Turning points are defined by the following sets of simultaneous equations:

\[
\begin{align*}
\phi &= 0, \quad \text{and} \quad \phi_u = r_u \cdot n_2 = 0, \quad \text{\( v \)-turning points} \\
\phi &= 0, \quad \text{and} \quad \phi_v = r_v \cdot n_2 = 0, \quad \text{\( u \)-turning points}
\end{align*}
\]  

(5.9)  

(5.10)

with \( n_2 \) the unit normal vector on surface \( q(s,t) \). Equations (5.9) and (5.10) provide identical results with equations (5.6) and (5.7). Note the similarity of the above two sets of equations with equations (5.1) and (5.2). The singular points are the points on the intersection curve at which the first partial derivatives of the distance function vanish

\[
\phi = \phi_u = \phi_v = 0
\]  

(5.11)

Most of the collinear normal points are critical points of the oriented distance function \( \phi \), where the gradient vector to the distance function is zero. By definition a critical point satisfies

\[
\phi_u = \phi_v = 0
\]  

(5.12)

with \( \phi \) not necessarily 0 there. Collinear normal points are critical points of the distance function \( \phi \), provided \( r(u,v) \) does not intersect the cut locus of \( q(s,t) \) and, therefore, we stay away from areas where the gradient of the distance function is ill-defined. An example where a collinear normal point is not a critical point of the distance function is seen in the third example in Figure 3-2. In that example, point \( p \) is not a critical point of the vector field defined on surface \( S_1 \), although the normal at point \( p \) is collinear with the normal at the
local extremum of surface $S_2$. However, this collinear normal pair is a critical point (point $p_2$) of the vector field defined on surface $S_2$ the swept surface, the distance function of which is shown on Figure 5-1.

**Figure 5-1:** Distance Function of Example 3 of Figure 3-2 Defined on the Swept Surface

![Diagram showing critical points $p_1$, $p_2$, and $p_3$ on $S_2$.]

$S_1$ $S_2$

$p_1$ $p_2$ $p_3$ Critical Points of Vector Field on $S_2$

Figure 5-2 presents an intersection curve in the parameter space of a parametric patch and identifies the various significant points on this curve. The algorithms proposed in this thesis do not require the computation of the turning points of the intersection curve in the parameter space of the second patch. These are usually different than the turning points in the parameter space of the first patch. If these points are required in the geometric representation of the intersection curve or for efficient processing of trimmed patches, they can be also computed with the same technique, described in the following sections, before the intersection computation. They can be also computed after completion of tracing by numerical iterative techniques, as good approximations for these points can then be readily obtained.

For the computation of the border and termination points of the intersection, a piecewise rational polynomial curve to piecewise rational polynomial surface intersection
**Figure 5-2:** Significant Points of an Intersection Curve

- • border point
- ★ termination point
- ◊ u turning point
- ○ v turning point
- ✽ singular and collinear normal point
- ◈ collinear normal point
capability is required. A combination of subdivision and numerical techniques is used in this thesis to compute these points with high accuracy. Alternative methods for the solution of the parametric curve to surface intersection problem can be found in [Barnhill 87, Chiang 89b].

Direct numerical techniques (minimization and Newton methods) were used for the computation of turning, singular and collinear normal points. These methods depend only on evaluations of $r(u,v)$, $q(s,t)$ and their partial derivatives and require initial approximations. The following section describes the techniques we developed for computing good initial approximations to the various types of significant points of the intersection curve.

5.2 Initial Approximations to Significant Points in Intersections of NURBS Patches

5.2.1 Turning and Singular Points

To obtain initial approximations to turning and singular points, we use a coarse initial subdivision (splitting) of the two parametric patches so that the control polyhedra provide a coarse approximation to the two surfaces. The use of bounding boxes assists in efficiently eliminating non-intersecting subpatches during this initial subdivision process. The control polyhedra faceted approximations of the two surfaces are subsequently intersected to provide a coarse approximation to the intersection curve. This coarse approximation does not necessarily reflect the true connectivity or the number of branches of the curve, but was found to provide good initial approximations for most turning and singular points. One level of initial subdivision (binary splitting) was adequate in most of the examples presented in this thesis to obtain starting points to most significant points. In some examples, two levels of initial subdivision provided better approximations to most significant points. The algorithm for the intersection of the faceted approximations of the two surfaces is described in more detail in Chapter 8.
For every two adjacent approximate intersection points in a triangular facet, the cross products \( r_v \times t \) and \( r_u \times t \) are computed at the approximate \( u, v \) and \( s, t \) parameter values of the approximate points. Vectors \( r_v, r_u \) and \( t \) are coplanar. As a result the direction of the cross products \( r_v \times t \) and \( r_u \times t \) changes sign as we march on the intersection curve away from a \( u \) or \( v \) turning point respectively (see also equations (5.6) and (5.7)). If the directions of these vectors at the two approximate intersection points differ by an angle larger than \( \pi/2 \), then we have a possibility of a cross-product direction reversal and a turning point in the vicinity of the two approximate points. We, therefore, use the middle point of the intersection segment as an initial approximation for \( u \) or \( v \) turning points depending on the cross-product which exhibits this behavior. Alternatively, at the two intersection points, the inner product \( r_v \cdot n_2 \) or \( r_u \cdot n_2 \) can be computed at the approximate \( u, v \) and \( s, t \) parameter values of the points, where \( n_2 \) is the unit normal vector on surface \( q(s,t) \). If any of these inner products changes sign (\( \phi_v \) and \( \phi_u \) change sign in equations (5.10) and (5.9)), then we have a possibility of a turning point in the vicinity of the two approximate points. We, therefore, use the middle point of the intersection segment as an initial approximation for \( u \) or \( v \) turning points depending on the inner-product which exhibits this behavior. Both methods to determine initial approximations to turning and singular points are equivalent. The latter method is more efficient to implement as it requires only inner products.

Figure 5-3 presents the coarse approximation of the intersection curve between a biquartic patch and a plane after two levels of initial subdivision. This approximation does not reflect the true connectivity of the curve. The starting points for the significant points of this intersection curve as computed from our method are also shown (dark bullets for \( v \) turning points and light bullets for \( u \) turning points). This intersection curve actually has ten turning points and three singular points [Prakash 88b]. Our method provides initial approximations for the three singular and the eight turning points. The significant points of this intersection as computed by the numerical techniques to be described in the following
section, together with the intersection in the u-v parameter space are shown in Figure 5-19. The actual three-dimensional intersection curve is shown in Figure 10-11. Figure 5-4 presents the coarse approximation of the intersection curve between two Bezier patches of degree six in each parametric direction after one level of initial subdivision. This intersection actually has twelve turning points, and our method provides initial approximations for all these points, as can be seen in this Figure. The computed significant points together with the intersection in the u-v parameter space are shown in Figure 5-20. The actual three-dimensional intersection curve is shown in Figure 10-17. Figure 5-5 presents the coarse approximation of the intersection curve between a bicubic Bezier patch and a quadratic-cubic Bezier patch after one level of initial subdivision. This intersection actually has three turning points and one singular point, and our method provides initial approximations for all these points as can be seen in this Figure. The computed significant points, together with the intersection in the u-v parameter space are shown in Figure 5-21. The actual three-dimensional intersection curve is shown in Figure 10-14. Figure 5-6 presents the coarse approximation of the intersection curve between a torus (represented as a rational bi-quadratic B-spline patch) and a bicubic Bezier patch after one level of initial subdivision. This intersection actually has six turning points, and our method provides initial approximations for five of these points. The computed significant points, together with the intersection in the u-v parameter space are shown in Figure 5-22. The actual three-dimensional intersection curve is shown in Figure 10-20. As can be seen from these examples, initial approximations for most turning and singular points can be obtained by intersection of a coarse approximation of the two surfaces. Figure 5-7 presents the coarse approximation of the intersection curve between two biquartic Bezier patches after two levels of initial subdivision. This intersection actually has sixteen turning points, and our method provides initial approximations for all these points as can be seen in this Figure. The computed significant points, together with the intersection in the u-v parameter space
are shown in Figure 5-23. The actual three-dimensional intersection curve is shown in Figure 10-18.

The methodology used to compute turning and singular points of the intersection set is a numerical approach and as a result there is no theoretical guarantee that in general
- at least one initial approximation exists for every significant point, nor that,
- the iterative technique will converge to the appropriate significant point.

In order to enhance the confidence in the intersection results, Chapters 6 and 7 describe some additional techniques, which may be used to verify the detection of all features of an intersection set.

The above methodology can be used to assist in the computation of initial approximations to turning and singular points for the general parametric to parametric intersection problem. In that case, the control polyhedra of the two surfaces are not defined. However, a lattice of points can be defined on each of the parametric surfaces, which provide a piecewise linear approximation of the surface. Intersecting these piecewise linear approximations of the two surfaces, we can test for the possibility of the existence of a turning point in a similar fashion. Finally, we can use the direct numerical techniques from this Chapter to compute these significant points accurately.

5.2.2 Border and Termination Points

A similar procedure is used to obtain starting points for border and termination points. A coarse approximation of each boundary curve of one patch obtained with subdivision (splitting) is intersected with a coarse approximation of the other patch and the resulting intersection points are computed and used as starting points for the numerical solution method described in the following section. Bounding boxes are used to eliminate non-intersecting curves and patches. One level of initial subdivision for the boundary curve and the patch provided good initial approximations of border and termination points for all the examples in this thesis.
Figure 5-3: Starting Points For Significant Point Computation of Intersection of
\[ w = -6u^4 + 21u^3 - 19u^2 - 6u^2v^2 + 11uv^2 + 3v^2 - 4v^4 = 0 \] with the Plane \( w = 0 \).
Two Levels of Initial Subdivision.
Figure 5-4: Starting Points For Significant Point Computation of Intersection Between Two Patches of Degree Six in Each Variable. One Level of Initial Subdivision.
Figure 5-5: Starting Points For Significant Point Computation for a Bicubic Bezier to Quadratic - Cubic Bezier Patch Intersection. One Level of Initial Subdivision.
Figure 5-6: Starting Points For Significant Point Computation for a Rational Biquadratic B-Spline to Bicubic Bezier Patch Intersection. One Level of Initial Subdivision.
Figure 5-7: Starting Points For Significant Point Computation of Intersection Between Two Biquartic Bezier Patches. Two Levels of Initial Subdivision.
5.2.3 Collinear Normal Points

[Sederberg 89c, Sederberg 89a] was the first to recognize the importance of collinear normals in detecting the existence of closed intersection loops in intersection problems. In the above references he suggested a method for verifying non-existence of collinear normal vectors in intersection problems using an interval type of analysis. He also recognized the importance of computing collinear normal points as opposed to determining their absence and suggested a subdivision method to detect these points. Recently [Markot 89] used Newton techniques to compute collinear normal points which lie on the intersection curve, (i.e. singular points). He uses the vector field defined by the gradient of the distance function to compute these points. His methodology does not provide a technique to automatically determine initial approximations for all these points. In this work, we develop an automatic method to compute initial approximations to collinear normal points and a method to accurately and efficiently compute these collinear normals using direct numerical techniques.

The computation of good initial approximations to the collinear normals is the first element in a successful iterative numerical solution method. Ideas from the theory of plane vector fields [Krasnoselskiy 66] are used to determine these initial approximations. The vector field of interest is the vector field defined by the gradient, $\nabla \phi$, of the oriented distance function $\phi(u,v)$ between two parametric surfaces introduced in Chapter 3 (see equation (3.8)). Most collinear normal points between the two surfaces are critical points of $\phi$, since they are minimum distance point pairs and the gradient vector of the distance function at these points is zero. There are some collinear normal points between two surfaces which are not critical points of $\phi$, as they are not minimum distance point pairs. In this context, the rotation number of the vector field $V = \nabla \phi$ along a closed curve $\gamma$ in the $u$-$v$ parameter space is important. The rotation number $W(V, \gamma)$ counts the number of rotations performed by the gradient vector $\nabla \phi(\gamma(t))$ while the point $\gamma(t)$ moves along the
curve $\gamma$ on the parameter space of $r(u,v)$. In the definition of the rotation number one needs to assume that $\nabla \phi \neq 0$ on the curve $\gamma$. The theorem used is the following:

**Theorem 2:** If the rotation number of the vector field along an arbitrary closed curve in the domain is non-zero, then there must exist at least one critical point of $\phi$ in the domain bounded by the closed curve. The theorem assumes that the vector field is continuous.

**Proof:** Let $D$ be a simply connected domain with boundary $\gamma$. If the continuous vector field $V$ is never zero on $D$, then the rotation $W(V, \gamma)$ is zero see eg. [Krasnoselskiy 66]. Hence if $W(V, \gamma) \neq 0$ then $V$ has at least one zero inside $D$. The boundary curve $\gamma$ need only be an oriented continuous curve.

The vector field $\nabla \phi$ is not continuous in a connected domain if the distance function $\phi$ is not proper at a point in the interior of the domain. For most of the intersection examples which are difficult to resolve (tangent, nearly tangent surfaces), we have found that, after a small level of initial subdivision the vector field is well defined at all points of the domain.

The first step in the determination of initial approximations to collinear normal points is the approximate computation of the vector field of the gradient of the oriented distance function on a lattice of points in the parameter space of the first patch, $r(u,v)$. We use the same coarse initial subdivision of the two parametric patches as with the turning and singular point computation to provide a coarse approximation to the two surfaces. The use of bounding boxes assists in efficiently eliminating non-intersecting subpatches during this initial subdivision process. For each control point $R_{ij}$ of the first surface, we calculate its Euclidean distance from all the control points of all the subpatches of the second surface $Q_{kl}$ and select the minimum distance pair. If the second surface is a plane, the minimum distance point can be computed exactly and easily by projecting the control point of the first surface on the plane to find the nearest point. The pair of minimum distance points is then identified in the parameter space of the two patches, to determine approximate parameter values of the two nearest points $(u, v, s, t)_{ij}$. The parameter values are the corresponding node values of the knot vectors of the two subpatches, i.e. $u = n_{u,i}$, $v = n_{v,j}$, $s = n_{s,k}$, $t = n_{t,l}$, where $n_{a}$ is the node vector in the a direction [Gordon 74]. The $s, t$ parameter values define
an approximate nearest point in the second surface to the point with \( u, v \) parameter values in the first surface. As a result the gradient of the distance function \( \phi \)

\[
\nabla \phi = V = [V_1, V_2]^T
\]

(5.13)
can be approximately computed at these points by evaluating \( V_1 = n_2(s,t) \cdot r_u(u,v) \) and \( V_2 = n_2(s,t) \cdot r_v(u,v) \). This computation is repeated for all control points in the subpatches of the first surface and the plane vector field of the oriented distance function between the two surfaces is thus approximated on a lattice of points. As an example, the top part of Figure 5-9 displays the vector field \( V \) of the distance function between the two surfaces shown in Figure 10-12 computed from the above methodology after one level of initial subdivision for both patches. In this figure the "normalized" vector field \( \nabla \phi/\| \nabla \phi \| \) is used, i.e. a unit vector of the same direction as the vector field. The "normalized" vector field has the same vector rotation properties as the original vector field see eg. [Krasnoselskiy 66].

Once the approximate vector field is determined, Theorem 2 may be used to determine initial approximations for the collinear normals. Every four neighboring points in the lattice of points form a quadrilateral, i.e. an oriented continuous closed curve. The rotation \( \rho \) of the vector field around this quadrilateral can be computed by computing the rotation of the vectors in the corner points of the quadrilateral. For each corner vector, the vector angle \( \theta \) is determined by computing

\[
\theta = \text{Arctan}(V_2/V_1) \quad \text{with} \quad -\pi < \theta \leq \pi
\]

(5.14)
The rotation \( \rho \) of the vector field around the quadrilateral is given by

\[
\rho = \sum_{i=0}^{3} \Delta \theta_i
\]

(5.15)

where \( \Delta \theta_i = \theta_{(i+1)\text{mod}4} - \theta_i \) and \( -\pi < \Delta \theta_i \leq \pi \)

where \text{mod} is the modulo operator. The rotation index of the quadrilateral is then defined by \( W = \rho/2\pi \). If the index is non-zero, the quadrilateral has a possibility of a collinear
normal (i.e. a critical point of $\phi$, if the oriented distance function is well-defined) in its interior and the parameter values of the center of the quadrilateral are used as initial approximations to compute this collinear normal as the following section describes. If the index is zero nothing is done. Figure 5-8 displays two such quadrilaterals with a possibility of a collinear normal point in their interior. This is only an approximate estimation of the rotation number of the vector field around the quadrilateral, but was found to be sufficient in all the examples of this thesis for an efficient computation of initial approximations to collinear normal points. Chapter 6 describes a verification test, which is applied to detect collinear normal points that might have not been identified in this approximate computation. If one of the computed vectors of the field is null the above calculation is unnecessary since the corresponding point can be used as an initial approximation to a collinear normal point.

![Diagram](image)

Rotation Index = 1  
Rotation Index = -1

**Figure 5-8:** Quadrilateral with Possibility of Collinear Normal Point in its Interior
Some examples illustrate the above methodology. The first thing to compare is the quality of the approximation of the vector field of the distance function between the two surfaces. By finding the minimum distance pairs of the control points of the two surfaces, we only approximate the vector field of the distance function. This approximation gets better and better if we increase the initial subdivision of the two surfaces since the control polyhedra approximate better the two surfaces. However, this results in an unnecessary increase in the computation time since the number of control points needed to be processed increases substantially. For all the examples in this thesis, one or two levels of initial subdivision were adequate to provide good approximation to the vector field and good initial approximations to collinear normal points. Once the approximate nearest points are obtained, they can be used as initial approximations to find the exact numerical nearest points in the two surfaces. That is for the point \( r_{ij} = r(u_i, v_j) \) on the first surface, we numerically minimize the objective function

\[
\text{Minimize} \quad \|r_{ij} - q(s, t)\|^2
\]

(5.16)

to determine the \( s, t \) parameter values of the point in the second surface nearest to point \( r_{ij} \). The \( (s, t) \) values associated with the nearest control point are used as initial values for the computation of the exact nearest point. Appendix C provides some details about the distance minimization between a point and a surface. The top part of Figure 5-9 displays the approximate vector field of the distance function between the two surfaces of Figure 10-12 after one level of initial subdivision as determined above and the lower part displays the accurate vector field determined using as initial approximation the approximate field. The comparison is good. Figure 5-10 displays another comparison example for the two surfaces shown in Figure 10-14. Again one level of initial subdivision is used for the approximate computation and the comparison is good. Small differences in the approximate vector field are not so important, since the important element in our computation is the rotation index of a quadrilateral in the field (an integer value), which
should not change. As can be seen from these examples the approximate vector field computation is quite accurate and avoids the additional numerical minimizations to compute nearest points which increase the computational time of the overall calculation. The verification test in Chapter 6 uses an accurate numerical computation of the vector field in order to detect collinear normal points, that may be missed in this initial computation.

Figures 5-11 through 5-17 present some examples from the computation of the collinear normal points using our methodology for a number of surface pairs. The top part of each Figure displays the approximate vector field of the distance function of the surface pair, with the initial approximations to the collinear normal points determined from our methodology (big filled circle), while the bottom part displays the same vector field with the exact numerical collinear normal points as computed using the numerical techniques outlined in the following section (big filled circle). Figure 5-11 displays the result for the surface pair shown in Figure 10-12 and the intersection shown in Figure 5-24. In this example there is one collinear normal point, which is computed by our method. The existence of a collinear normal assists in rapidly identifying the small intersection loop. Figure 5-12 displays the result for the surface pair shown in Figure 10-14 and the intersection shown in Figure 5-21. In this example there are two collinear normal points, which are computed by our method. One of the collinear normal points is also a singular point of the intersection curve (right point), while the other identifies the existence of an intersection loop (left point). Figure 5-13 displays the result for the surface pair shown in Figure 10-19. In this example there are two collinear normal points, which are computed by our method. This example has a special interest since the two surfaces are nearly coincident. Figure 5-14 displays the result for the surface pair shown in Figure 10-11 and the intersection shown in Figure 5-19. In this example, there are nine collinear normal points of which seven are computed by our method using one level of initial subdivision (splitting). Three of these collinear normal points are also singular points as can be seen in
Figure 5-9: Comparison of Approximate and Accurate Vector Fields of Distance Function Between Surfaces in Figure 10-12

Approximate Vector Field

Accurate Vector Field
Figure 5-10: Comparison of Approximate and Accurate Vector Fields of Distance Function Between Surfaces in Figure 10-14

Approximate Vector Field

Accurate Vector Field
Figure 5-19. Figure 5-15 displays the result for the surface pair shown in Figure 10-17 and the intersection shown in Figure 5-20. In this example there are five collinear normal points. There are five initial approximations provided but only four of the collinear normals are computed by our method using one level of initial subdivision (splitting). More details on how to resolve situations like this are provided in Chapter 6. Three of these collinear normal points are interior to and identify the three loops in this intersection. Figure 5-16 displays the result for the surface pair shown in Figure 10-18 and the intersection shown in Figure 5-23. In this example there are nine collinear normal points. Initial approximations are provided for all these points using one level of initial subdivision (splitting). Four of these collinear normal points are interior to and identify the four small loops in this intersection. Figure 5-17 displays the result for the surface pair shown in Figure 10-22 of two torii leading to a near singular intersection. In this example, there are nine distinct collinear normal points (four appear twice in the Figure since the torus is a closed surface). There are many initial approximations provided since there are several quadrilaterals satisfying the rotation index condition of Theorem 2 but not all of them contain critical points of the oriented distance function. This intersection example is one case, where the vector field on one of the torii is not well defined at all points. There are points on one torus, which are on the cut locus of the other torus, similar to example one in Figure 3-2. This is the reason for the many starting points to collinear normal points in this example, since each of the ill-defined points triggers the rotation index criterion. The final points are only the actual collinear normal points.

As can be seen from the above examples, our methodology is very good in obtaining initial approximations for most (usually all) collinear normal points in a number of complicated examples with minimal initial subdivision (splitting). Most initial approximations are very close to the actual collinear normal points. More information on the identification of all collinear normal points between two surfaces is provided in Chapters 6 and 7.
The above methodology can be easily extended to the general parametric to parametric surface intersection problem. In that case, we would evaluate a lattice of points on the two surfaces and use these points to determine the closest distance pairs. The vector field of the gradient of the oriented distance function can then be approximated at these nearest distance pairs and the determination of initial approximations for the collinear normal computation can be similarly performed. The numerical techniques of the following section can then be used to accurately compute the collinear normal points.

The automatic determination of initial approximations for collinear normal points between two surfaces is equivalent to automatic determination of initial approximations for the extrema (maxima, minima) and saddle points of an objective height function. As was also mentioned in Chapter 3 the vector field of the oriented distance function for this case is well-defined at all points in the domain. In addition, all collinear normal points are extrema or saddle points of the height function and are therefore critical points of the vector field of the oriented distance function. One of the difficult problems in non-linear constrained optimization is the identification of all extrema points of a non-linear function in order to identify its global minimum and maximum [Gill 81]. The example shown in Figures 5-3, 5-19 and 10-11 is indicative of the power of our technique. In this case we have a biquartic explicit Bezier patch defined over plane z=0, a non-linear height function of two variables, and we compute initial approximations to all the critical points of this function and use these to accurately compute the actual critical points of the function. Our methodology is able to identify correctly initial approximations to the nine critical points of this function (3 saddle points, 3 minima and 3 maxima points). Figure 5-18 displays these computed critical points. For clarity, the biquartic Bezier surface is translated in the z direction. The present methodology using the plane vector field of the oriented distance function to identify collinear normal points is applicable for height functions of two variables. It is interesting to investigate the extension of this methodology to objective functions of three or more variables by developing similar ideas for multidimensional vector fields.
The following section describes the techniques developed for accurately computing the various types of significant points of the intersection curve using direct numerical techniques once the initial approximations to these points are obtained as described in this section.

5.3 Direct Numerical Solution Methods

Each of the initial approximations for the border, termination, turning, singular and collinear normal points is used to iteratively compute these points accurately using the geometric or algebraic conditions that these points satisfy, as described in Section 4.1. Since we are interested in robust algorithms which can solve for some high order singularities, the solution method must be able to handle the presence of points in the domain where the Jacobian of a function is singular. These situations can be handled by modified Newton algorithms or as minimization problems using a modified Gauss-Newton method [NAG 89]. In both minimization and Newton methods the required derivatives for the solution should be explicitly available at all points in the domain to contribute to the reliability of these schemes. Computation of the required derivatives using numerical differentiation should be avoided, since errors in numerical differentiation can adversely affect the stability of numerical computations [Gill 81]. In our implementation, all required derivatives are explicitly available at all points in the parametric domain of the two surfaces. For the implementation of the minimization problem, the Hessian of the objective function at each step of the iteration is explicitly available to improve the reliability of convergence to the solution.

In numerical experimentation with both Newton and minimization methods on a large number of examples selected for their complexity and diversity and with a coarse initial subdivision, it was found that the minimization scheme was stable in that it always produced a solution very close to the correct solution but usually with reduced accuracy.
Figure 5-11: Initial Approximations and Numerically Computed Collinear Normal Points for Surfaces in Figure 10-12

Initial Approximate Points

Final Numerically Computed Points
Figure 5-12: Initial Approximations and Numerically Computed Collinear Normal Points for Surfaces in Figure 10-14

Initial Approximate Points

Final Numerically Computed Points
Figure 5-13: Initial Approximations and Numerically Computed Collinear Normal Points for Surfaces in Figure 10-19

Initial Approximate Points

Final Numerically Computed Points
Figure 5-14: Initial Approximations and Numerically Computed Collinear Normal Points for Surfaces in Figure 10-11

Initial Approximate Points

Final Numerically Computed Points
Figure 5-15: Initial Approximations and Numerically Computed Collinear Normal Points for Surfaces in Figure 10-17

Initial Approximate Points

Final Numerically Computed Points
Figure 5-16: Initial Approximations and Numerically Computed Collinear Normal Points for Surfaces in Figure 10-18

Initial Approximate Points

Final Numerically Computed Points
Figure 5-17: Initial Approximations and Numerically Computed Collinear Normal Points for Surfaces in Figure 10-22

Initial Approximate Points

Final Numerically Computed Points
Figure 5-18: Critical Points of Quartic Bezier Patch in Figure 10-11
The Newton technique on the other hand provided a much better accuracy in the solution compared to the minimization method but occasionally failed to converge if the initial approximation was not close to the significant point. Similar observations have been also reported in [Prakash 88b, Patrikalakis 90a]. Hence, in order to exploit both the stability of the minimization method and the increased accuracy provided by the Newton method, the minimization method was used first, followed by the Newton method. This proved successful in a variety of complex cases and led to accurate computation of significant points to the intersection.

5.3.1 Constrained Minimization Techniques

In this section we summarize the modified Gauss-Newton method for constrained minimization problems [Gill 74]. The usual problem to be solved is the minimization of an objective function $F(u,v,s,t)$ subject to the constraint that the variables (parameters) are within the parametric domain of the parametric patches. The condition for a minimum of a function of four variables is that the gradient of the function vanishes ($F_u = F_v = F_s = F_t = 0$) at the minimum. The numerical solution is iterative and starts with an initial approximation vector $r^0 = [u^0, v^0, s^0, t^0]^T$. At each iteration step a correction vector is determined, which is added to the current approximation. The k-th iteration is governed by equations obtained from a Taylor expansion of the conditions for the minimum of the objective function. At the k-th iteration step we have:

$$r^{k+1} = r^k + \alpha^k \delta^k$$  \hspace{1cm} (5.17)

where $r^k$ is the solution vector at iteration k, $\delta^k$ is the direction of the step to be taken at the k-th iteration and $\alpha^k$ is the step length. The vector $\delta^k$ is obtained as the solution of the following system:

$$H^k \delta^k = -g^k$$  \hspace{1cm} (5.18)
where $g^k$ is the gradient vector of $F(u,v,s,t)$ (including the first derivatives of $F(u,v,s,t)$), and $H^k$ is the Hessian matrix of second derivatives of $F(u,v,s,t)$ both evaluated at $r^k$. The solution for the correction $\delta^k$ is obtained by a modified Cholesky factorization of $H^k$ where the modification handles the case of the Hessian being indefinite or singular [NAG 89]. Another important modification is the change in the solution procedure whenever the gradient of the function becomes less than a certain tolerance and the matrix $H$ is not positive definite as in the case of saddle points of $F(u,v,s,t)$. This method of solution requires the use of the second derivatives of the objective function, which means the use of up to the third partial derivatives of the parametric surfaces. The step length $\alpha^k$ is chosen so that $F(r^{k+1}) < F(r^k)$ and is computed using an one-dimensional optimization technique based on quadratic or cubic safeguarded polynomial approximation [NAG 89].

Various **objective functions** were studied for the minimization problem to compute border, termination, turning, singular and collinear normal points. It is good practice in a minimization problem, to select objective functions and variables that provide a sensible scaling in the neighborhood of the solution [Gill 81]. In our implementation, we tried to select objective functions that belong in the interval [0, 1] in the neighborhood of the solution. We also reparametrize the surfaces, so that their parameters take values in the interval [0, 1]. In addition, since most objective functions contain the distance vector between the two surfaces before performing any intersection computation, the two parametric patches are translated and scaled to fit in a cube [-1,1]x[-1,1]x[-1,1]. The objective functions are chosen so that at the computed significant points are zero. The objective functions selected for the various problems are as follows.

For the computation of **border and termination points** in the $u$-$v$ parameter space the objective function is:
\[
\begin{align*}
\text{Minimize} & \quad F(s,t,\tau) = \|q(s,t) - r(\tau)\|^2 \quad \text{for border points or} \\
\text{Minimize} & \quad F(u,v,\tau) = \|r(u,v) - q(\tau)\|^2 \quad \text{for termination points} \\
\end{align*}
\] (5.19) (5.20)

where \(q(\tau)\) and \(r(\tau)\) represent the boundary curves in each of the two cases. For example \(r(\tau) = r(0,v)\) or \(r(1,v)\) or \(r(u,0)\) or \(r(u,1)\).

For the computation of turning and singular points in the \(u-v\) parameter space the objective function is chosen as follows.

For \(u\) turning points:

\[
\begin{align*}
\text{Minimize} & \quad F(u,v,s,t) = \|r - q\|^2 + \left(\frac{r_v \cdot n_2}{\|r_v\|}\right)^2 \quad \text{or} \\
\text{Minimize} & \quad F(u,v,s,t) = \|r - q\|^2 + 1 - \left(\frac{r_v \cdot t}{\|r_v\| \|t\|}\right)^2 \\
\end{align*}
\] (5.21) (5.22)

and for \(v\) turning points,

\[
\begin{align*}
\text{Minimize} & \quad F(u,v,s,t) = \|r - q\|^2 + \left(\frac{r_u \cdot n_2}{\|r_u\|}\right)^2 \quad \text{or} \\
\text{Minimize} & \quad F(u,v,s,t) = \|r - q\|^2 + 1 - \left(\frac{r_u \cdot t}{\|r_u\| \|t\|}\right)^2 \\
\end{align*}
\] (5.23) (5.24)

where \(\|\cdot\|\) is the magnitude of the vector, \(\cdot\) denotes inner product and \(n_2\) is the unit normal vector on the second surface. If \(t\) or both second terms in equations (5.21) and (5.23) approach zero, as near a singular point, the minimization automatically shifts to the following objective function

\[
\begin{align*}
\text{Minimize} & \quad F(u,v,s,t) = \|r - q\|^2 + \left(\frac{r_v \cdot n_2}{\|r_v\|}\right)^2 + \left(\frac{r_u \cdot n_2}{\|r_u\|}\right)^2 \quad \text{or} \\
\text{Minimize} & \quad F(u,v,s,t) = \|r - q\|^2 + 1 - (n_1 \cdot n_2)^2. \\
\end{align*}
\] (5.25) (5.26)

where \(n_1\) is the unit normal vector on the first surface. The \(1 - (n_1 \cdot n_2)^2\) term in equation (5.26) specifies that at a singular point the two surfaces have collinear normals.
For the computation of **collinear normal points** in the u-v parameter space the objective function is:

\[
\text{Minimize} \quad F(u,v,s,t) = \|r - q\|^2 - [(r - q) \cdot n_1]^2 + 1 - (n_1 \cdot n_2)^2
\]  
\[\text{(5.27)}\]

The first two terms in the right hand side of equation (5.27) identify an intersection point, or points with difference vector \(r - q\) parallel to \(n_1\), while the remaining terms identify points with parallel normal vectors. When the objective function is zero at a pair of points, a collinear normal point pair is determined. This objective function does not only compute the critical points of the oriented distance function, but also can compute collinear normal points between the two surfaces which do not form a closest distance pair. The first and second derivatives of the objective functions in equations (5.21), (5.23), (5.25) and (5.27) are given in Appendix A.

With the alternative definition of the significant points using the distance function \(\phi\), the objective function for the computation of **singular points** could be defined by

\[
\text{Minimize} \quad F(u,v) = \phi^2 + \phi_u^2 + \phi_v^2
\]  
\[\text{(5.28)}\]

where \(F(u,v) = 0\) at a minimum for a singular point. Similarly, an objective function for the computation of **critical points** of the oriented distance function could be defined by

\[
\text{Minimize} \quad F(u,v) = \phi_u^2 + \phi_v^2
\]  
\[\text{(5.29)}\]

where \(F(u,v) = 0\) at a minimum for a critical point. For the evaluation of the oriented distance function and its derivatives, the \(s, t\) parameters of the minimum distance point to a point with parameter \(u\) and \(v\) need to be determined. In the event of encountering higher order singularities of multiplicities greater than two it may be required to modify the objective function with the vanishing derivatives near such points. Whenever all derivatives of a certain order vanish in the converging sequence approaching the solution, the minimization function could be adaptively augmented by an additional derivative (see also [Prakash 88a, Prakash 88b]). For example, near triple point singularities all second order
derivatives vanish in addition to the first order derivatives. In such a case the objective function would be modified to

\[ \text{Minimize } F(\mu, \nu) = \phi^2 + \phi_{\mu}^2 + \phi_{\nu}^2 + \phi_{\mu\mu}^2 + \phi_{\mu\nu}^2 + \phi_{\nu\nu}^2 \]  

(5.30)

to ensure convergence at these points. We have constructed analytic expressions for the second and higher derivatives of the oriented distance function required in the numerical computations for minimization, in terms of derivatives of the surface equations. This is remarkable because the distance function is generally thought of as procedural requiring extensive numerical computations and iteration. The second derivatives of the distance function can be found in Appendix B. An additional (fourth) parametric derivative of the two surfaces will be required in this case for the computation of the second order partial derivatives of the objective function in equation (5.30).

5.3.2 Newton Iteration

The problem to be solved is the solution of a set of \( n \) non-linear equations \( F \) for \( n \) unknowns. The Newton method is an iterative scheme for determining a solution to a system of non-linear equations. From an initial approximation to the solution, we march to the solution using local Taylor expansions of the non-linear equations. At the \( k \)-th iteration step of the second order Newton method, a correction \( \delta^k \) is determined and the current solution vector \( r^k \) is updated using

\[ r^{k+1} = r^k + \frac{\delta^k}{2} \]  

(5.31)

\[ \delta^k = [G(r^k)]^{-1} F(r^k) \]  

(5.32)

\( G \) is the matrix of first derivatives of the non-linear functions \( F \) with respect to the unknown variables evaluated at the previous iteration step and superscript -1 denotes matrix inverse.

For the Newton iteration in the computation of the border and termination points of
the intersection curve we need to solve the system of equations \( Q(s,t) - R(\tau) = 0 \) or \( R(u,v) - Q(\tau) = 0 \). This is a system of three equations for the three unknown parameters in each case.

For the Newton iteration in the computation of the turning and singular points of the intersection curve, we need a system of four equations for the four unknown parameters \( u, v, s \) and \( t \). Three of the equations come from the three components of equation (3.6) \((r(u,v) - q(s,t))\) and the fourth equation comes from the second and third terms in the objective functions for the minimization problem, equations (5.21), (5.23) and (5.25). For objective functions (5.21), (5.23), the fourth equation is \( F_4 = \frac{r_u}{\|r_u\|} \cdot n_2 \) and \( F_4 = \frac{r_v}{\|r_v\|} \cdot n_2 \) respectively.

For objective function (5.25) the fourth equation is of the form \( f^2 + g^2 \). This function has a global minimum at a point with \( f(u,v,s,t) = 0 \) and \( g(u,v,s,t) = 0 \). For an extremum \( \nabla(f^2 + g^2) = 0 \) and, consequently, when \( f = 0 \) and \( g = 0 \) for a singular point, the fourth equation has always at least a double point. A second order Newton iteration on such a function will converge at most linearly. To remedy this, the contribution of the fourth equation to the system of equations (value of fourth equation \( F_4 \)) during the Newton iteration is always doubled to ensure faster convergence to the solution [Dahlquist 74], i.e. \( F = [F_1, F_2, F_3, 2F_4]^T \). The same situation is present for the objective functions in equations (5.22), (5.24) and (5.26). For these cases the fourth equation is of the form \( 1 - f^2 \), where \( |f| \leq 1 \) and \( f \) has a global minimum at a point with \( f = 1 \) or \(-1 \) i.e. a turning or singular point. The same remedy is also used in this case.

For the Newton iteration in the computation of the collinear normal points of the two surfaces, we need a system of four equations for the four unknown parameters \( u, v, s \) and \( t \). Four independent equations which a collinear normal line satisfies are the following [Sederberg 89c]:
\begin{align*}
F_1 &= (r - q) \cdot r_u = 0 \\
F_2 &= (r - q) \cdot r_v = 0 \\
F_3 &= (r_u \times r_v) \cdot q_s = 0 \\
F_4 &= (r_u \times r_v) \cdot q_t = 0
\end{align*}

and \( \times \) denotes cross-product. Equations (5.33) and (5.34) specify that the normal vector to surface \( r \) and the distance vector \( (r - q) \) are collinear, while equations (5.35) and (5.36) specify that the normal vectors in the two surfaces are parallel and, therefore, also collinear.

### 5.4 Examples

The above method of obtaining initial approximations and computing significant points is very useful because it does not depend on external information to initiate the procedure. In addition, it allows computation of most turning, singular and collinear normal points with very little initial subdivision and rapidly convergent minimization and Newton-like iteration. The key difference between many marching techniques used in intersection solutions and the method developed here is that no knowledge of initial starting points on every segment of the curve is required by our procedure. The lattice and marching intersection methods of [Varady 83, Barnhill 87, Rossignac 87, Bajaj 88, Chen 88] could profit from the above turning, singular and collinear normal point computation, as described in Chapter 9.

Some final results from the computed significant points for some of the examples presented in this Chapter are presented next. In the following figures the intersection in the parameter space of the first patch is shown together with the significant points of the intersection computed by the numerical techniques described in this Chapter, which are shown using circles. Small dark filled circles are used for \( v \) turning points and singular points, small light filled circles are used for \( u \) turning points, border and termination points, and small unfilled circles are used for collinear normal points which are not singular points.
Figure 5-19 presents the result of intersecting a biquartic Bezier patch with a plane also shown in Figure 5-3 and Figure 10-11. In this example and with two levels of initial subdivision (splitting), we are able to find the three singular points of the intersection, eight of the ten turning points and the six additional collinear normal points (two points outside the intersection curve are not shown). Figure 5-20 presents the result of intersecting two Bezier patches of degree six in each parametric direction also shown in Figure 5-4 and Figure 10-17. In this example and with one level of initial subdivision, we find ten of the twelve turning points and four of the five collinear normal points. Figure 5-21 presents the result of intersecting a bicubic Bezier patch with a quadratic-cubic Bezier patch also shown in Figure 5-5 and Figure 10-14. In this example and with one level of initial subdivision, we find the singular point, the three turning points, the two border points and the collinear normal point of the intersection. Figure 5-22 presents the result of intersecting a torus (rational biquadratic B-spline patch) and a bicubic Bezier patch also shown in Figure 5-6 and Figure 10-20. In this example with one level of initial subdivision, we find five of the six turning points, and the three collinear normal points of the intersection. Figure 5-23 presents the result of intersecting two biquartic Bezier patches also shown in Figure 5-7 and Figure 10-18. In this example and with two levels of initial subdivision, we find the sixteen turning points and the nine collinear normal points of the intersection between the two surfaces. Figure 5-24 presents the result of intersecting two bicubic Bezier patches also shown in Figure 5-11 and Figure 10-12. In this example and with one level of initial subdivision, we find all four turning points, and the one collinear normal point of the intersection. An additional level of initial subdivision (splitting) assists in determining the significant points that are missed in the above examples. More details on the verification of significant point computation is provided in Chapters 6 and 7.
5.5 Algorithmic Issues

The determination of initial approximations to significant points and the computation of these points using the direct numerical techniques presented in this Chapter provide a very efficient way to identify features of the intersection curve to assist in the actual tracing of this curve. To provide some insight on the computational effort required for the computation of these points, several test calculations have been performed. The determination of starting points to turning and singular points requires the intersection of the facets of the coarsely subdivided surfaces. If the number of control points in a Bezier patch is n, this calculation requires 2(n-1) edges of the first patch to be intersected with the 2(n-1) triangular facets of the second patch. More details on this intersection will be given in Chapter 8. For k levels of initial subdivision this requires a maximum of 4(n-1)^2 16^k line to plane intersections. This assumes that bounding boxes are not used to eliminate non-intersecting subpatches. The operation count decreases significantly when k increases and bounding boxes are used. Ten of the intersection examples presented in this thesis were analyzed to determine CPU times for the execution of this algorithm. The computer used is a Silicon Graphics Iris-4D 120GTX Graphics Workstation with only one processor operating. For such a machine, the aggregate time required to perform 4 million floating point operations (1 million of each additions, subtractions, multiplications and divisions) was tested to be around 6 CPU seconds. The rating of a single processor of the Iris-4D 120GTX is 1.54 MFLOPS (million floating point operations per second). It should be mentioned that one addition or one subtraction are considered one floating point operation, while one multiplication is considered four floating point operations and one division is considered six floating point operations. For the intersection examples with biquadratic and bicubic patches the average time for the computation of all initial approximations to turning and singular points was 1.67 CPU seconds. For the degree six surfaces in Figure 10-17 the computation time was 18.5 CPU seconds. The computation time required for the
computation of turning and singular points using the direct numerical techniques (minimization and Newton iteration) of this Chapter are as follows. For biquadratic and bicubic patches, the average time per turning or singular point is 0.5 CPU seconds. This average is computed from a sample of 65 points. The above average increases to 1.82 CPU seconds/per point for the degree six surfaces in Figure 10-17. This average is computed from a sample of 12 points. The majority of the time is spent in evaluating the position and the partial parametric derivatives of the patch. In our implementation, B-splines are evaluated using the recursive formula in equation (3.17) for which the cost of computing all derivatives is equal to the cost of computing only one derivative.

The determination of initial approximations to collinear normal points requires the computation of distances between the control points of the two surfaces, evaluation of the vector field of the distance function at the control points of the first patch and computation of the rotation index of the resulting quadrilaterals. If the number of control points in a Bezier patch is $n$, this calculation requires $n^2$ point to point distance computations and comparisons and $n$ evaluations of the vector field for each pair of subpatches. For $k$ levels of initial subdivision this requires a maximum of $16^k n^2$ point to point distance computations and $4^k n$ vector field evaluations. This also assumes that bounding boxes are not used to eliminate non-intersecting subpatches. The operation count decreases significantly when $k$ increases and bounding boxes are used. The same intersection examples were analyzed to determine CPU times for the execution of this algorithm. For the intersection examples with biquadratic and bicubic patches, the average time for the computation of all initial approximations to collinear normal points was 0.56 CPU seconds. For the degree six surfaces in Figure 10-17 the computation time was 3.66 CPU seconds. The computation time required for the computation of the collinear normal points using the direct numerical techniques (minimization and Newton iteration) of this Chapter are as follows. For biquadratic and bicubic patches, the average time per collinear normal point is
0.3 CPU seconds. This average is computed from a sample of 45 points. The above average increases to 1.3 CPU seconds per point for the degree six surfaces in Figure 10-17. This average is computed from a sample of 5 points. It is important to note that the computation of collinear normal points is considerably faster than the computation of turning and singular points, since it does not require the computation of an initial approximation to the intersection curve and the initial approximations to the collinear normal points are very close to the actual points due to the local nature of the rotation number computation.

Figure 5-19: Intersection and Significant Points for Surface
\[ w = -6u^4 + 21u^3 - 19u^2 + 6uv^2 + 11uv^2 + 3v^2 - 4v^4 = 0 \]
with the Plane \( w = 0 \)
Shown in Figure 10-11 Two Levels of Initial Subdivision.
**Figure 5-20:** Intersection and Significant Points for Two Patches of Degree Six in Each Variable Shown in Figure 10-17. One Level of Initial Subdivision.

- Collinear Normal Points
- V-Turning Points or Singular Points
- U-Turning Points
Figure 5.21: Intersection and Significant Points for a Bicubic Bezier Patch with a Quadratic - Cubic Bezier Patch Shown in Figure 10.14. One Level of Initial Subdivision.
**Figure 5-22:** Intersection and Significant Points for a Rational Biquadratic B-Spline Patch with a Bicubic Bezier Patch Shown in Figure 10-20. One Level of Initial Subdivision.
Figure 5-23: Intersection and Significant Points for Two Biquartic Bezier Patches Shown in Figure 10-18. Two Levels of Initial Subdivision.
Figure 5-24: Intersection and Significant Points for Two Bicubic Patches Shown in Figure 10-12. One Level of Initial Subdivision.
Chapter 6

FEATURE DETECTION USING TOPOLOGY PROPERTIES OF THE VECTOR FIELD OF THE DISTANCE FUNCTION

6.1 Motivation

This and the following Chapter present criteria, which may be used to detect the various features of an intersection curve between two surfaces. The problem addressed in these Chapters is robustness or reliability of the intersection algorithm. There are two main methods for numerically tracing the curve of intersection of two surfaces. These are marching and subdivision methods, which were reviewed in detail in Chapter 2. A major difficulty of marching methods is that in order to detect and trace all branches of an intersection curve at least one starting point is required in each intersection branch. Subdivision methods do not require starting points to intersection branches, but require substantial computation time to trace small features of an intersection (small loops) as well as to resolve closely spaced features of an intersection (singularities, near singularities and tangencies of two surfaces along a curve).

One of the major contributions of this thesis is the development of techniques to automatically compute approximations of significant points of an intersection set, i.e. border, termination, turning, singular and collinear normal points, and then use numerical techniques to compute these points very accurately. The details of these techniques are presented in Chapter 5. These techniques have been found to usually provide starting points in all intersection branches and, consequently, improve the robustness of marching methods. In addition, these techniques identify the singular points and the small intersection loops which slow down subdivision methods and, as a result, improve their efficiency by reducing the level of subdivision required to accurately trace the various intersection branches.
However, due to the numerical nature of our approach to compute the significant points of an intersection curve, there is no guarantee that in general

- at least one initial approximation exists for every significant point, nor that,
- the iterative technique will converge to the appropriate significant point.

Some significant point computation examples where significant points were missed are described in Chapter 5.

Chapter 5 presented two methodologies for the automatic computation of significant points of the intersection curve. The first methodology computes turning and singular points of the intersection using a coarse approximation of the intersection curve to provide approximations to these points. The second methodology computes collinear normal points (which include all singular points) of the two surfaces using an approximation of the vector field of the gradient of the oriented distance function defined on the parameter space of one of the surfaces to provide approximations to these points. The collinear normal points of interest are the critical points of the vector field of the distance function. The objective of both computations is to automatically provide at least one point on or inside all intersection loops and identify all singular points of the intersection. This information together with the additional computation of border and termination points of the intersection, provides sufficient data to assist in tracing the intersection curve correctly.

In this manner, all border, termination, turning and singular points of the intersection give enough information to trace the complete intersection set. This is also true for the set of all border and termination points and the critical points of the vector field of the gradient of the oriented distance function. This was alluded to in Chapter 3. A closed loop of the intersection of two surfaces, corresponds to a closed level curve $c$ of the distance function $\phi$, i.e. $\phi(u,v) = 0$. The parameter domain enclosed by such a curve contains at least one extremum of function $\phi(u,v)$ and consequently a critical point of $\phi$, i.e. a point where $\nabla \phi = 0$. This is a well-known theorem in topology that appears in various forms see eg.
[Krasnoselskiy 66]. The only requirement of the theorem is that the vector field \( V = \nabla \phi \) should be continuous at all points inside the closed domain. This requirement might cause trouble in some of the cases presented in Chapter 3, where the oriented distance function is ill-defined at certain points in the domain. For \( C^1 \) surfaces with Lipschitz continuous derivatives, these situations occur away from actual intersections and the use of subdivision is sufficient to eliminate the presence of these regions as will be also explained below.

The objective of this Chapter is to develop a technique that verifies the computation of critical points of the oriented distance function. The method detects missed critical points and then computes these missed critical points by locally and adaptively applying the minimization technique of Chapter 5. Under a set of conditions, the method developed in this Chapter provides an efficient way to compute the critical points of the vector field, increasing substantially the confidence in the results of the intersection computation. This technique is applicable for intersections with only a finite number of critical points of the oriented distance function.

[Sederberg 89c] identified the importance of collinear normal points in identifying the topology of the intersection curve and proved a theorem about the presence of collinear normal points in the interior of a loop of an intersection curve. [Sederberg 89c] suggests the use of interval arithmetic to verify the non-existence of collinear normals. Absence of collinear normals implies absence of intersection loops. He applies interval arithmetic to equations (5.33) to (5.36) to examine the non-existence of collinear normal points. As can be seen in Chapter 7 this process requires substantial levels of subdivision to isolate and remove the collinear normal points and is slow in resolving singularities. [Sederberg 89c] realized the importance of computing collinear normal points as opposed to determining their non-existence and also suggested a linearly convergent subdivision technique to detect these points. According to that technique, the two surfaces are continuously subdivided in a binary fashion until two sufficiently small patches are
obtained which simultaneously satisfy equations (5.33) to (5.36). This methodology to compute collinear normal points is slow and requires substantial subdivision computation before any intersection tracing is actually performed. The technique proposed in this work, computes collinear normal points with quadratically convergent techniques and is very efficient in verifying their computation with occasional little amount of subdivision.

6.2 Results from the Theory of Plane Vector Fields

There are several results from the theory of plane vector fields, which are useful in the development of our method. Some of these, such as the rotation number of a vector field along a closed curve and Theorem 2 were introduced in Section 5.2.2. Theorem 2 is very useful and is used to identify cases where our numerical calculations have missed a critical point of the vector field of the distance function. According to this theorem, if the rotation number of the vector field \( V = \nabla \phi \) along a closed curve in the domain is non-zero, there must exist at least one critical point of \( \phi \) in the domain bounded by the closed curve. This theorem was based on another theorem see eg. from [Krasnoselskiy 66] according to which:

**Theorem 3:** Suppose that a continuous vector field \( V \) in a closed region \( D \) has no null vectors (corresponding to critical points), then the rotation \( W(V, \gamma) \) of the field on the boundary \( \gamma \) of the region is zero.

The converse of this theorem, i.e. if the rotation of the field on the boundary of a region is zero, then there are no critical points in the interior of the region, could provide a sufficient condition for the verification of the computation of all critical points of \( \phi \) but unfortunately it is not always true for the reasons to be outlined below.

Each of the isolated critical points of a vector field has an **index** associated with it. This index measures the rotation number of the vector field on a curve that encloses this and only this critical point. The **Poincare index theorem** see eg. [Krasnoselskiy 66] specifies that:

**Theorem 4:** The rotation number (index) of a continuous vector field on the boundary of a simply connected region equals the sum of the indices of all isolated critical points in this region.
It can be shown that the index of the gradient field of a function $\phi$ at an isolated extremum of that function is 1, while the index of the gradient field of that function at an isolated saddle point is -1. The index of a simple critical point can be easily determined, since it has the sign of the determinant of the Hessian of the oriented distance function $\phi$, provided this determinant is not zero. For a region containing one saddle and one extremum point of the vector field, the rotation of the field on the boundary of the region according to the Poincare index theorem is $1 + (-1) = 0$, which proves that the converse of Theorem 3 does not hold.

However, the numerical techniques presented in Chapter 5 usually compute all critical points and seldom miss one or more of these points as was also shown in the examples of Chapter 5. Our numerical experiments suggest that it is very unlikely that the numerical technique of Chapter 5 misses two neighboring critical points of the vector field with opposite index signs. As a result, the use of the converse of Theorem 3, although it does not in general guarantee the result of the computation, it increases our confidence in the results of the numerical computation of critical points.

From the intersection test cases examined in this work, in one special example our verification computation failed to identify a critical point of the intersection curve. This example, shown in Figure 10-15, is the ramphoid algebraic curve which has a cusp in the center of the parametric domain of the surface representing the ramphoid. Figure 6-1 presents the exact vector field of the oriented distance function for this surface as computed from the techniques of Chapter 5 and Figure 6-2 shows the intersection curve with all significant points. This field has two critical points, one of which (interior to intersection loop) is computed by our method. The cusp at the center of the domain is never computed by this algorithm, since the rotation of the vector field on any closed curve surrounding this critical point (and not enclosing the other critical point) is zero. This is an example of a degenerate critical point. At the cusp the determinant of the Hessian of the distance
function is zero, i.e. \( \phi_{uu} = \phi_{uv} = 0 \), but \( \phi_{vv} \neq 0 \). This singular point of the intersection curve has a single tangent direction (\( v = 0.5 \)). Functions with degenerate critical points must be treated by higher than second order techniques. It should be mentioned that our alternative methodology to compute turning and singular points of the intersection using an approximation of the intersection curve computes this cusp correctly, as can be seen in Figure 6-2, which displays all significant points for this intersection.

6.3 Verification Method

Table 6-I outlines the steps of the verification method implemented in this work. These are described in detail below. The first step after the initial computation of collinear normal points of the two surfaces using the techniques of Chapter 5 is to subdivide the two parametric surfaces at these points. This subdivision splits the first patch to a number of subpatches each of which will be examined for a possibility of a collinear normal point in its interior. It should be remembered that some of these subpatches have critical points of the oriented distance function at their corners. The bounding box of each of the subpatches of the first surface is compared with the bounding boxes of the subpatches of the second surface to determine which of the subpatches have no intersection. This is an important step of the algorithm to reduce the number of verification tests. The subpatches of both surfaces are kept in trees for speedy access as described in Chapter 8.

Each of the intersecting subpatches of the first surface is then examined for the existence of a critical point of the oriented distance function in its interior. For this test the rotation number of the vector field on the boundary of the subpatch needs to be determined. If the rotation number is non-zero, then there is a possibility of a missed critical point and we need to search for this critical point. This is the test we apply at all the intersecting subpatches. If the rotation number is zero, the present test is inconclusive, although, as we found, such a condition is rarely associated with a critical point.
Figure 6-1: Exact Vector Field for Ramphoid Surface in Figure 10-15
Figure 6-2: Intersection and Significant Points for Ramphoid Surface
\[ w = u^4 - 2u^2v + u^2v^2 - uv^2 + v^2 = 0 \] with the Plane \( w=0 \).
Two Levels of Initial Subdivision
The rotation number on a closed curve can be determined using the Poincare formula [Krasnoselskiy 66]:

$$W(V, \gamma) = \frac{1}{2\pi} \int_{\gamma} \frac{\chi d\psi - \psi d\chi}{\chi^2 + \psi^2}$$  \hspace{1cm} (6.1)$$

where \(V(u,v) = (\chi, \psi)\) is the vector field, and \(\gamma\) is the closed curve. To compute the rotation from equation (6.1) we can use numerical quadrature. Since it is known a priori that the rotation along a closed curve is an integer, we only need to evaluate this integral to within an error of less than 0.5 to determine the rotation. Alternatively, we can evaluate the angle \(\theta\) of the vector field at \(n\) points in the boundary curve using

$$\theta = \text{Arctan}(\psi/\chi)$$ \hspace{1cm} (6.2)$$

where the arc tangent assumes values in the range \(-\pi < \theta \leq \pi\). The rotation \(W\) of the vector field around the quadrilateral is then given by

$$W = \frac{1}{2\pi} \sum_{i=0}^{n-1} \Delta \theta_i$$ \hspace{1cm} (6.3)$$

where \(\Delta \theta_i = \theta_{(i+1)\text{mod}n} - \theta_i\) and \(-\pi \leq \Delta \theta_i \leq \pi\)

where mod is the modulo operator and is used to account for the first point which is also the last point at which the vector field is evaluated. The absolute angle change \(\Delta \theta\) in the vector field between two adjacent points in the closed curve is not allowed to be larger than \(\pi\). For most of the examples in this thesis twenty points per boundary curve were adequate for estimation of the rotation of the vector field around the boundary of a subpatch. If any point on the boundary of the subpatch (except the corners) has a null vector, the above computation cannot be performed and the test is assumed to have failed. In order to avoid integrating over critical points which are already computed and are at corners of subpatches, the boundary curve is modified as shown in Figure 6-3 using a small tolerance to avoid the critical point. This Figure displays an example of a boundary curve with rotation number 1. This modification is acceptable, since we want to address intersection
examples with isolated critical points. This tolerance indicates the distance between critical points that we want to resolve. The vector field of the distance function at the various points on the closed boundary curve in the u-v parameter space is evaluated using a numerical technique to determine the point on the second surface which is the minimum distance from a point on the boundary curve. This numerical technique was also described in Chapter 5 and is explained in Appendix C. A better alternative to this computation is the use of the differential equations describing the orthogonal projection of the boundary curve on the second surface to determine the minimum distance points on the s-t parameter space and compute the vector field at these points. The differential equations describing the orthogonal projection of a curve on a surface can be found in [Pegna 89] and one of their variations is discussed in Chapter 9, where tensorial differential equations are used to trace the intersection curve.

**Figure 6-3:** Example of Rotation Computation on Patch Boundary Curve

If the rotation number on the boundary curve is non-zero, we search locally in the
subpatch for a missed critical point. This search is performed using the methodology of Section 5.2.2 for the computation of critical points. First, the approximate vector field defined as the gradient of the oriented distance function is evaluated at a lattice of points (for each control point of the subpatch) and then initial approximation points are determined for the interior of this subpatch using the methodology of Section 5.2.2. At least one initial approximation is expected to be determined in this case and the critical point corresponding to this point is determined using the numerical techniques of Section 5.3 and the objective function given by equation (5.27) within the domain of the subpatch. Once a new critical point is determined, the subpatch is subdivided at this point and the verification test is performed recursively in the new subpatches until it is satisfied in all of them. This process is repeated until all the subpatches of the first patch satisfy this test.

6.4 Applications

The verification methodology described in the previous sections has proven successful in the large number of complex intersection examples examined in this thesis. It enables identification and computation of collinear normal points which are missed in the preliminary significant point computation and improves substantially the confidence in the results of the intersection computation. Figure 6-4 presents an example of the application of this method to the intersection of the surfaces in Figure 10-17. The significant points of this intersection case were computed in Chapter 5 and were shown in Figure 5-20. For this example the methodology of Chapter 5 was able to determine the four of the five collinear normal points of the intersection (using one level of initial binary subdivision for the two patches also shown in Figure 6-4). Thus initially the intersection domain is split only at the four collinear normal points. The result is shown at the top part of Figure 6-4 which displays the intersection, the significant points and the intersection domain for this case. When the verification test developed above is used, the subdomain with the missing
Table 6-I: Algorithm for Verification of Computation of Critical Points of the Oriented Distance Function Between Two Surfaces

Subdivide both surfaces at each of the available significant points

For each subpatch of the first surface:
{
    1. Determine if it has possibility for intersection, by comparing its bounding box, with the bounding boxes of all subpatches of the second surface. If there is no possibility of intersection, skip the next three steps.

    2. Evaluate the rotation number of the vector field of the distance function on the boundary curve of the subpatch. Use minimization to evaluate the vector field on the boundary curve and numerical quadrature to evaluate the rotation number.

    3. If the rotation number is zero, the verification test is inconclusive.

    4. else if the rotation number is non-zero, the verification test fails.

    { 
        Evaluate initial approximations to the critical points of the vector field in the subpatch using the methodology of Section 5.2.2.

        Compute the missed critical point(s) using the direct numerical techniques of Section 5.3.

        Subdivide both surfaces at the new critical points.

        For each of the new subpatches of the first surface apply recursively the verification test starting in step 1.
    }
}
}
collinear normal point is identified and the collinear normal computation methodology is applied locally to determine the missing collinear normal point. The result is shown in the lower part of Figure 6-4 (small unfilled circles specify the collinear normal points of the intersection).

The remaining set of examples illustrate some of the special cases encountered in the application of this methodology. These concern cases where the gradient of the oriented distance function is not well defined. These situations were described in Section 3.2 and were illustrated in Figure 3-2. The first type of special situation is the fourth example of Figure 3-2. This situation is the result of the definition of the oriented distance function in a bounded domain in which \( n_2 \) and \( r - q \) for two minimum distance points in the sense of Section 3.2 are not collinear. These special points occur always away from the intersection, unless they are adjacent to border or termination points which have already been computed. In the application of our verification test in a sub-domain these points are not specifically identified but are used like proper minimum distance points and the vector field is approximately evaluated at the parameter values of these minimum distance points and is used to determine the rotation of the vector field on the boundary curve. Sometimes, these points might cause the rotation to be non-zero. The verification test then fails and the local collinear normal computation is triggered. An initial approximation is determined but this point fails to converge, since there is no actual collinear normal point in the subdomain. For these cases we split the two surfaces at the initial approximation and then apply the verification criterion in the new subpatches. Since these special regions are not near the actual intersection, the bounding box intersection test for the new subpatches allows us to eliminate the subpieces which cause trouble and enables satisfaction of the verification test. Figure 6-5 shows an example from such a situation. In this Figure the intersection curve, the significant points and the intersection domain are shown for the surfaces shown in Figure 10-13. This intersection curve, has two termination points, two turning points and
Figure 6-4: Verification Computation for Intersection Example in Figure 10-17
one collinear normal point. One level of initial subdivision was used to determine these points. The top part of Figure 6-5 shows the intersection domain, without the application of the verification test, while the bottom part shows the intersection domain after the application of this test. In two regions of this domain, the topology criterion is not satisfied due to the situation described above. As shown in this Figure a single additional local level of subdivision is adequate to resolve these situations using the bounding boxes of the new subpatches.

The third example illustrates the other type of difficulty with the definition of the oriented distance function. This is the situation illustrated in the first three examples of Figure 3-2. For these examples, points on the parameter space of one surface are on the cut locus of the other surface and as a result there are more than one equal minimum distance points. At these points the vector field of the gradient of the oriented distance function is ill-defined. For $C^1$ surfaces with Lipschitz continuous derivatives the cut locus stays away from the surface [Wolter 85] and, therefore, these points occur away from the actual intersection. It is usually the case, that these points create an effect similar to a critical point in their neighborhood, i.e. the rotation number of the vector field on a closed curve around these points is non-zero. The second and third examples of Figure 3-2 exhibit this phenomenon. In the application of our verification test in a sub-domain such points in the interior of a sub-domain cause the verification test to fail and trigger the local collinear normal computation. A starting point is determined but this point fails to converge, since there is no proper collinear normal point in the subdomain. For these cases the remedy of the previous example is still applicable. Again we split the two surfaces at the starting point which is close to the undefined vector field point and then apply the verification criterion in the new subpatches. Since these points are not near the actual intersection, the bounding box intersection test for the new subpatches allows us to eliminate the subpieces which cause trouble and enables satisfaction of the verification test. Figure 6-6 shows an
Figure 6-5: Verification Computation for Intersection Example in Figure 10-13
example from such a situation in the intersection of two cylinders of equal radius at right angles to each other. In this Figure the intersection curve, the computed significant points and the intersection domain are shown for the cylinders also illustrated in Figure 10-23. The intersection curve has two singular points (two are identical in the above Figure). One level of initial subdivision was used to determine these points after splitting of the rational B-spline surfaces representing the cylinders in their rational polynomial Bezier elements. The top part of Figure 6-6 shows the intersection domain, without the application of the verification test, while the bottom part shows the intersection domain after the application of this test. In two regions of this domain, the topology criterion is not satisfied due to the undefined vector field point. As shown in this Figure the first point is resolved using one additional local level of subdivision, while the second point is resolved using two additional local levels of subdivision.

The application of this methodology to verify the computation of all relevant collinear normal points of an intersection curve provides a very effective and efficient way to identify features of the intersection curve and improve the confidence in the results of the intersection. To provide some insight on the computational effort required for the application of this test several calculations have been performed. In order to apply this verification test in a subpatch, the rotation of the vector field on the boundary curve of the subpatch needs to be determined. This requires the evaluation of the vector field of the distance function at N points in the boundary of the subpatch to assist in the computation of the integral of the vector field on this curve. A point to surface minimum distance problem needs to be solved for each of these points. This is solved using the minimization technique described in Appendix C. On the average twenty points per boundary curve were sufficient for the accurate computation of the rotation number of the vector field. The verification test has to be applied to all the subpatches of the first surface which have a possibility of an intersection. The number of subpatches in the intersection domain of the first patch depends
Figure 6-6: Verification Computation for Intersection Example in Figure 10-23
linearly on the number of significant points used to split this domain as will be described in more detail in Chapter 8. Twelve of the intersection examples presented in this thesis were analyzed to determine CPU times for the execution of this verification test. For the intersection examples with biquadratic and bicubic patches the average time for the verification of one subpatch was 0.7 CPU seconds (sample of 219 subpatches). For the degree six surfaces in Figure 10-17 the average time for verification of a subpatch was 2.61 CPU seconds (sample of 20 subpatches). Table 9-III in Chapter 9 presents the total time required for this verification test in a number of examples.

The verification test described in this Chapter provides a powerful technique to substantially improve the confidence in the identification of all segments of the intersection of two surfaces by determining all relevant collinear normal points of the intersection. In contrast to the methods of the following Chapter, this methodology does not provide a guarantee of detection of all features of the intersection, but increases substantially the probability of detection of all these features at only a small computational cost. A combination of the methodologies of this and the following Chapter is also possible in an effort to improve the reliability of the significant point computations.
Chapter 7

FEATURE DETECTION USING CONVEXITY PROPERTIES

7.1 Introduction

The previous chapter developed a technique to verify the computation of critical points of the oriented distance function to enable detection of all singularities and small loops in an intersection set. This method detects missed critical points using topology properties of the vector field of the gradient of the oriented distance function and then computes these missed critical points by locally and adaptively applying the minimization technique of Chapter 5. This method is a powerful and not costly technique in the identification of all segments of the intersection of two surfaces, although it does not provide a theoretical guarantee of detection of all features.

This Chapter describes criteria which provide a theoretical guarantee of detection of all loops and singularities in an intersection set. These criteria are based on convexity properties and the determination of bounds for the partial derivatives and normal vectors of rational B-spline surfaces. This Chapter describes techniques utilizing these criteria that can be used to isolate small features of an intersection set such as singular points and small loops or that can be used to enhance the confidence in the significant point computation and the detection of all features of an intersection set.

7.2 Loop Detection Using Convexity Properties of Rational Spline Surfaces

The first set of techniques described in this Chapter provide a condition, which if satisfied guarantees that there is no missing intersection loop in the interior of an intersection domain. The intersection algorithms developed in this work, use the significant
points of the intersection which are computed with the methods of Chapter 5 to adaptively subdivide the two surfaces at these points. The satisfaction of the loop detection condition for the subpatches resulting from adaptive subdivision provides a theoretical verification of the success of the numerical process used to determine all singular points and to identify all intersection loops. In addition, binary subdivision based on the loop detection condition can be used to identify singular points and regions with small loops or closely spaced features of an intersection set. This is an alternative method providing initial approximations to singular points or collinear normal points which identify small features of the intersection.

There has been a recent interest in developing techniques to detect loops in intersection curves [Sinha 85, Prakash 88b, Sederberg 88a, Sederberg 89c, Cheng 88, Patrikalakis 90a]. These methods are reviewed extensively in Chapter 2, and some relevant information will be repeated here. [Sinha 85] proved that if two at least $C^1$ surfaces intersect in a closed loop, then there exists a normal vector on one surface which is parallel to a normal vector in the other surface (parallel normal vectors). [Sinha 85] suggested use of this condition to assist in selective subdivision of two surfaces that are intersected to identify all intersection segments. If bounds to the normal vector directions of two subpatches do not intersect, these subpatches do not intersect in an interior loop and further subdivision is unnecessary.

[Sederberg 88a] extended the above work and derived a condition under which an intersection curve must contain a point on at least one patch boundary curve. He proved that if two at least $C^1$ surfaces intersect in a closed loop (in both parametric spaces), then there exists a normal vector on one surface which is perpendicular to a tangent vector on the other surface. To be able to test for this condition, [Sederberg 88a] introduced cones that bound the tangent directions of all curves of constant $u$ or $v$ in a rational B-spline patch ($u, v$ bounding cones), and also cones that bound all of the normal vectors on a rational B-
spline patch *(normal cone)*. Using the normal cone, the *tangent plane cone* was also defined by the following property. If its vertex is translated to any point on the surface, the tangent plane at that point will not cut the tangent plane cone and no other point on the surface lies within the cone. The tangent plane cone for a surface has the same axis as the normal cone, and its half angle is π/2 minus the half angle of the normal cone. Figure 7-1 adapted from [Sederberg 88a] shows the tangent plane cone for two bicubic patches. The tangent cone provides a measure for the *flatness* of a patch. The larger the half angle of the tangent cone, the flatter the patch. The criterion used to evaluate the existence of closed loops in the intersection curve on both parametric domains is as follows: If the u or v cone from one of the surfaces lies within the tangent plane cone of a second surface, then all v or u isoparameter curves of the first surface intersect the second surface at most once. This *guarantees* single valued intersection curves in v or u. Figure 7-2 adapted from [Sederberg 88a] shows two surfaces along with the u cone for the "vertical" surface patch and the tangent plane cone for the "horizontal" surface patch. Since the u cone lies entirely within the tangent cone, from the condition the intersection curve passes through the boundary curves of the surface. The application of this criterion by [Sederberg 88a] is as follows: The u and v cones from each surface are tested with the tangent plane cone of the other surface with respect to this condition. If the condition is not satisfied, one of the patches is subdivided (the one with the largest bounding cone). Bounding boxes are used to eliminate non-intersecting surface pairs and the criterion is evaluated again. If the condition is satisfied for one of the patches there is no need for further subdivision and marching is used to trace the intersection curve.
7.3 Application of Feature Detection Techniques to Rational Spline Surface Intersections

[Sederberg 88a] observed that, in actual practice, the above feature detection test required very large amount of computation even for simple intersections. [Sinha 85] did not report on the efficiency of the application of his technique. Closer study of these promising methods allows identification of important factors affecting their efficiency and ways to enhance their performance.

The first issue is the type of subdivision used to split the subpatches, which do not satisfy the detection criterion. [Sederberg 88a] used subdivision of one of the surfaces at a time. He selected to subdivide the patch with the largest first partial derivative fluctuation
as determined using the $u$ and $v$ bounding cones. The rationale behind such a selection was the reduction in the amount of subdivision required. However, our experimentation with such a technique indicates that the amount of subdivision may be actually increased even in simple cases when this rationale is used. Such performance may be also related to the bounding box which is used in the subdivision method to efficiently eliminate non-intersecting subpatches. [Sederberg 88a] used a min / max bounding box to assist in the subdivision method. Figure 7-3 illustrates a simple example of an inclined straight line and a planar curve, where this problem can be identified. If we only split the curve (obviously involving the largest derivative fluctuation), the bounding rectangle of the curve decreases in size although it is still enclosed within the bounding rectangle of the straight line. In order to satisfy the single intersection point criterion (analogous to the loop detection criterion for surfaces), we would have to subdivide the curve in a binary fashion until we
reach point A in Figure 7-3. If both curves are split simultaneously, the absence of an intersection point can be detected with little subdivision. The use of a bounding box naturally oriented to the surface geometry, as the rectangular bounding box described in Chapter 4 alleviates part of this problem. Still, splitting of both surfaces which fail to satisfy the feature detection criterion allows in many cases faster elimination of non-intersecting subpatches and is the most appropriate method.

**Figure 7-3: Inefficiency of Subdivision of Only One Curve**

The feature detection techniques employ binary subdivision, slowly bracketing the features of the intersection in smaller scales. The presence of singularities in the intersection considerably affects the efficiency of this subdivision process, since this process leads to a linear convergence to the singularity. The subdivision process is stopped, when the condition is satisfied or the limiting subpatch size is reached. The limiting subpatch size can be expressed as a tolerance in parameter space on the size of the smallest subdomain allowable or as a tolerance in three dimensional space on the actual size of the bounding box of the subpatch. A tolerance measuring the planarity of a subpatch can be
also used to stop this subdivision process, i.e. when a subpatch becomes planar within the tolerance it is not subdivided further.

An additional issue is the construction of tight bounds to the entities (position, u, v parametric derivatives, normal vectors, tangent planes) which are compared in the various feature detection conditions described above. If these bounds are not tight, the feature detection process is slowed down. This becomes a real problem near singularities and in small loops, where the normal vectors to the two surfaces are very close to each other and the bounds to these entities intersect. [Sinha 85] does not provide a methodology to compute bounds to the normal directions of a surface patch. [Sederberg 88a] used cones to bound the u or v parametric derivative directions and the tangent planes of a surface patch. Chapter 4 described the construction of rectangular pyramids as an alternative to cones to bound surface parametric derivatives and normal vectors. These result in tighter bounds for these entities as was shown in Chapter 4 and may improve satisfaction of the feature detection criterion. The rectangular pyramid bounds were used in our implementation of these conditions.

In what follows, some of the results and possible applications of the feature detection criteria will be illustrated. As with the algorithm of [Patrikalakis 90a] which used a similar technique for the implicit algebraic to parametric surface intersection problem, we can capitalize on the a priori computation of "most" significant points (border, turning, singular and collinear normal points) and splitting of the surfaces at these points to tightly bracket small loops and singular points. The feature detection conditions and subdivision can then be used to ensure computation of all such features. In practice, most significant points are located a priori and the condition is usually satisfied and little additional subdivision is required. For the cases of failure to satisfy the condition, subdivision is employed until verification is achieved.

In the application of [Sinha 85]'s and [Sederberg 88a]'s conditions, the first step is
the subdivision (splitting) of the two surfaces at a sufficient subset of available significant points (border, termination and collinear normal points or border, termination, turning and singular points) which are computed using the methods presented in Chapter 5. Next, the rectangular bounding boxes to each of the subpatches described in Chapter 5 are compared and used to rapidly eliminate non-intersecting subpatches, thereby speeding up the execution of this algorithm.

For [Sinha 85]'s feature detection criterion using the tighter rectangular pyramid derivative bounds we have the following approach. For each pair of subpatches with possibility of intersection, the rectangular pyramids which bound the normal vector directions of each surface are compared. If these pyramids intersect [Sinha 85]'s condition is not satisfied and the two subpatches are subdivided further. If the pyramids do not have a common region of intersection the condition is satisfied and there is no need for further subdivision. Once the condition is satisfied for all subpatches, the actual tracing of the intersection curve can be performed using for example the methods of Chapters 8 or 9.

For an application of [Sederberg 88a]'s feature detection condition, we need to define also a bound to the tangent planes of a surface patch. The definition of a tangent plane pyramid from the normal pyramid is not straightforward. As a result we enclose the normal pyramid in the smallest possible cone and, then define the tangent plane cone in a manner similar to [Sederberg 88a]. The tangent plane cone of a surface patch is the entity, which if its vertex is translated to any point on the surface, the tangent plane to the surface at that point will not cut the tangent plane cone. The tangent plane cone has the same center vector as the normal cone, while its half angle is $\pi/2 - \phi$, where $\phi$ is the half angle of the normal cone. If $\phi > \pi/2$, then the tangent plane cone is not defined and subdivision is required to reduce $\phi$. The smallest normal cone, which circumscribes the normal pyramid has the same center vector as the rectangular pyramid and its half angle is given by the angle between the center vector and any edge vector of the rectangular pyramid ($\phi =$
Arcos(C_p \cdot C_i), where C_p is the center unit vector, and C_i an edge unit vector of the normal directions rectangular pyramid). The criterion developed by [Sederberg 88a] is extended to the case of pyramids and is used to evaluate the existence of closed loops in the intersection curve as follows. For each pair of intersecting subpatches, the rectangular pyramids which bound the u, v parametric derivative directions of each subpatch and the tangent plane cone defined above are computed. The following condition is then evaluated. If the u or v rectangular pyramid from any of the subpatches lies completely within the tangent plane cone of the second subpatch, then all v or u isoparameter curves of the first patch intersect the second patch at most once and there are no closed loops in the intersection of these two subpatches. If this condition is not satisfied, the two subpatches are subdivided and the condition is checked again for the new subpatches. Once the condition is satisfied for all subpatches, the actual tracing of the intersection curve can be performed using for example the methods of Chapters 8 or 9.

Testing of a u or v rectangular pyramid of a patch for inclusion inside the tangent plane cone of another patch is performed by examining the inclusion of all sides of the pyramid in the same region of the tangent plane cone. For example, if C_{tp} is the center unit vector of the tangent plane cone and \phi_{tp} is the half angle and C_i are the side unit vectors of the rectangular pyramid which is tested for inclusion, then if for all C_i, \phi = Arcos(C_{tp} \cdot C_i) belongs in one of the two intervals \phi \leq \phi_{tp} + \epsilon or \phi \geq \pi - \phi_{tp} - \epsilon (for the other side of the cone) with \epsilon \geq 0 tolerance, the rectangular pyramid is included in the tangent plane cone of the other patch. Non-inclusion can be determined faster if one of the side edge vectors results in a \phi outside the two intervals above. The tolerance \epsilon is used to account for imprecision in the computation of significant points particularly singular points and in order to reduce the amount of subdivision required in cases, where there is a very small common part of the u or v pyramid with the tangent plane cone near singularities and small loops. This tolerance is analogous to the tolerance used to identify planarity in a subpatch and
makes application of the feature detection criteria more practical. For our implementation $\varepsilon = 0.001$ to 0.01 radians.

The above methodologies can be also used in the implicit algebraic to rational polynomial parametric surface intersection. In this case the intersection is represented as an algebraic curve in the parameter space of the parametric patch. This algebraic curve can be represented as the result of the intersection of an integral Bezier surface patch and a control plane $w=0$ as shown in [Prakash 88a, Patrikalakis 90a]. The tangent plane cone for a plane $w=0$ is the half-space $w > 0$ or $w < 0$. In this case, the feature detection criterion of [Sederberg 88a] is satisfied, if the $u$ or $v$ bounding pyramid ($u$ or the $v$ hodograph) of the parametric representation of the algebraic curve in terms of $w = F(u,v)$ belongs in the half-space $w > 0$ or $w < 0$, i.e. it does not intersect the control plane $w=0$ at any other point but the origin. For this problem, the control surface $T(u,v)$ representing the algebraic curve of intersection, $w = F(u,v)$, which was described in Section 3.5.3, is first partitioned using subdivision at all available border, turning and singular points. Next, each of the resulting subpatches with a possibility of an intersection (i.e. with a bounding box intersecting the $w=0$ plane) is examined for satisfaction of the feature detection criterion. In this case the feature detection condition is satisfied if one of the $u$ or $v$ derivative surface control polyhedron for each subpatch (the $u$ or $v$ hodograph) does not intersect the plane $w=0$. This guarantees that there are no multiple intersection points (more than one intersection point) in an isoparametric curve of the subpatch in the direction which satisfies the criterion. If this criterion is not satisfied further binary subdivision of the subpatch is employed, and the above process is again repeated. This technique is a variation of the methodology suggested by [Prakash 88b, Patrikalakis 90a].
7.4 Examples

Some examples from application of this methodology are presented next. Figure 7-4 illustrates the subdivision in the parameter space of one of the patches required to satisfy the feature detection criterion for the intersection of the bicubic patch with the biquadratic patch also shown in Figure 10-13. The termination and turning points of the intersection curve are used to initially split the domain. Figure 7-5 illustrates the subdivision required to satisfy the feature detection criterion for the intersection of a bicubic Bezier patch and a rational biquadratic B-spline patch (torus) also shown in Figures 10-20 and 5-22. The computed collinear normal points are used to initially split the domain. Some of the other significant points are also shown in this Figure. Figure 7-6 illustrates the subdivision required to satisfy the feature detection criterion for the same example, when the significant point information is not used to initially split the domain. This is analogous to the application of the feature detection condition by [Sederberg 88a] using binary subdivision. As can be seen, in intersection examples with absence of singular points or small loops the two methods are comparable.

Figure 7-7 illustrates the subdivision required to satisfy the feature detection criterion for the intersection of two bicubic patches also shown in Figures 10-12 and 5-24. The computed collinear normal point identifying the small intersection loop is used to initially split the domain. The turning points of this intersection curve are also shown in this Figure. Figure 7-8 illustrates the subdivision required to satisfy the feature detection criterion for the same example, when binary subdivision is used. The same tolerance in the feature detection condition is used for both these examples. As can be seen in this case binary subdivision results in fewer subdomains. This is attributed to the more balanced subdivision (square subdomains) resulting from a binary subdivision, which allows faster isolation of the region around the collinear normal point. This inefficiency of splitting at significant points of small loops is also observed in some of the Figures of this Chapter.
described below. To reduce the amount of subdivision required when significant points are used it may be appropriate to use non-uniform subdivision in the subpatches with a significant point in one of their corners, i.e. instead of binary (center) splitting the subpatches which do not satisfy the loop detection criterion, splitting should be closer to the significant point (off-center) to allow for faster isolation of the intersection loop and satisfaction of the feature detection condition.

An advantage of the precomputation of significant points using the results of Chapter 5 is the identification of singular points of the intersection and splitting at these points. The alternative is to search for these points using binary subdivision. Figures 7-9 and 7-10 illustrate this case for the intersection of a bicubic patch with a plane also shown in Figure 10-10. In both Figures the subdivision required to satisfy the feature detection condition in one of the patches is shown. When the computed singular point in Figure 7-9 is used to split the intersection domain, the feature detection condition is satisfied immediately and no further subdivision is required. When binary subdivision is used, we observe a linear convergence to the singular point. Figures 7-11 and 7-12 illustrate the same case for the intersection of a biquartic patch with a plane also shown in Figure 10-11 and Figure 5-19. In both Figures the subdivision required to satisfy the feature detection condition in one of the patches is shown. When the computed collinear normal and singular points in Figure 7-11 are used to split the intersection domain little subdivision is required to satisfy the feature detection condition around the singular points. When binary subdivision is used, more subdivision effort is required near the singular points to bracket these points. Figures 7-13 and 7-14 illustrate the same case for the intersection of a bicubic patch with a quadratic - cubic patch also shown in Figure 10-14 and Figure 5-21. In both Figures the subdivision required to satisfy the feature detection condition in one of the patches is shown. When the computed collinear normal and singular points in Figure 7-13 are used to split the intersection domain little subdivision is required to satisfy the feature detection
condition around the singular point (the rightmost point). When binary subdivision is used, more subdivision effort is required near the singular point to bracket this point.

As can be seen in the above Figures the feature detection conditions can be applied to identify regions in the parameter domain with singular points or small loops. These regions could be used to provide starting points to singular points or to collinear normal points identifying small loops. A numerical method could be used to compute these points accurately. This is an alternative to the method described in Chapter 5 for the computation of initial approximations to singular and collinear normal points. Near singularities, small loops and constrictions, a binary subdivision based on the feature detection conditions would result in smaller subdomain size as can be seen in Figures 7-10, 7-12 and 7-14. Figure 7-15 presents another example of the binary subdivision resulting from application of the feature detection condition in one of the patches for the intersection between two biquartic patches also shown in Figures 10-18 and 5-23. In this intersection curve we have four small intersection loops, which are identified by the binary subdivision method by the concentration of subdivision in four areas of the domain. It is envisioned, that a method to recognize the pattern of the resulting subdivision for satisfaction of the feature detection criteria may provide good initial approximations to singular and collinear normal points of small loops or constrictions. This method could also be used initially in an intersection algorithm to determine if the intersection can be considered "simple" so that an efficient algorithm is used to trace the intersection. If it is classified as a "difficult" intersection with closely spaced features a more elaborate and less efficient intersection algorithm can be used to actually trace the intersection curve. This could be advantageous in a solid modeling system, since the classification of an intersection as "simple" can be performed fast with little subdivision, while only difficult intersections would be slowed down. One issue, still to consider; is that the methods described above depend on numerical computations and the use of tolerances to limit the amount of subdivision performed and do
not necessarily provide a firm numerical guarantee on the computation of all features of an intersection curve. Figure 7-12 may illustrate a possible problem with such an approach. In this Figure one of the singular points of the intersection (the one with $v = 0.5$) is not very tightly bracketed by binary subdivision. This is attributed to the fact that the domain is split near this point and the feature detection condition is satisfied within the tolerance used at this point. A method to recognize regions around singular points may fail to identify such a region.

Table 7-I presents an estimate of the computational time required for satisfaction of the loop detection criterion in a number of examples from this thesis, when binary subdivision is used to isolate the features of the intersection curve. In this computational time, we include the subdivision required, the evaluation of the bounds to the subpatches (position and derivatives) and the application of the feature detection condition. As can be seen in this Table, for "simple" intersection cases a small computational cost is attributed to this test, while for "difficult" intersection cases the computational cost increases.

Table 7-I: Computational Times for the Feature Detection Test Using Binary Subdivision

<table>
<thead>
<tr>
<th>Intersection Example</th>
<th>Computational Cost (CPU sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 10-10</td>
<td>1.0</td>
</tr>
<tr>
<td>Figure 10-11</td>
<td>6.7</td>
</tr>
<tr>
<td>Figure 10-12</td>
<td>2.7</td>
</tr>
<tr>
<td>Figure 10-14</td>
<td>12.2</td>
</tr>
<tr>
<td>Figure 10-13</td>
<td>0.8</td>
</tr>
<tr>
<td>Figure 10-20</td>
<td>3.3</td>
</tr>
<tr>
<td>Figure 10-17</td>
<td>17.5</td>
</tr>
<tr>
<td>Figure 10-18</td>
<td>30.9</td>
</tr>
</tbody>
</table>
7.5 Summary

As we have seen in this Chapter the use of the feature detection conditions allows one to verify the detection of all features of an intersection curve. For intersection examples with little complexity these conditions are satisfied with little subdivision and the tracing schemes of Chapter 8 and 9 can be used to trace the intersection curve. For many complex examples involving closely spaced features or nearly tangent surfaces, the condition is occasionally *conservative* and somewhat inefficient requiring several levels of additional subdivision even though all significant points are obtained in advance and are used to split the domain. For these examples, the bounding box intersection test for the subpatches described in Chapter 4, is a very important step in the algorithm since it enables elimination of subpatches with unresolved loop detection criteria but no actual intersection. In some cases of intersection curves with closely spaced features, where the u or v iso-parametric lines of one surface intersect in two closely spaced points of the other patch, the verification process requires substantial subdivision to satisfy the loop detection condition.

The feature detection conditions in association with binary subdivision can also be used to identify regions in the intersection domain around singular points and small loops. This provides an alternative way to compute initial approximations to significant points to identify the features of the intersection curve.
Figure 7-4: Domain Subdivision for Feature Detection Condition Satisfaction for Surfaces in Figure 10-13. Use of Significant Points to Initially Split Domain.
Figure 7-5: Domain Subdivision for Feature Detection Condition Satisfaction for Surfaces in Figure 10-20. Use of Significant Points to Initially Split Domain.

Figure 7-6: Domain Subdivision for Feature Detection Condition Satisfaction for Surfaces in Figure 10-20. Binary Subdivision.
**Figure 7-7:** Domain Subdivision for Feature Detection Condition Satisfaction for Surfaces in Figure 10-12. Use of Significant Points to Initially Split Domain.

**Figure 7-8:** Domain Subdivision for Feature Detection Condition Satisfaction for Surfaces in Figure 10-12. Binary Subdivision.
**Figure 7-9:** Domain Subdivision for Feature Detection Condition Satisfaction for Surfaces in Figure 10-10. Use of Significant Points to Initially Split Domain.

**Figure 7-10:** Domain Subdivision for Feature Detection Condition Satisfaction for Surfaces in Figure 10-10. Binary Subdivision.
Figure 7-11: Domain Subdivision for Feature Detection Condition Satisfaction for Surfaces in Figure 10-11. Use of Significant Points to Initially Split Domain.

Figure 7-12: Domain Subdivision for Feature Detection Condition Satisfaction for Surfaces in Figure 10-11. Binary Subdivision.
Figure 7-13: Domain Subdivision for Feature Detection Condition Satisfaction for Surfaces in Figure 10-14. Use of Significant Points to Initially Split Domain.

Figure 7-14: Domain Subdivision for Feature Detection Condition Satisfaction for Surfaces in Figure 10-14. Binary Subdivision.
Figure 7-15: Domain Subdivision for Feature Detection Condition Satisfaction for Surfaces in Figure 10-18. Binary Subdivision.
Chapter 8

TRACING USING SUBDIVISION AND NEWTON TECHNIQUES

8.1 Partitioning Scheme for Intersection Domain

The significant points of the intersection curve computed using the methodologies of Chapter 5 are used to partition (split) the intersection domain. In [Prakash 88a]'s algebraic to rational polynomial parametric surface intersection algorithm the first step of the tracing process was to split the control patch (intersection domain) along parametric lines passing through all significant points. In this way, significant points were guaranteed to be on a corner of one of the subpatches and if all turning and singular points were computed in advance, the intersection segments in each of the subdomains were monotonic. The above scheme results in a quadratic growth in the number of subpatches, $O((r+1)^2)$, which need to be intersected, where $n$ is the number of significant points.

There is an alternative way to partition the intersection domain in a smaller number of subpatches and still ensure that intersection segments are monotonic and significant points are on a corner of one of the subpatches. A subdivision tree structure is required to achieve this more economic partitioning of the intersection domain, see Figure 8-1. The significant points are used in sequence to partition the domain, rather than simultaneously as in the earlier method. For each significant point on the border of a subdomain, we partition the subdomain in two pieces using an isoparametric line orthogonal to that border and passing through this point. For each significant point which is internal to an intersection subdomain, we identify the subdomain and we partition it at the significant point in four pieces using isoparameter lines. Thus, each of the significant points becomes a node in the tree with two or four "leaves". In the end of partitioning, the final leaves of the tree correspond to subdomains potentially containing intersection segments. The
subpatches corresponding to each of the subdomains are determined using subdivision. This scheme results in a linear growth in the number of subpatches which need to be intersected, $O(4n)$, where $n$ is the number of significant points.

For the algebraic to rational polynomial parametric surface intersection, the control surface representing the algebraic curve of intersection is partitioned using the significant points computed by our method. For the rational polynomial parametric to rational polynomial parametric surface intersection, after the initial subdivision needed for obtaining good approximations of significant points and following the computation of significant points, both parametric patches are subdivided at the parameter values of a sufficient set of these points. There are two alternative sufficient sets of significant points to use in the splitting of the domain. The first sufficient set is all border, termination, turning and singular points, while the second sufficient set is all border, termination and collinear normal points as also described in Chapter 5. In order to avoid accumulation of numerical error in the subdivision process, each of the subpatches is determined from a single subdivision step from the original surface data. If one of the parametric patches is a plane, it is not subdivided at any point in order to improve the efficiency of the method. Figure 8-1 displays one example of this partitioning scheme for one of the surfaces in a parametric to parametric surface intersection problem. Figure 8-1 illustrates the partitioning for an intersection curve with one border point, one termination point and four turning points. One level of initial binary subdivision is used in this illustration. Such subdivision is normally used to compute good initial approximations of significant points. The resulting tree containing all the partitioning information is shown in the same Figure. The partitioning scheme outlined above allows our method to closely bracket the intersection and to efficiently concentrate all computing resources in places where there are intersection segments. Some additional examples of this localized partitioning scheme have been shown in Chapter 6 in Figures 6-4, 6-5 and 6-6.
In order to efficiently access a leaf of the tree during the tracing stage a depth-first-search of the partitioning tree is used. Once the tracing stage of our algorithms is performed for each subpatch of the partitioning tree, an algorithm for the connection of the intersection segments is used relying on traversing the partitioning tree of one of the parametric patches from the leaves to the root. At the top node the result is a list of connected intersection segments, which can be broken into a list of monotonic segments in the parameter space of the first parametric patch.

8.2 Tracing Scheme

8.2.1 Motivation

This and the following Chapter present two methodologies for tracing the intersection curve of two surfaces, once the significant points are determined. The first methodology described in this Chapter is a subdivision and faceting technique exploiting the convexity properties of B-splines in combination with a local Newton refinement of the intersection. The second methodology described in the following Chapter is a marching technique and is based on the solution of a set of differential equations describing the zero level curves of the distance function between two surfaces. In what follows, the subdivision and faceting technique as applied to the rational polynomial parametric surface intersection problem is described. For the algebraic to rational polynomial parametric surface intersection, a methodology has been developed which is similar but more efficient than the method proposed by [Prakash 88a], as described in Section 8.1. Our algebraic to rational polynomial parametric surface intersection method has been described in [Kriezis 90]. The method described in this Chapter addressing rational polynomial parametric surface intersections has also been summarized in [Kriezis 89].

Standard subdivision methods for tracing intersection curves subdivide (split) the parametric patches until a planarity condition is satisfied. This is followed by intersection of
Figure 8-1: Partitioning Scheme and Tree Structure

A = Border Point
B = Termination Point
C, D, E, F = Turning Point
the resulting triangular faceted approximations of the subpatches to determine intersection curve points. Bounding boxes are used to rapidly exclude non-intersecting subpatches. This type of methods requires substantial subdivision, generates large amounts of intersection data and is not designed to treat intersections with singular points effectively. The proposed subdivision, faceting and Newton based tracing algorithm developed in this thesis reduces the amount of subdivision required and the amount of intersection data obtained, has built-in features to account for complex intersections such as singular points and near singular points and ensures high accuracy in the intersection points at small incremental expense.

In contrast to standard subdivision techniques, our tracing scheme does not subdivide the parametric patches until a planarity condition is satisfied, before intersecting triangular faceted approximations of the subpatches. Our tracing scheme starts after the two parametric surfaces are subdivided at the significant points and any one of the verification criteria outlined in the previous Chapters is used. Approximate tracing of curve segments in a subdomain is based on approximation of subpatches by their control polyhedra and intersection of these approximations with each other. This is the technique also used by [Prakash 88a]. The approximation of the curve of intersection arising from the intersection of the two polyhedra is refined using an efficient Newton technique. The use of tight rectangular bounding boxes, described in Chapter 4, ensures efficient elimination of non-intersecting subpatches.

8.2.2 Polyhedral Intersections

The information on the subpatches of each parametric patch is held in a tree as was explained in Section 8.1. The tree of the first patch is traversed using a depth-first search until a leaf is reached, and the subpatch on this leaf is intersected with all the subpatches associated with the terminal leaves on the tree of the second patch. The rectangular bounding boxes described in Chapter 4 are used to eliminate unnecessary computations by
identifying non-intersecting subpatch pairs. In order for the control polyhedron of the subpatches to provide a better approximation to a subpatch from the initial step, a small number of simple knots (usually two or three knots), distributed uniformly in both directions, is added in each of the subpatches using a subdivision algorithm [Lyche 85]. An additional level of binary subdivision (splitting) of the intersection domain of both surfaces can be used to replace this knot refinement step and provide a better approximation to the two surfaces. Once the refinement of each pair of subpatches is performed, the edges of the control polyhedron of the subpatch in the first tree are intersected with a triangular faceting of the control polyhedra of the subpatches in the second tree.

This computation is the most computationally intensive part of the tracing scheme and is based on intersection of the edges of the quadrilaterals of the control polyhedron of the first subpatch with each of the planar triangular facets of the control polyhedron of the second subpatch. Figure 8-2 displays the intersection of one quadrilateral from the first patch with a faceted approximation of the second patch. All the edges of the control polyhedron of the first patch are intersected once with each of the triangular facets of the control polyhedron of the second patch and the intersection information for the edges is kept in a data structure designed to assist in the connection phase of the intersection segments, see Figure 8-3. If the degree of both rational polynomial surfaces is m in each of the u, v and s, t parameters and we have added q interior knots in each of the subpatches, we need to intersect 2 (m+q+1) (m+q) edges (for (m+q+1)^2 control points in the control polyhedron) in the first patch with each of the 2 (m+q)^2 triangular facets of the control polyhedron of the second subpatch. This results to an \(O((m+q)^4)\) count for the number of line to plane intersections to be performed for each pair of intersecting subpatches. In order to substantially reduce this potentially large number of simple intersection computations, bounding boxes are used to eliminate most of the line to plane intersections. For example, each of the edges is examined for intersection with the rectangular bounding box of the
second subpatch and then it is also examined for intersection with the min / max bounding box of each quadrilateral (two triangular facets) of the second subpatch.

**Figure 8-2: Quadrilateral Intersections**

The type of intersection segment in each quadrilateral of the first patch is determined by the number of edges of the quadrilateral that have an intersection, and the number of vertices that lie on the control polyhedron of the second patch. Each edge is allowed to intersect the control polyhedron of the second patch in one interior point or to lie on the control polyhedron. If there are more distinct intersection points, as may be the case for closely spaced features or inadequately refined subpatches, knots are added in both subpatches and the control polyhedron is refined, until such a problem is resolved. The end of an edge is considered to lie on the control polyhedron of the second patch, if its distance from a triangular facet on the control polyhedron is less than a tolerance $\epsilon$. This is an important tolerance to assist in identifying intersection points on the corner of a subpatch, which correspond to significant points of the intersection and to ensure proper connectivity. Its value for all the examples in this thesis is $10^{-10}$ to $10^{-12}$. Figure 8-3 presents a diagram
of the quadrilateral and edge data structures used for representing the intersection information to assist in the connection phase of this algorithm. For each edge of the control polyhedron of the first patch, the number of intersection points with the control polyhedron of the second patch, the three-dimensional coordinates of the intersection points and the approximate parameter values of each intersection point in both parametric spaces are stored. For each quadrilateral of the control polyhedron of the first patch, the number of intersection points of its edges, the number of vertices which are intersection points and the intersection information associated with each edge are stored. The information in the quadrilateral data structure is sufficient to properly connect the various intersection segments.

The intersection of a polyhedron edge with a triangular facet is computed as a line-plane intersection. Specifically, let points \( T_1, T_2 \) and \( T_3 \) be the three vertices of a triangular facet and points \( P_1 \) and \( P_2 \) be the two vertices of a polyhedron edge. The intersection point between edge \( P_1P_2 \) and the triangle \( T_1T_2T_3 \) is found by solution of the following system of simultaneous equations for \( w, t_1, t_2 \):

\[
(P_2 - P_1)w + (T_1 - T_2)t_1 + (T_1 - T_3)t_2 + (P_1 - T_1) = 0
\] (8.1)

If \( 0 \leq w \leq 1, t_1, t_2 \geq 0 \) and \( t_1 + t_2 \leq 1 \) the intersection point is in the interior of the edge and the triangular facet. If the set of equations (8.1) is singular (when the determinant of the matrix multiplying the unknowns is less than a small tolerance), then the edge is parallel or on the triangle. An edge intersection point is considered to be on the vertex of the edge if parameter value \( w \) is within \( \varepsilon \|P_2 - P_1\| \) of 0 or 1, where \( \varepsilon \) is a small tolerance. Once an approximate intersection point is computed in three-dimensional space, approximate parameter values are determined for this point in the parameter space of the two patches, using linear interpolation of the node values corresponding to the ends of the intersecting edge and the vertices of the intersecting triangle facet. Thus, the \( u \) and \( v \) parameter values of the intersection point are determined using parameter \( w \) and the node values
Figure 8.3: Data Structures for Representation of Intersection Information
corresponding to control points $P_1$ and $P_2$, while the $s$, $t$ parameter values of the intersection point are determined using barycentric coordinates $t_1, t_2$ and the node values corresponding to the vertices of the triangle.

In the majority of cases, there is a single intersection segment per quadrilateral and connectivity is easy to determine. The quadrilaterals with intersections in all edges suggest the existence of a closely spaced feature (constriction). Whenever such a situation is detected in the interior of the control polyhedron, knots are added (or binary splitting may be used) in the interior of the quadrilateral with intersections in all four edges, until all these situations are resolved and correct connection can be achieved unambiguously. This is always possible in a finite number of steps if all singular points are found a priori and used to split the intersection domain. This is the method also used by [Prakash 88a]. A final verification of the consistency of the intersection information in the data structure is performed to ensure that intersections in the vertices of the quadrilateral edges are correctly identified in the edge data structure. For example, if the vertex of an edge is considered as an intersection point in this edge, all the edges adjacent to this vertex should have their intersection information corrected to identify the vertex as an intersection point.

8.2.3 Connection Phase

The intersection phase gives a number of unconnected linear segments in each quadrilateral whose end points are close to the intersection curve. The quality of approximation of the intersection by these points is evaluated and is refined after they are connected to form piecewise linear splines between significant points. The connection is performed in the parametric space of the first subpatch using the intersection information in the geometry data structure.

The first step of the connection is to determine all possible starting points in the borders of the subpatch. We search for starting points on all boundary edges of the control
polyhedron. Figure 8-4 illustrates some of the types of starting points expected. Usually starting points exist in the interior of the boundary, see points A and C in Figure 8-4. The starting quadrilaterals and points S1 and S4 are adequate to determine the starting point information in these cases. Starting points at the corners of the subdomain usually indicate significant points at these corners. If the corner is a singular point we usually expect a corner quadrilateral with a closely spaced feature. Point B in Figure 8-4 is a common case and results in two starting points.

![Diagram showing starting point examples with points A, B, C, S1, S2, S3, S4](image)

**Figure 8-4: Starting Point Examples**

The case of a corner quadrilateral of one subpatch on the control polyhedron of a subpatch of the second surface requires special treatment. It usually signifies a higher order singularity, with possibly more than two segments starting from the corner. Figure 8-5 illustrates the situation in a case with one quadrilateral on the control polyhedron of the second patch next to a corner. First a list of edges which surrounds the "shaded" domain of the patch is created. In the example shown, four edges form the "envelope" of this quadrilateral. The list of edges is checked for possible intersection points. The number of
intersection points on this list of edges signify the number of segments starting from the corner of the patch. A linear initial approximation to the intersection segments from the intersection points to the corner is assumed and a number of points are added on this linear approximation depending on the distance of points S1, S2, S3 from the corner. The accuracy of this linear approximation is increased during the Newton iteration used to increase the accuracy of individual intersection points as explained in the next section.

Figure 8-5: Corner Resolution Scheme for Quadrilaterals on the Control Polyhedron

Once all starting points are found and starting quadrilaterals are determined, the connection phase starts from a starting point and proceeds connecting the intersection segments in neighboring rectangles until a quadrilateral is reached which corresponds to another starting point or to an abrupt end of the list in the interior of the domain, see Figure 8-6. The interior of the domain is also checked for possible intersection lists which start and end abruptly in the domain. The abrupt ending lists correspond to local termination points in the intersection of the two subpatches. In the significant point computation we only compute termination points for the intersection of the two whole surfaces and not for
every pair of subpatches from the two surfaces. This is the reason for the presence of the abrupt ending lists in the intersection information for two subpatches. In this way, approximations to the separate pieces of the intersection curve are obtained in each of the subpatches. After each subpatch of the first tree is intersected with all the subpatches of the second tree and the Newton iteration of Section 8.2.4 is applied, the resulting lists are linked together to allow only for lists that start and end in the boundary of the subdomain. This process is illustrated below.

Figure 8-6 illustrates various types of intersection segments in the parameter space of a subpatch from the first surface. The grid corresponds to the "projection" of the control polyhedron of the subpatch in the parametric domain (at the corresponding node values). This subpatch of the first surface is intersected by two subpatches from the other surface. The resulting intersection segments from one subpatch are curves 1 and 2, while the resulting intersection segments from the second subpatch are curves 3 and 4. The gaps in the intersection information are caused by the fact that we intersect the edges of the control polyhedron of one of the subpatches with the faceted approximation of the second subpatch to save on computations, instead of intersecting the two faceted approximations of the two subpatches to determine the local termination points in the interior of a quadrilateral of the first subpatch (quadrilaterals containing A in Figure 8-6). After the final linking step in the connection phase the subpatch shown results in three intersection curves that start and end in the boundary of the subdomain. This can be seen in the lower part of Figure 8-6.

8.2.4 High Accuracy Computation of Intersection Points

The connected lists of points in each subpatch, determined as in the previous section, represent points which are close to the intersection curve in the parametric space of both surfaces. The application of a direct numerical technique such as Newton iteration at each of these points allows very fast evaluation of accurate intersection points. This has been implemented to efficiently improve the accuracy of the initial piecewise linear segments.
Figure 8-6: Types of Intersection Segments

Initial connection in subdomain

A = abrupt end

Completed connection in subdomain
The intersection curve of two parametric patches is defined in equation (3.6). This represents a system of three equations in four unknowns u, v, s and t. To solve this system, we freeze one of the unknowns as follows. From a point \((u_i, v_j)\) and a point \((s_i, t_j)\) close to the intersection curve we move in the u or the v direction of the first parametric patch depending on the tangent direction to the intersection curve calculated using the approximate intersection point parameter values. We compute the angles \(\phi_1\) and \(\phi_2\) between the tangent direction to the intersection curve and the u and v parametric derivatives of the patch:

\[
\phi_1 = \min\{ \arccos\left( \frac{\vec{R}_u \cdot \vec{t}}{||\vec{R}_u|| \cdot ||\vec{t}||} \right), \ \pi - \arccos\left( \frac{\vec{R}_u \cdot \vec{t}}{||\vec{R}_u|| \cdot ||\vec{t}||} \right) \} \quad (8.2)
\]

\[
\phi_2 = \min\{ \arccos\left( \frac{\vec{R}_v \cdot \vec{t}}{||\vec{R}_v|| \cdot ||\vec{t}||} \right), \ \pi - \arccos\left( \frac{\vec{R}_v \cdot \vec{t}}{||\vec{R}_v|| \cdot ||\vec{t}||} \right) \} \quad (8.3)
\]

where \(\arccos\) is defined in the range 0 to \(\pi\) and \(\phi_1, \phi_2 \in [0, \pi/2]\). If the angle \(\phi_1 < \phi_2\), the most appropriate Newton iteration direction is along constant v parametric lines. If the angle \(\phi_2 < \phi_1\), the most appropriate Newton iteration direction is along constant u parametric lines. Figure 8-7 shows the two marching directions for the Newton iteration.

**Figure 8-7**: Marching Directions for Newton Iteration
For ordinary points, a second order Newton iteration is used, see equations (5.31) and (5.32). The iteration converges if the correction vector is below a certain tolerance $\varepsilon$. This tolerance for all intersection examples was set at $\varepsilon = 10^{-12}$. If the iteration is slow to converge, as is the case close to singularities, we use a modified Newton iteration to accelerate convergence [Ortega 70]

$$x^{k+1} = x^k - J(x^k)^{-1} F\{x^k + F(x^k - J(x^k)^{-1} F(x^k))\}$$

where $F$ is the function vector from equation ((3.10)), $J$ the Jacobian matrix, $x^k$ is the solution vector at the $k$th iteration containing the parameter values $u$ or $v$, $s$ and $t$ and superscript $^{-1}$ denotes matrix inverse. If the direction of Newton iteration selected is along constant $v$ parametric lines, the Jacobian columns contain vectors $r_u$, $-q_s$ and $-q_t$. Vector $r_u$ is replaced by vector $r_v$ if the direction of Newton iteration selected is along constant $u$ parametric lines.

In the above Newton iteration, we allow a limited number of iterations only. Since the curve is defined in a bounded domain, the iteration is also not allowed to move beyond the domain. If the iteration does not converge in the allowable number of iterations, the direction of marching is reversed and the iteration starts again from the initial starting point. This is usually encountered in cases when our initial approximate selection of marching direction is inappropriate and when we are very close to singularities. If the iteration does not converge in the other direction also, then the initial starting point is returned and the iteration results are discarded. Since these points are inaccurate, the remaining accuracy tolerance checks in parameter space and three-dimensional space are not satisfied. For these points knots are added and the control polyhedron is further subdivided to improve its resolution. The connection phase is then repeated until all accuracy tolerances are satisfied. In the initial subdivision, it is possible to have spurious and extraneous segments appearing in the polyhedral intersection, which do not form part of the real intersection. These may
arise near singular points, particularly cusp points, and also if the initial subdivision in a subpatch is very coarse to describe properly the intersection curve and results in extraneous loops and segments. These are not present in the actual intersection and, therefore, are not identified by significant points on the intersection curve. These points do not converge during the Newton iteration and further subdivision and more accurate approximation of the intersection curve removes them from the intersection polyhedron.

Once the accurate intersection segment ends are obtained, a simple procedure is used to ensure that we have adequate intersection points in a list, so that the resulting linear segments are accurate approximations at points between the two ends. For each linear segment corresponding to the two parametric spaces, the middle point is checked to determine if it is close to the intersection curve. If the distance of the computed points on the two patches in three dimensions corresponding to the middle point exceed a certain accuracy tolerance, a point is added in the midpoint and the Newton iteration is employed to improve its accuracy. Once this point is obtained accurately, it is added in the list of intersection points. This procedure is repeated until all lists are satisfactory. This procedure is illustrated in Figure 8-8. Alternatively, the piecewise linear approximation to the intersection may be fitted with higher order splines and then these could be used to apply the above test. Such a technique is likely to reduce the data necessary to accurately represent an intersection curve segment. [Sederberg 89a] recently proposed a method to generate a $G^2$ piecewise approximation to the intersection curve of two parametric patches computed directly from two points in the intersection curve. This method could be applied in the intersection results to improve the approximation and reduce the data necessary to describe the intersection.

Obtaining high accuracy using the above Newton iteration method has proved very successful in speeding the tracing part of the algorithm for many complex and diverse examples.
8.2.5 Final Connection Phase

For each of the subpatches of the first surface in the intersection tree, the control polyhedron edge intersections, the connection procedure and the high accuracy Newton computation of the intersection points described in the previous sections are performed. These result in a number of accurate intersection lists in the domain of each of the subpatches at the leaves of the partitioning tree of Section 8.1. The final connection step, traverses the partitioning tree of the first surface from the leaves to the root, connecting at each node the lists with common ends, which are not significant points. Figure 8-9 illustrates one level of this process. The numbers at the ends of the intersection lists identify the side of the subdomain at which an intersection list starts or ends. Number 5 identifies a significant point. The result after the connection at this node is shown at the lower part of this Figure. The corresponding tree structures are also shown in this Figure. At the root of the tree the result is a number of intersection segments with ends at significant points of the intersection. These intersection segments are monotonic segments in the parameter space of the first patch, provided that all turning points of the intersection curve in the parameter
space of this patch have been obtained by the method of Chapter 5. This final connection phase provides an additional verification of the consistency of the intersection results. The various steps in the subdivision, faceting and Newton tracing technique are also outlined in Table 8-I.

8.3 Implementation Issues

The application of the subdivision, faceting and Newton scheme described before has proven reliable in tracing a large number of intersection examples. To provide some insight on the computational effort required for the application of this scheme several calculations have been performed. It has already been mentioned that the most time-consuming part of the algorithm is the faceting intersection step (line-plane intersection) which is \( O(n^2) \), with \( n \) the number of control points of a refined subpatch for each pair of intersecting subpatches. This operation count is reduced substantially with the use of bounding boxes to eliminate non-intersecting line segments and triangular facets. Twelve of the intersection examples presented in this thesis were analyzed to determine CPU times for the execution of this tracing algorithm. The characteristics of the machine used were presented in Chapter 5. For the intersection examples involving rational biquadratic and bicubic patches, the average time for the computation of the intersection of a pair of subpatches was 0.7 CPU seconds (sample of 724 pairs of intersecting subpatches). For the degree six surfaces in each parameter shown in Figure 10-17, the average time for the computation of the intersection of a pair of subpatches was 4.85 CPU seconds (sample of 67 pairs of intersecting subpatches). The total computation time for the tracing phase for this algorithm varied with the complexity of the intersection ranging between 10 and 350 CPU seconds with the majority of the rational biquadratic and bicubic examples in the 20 to 70 CPU seconds range. Table 9-III in the following Chapter compares the tracing time of several intersection examples using the methodologies of this and the following Chapter.
Table 8-1: Algorithm for Tracing of Intersection Curves Using Subdivision and Newton Methods

Partition both surfaces at each of the significant points and construct the subdivision tree.

For each subpatch of the first surface with a possibility of an intersection with the second surface:
{
    Add a few (2 or 3) interior knots using subdivision to refine the control polyhedron.

    For each subpatch of the second surface with a possibility of an intersection with the first surface:
    {
        1. Check rectangular bounding boxes for intersection. If there is no possibility of an intersection, skip steps 2-7.

        2. Add a few (2 or 3) interior knots using subdivision to refine the control polyhedron of the subpatch of the second surface.

        3. Intersect faceted approximations of the two subpatches and create the edge and quadrilateral data structures.

        4. Check for constrictions in the intersection data structures and resolve these by locally adding interior knots in both surfaces and go to step 3, else continue

        5. Determine starting points for intersection lists and connect intersection points

        6. Apply local Newton iteration to intersection points

        7. If there are points that fail to satisfy the accuracy criterion, add more interior knots in both surfaces and go to step 3, else continue
    }

    Connect abrupt ending lists for this subpatch, so that there are only lists, which start and end in the boundary of this subpatch.
}

Connect the intersection lists at the leaves of the partitioning tree.

Add points in the middle of intersection segments that poorly approximate the intersection curve.
The faceting and line-plane intersection part of this tracing algorithm takes approximately 90% of the total computation time, while the connection phase and the Newton iteration take the remaining 10%.

Some of the ideas of the subdivision, faceting and Newton tracing methodology presented in this Chapter can be readily extended for use in the general parametric to general parametric surface intersection problem. For a general parametric surface, the absence of a control polyhedron can be replaced by a lattice of points evaluated to be on the parametric surface. The significant points of the intersection computed with a method similar to Chapter 5 as was also described there are points on this lattice. The subdivision process also needs modification and can be performed by evaluation of additional points.
refining the lattice of points on the surface. The polyhedral approximations to the two surfaces can then be intersected to determine approximate intersection points and the Newton method can be used to compute these points accurately. [Barnhill 89] suggests a similar method for computation of initial approximations to intersection segments, before using marching to trace these segments. Tight bounding boxes for subpieces of the parametric surface (subpieces of the lattice of points) need to be constructed to improve the efficiency of such a scheme.
Chapter 9

TRACING USING TENSORIAL DIFFERENTIAL EQUATIONS

9.1 Motivation

Marching methods involve generation of sequences of points of an intersection curve branch by stepping from a given point on the intersection curve in a direction prescribed by the local differential geometry of the curve. Marching methods are known to be a more efficient alternative to subdivision methods, as they are simple to implement and do not generate large amounts of data during the solution process. The first difficulty with marching methods, in contrast to subdivision methods, is that they require starting points for every intersection branch. An important contribution of this thesis is the development of techniques to determine these starting points on all connected intersection components and to identify the singular points of the curve. The second difficulty with marching methods is the selection of the step size. Incorrect step size may lead to erroneous connectivity of solution branches or even to endless looping in the presence of closely spaced features [Geisow 83]. Most marching methods make use of curvature analysis or power series expansions about each point of the intersection solution to control the step size. Chapter 2 reviewed some existing techniques proposed to control the marching step size.

In this Chapter an alternative marching technique is developed, which transforms the intersection curve following (marching) problem to an equivalent initial value ordinary differential equation (ODE) problem. There is a large amount of literature dealing with the numerical solution of initial value problems and there are some very good algorithms for the robust integration of the associated systems of ODE [Gear 71, Hall 76, NAG 89]. These algorithms use adaptive techniques to solve the problem of the selection of the proper integration (marching) step size. The technique developed in this work exploits orthogonal
projections of curves on surfaces useful for the construction of blending surfaces [Pegna 89] and for mapping trimming curves on surfaces [Pegna 90]. An additional contribution of this Chapter is tracing of an intersection curve with infinite singular points, i.e. tracing of intersection curves at which two surfaces are mutually tangent to each other (also called non-transversal intersections). Two algorithms based on marching and utilizing the ideas from earlier Chapters of this thesis are described and compared with the tracing scheme of Chapter 8.

9.2 Tensorial Differential Equations

In section 3.2, the intersection curve between two surfaces was considered as the zero level curve $\phi(u,v) = 0$ of the oriented distance function between the two surfaces. In addition, if a curve $\gamma(w)$ on a surface $r(u,v)$ and its orthogonal projection on another surface $q(s,t)$ are identical, then curve $\gamma(w)$ is an intersection curve of the two surfaces. These two views of an intersection curve provide the tools for the development of differential equations describing the intersection curve of two surfaces. The tensorial differential equations for $\phi(u,v) = 0$ will be defined as a specialization of the tensorial differential equations describing the orthogonal projection of a curve on a surface and the differential equations describing an iso-distance curve of one surface from another surface.

When a point $R$ of $r(u,v)$ describes a differentiable curve $\gamma$, parameterized by $w$, the following equations describe the orthogonal projection of this curve on the second surface.

$$\frac{d}{dw}((r(u,v) - q(s,t)) \cdot q_s) = 0 \quad (9.1)$$

$$\frac{d}{dw}((r(u,v) - q(s,t)) \cdot q_t) = 0 \quad (9.2)$$

Expanding the differentiation in the above equations and using the chain rule we obtain the differential equation describing the projection curve of $\gamma(w)$ in the parametric domain of surface $q(s,t)$. 

\[
\left( \frac{ds}{dw}, \frac{dt}{dw} \right)^T = [K_{ij}]^{-1} \left[ G_{ab} \right] \left[ \frac{du}{dw}, \frac{dv}{dw} \right]^T
\] (9.3)

Appendix B provides a more detailed derivation of this equation. \([K_{ij}]\) is the orthogonal projective tensor of surface \(q(s,t)\) given by

\[
K_{ij} = q_i \cdot q_j - (r-q) \cdot q_{ij} \quad \text{with} \quad i, j = s, t
\] (9.4)

a function of the first and second fundamental forms of surface \(q(s,t)\). Subscripts in \(r\) and \(q\) denote partial derivatives. \([G_{ab}]\) is given by

\[
G_{ab} = r_b \cdot q_a \quad \text{with} \quad a = s, t \quad \text{and} \quad b = u, v
\] (9.5)

Equation (9.3) assumes the knowledge of curve \(\gamma(w)\) on surface \(r(u,v)\). We are interested in the intersection curve of the two surfaces, which is an iso-distance curve of the oriented distance function defined in Section 3.2, i.e. \(\phi(u,v) = \text{constant}\). This can be expressed as the solution of

\[
\frac{d\phi}{dw} = \phi_u \frac{du}{dw} + \phi_v \frac{dv}{dw} = 0
\] (9.6)

The general solution of equation (9.6) is

\[
\left[ \frac{du}{dw}, \frac{dv}{dw} \right] = \pm \left[ \frac{\phi_v}{\sqrt{\phi_u^2 + \phi_v^2}}, \frac{-\phi_u}{\sqrt{\phi_u^2 + \phi_v^2}} \right]
\] (9.7)

where parameter \(w\) corresponds to arc length in the parameter space of surface \(r(u,v)\). The \(\pm\) signs denote the two possible directions of marching on an iso-distance curve of the surface.

Equations (9.3) and (9.7) are a four dimensional first order differential equation system describing a curve on the second surface which is equidistant to curve \(\gamma(w)\) on the first surface. The intersection curve between the two surfaces is a special case of this differential equation system. In this case, the second term in equation (9.4) (the contribution of the second fundamental form of surface \(q(s,t)\)) is zero, since for an
intersection point \( r - q = 0 \). From an initial value on the intersection curve, these equations are numerically integrated to trace the intersection curve. The only other requirement on the at least C\(^1\) parametric surfaces used in this system of differential equations is that their first parametric partial derivatives do not vanish simultaneously at an intersection point. This tensorial differential equations methodology is not limited to piecewise rational polynomial parametric surfaces but can deal easily with general parametric surfaces.

9.3 Marching from Singularities

The presence of singularities in an intersection curve poses a problem to a marching method which traces the intersection curve through each of these singularities, since there can be more than one direction of tracing through such a point. [Bajaj 87] used a desingularization method based on birational transformations to rectify this type of problem in marching methods for algebraic curves and, thus, offered a procedure to connect the proper branches by marching across singular points. In this section, we present a procedure based on tensorial differential equations for marching from singular points of an intersection curve computed using the methods described in Chapter 5.

At a singular intersection point, the gradient of the oriented distance function \( \phi(u,v) \) between two surfaces is null, \( \phi_u = \phi_v = 0 \). In this case the system of first order differential equations in equations (9.3) and (9.7) has zero right hand side. A higher order expansion of the distance function is required at such points to determine the direction of an iso-distance curve between the two surfaces [Krasnoselskiy 66]. For an iso-distance curve in this case \( \frac{d\phi}{dw} \) is identically zero and we need:

\[
\frac{d^2\phi}{dw^2} = \left[ \frac{du}{dw}, \frac{dv}{dw} \right] [H_{ij}] \left[ \frac{du}{dw}, \frac{dv}{dw} \right]^T = 0
\]  
(9.8)

where \( H_{ij} = \phi_{ij} \) with \( i, j = u, v \), the Hessian matrix of the oriented distance function between
the two surfaces. The Hessian of the oriented distance function is defined in Appendix B. The solution of equation (9.8) determines the tangent direction(s) of the intersection curve at the singular points. For a minimum of the distance function which is also an intersection point, equation (9.8) has no real solution. In this case det(H) > 0 and we have an isolated point of the intersection with no real marching directions. For a saddle point of the distance function which is also an intersection point, equation (9.8) has two real and distinct solutions. In this case det(H) < 0 and we have a self-intersection point of the intersection curve with two distinct marching directions. For a cusp point of the intersection curve, equation (9.8) has two real identical solutions. In this case det(H) = 0 (H is not a null matrix) and we have a single marching direction from such a singular point.

The solution of equation (9.8) for det(H) ≤ 0 is given by:

\[
\begin{align*}
\left[\frac{du}{dw}, \frac{dv}{dw}\right]^T &= \lambda_1 \left[\phi_{uv} \pm \sqrt{-\text{det}(H)}, \phi_{uu}\right]^T \text{ or } \\
\left[\frac{du}{dw}, \frac{dv}{dw}\right]^T &= \lambda_1 \left[\phi_{vv}, -\phi_{uv} \mp \sqrt{-\text{det}(H)}\right]^T
\end{align*}
\]  

(9.9) (9.10)

with \(\text{det}(H) = \phi_{uu}\phi_{vv} - \phi_{uv}^2\) and \(\lambda_1\) a parameter determined by the parametrization of the intersection curve. Equations (9.9) and (9.10) result in the same marching directions, as can be verified by examining the slope of the corresponding directions, i.e.

\[
\frac{-\phi_{uv} + \sqrt{-\text{det}(H)}}{\phi_{uu}} = \frac{\phi_{vv}}{-\phi_{uv} - \sqrt{-\text{det}(H)}}
\]

(9.11)

Since parameter \(w\) represents arc length in the parameter space of the first patch, the left hand side of equations (9.9) or (9.10) is normalized to be a unit vector (\(dw^2 = du^2 + dv^2\)). This uniquely determines parameter \(\lambda_1\). In this manner, the simplified version of equations (9.3) (after substituting \(r-q = 0\)), and equations (9.9) or (9.10) (instead of equations (9.7)) define the updated tensorial differential equations to describe the intersection curve at singular points. Marching from singular points is therefore accomplished by using this set
of equations to determine a non-singular point on the intersection curve near the singular point in a specified direction, where equations (9.7) and (9.3) can be used to trace the intersection segment. The above methodology is applicable for marching from ordinary singular points of degree 2. It can be extended to allow marching from higher order singularities by using additional higher partial derivatives of \( \phi(u(w), v(w)) \), and classifying the corresponding marching directions.

9.4 Surfaces Tangent Along General Curves

An important special intersection problem, that has received little attention in the literature, is the computation of the intersection of two surfaces which are tangent or have the same normal vector direction along a general curve, see [Barnhill 89]. In this case we have \textit{infinite singular points} on the intersection curve. As an example, this type of intersection curve occurs in geometric models involving blending and filleting surfaces. It is very difficult to compute such an intersection curve accurately with a subdivision method, since an infinite amount of subdivision is required to identify this curve properly. An ordinary marching method, similarly, breaks down in such cases. Higher derivatives of local surface differential properties and more complex power expansions at intersection points are required. The tensorial differential equation method presented in this chapter can be very easily extended to treat such intersection cases in an automated manner.

In this respect, the results of the previous section are applicable. At any point on such an intersection curve the gradient of the distance function between two surfaces is null, \( \phi_u = \phi_v = 0 \). In this case the system of first order differential equations in equations (9.3) and (9.7) has zero right hand side. The higher order expansion of the distance function in equation (9.8) is required at such points to determine the direction of an iso-distance curve between the two surfaces. The Hessian of the distance function at any point on the intersection curve of two mutually tangent surfaces has \( \det(H) = 0 \) (\( H \) is not a null matrix)
and there is a single marching direction from such a singular point. The general solution of equation (9.8) in this case is given by:

\[
\begin{align*}
\left(\frac{du}{dw}, \frac{dv}{dw}\right)^T &= \lambda_1 \left[ -\phi_{uv}, \phi_{uu}\right]^T \quad \text{or} \\
\left(\frac{du}{dw}, \frac{dv}{dw}\right)^T &= \lambda_1 \left[ \phi_{vv}, -\phi_{uv}\right]^T
\end{align*}
\] (9.12) (9.13)

with \(\lambda_1\) parameter determined by the parametrization of the intersection curve. Since parameter \(w\) represents arc length in the parameter space of the first patch, the left hand side of equations (9.12) or (9.13) has to be a unit vector (\(dw^2 = du^2 + dv^2\)). At \(u\) or \(v\) turning points of such an intersection curve, one of equations (9.12) or (9.13) has a null right hand side. As a result, the appropriate equation from (9.12) and (9.13) should be selected to form the tensorial differential equation system to avoid the right hand side of this system to be null.

In this manner the simplified version of equations (9.3) after substituting \(r-q = 0\) and equations (9.12) or (9.13) define the revised tensorial differential equations to describe an intersection curve along the two surfaces which are mutually tangent. These equations may be numerically integrated to determine such an intersection curve.

9.5 Tracing Scheme

9.5.1 Numerical Integration Scheme

There is a large amount of literature dealing with the numerical solution of initial value ODE problems. There are several techniques for integrating an initial value ODE problem, such as Taylor series methods, Runge-Kutta methods, linear multistep methods and various predictor-corrector methods [Hall 76]. Different methods use different stepping size and order of approximation. In our implementation, we selected to use the Adams method, which is a linear multistep method. These type of methods allow variable
order and variable step size for the integration computation and there are several good algorithms using these methods [NAG 89]. As an illustration of these methods it is convenient to consider the problem of a single first order differential equation

\[ \frac{du}{dw} = f(w, u), \quad \text{with} \quad u(a) = s \]  

(9.14)

with the solution required at certain values of \( w \) in a range \([a, b]\). For the Adams methods we integrate equation (9.14) to obtain

\[ u(w_{n+1}) = u(w_n) + \int_{w_n}^{w_{n+1}} f(w, u(w))dw \]  

(9.15)

If we denote the minimum degree polynomial which interpolates the values \( f_n, f_{n-1}, \ldots, f_{n-k+1} \) by \( P_{k,n}(w) \), where \( f_r = f(w_r, u_r) \), then we obtain the k-step k-order Adams-Bashforth formula

\[ u_{n+1} = u_n + \int_{w_n}^{w_{n+1}} P_{k,n}(w)dw \]  

(9.16)

The Adams formulae possess the important property of being strongly stable [Hall 76]. The characteristic of the Adams method is that it requires information from a number of back integration values in order to integrate forward. Equation (9.16) is a constant step size formula. The particular formula used in our implementation uses the divided differences form of the Adams method instead of the interpolation form to allow for variable step size and variable order in the intersection scheme. This is considered to be one of the better choices for integration scheme for system of ODE's of moderate complexity [Hall 76]. The selected technique allows the development of a robust tracing scheme, which automatically adjusts the step size and the approximation order in regions with rapid variations in the intersection curve and near constrictions (in the vicinity of near singularities).

Two algorithms have been developed for tracing the intersection curves of two surfaces and using the tensorial differential equations described above. The first algorithm
is an efficient basic algorithm with limited verification. The second algorithm is more comprehensive and utilizes the verification test of Chapter 6 to improve the confidence in the intersection results.

9.5.2 Algorithm 1

The various steps of the first ODE tracing algorithm are shown in Table 9-I, while the steps of the second ODE tracing algorithm are shown in Table 9-II. The first step of the first algorithm is the computation of border, termination, turning and singular points of the intersection curve using the techniques of Chapter 5. The second step is the characterization of each of these points to determine the number of curve segments passing through each point and the different tangent directions of curve segments emanating from each of these points. This is a very important step to assist in properly tracing all intersection segments. Figure 9-1 illustrates possible significant point directions accounted for in our algorithm. To perform the characterization of the significant points the distance function and its derivatives are used.

For a border point (which is not a turning point), there is a single intersection segment and a single direction to the interior of the u-v parametric domain. This direction is determined by evaluating equations (9.7) at this point. If the first step of equations (9.7) at that point leads to a point in the interior of the u-v domain, this integration direction provides the correct marching direction. If it leads to a point in the exterior of the u-v domain, the integration direction is reversed (change of sign in the right hand side of equation (9.7)). For a termination point there is a single intersection segment and a single direction which is determined by evaluating equations (9.3). If the first step of equations (9.3) at that point leads to a point in the interior of the s-t domain, this integration direction provides the correct marching direction. If it leads to a point in the exterior of the s-t domain, the integration direction is reversed (change of sign in right hand side of equation
(9.7)). For a turning point there is a single intersection segment and two directions to march from or to (opposite directions). A special case occurs, when a turning point occurs in the boundary of one of the parametric domains. In that case, the type of turning point and the boundary curve at which it lies determine the number of intersection directions emanating from this point to the interior of the domain. Both possible directions are examined by evaluating both signs of equations (9.7). If the first step of one of the signs of equation (9.7) at that point leads to a point outside the intersection domains u-v or s-t, the corresponding direction is discarded. Figure 9-1 illustrates one such case (Point B). For singular points, the Hessian of the distance function is used to determine the possible marching directions as was described in section 9.3. If det(H) > 0 at a singular point, we have an isolated point and no intersection segment emanating from this point (Point H in Figure 9-1). If det(H) < 0 at a singular point, we have a self-intersection point and there are two intersection segments and four directions to march from or to (Point F in Figure 9-1). If this point is also in the boundary, only the directions that lead to points inside the domain are accepted. If det(H) = 0 (H not a null matrix), as occurs at a cusp, there is a difficulty in identifying all the possible segment combinations, since there can be two segments with the same direction (Point G in Figure 9-1) or four segments with two opposite directions emanating from this point (Point I in Figure 9-1). Since, it is not very common for two such points to occur in the same loop with no other turning points between them, we resolve this situation by allowing marching only towards these points (and not from these points). A similar procedure can be used to characterize the border, termination and turning points for the intersection curve of two surfaces which are mutually tangent along this curve.

After characterization of all computed significant points, the allowable directions of marching at each of these points are placed in a list and are activated. Each of the points in the list and one of its active marching directions are selected as starting conditions for the
integration of the tensorial differential equations. These are integrated using the Adams method described in the previous section. The integration is stopped, when the neighborhood of another significant point is detected. This is done as follows. If during the integration we arrive at an exterior point of one of the intersection domains u-v or s-t, the integration is stopped and we search for the border point from our list of significant points, which is closest to this point. If this point is within a small tolerance to our approximate intersection point on the border of the domain, it is included in the intersection segment and its corresponding marching direction is deactivated. If during the integration we arrive at a point at which \( \frac{du}{dv} \) or \( \frac{dv}{dw} \) changes sign or become very small the integration is stopped and we search for the u-turning or v-turning point or singular point from our list of significant
points which is in the neighborhood of this point. If this point is within a small tolerance to our current intersection point, it is included in the intersection segment and its corresponding marching direction is deactivated. The above process is repeated for all the active directions of the significant points computed in our method, until all of them are treated. If during the integration method we reach a significant point (other than singular point) which was not computed by the numerical method of Chapter 5, we can compute this point accurately using a local iterative numerical method and add it in the list of significant points with its remaining active marching directions. The cusps in the intersection curve are always the last points of the significant point list and it is expected that in most cases by the time we have to treat these points they are already deactivated using the intersection segments emanating from the other significant points.

Once this process is finished, we obtain a number of monotonic piecewise linear intersection segments which are usually very accurate. To further enhance the accuracy of all these points, the local Newton iteration technique described in section 8.2.4 may be used. One to two iteration steps of this Newton technique at each point were adequate to achieve a high intersection accuracy of $10^{-12}$. In order to improve the accuracy of the piecewise intersection segments at points in the middle of these segments, the technique used in section 8.2.4 is also used here to add more points in regions with a poor piecewise approximation to the intersection curve. Parameter $w$ in the tensorial differential equations represents arc length in the u-v parameter domain and can be easily used as the parametrization parameter for fitting of a higher order curve approximating the piecewise linear intersection segments.

The above method depends on computation of all singular points, at least one border or termination point for each intersection segment which is not a loop, and at least one turning point for each intersection loop. In a basic implementation of this algorithm, we did not incorporate a verification of the significant point computation. The feature detection
techniques described in Chapter 7 can be used to verify the detection of all intersection segments if this is desired. However, these techniques may substantially increase the computational cost of the intersection.

9.5.3 Algorithm 2

The second algorithm developed for tracing the intersection curve using the tensorial differential equations utilizes some of the ideas of the subdivision technique of Chapter 8 and the verification test described in Chapter 6. It is a comprehensive algorithm and provides additional confidence in the intersection results.

The first step of the second algorithm is the computation of border, termination and the collinear normal points between the two intersecting surfaces using the techniques of Chapter 5. The second step is the creation of the partitioning trees of the intersection domain for both surfaces using these points in a manner similar to Section 8.1. For each subpatch of the first surface, the feature detection test of Chapter 6 is applied to determine if a critical point (collinear normal point) of the oriented distance function between the two surfaces is missing. If there is a missing critical point, it is computed numerically using the technique of Section 6.3 and the domain is further subdivided at this new point. This verification step, although not providing a firm guarantee, as it is based on a necessary condition, improves our confidence that all singular points and all collinear normal points of the intersection curve have been identified. For each subpatch of the first surface satisfying this verification test, its boundary curves are intersected with the other surface to determine the local border points. This curve to surface intersection is performed as follows. The control polygon of the boundary curve of the first patch is intersected with the control polyhedra of the subpatches of the second patch which have a possibility of an intersection to determine starting points for the border point computation. Then, a minimization or Newton iteration can be utilized to evaluate these local border points
### Table 9-I: Algorithm 1 for Tracing the Intersection Curve Using Tensorial Differential Equations

1. **Compute border, termination, turning and singular points**

2. **Characterize each of the significant points and determine the marching directions from or to these points**

3. **For Each significant point with an active marching direction**
   
   - Deactivate the selected marching direction
   
   - Use the Adams method to numerically trace the intersection, until the neighborhood of a different significant point is detected
   
   - If the detected significant point is in our list, deactivate its corresponding marching direction and go to the next significant point
   
   - Else if the detected significant point is not in our list, compute it accurately using the numerical techniques of Chapter 5 and add it in the list of significant points.

4. **Else if the detected significant point is not in our list, compute it accurately using the numerical techniques of Chapter 5 and add it in the list of significant points.**

If desired, apply Newton iteration at all intersection points to improve their accuracy, and add points in the middle of intersection segments that poorly approximate the intersection curve.
accurately in a manner similar to the method described in Chapter 5. The significant points computed earlier are also corner border points in these subpatches. Each of the local border points of a subpatch is characterized using the technique of the first algorithm and the possible marching directions are identified. For each subpatch numerical integration is used to trace all its intersection segments using the local border points to march to or from, similarly to the previous algorithm. The above process is repeated for all subpatches of the first surface. The final connection step of Section 8.2.5 is then used to connect the intersection lists in the subpatches at the leaves of the partitioning tree. The Newton iteration and the refinement of the intersection segments may also be used in this algorithm to improve the accuracy of the final intersection segments. A summary of the second algorithm can be found in Table 9-II.

9.6 Comparison of Efficiency of Tracing Methods

The tracing methods based on the tensorial differential equations described in this Chapter and the subdivision type tracing scheme described in the previous Chapter, have proven reliable in tracing a large number of difficult and diverse intersection examples. All methods were designed to result in reliable and accurate intersections. A comparison of the computational CPU times required by each of these tracing algorithms in a number of intersection examples analyzed in this work is shown in Table 9-III. The characteristics of the machine used were presented in Chapter 5. It should be pointed out, that Table 9-III presents the computational time of only the tracing part of our algorithms. In the next Chapter the total time of our intersection algorithms will be compared. As can be seen in this Table, the first algorithm of this Chapter is the most efficient algorithm in the intersections that do not involve a planar surface, while the subdivision scheme of the previous Chapter is a more time consuming algorithm. This is primarily attributed to the high computational cost of intersecting the faceted approximations of the two surfaces in
**Table 9-II: Algorithm 2 for Tracing the Intersection Curve Using Tensorial Differential Equations**

- Compute border, termination and collinear normal points
- Create the partitioning trees for the two surfaces at these points.

**For Each** subpatch of the first tree with a possibility of an intersection with the second surface

```plaintext
1. Apply the topological verification criterion of Chapter 6 and if it is not satisfied compute missed critical point, subdivide the intersection domain and repeat this step in the new subpatches.

2. Compute local border points for the subpatch

3. Characterize these points and account for significant points at the subpatch corners

4. **For Each** border point with an active marching direction in a subpatch

   ```plaintext
   a. Deactivate the selected marching direction
   b. Use the Adams method to numerically trace the intersection, until another border point is approached.
   ```
```

Connect the intersection lists at the leaves of the partitioning tree.

Apply Newton iteration at all intersection points to improve their accuracy, and add points in the middle of intersection segments that poorly approximate the intersection curve.
the subdivision scheme as compared to the efficient numerical integration of the tensorial differential equations describing the intersection curve. In the intersection examples with a planar surface, the planar surface is not subdivided and as a result the computational cost of intersecting the faceted approximations of the two surfaces decreases substantially. The verification test of Chapter 6 which is used in the subdivision and the second marching algorithm moderately increases the total computational cost. The computational cost of this verification test is also shown in Table 9-III for the same intersection examples.

An additional issue in the implementation of a tracing technique is the complexity of the computer code required. In this respect, the subdivision technique of Chapter 8 is the most complex technique to implement since it requires examination of a number of special cases and requires very careful attention in the setting of tolerances to identify intersection points and in treating of special conditions.
Table 9-III: Computational Times for the Various Tracing Methods in the Parametric-Parametric Surface Intersection

<table>
<thead>
<tr>
<th>Figure Number</th>
<th>Subdivision(^a) (CPU sec)</th>
<th>Marching 1 (CPU sec)</th>
<th>Marching 2(^a) (CPU sec)</th>
<th>Verification Test(^b) (CPU sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10-11</td>
<td>18.6</td>
<td>50.6</td>
<td>85.3</td>
<td>10.8</td>
</tr>
<tr>
<td>10-12</td>
<td>29.1</td>
<td>9.2</td>
<td>20.0</td>
<td>9.6</td>
</tr>
<tr>
<td>10-14</td>
<td>57.8</td>
<td>24.6</td>
<td>43.9</td>
<td>9.6</td>
</tr>
<tr>
<td>10-13</td>
<td>21.3</td>
<td>4.3</td>
<td>12.7</td>
<td>8.3</td>
</tr>
<tr>
<td>10-19</td>
<td>46.2</td>
<td>10.5</td>
<td>29.3</td>
<td>10.2</td>
</tr>
<tr>
<td>10-20</td>
<td>70.7</td>
<td>33.0</td>
<td>55.7</td>
<td>10.9</td>
</tr>
<tr>
<td>10-24</td>
<td>47.6</td>
<td>24.9</td>
<td>48.2</td>
<td>14.6</td>
</tr>
<tr>
<td>10-22</td>
<td>176.9</td>
<td>63.7</td>
<td>121.5</td>
<td>56.2</td>
</tr>
<tr>
<td>10-17</td>
<td>362.8</td>
<td>112.2</td>
<td>219.9</td>
<td>52.2</td>
</tr>
</tbody>
</table>

\(^a\) The computational time for the verification test is not included.

\(^b\) The verification test follows Chapter 6.
Chapter 10

INTERSECTION ALGORITHMS AND APPLICATIONS

10.1 Summary of Intersection Algorithms

In section 3.3 a number of important issues were identified for the design of a good intersection algorithm, such as reliability, accuracy, automation and efficiency. The aim of the ideas presented in the previous Chapters of this thesis was to address each of these issues and provide techniques to achieve the good performance required from an intersection algorithm. A number of intersection algorithms have been developed and tested to verify their reliability and efficiency. An outline of the intersection algorithms developed in this work and several numerical experiments from their application are presented in this Chapter. A comparison of the computational efficiency of the different algorithms in a number of examples is also provided in Table 10-III.

10.1.1 Rational Polynomial Parametric - Implicit Algebraic Surface Intersection

The first algorithm developed addresses the rational polynomial parametric to implicit algebraic surface intersection problem. This algorithm is an extension of the algorithm developed by [Prakash 88a] to treat higher order intersections (up to degree 12 algebraic curves of intersection in each parameter have been computed) and introduces efficiency enhancements in the partitioning of the intersection domain and in tracing of the resulting intersection segments using adaptive subdivision and Newton techniques. A description of the algorithm is given in Table 10-1. More details on this algorithm can be found in [Kriezis 90]. As implemented, this algorithm allows computation of the intersection of any order algebraic surface with a rational biquadratic or bicubic patch. The highest order example computed involved the intersection of a torus with a bicubic patch (intersection curve of degree 12 in each parameter variable).
10.1.2 Rational Polynomial Parametric Surface Intersection

The second set of algorithms developed addresses the intersection of two rational polynomial parametric surfaces. All the algorithms in this set have a similar structure which is described in Table 10-II. As implemented, our algorithms allow computation of the intersection of any order rational B-spline patches. The highest order example computed involved the intersection of two Bezier patches of degree 6 in each parameter.

There are three different algorithms compared in this work and there are three distinguishing characteristics for each of these algorithms. The first characteristic is the subset of significant points, which are used in the domain partitioning scheme. There are two subsets which completely describe the topology of the intersection curve. The first is composed of border, termination, turning and singular points of the intersection, while the second is composed of border, termination and collinear normal points (including singular points) of the intersection. An algorithm can compute one of the two subsets of significant points or even both subsets in case increased reliability is required for the identification of all singular points of the intersection (i.e. compute these points using two independent methods). The latter case might be of use in the cusp example of Section 6.2 also shown in Figure 10-15, when the cusp is not computed using the collinear normal point computation of Chapter 5 and is not identified in the topological verification of Chapter 6. However, this cusp point is computed in the singular point computation of Chapter 5.

The second characteristic in the design of these algorithms is the use of the feature detection methodologies described in Chapters 6 and 7. The requirements vary from application to application concerning the level of guarantee of intersection feature detection. The feature detection technique using the convexity properties of the bounds to the parametric derivatives of the B-splines provides a condition that is necessary and sufficient for the detection of all features of an intersection. Its disadvantage is that it may become slow for the difficult intersections that one may wish to analyze. The feature
Table 10-I: Algorithm for NURBS / Algebraic Surface Intersection

**INPUT:** A NURBS surface patch and an algebraic surface.

Split NURBS surface patch to its rational polynomial elements.

**For each** rational polynomial subpatch of NURBS surface patch

1. Direct computation of Bernstein coefficients of algebraic curve of intersection in parameter space of subpatch.
2. Check algebraic curve coefficients for possibility of an intersection
   **If** there is a possibility of an intersection
3. Compute border points, and iso-parametric lines of intersection.
4. Compute turning and singular points of the algebraic curve.
5. Tree partitioning of the intersection domain at significant points and verification of significant point computation using convexity properties.
6. Tracing of the intersection curve at each subdomain with a possibility of an intersection, using subdivision - Newton technique.
7. Connection of intersection lists in the subdivision tree.

**OUTPUT:** A number of connected intersection lists, providing a piecewise linear approximation of the intersection curve in parameter space and three-dimensional space.
**Table 10-II: Algorithm for NURBS / NURBS Intersection**

**INPUT:** Two NURBS surface patches

1. Translation and scaling of both surfaces to fit in box $[-1,1] \times [-1,1] \times [-1,1]$.

2. Coarse subdivision (splitting) of both parametric surfaces (1 or 2 levels).

3. Compute border, termination, turning, singular, collinear normal points or any adequate subset of these.

4. Tree partitioning of the intersection domain of both surfaces at an adequate subset of the significant points.

5. Application of feature detection technique for each subdomain. Based on the topological properties of the vector field of the gradient of the oriented distance function or on the convexity properties of the derivatives of the subpatches.

6. Tracing of the intersection curve at each subdomain with a possibility of an intersection, using a subdivision - Newton technique or using marching with the tensorial differential equations of the intersection.

7. Connection of intersection lists in the subdivision tree.

8. Map intersection points to actual location in three-dimensional space.

**OUTPUT:** A number of connected intersection lists, providing a piecewise linear approximation of the intersection curve in the parameter space of both patches and in three-dimensional space.
detection using the topological properties of the vector field of the distance function provides a condition that is necessary but not sufficient for the detection of all features of an intersection. The advantage of this methodology is its efficiency as compared to the previous methodology. In combination with the computation of collinear normal points, it is our experience that this methodology is very reliable in identifying all collinear normal points of two parametric surfaces in the majority of intersection cases.

The third characteristic in the design of these algorithms is the selection of the tracing scheme to use for the actual computation of the intersection curve. In Chapters 8 and 9, two alternative tracing schemes were examined. The subdivision / Newton scheme of Chapter 8 improves on the efficiency of current subdivision schemes, by reducing the amount of subdivision required to achieve high accuracy. However, as shown in Chapter 9 its computational cost is still higher than the computational cost of marching methods to trace intersections. The marching technique of Chapter 9 using the tensorial differential equations describing the intersection curve, in association with the computation of significant points, provides an efficient and reliable method to trace the intersection of two parametric surfaces.

Table 10-III provides a comparison of the total computational cost of four different algorithms on a number of intersection examples. The characteristics of the machine used were presented in Chapter 5. The first algorithm is the algebraic to rational polynomial parametric surface intersection algorithm of the previous section and applies to some of the intersection examples of this thesis. The other three algorithms are specific to the rational polynomial parametric to rational polynomial parametric surface intersection problem. The second algorithm computes the collinear normal points, applies the topological detection criterion and uses adaptive subdivision and Newton techniques to trace the intersection as was described in Chapter 8. The third algorithm computes collinear normal points, applies the topological detection criterion and uses the second marching algorithm described in
Chapter 9 to trace the intersection. The fourth algorithm computes turning and singular points, does not use a feature detection criterion and uses the first marching algorithm described in Chapter 9 to trace the intersection. The following observations can be made on this table. If the intersection can be represented as an algebraic curve in the parameter space of the first patch of moderate degree (up to degree 9 in each parameter) algorithm A is the most efficient. The marching algorithms increase in time complexity with an increase in the complexity of the intersection curve since they depend primarily on the number of intersection segments needed to be traced, while the subdivision / faceting algorithm (algorithm B) depends less on the complexity of the intersection and is governed primarily by the number of intersecting subpatches of the two surfaces and the number of line to plane intersections performed. The verification test in algorithms B and C accounts for 10 to 25 % of the total computational time as was also seen in Table 9-III. Marching algorithm D is the most efficient of the parametric to parametric surface intersection algorithms.

10.2 Numerical Experiments

10.2.1 Intersections of Algebraic Surfaces with Rational Biquadratic and Bicubic Patches

All examples presented in [Prakash 88a, Prakash 88b] have been recomputed with the new algebraic to parametric surface intersection algorithm. A number of additional examples were also computed to illustrate some of the new features of the algorithm. The highest order example computed with the algebraic to parametric surface intersection algorithm involved the intersection of a torus with a bicubic patch (intersection curve of degree 12 in each parameter variable).

The tolerances for all the examples shown in this and the following section had the following values. The relative tolerance used to distinguish two points in parameter space was $\varepsilon_{\text{sort}} = 10^{-10}$. The relative tolerance used to evaluate if a point can be considered to be
Table 10-III: Total Computational Times for Four Intersection Algorithms

Algorithm A = Parametric / algebraic intersection, tracing using subdivision and Newton without verification.
Algorithm B = Parametric / parametric intersection using topological feature detection and tracing using subdivision and Newton.
Algorithm C = Parametric / parametric intersection using topological feature detection and tracing using the second marching algorithm.
Algorithm D = Parametric / parametric intersection, tracing using the first marching algorithm without verification.
N/A = Not applicable.

<table>
<thead>
<tr>
<th>Figure Number</th>
<th>Algorithm A (CPU sec)</th>
<th>Algorithm B (CPU sec)</th>
<th>Algorithm C (CPU sec)</th>
<th>Algorithm D (CPU sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10-11</td>
<td>13.7</td>
<td>41.2</td>
<td>105.0</td>
<td>65.3</td>
</tr>
<tr>
<td>10-12</td>
<td>N/A</td>
<td>44.0</td>
<td>31.5</td>
<td>14.0</td>
</tr>
<tr>
<td>10-14</td>
<td>N/A</td>
<td>77.5</td>
<td>57.3</td>
<td>32.3</td>
</tr>
<tr>
<td>10-13</td>
<td>N/A</td>
<td>34.7</td>
<td>24.7</td>
<td>6.6</td>
</tr>
<tr>
<td>10-19</td>
<td>N/A</td>
<td>102.4</td>
<td>42.9</td>
<td>13.8</td>
</tr>
<tr>
<td>10-20</td>
<td>51.9</td>
<td>101.5</td>
<td>79.4</td>
<td>42.2</td>
</tr>
<tr>
<td>10-23</td>
<td>4.7</td>
<td>85.2</td>
<td>61.6</td>
<td>21.2</td>
</tr>
<tr>
<td>10-1</td>
<td>72.9</td>
<td>248.7</td>
<td>182.6</td>
<td>107.6</td>
</tr>
<tr>
<td>10-17</td>
<td>N/A</td>
<td>467.5</td>
<td>297.0</td>
<td>160.6</td>
</tr>
</tbody>
</table>
on the control plane (it reflects the accuracy of the intersection points in parameter space) was $\epsilon_{\text{plane}} = 10^{-12}$. The parameter space of the parametric surface was normalized to $[0,1] \times [0,1]$, and the largest absolute value weight, $|w_{ij}|$, in the algebraic curve of intersection was also normalized to 1. The tolerance for the distance accuracy of an approximate intersection point in 3-D was $\epsilon_{3D} = 10^{-10}$. All computations were performed in double precision.

Figure 10-1 illustrates the intersection of two torii of equal radii (larger radius 200 and smaller radius 50) with equatorial planes located at right angles. One of the torii is represented as a rational biquadratic B-spline patch, while the other torus is represented as a degree 4 algebraic surface. The intersection has 4 singular points which are identified by our algorithm. To examine the robustness of treating such high order intersection curves, we also increased the smaller radius of one of the torii to 50.01. The result is shown in Figure 10-2. As can be seen the singular points disappear, and connectivity is different and consistent with what is expected.

Figure 10-3 presents one example of intersection of a torus represented as an algebraic surface with a bicubic Bezier patch. The intersection is composed of two loops. Figure 10-4 presents one example of intersection of a bicubic B-spline patch representing one blade of a propeller with a cylinder to determine a cylindrical section of the propeller blade.

### 10.2.2 Algebraic Curves

A number of additional algebraic curve examples were also computed. These algebraic curves are represented in the Bernstein form in a bounded domain. Figure 10-5 presents the two-sided offset curve of the parabola $y = x^2$ by a distance of 1.0 in the domain $(-2.5, 2.5) \times (-1.0, 2.5)$. This algebraic curve is of order 5 in the $y$ and order 7 in the $x$ direction. This curve was obtained from fitting an algebraic curve of the above orders to
points on the two-sided offset of the parabola using singular value decomposition as explained in [Patrikalakis 88]. The singular value decomposition technique in this case leads to an exact fit, see also [Farouki 89].

Figure 10-6 illustrates a reducible algebraic curve with equation

\[ f(x,y) = (x - 0.7y - 0.1) (1.2x + 2y - 1.6) (x - y - 0.056338029) = 0 \]

with a triple point at (0.521126761, 0.464788732). Figure 10-7 illustrates a reducible algebraic curve with equation

\[ f(x,y) = (y - 0.2x - 0.1) (y - 0.15x - 0.15) (y - 0.1x - 0.2) = 0 \]

with a triple point at (1.0, 0.3).

Figure 10-8 presents an algebraic curve which is composed of three parametric lines, one with multiplicity 2 and illustrates the treatment of such cases by our algorithm. The algebraic curve illustrated in this Figure has the equation

\[ f(x,y) = (x - 0.25)^2 (y - 0.75) (y - 0.22) = 0. \]

Figure 10-9 illustrates the result of a perturbation in this algebraic curve equation by 10^{-5}, where the topology of the curve changes dramatically. The algorithm is able to resolve the resulting closely spaced features accurately.
Figure 10-1: Torus-Torus Intersection, Equal Radii.
Figure 10-2: Torus-Torus Intersection, Smaller Radii 50.0 and 49.99, Equal Larger Radii.
**Figure 10-3:** Torus-Bicubic Patch Intersection, Two Loops.

**Figure 10-4:** Cylindrical Section of a Propeller Blade Represented as a Bicubic B-Spline Patch.
**Figure 10-5:** Two-Sided Offset of Parabola $y - x^2 = 0$, by a Distance of 1.0.

**Figure 10-6:** Three Lines with a Triple Point.
Figure 10-7: Three Lines with a Triple Point.

Figure 10-8: Multiple Parametric Lines, One With Multiplicity 2.
Figure 10-9: Perturbation of Multiple Parametric Lines Equation With Line of Multiplicity 2 by $10^{-5}$. 
10.3 Intersections of Rational Polynomial Patches

The rational polynomial parametric to rational polynomial parametric surface intersection algorithms have been implemented and tested to verify their reliability and accuracy. Many complex and diverse examples initially analyzed in [Prakash 88a, Prakash 88b, Patrikalakis 90a] have been recomputed with the new more general algorithms. Additionally intersections of general rational B-spline surface patches were also analyzed.

As implemented, the new algorithms allow automatic computation of the intersection of two arbitrary order non-uniform rational B-spline patches. The tolerances for all the examples in this section had the following values. The tolerance used to distinguish two points in parameter space was $\varepsilon_{\text{sort}} = 10^{-9}$. The tolerance used to evaluate when a point is considered to be on the control polyhedron (it reflects the accuracy of the intersection points in parameter space) was $\varepsilon_{\text{param}} = 10^{-10}$. The tolerance for the distance accuracy of an approximate intersection point in 3-D was $\varepsilon_{3D} = 10^{-10}$.

Figure 10-10 presents the intersection of a bicubic Bezier patch with a plane. This intersection has one singular point and four border points. Figure 10-11 presents the final result of the approximate intersection illustrated in Figure 5-3 of a biquartic Bezier patch intersected with a plane. This intersection has ten turning points, three singular points and nine collinear normal points.

Figure 10-12 presents the intersection of two bicubic Bezier patches. The intersection in this case is a very small loop in the parameter space of both patches (four turning points, one collinear normal point). Figure 10-13 presents the intersection of a biquadratic and a bicubic Bezier patch. The intersection in the parameter space of the bicubic patch has two turning points, and two termination points, while the intersection in the parameter space of the biquadratic patch has two turning points and two border points. A collinear normal point exists between these two surfaces. Figure 10-14 presents the
intersection of a bicubic patch with a patch which is cubic in the u direction and quadratic in the v direction. The resulting intersection has one singular point, four turning points, two border points and two collinear normal points in the parameter space of the first patch. Figure 10-15 presents the intersection of a Bezier patch which is quartic in the u direction and quadratic in the v direction with a plane. The resulting intersection has two turning points and one singular point (cusp) and two collinear normal points. Figure 10-16 presents the intersection of a rational quadratic-linear B-spline patch (representing a cylinder) with a biquadratic Bezier patch representing an elliptic paraboloid. The resulting intersection has four border points and one singular (collinear normal) point (cusp). Figure 10-17 presents the final accurate result of the approximate intersection presented in Figure 5-4 between two Bezier patches of degree six in each variable (nearly coincident surfaces). The resulting intersection results in three loops in the parametric space of both patches. Figure 10-18 presents the final accurate result of the approximate intersection presented in Figure 5-7 between two biquartic Bezier patches (nearly coincident surfaces). The resulting intersection results in four small loops in the parametric space of both patches. Figure 10-19 presents the intersection of two nearly coincident biquadratic Bezier surfaces. One of the surfaces is obtained from a small random perturbation of the control points of the other surface. The resulting intersection has four border and termination points, two turning points and two collinear normal points. Figure 10-20 presents the intersection of a bicubic Bezier patch with a torus which is represented as a rational biquadratic patch. The resulting intersection has six turning points and three collinear normal points. Figure 10-21 presents the intersection of a bicubic B-spline patch representing a propeller blade with a rational biquadratic B-spline patch representing the hub of the propeller. The intersection in 3-D and in the two parametric domains along with the subdivision of each domain are shown in the above figures.

The intersection of two torii of equal radii (larger radius 200 and smaller radius 50)
with equatorial planes located at right angles shown in Figure 10-1 was also analyzed with
the rational polynomial parametric to rational polynomial parametric surface intersection
problem. Both torii were represented as rational biquadratic B-spline patches in this case.
To examine the robustness of our method in processing near singular cases, we translated
one of the torii by 0.01 in one of the coordinate axes. The result is shown in Figure 10-22.
As can be seen the singular points disappear, and connectivity is different. Figure 10-23
illustrates the intersection of two cylinders of equal radii (radius 2.5) with axis at right
angles to each other. Both cylinders are represented as rational B-spline patches. This
intersection has two singular points. If we translate one of the cylinders by 0.001 in one of
the coordinate axis, the singular points disappear and the result can be seen in Figure 10-24.

Figures 10-25 and 10-26 present two examples with a non-transversal intersection
involving infinite singular points. Figure 10-25 presents the intersection of two biquartic
Bezier patches. The biquartic surfaces are tangent with each other along a circular segment
in parameter space. Figure 10-26 presents the intersection of two biquartic Bezier patches,
which are tangent with each other along a closed loop intersection, which has the form of
an ellipse in the parameter space of both surfaces.
Figure 10-10: Bicubic Patch with Plane Intersection.

Figure 10-11: Biquartic Patch with Plane Intersection.
Figure 10-12: Bicubic - Bicubic Patch Intersection - One Small Loop

Figure 10-13: Bicubic - Biquadratic Patch Intersection
Figure 10-14: Bicubic - Cubic / Quadratic Patch Intersection

Figure 10-15: Quartic / Quadratic Bezier Patch - Plane Intersection
Figure 10-16: Cylinder - Elliptic Paraboloid Intersection

Figure 10-17: Intersection of Two Patches of Degree Six in Each Parameter - Three Loops
Figure 10-18: Intersection of Two Biquartic Bezier Patches - Four Small Loops.
Figure 10-19: Intersection of Two Nearly Coincident Biquadratic Patches.
Figure 10-20: Intersection of a Bicubic Patch and a Torus

Figure 10-21: Intersection of a Propeller Blade Represented as a Bicubic B-spline Patch and a Propeller Hub Represented as a Rational Biquadratic Patch
Figure 10-22: Torus-Torus Intersection, One Torus
Translated by 0.01
Figure 10-23: Cylinder - Cylinder Intersection, Equal Radii (2.5)
Figure 10-24: Cylinder-Cylinder Intersection, One Cylinder
Translated by 0.001
Figure 10-25: Biquartic - Biquartic Patch Intersection Involving Tangency Along a Curve.
Figure 10-26: Biquartic - Biquartic Patch Intersection Involving Tangency Along a Closed Loop.
Chapter 11

CONCLUSIONS AND RECOMMENDATIONS

In this chapter, we summarize the major results of this thesis. We also identify other potential applications and related future research topics on the computation of surface to surface intersections.

In this thesis, we develop automated algorithms for the computation of rational B-spline surface to surface intersections. These algorithms are reliable in treating complex intersections and lead to intersection curve solutions of high accuracy at small computational expense. These algorithms are ideal for computational geometry and geometric modeling applications, since they provide automated solutions and the ability to describe all features of an intersection set.

In this thesis, we introduce a new formulation of surface to surface intersections as the problem of tracing the zero level curve of the oriented distance function of one surface from another surface. The vector field of the gradient of the oriented distance function is defined and some of its properties are described. Characterization of the critical set of the distance function between two surfaces is equivalent with identification of all features of the intersection curve between the two surfaces. Some of the properties of this critical set are used to identify the features of the intersection curve.

In this thesis, new algorithms are developed for the computation of tight bounds for the position, first parametric partial derivatives and the normal vectors of rational B-spline surface patches. Naturally oriented rectangular boxes are used to bound the position of patches, while rectangular pyramids are used to bound the first parametric derivatives and the normal vectors of the patches. These bounds are useful in subdivision methods for checking absence of intersections between two surfaces before performing any intersection
computations thus improving the efficiency of these computations. These bounds are
tighter than currently available bounds and are equally efficient in their computation. A
new efficient algorithm for the detection of absence of intersection of two arbitrarily
oriented rectangular bounding boxes of two surfaces is also described.

One of the major contributions of this thesis is the development of techniques to
automatically compute significant points of an intersection set of two rational spline
parametric surfaces, i.e. border, termination, turning, singular and collinear normal points,
to assist in identifying all features of an intersection set. Two methodologies are developed
to automatically determine initial approximations to significant points. The first uses an
initial subdivision / faceting technique and the convexity properties of B-splines to
determine initial approximations to border, termination, turning and singular points. The
second technique uses topological properties of the vector field of the gradient of the
oriented distance function between the two surfaces to determine initial approximations to
collinear normal points. Numerical techniques based on minimization and Newton iteration
are then used to accurately compute the significant points. Significant points provide
starting points in all intersection branches and improve the robustness of marching
methods. In addition, these techniques identify the critical points which slow down
subdivision methods and, as a result, improve their efficiency by reducing the level of
subdivision required to accurately trace the various intersection branches.

In this thesis, we develop a new necessary condition for the identification of collinear
normal points of two surfaces using topological properties of the vector field of the gradient
of the oriented distance function of one parametric surface from another parametric surface
and describe an algorithm for an efficient application of this condition. Since a numerical
procedure is used for the computation of the collinear normal points of an intersection set,
there is no guarantee that all these points are identified to enable detection of all
intersection loops. The topological loop detection condition at the subpatches resulting from
an adaptive subdivision (splitting) of the intersection domain at available significant points enables the development of an adaptive procedure to detect collinear normal points, potentially missed in the initial collinear normal computation procedure. This technique is an efficient way to improve the confidence in the results of the intersection computation. The rotation of the vector field of the gradient of the oriented distance function on the boundary of subpatches resulting from the above subdivision (splitting) is instrumental in application of this collinear normal detection condition.

In this thesis, we also present two necessary and sufficient conditions for the detection of all features of an intersection curve using subdivision and convexity properties of the parametric partial derivatives of the two surfaces. These conditions are expressed in terms of tight rectangular pyramid bounds for the partial derivatives and the normal vectors of a surface which are developed in this work. We also describe two new algorithms for an efficient implementation of these conditions. The satisfaction of the feature detection conditions at the subpatches resulting from an adaptive subdivision of the intersection domain provides a theoretical verification of the success of the numerical process used to identify intersection loops and singular points.

In this thesis, several new algorithms are developed for tracing the intersection curve of two surfaces.

The first method is based on a coarse subdivision and faceting technique exploiting the convexity properties of B-splines, and their approximation by the control polyhedron in combination with a local Newton-type refinement of intersection points. An adaptive partitioning scheme for the intersection domain at the significant points of the intersection is used with this method. The proposed algorithm reduces the amount of subdivision required in existing subdivision techniques, computes complex intersections containing singularities and near singularities and ensures high accuracy in the intersection points.

The second method is a marching technique and is based on the solution of a set of
tensorial differential equations describing the zero level curves of the oriented distance function of one parametric surface from another surface. This method relies on a new characterization procedure for significant points, which allows computation of tangent directions to the intersection curve at such points. The significant points of the intersection curve are used as starting points for the marching technique and robust numerical integration is used to trace the intersection curve. The tensorial differential equations describing the intersection curve are extended to also trace non-transversal intersections between surfaces (surfaces mutually tangent along a general curve). Two algorithms are developed to apply this marching technique.

A large number of intersection examples, chosen for their complexity and diversity, are presented to illustrate the properties of the algorithms developed in this work. The computational efficiency of all algorithms developed in this thesis is also compared in some of these illustrative examples.

Based on the results of this thesis, the following topics are recommended for future research.

Extension of some of the ideas of this thesis, such as the automatic significant point computation and the tracing technique using tensorial differential equations to the problem of computation of general parametric to parametric surface intersections is envisioned. Many of these extensions are straightforward in nature, since the majority of the computations involve evaluation of surface position and its partial derivatives and were alluded to at appropriate points in earlier chapters of this thesis. The major difference of a general parametric surface and a rational B-spline surface is the presence of a control polyhedron and the associated convexity properties in the latter.

New and more efficient, necessary and sufficient conditions for computation of features of an intersection curve, as well as for verification of computation of all intersection segments need to be further developed. In this respect, differential geometric
and topological methods based on the properties of the vector field of the oriented distance function seem promising. Interval type methods to determine bounds to certain geometric operations may be also of value. Additional work needs to be performed in the efficient and accurate evaluation of the rotation number of the vector field of the gradient of the oriented distance function on a closed curve in the parameter space of a surface patch using the orthogonal projection of a curve on a surface. Additional work is also required to examine potential problems with regions where the oriented distance function is not well defined to ensure elimination of such regions.

The development of general bounding box based intersection methods for general parametric surfaces is another topic of interest. It is standard practice in subdivision methods, before actual computation of the intersection between two subpatches, to check for intersection of boxes that bound these subpatches. This eliminates unnecessary computations and improves the efficiency of the computation. The idea is to reduce the complexity of the search problem by restricting the area which can possibly contain components of the intersection set. The convex hull property of rational B-spline surface patches and their control polyhedra assist in easily computing bounds for these. Similar bounds need to be determined for the parametric derivatives and the normal vectors of the surfaces as was seen in this thesis. In general bounding boxes for position and partial parametric derivatives of parametric surfaces are not so easily computed. New methods need to be developed to efficiently construct linear or non-linear bounding boxes for general parametric surfaces.

Another important degenerate intersection problem is the detection of surface overlap of two parametric surfaces (ie. when the intersection of two surfaces is a surface patch). This is an important step in developing Boundary Representations of complex objects. The vector field ideas and the computation of the critical points of the oriented distance function developed in this work may be useful in solving this complex problem.
The extension of the tensorial differential equation method to trace higher order non-transversal intersections, such as surfaces which meet with curvature continuity, is also possible using the third derivative of the distance function between the two surfaces. The technique to march from singular points of the intersection curve can also be extended in a similar fashion to treat higher than second order singularities.

The automatic computation of collinear normal points between two surfaces using the theory of plane vector fields may be also of use in optimization algorithms. As was seen in this thesis, the computation of all collinear normal points between two surfaces is equivalent to the determination of all extrema points of a height (objective) function of two variables. The extension of this method to more variables requires more general theories for multi-dimensional vector fields and could be an area of significant future research. The computation of all local extrema of an objective function and, consequently, the global extremum is a fundamental problem in optimization.

Another area, which received little attention in this work, is the problem of fitting of the piecewise linear intersection curve results with higher order piecewise polynomials to reduce the data needed to represent the intersection segments and accurately approximate the intersection curve.

Many of the vector field ideas used in this work for the surface to surface intersection problem can be also applied to the curve to surface intersection problem to assist in developing efficient and robust algorithms to treat this simpler problem, which is very common in solid modeling applications.

In implementing intersection algorithms in floating point arithmetic, logical decisions based on proximity of a distance or another quantity to zero are frequently made. Numerical inaccuracies, particularly over a sequence of geometric operations such as intersections, may lead to a sequence of logical decisions that are inconsistent. This may lead to failure of geometric algorithms. Research towards alleviating such tolerance related problems is recommended.
The implementation of some of the algorithms of this thesis in a parallel environment is another issue requiring further research. Many of the ideas of this thesis could be profitably applied in a parallel processing environment. Some of these are the following. The computation of initial approximations to significant points is performed using many repetitive local computations, while the significant points are computed separately using numerical techniques. The verification tests described in Chapters 6 and 7 are local tests applied to a number of subpatches. From the tracing algorithms the subdivision, faceting and Newton technique is composed of a large number of simple computations (line plane intersection), while the second marching algorithm is a local algorithm operating independently in each subpiece of the intersection domain.

Another area, where the present work may be of assistance is in the representation of trimmed surface patches needed in the Boundary Representation method. The computation of the significant points of the intersection curve allows a representation of the topology of trimmed surface patch boundaries using monotonic intersection branches. Trimmed patches represented in this manner may be easier to interrogate.

In addition, to the need for intersection computations in geometric modeling and computational geometry to perform combinations of objects and contouring of complex surfaces, there are some new applications of the ideas presented in this thesis. The first application may be surface localization for the verification of shape conformance of a manufactured part with a tolerated geometric description of an object [Patrikalakis 90c]. The second application is in feature detection using the medial axis transform [Patrikalakis 90b, Gursoy 89]. In this application, an object is represented by its medial surface and the associated radius function, which allows determination of a number of important features of the object. For the computation of the medial surface, intersections of surfaces equidistant from two faces of the object are required. The third application is in automatic finite element mesh, or automatic hydrodynamic or aerodynamic panel mesh generation. For
many automatic mesh generation schemes, planar, cylindrical or other more complex sections of mechanical objects need to be computed.

The general parametric surface to parametric surface intersection problem is a formidable and very useful computational problem and there is still a lot of work that needs to be performed in trying to reconcile the conflicting goals of robustness, accuracy and efficiency of an algorithm solving this problem.
Appendix A

DERIVATIVES FOR THE MINIMIZATION OBJECTIVE FUNCTIONS

A.1 Partial Derivatives for Rational Surfaces

The partial derivatives of a rational parametric surface up to order three are required in the numerical computation of the significant points of the intersection curve as well as in other numerical computations of this work. If the parametric surface is specified in the rational form \( r(u,v) = R(u,v)/w(u,v) \), the following recursive equations from [Farouki 86b] are an efficient means for computing the partial derivatives of the surface equation.

\[
\begin{align*}
    r_u &= (R_u - w_u r)/w \\
    r_v &= (R_v - w_v r)/w \\
    r_{uu} &= (R_{uu} - 2w_u r_u - w_{uu} r)/w \\
    r_{uv} &= (R_{uv} - w_u r_v - w_v r_u - w_{uv} r)/w \\
    r_{vv} &= (R_{vv} - 2w_v r_v - w_{vv} r)/w \\
    r_{uuu} &= (R_{uuu} - 3w_u r_{uu} - 3w_{uu} r_u - w_{uuu} r)/w \\
    r_{uuv} &= (R_{uuv} - 2w_u r_{uv} - w_v r_{uu} - 2w_{uv} r_u - w_{uuu} r_v - w_{uuv} r)/w \\
    r_{uvv} &= (R_{uvv} - 2w_v r_{uv} - w_u r_{vv} - 2w_{uv} r_v - w_{vvv} r_u - w_{uvv} r)/w \\
    r_{vvv} &= (R_{vvv} - 3w_v r_{vv} - 3w_{vv} r_v - w_{vvv} r)/w
\end{align*}
\]  

A.2 Partial Derivatives of the Surface Normal

The derivatives of the surface normal of both surfaces are required in the evaluation of the Jacobian and the Hessian of the objective functions needed for the computation of turning points, singular points and collinear normal points. These derivatives will be
presented next for a rational parametric surface \( \mathbf{r}(u,v) \). We employ the symbol \( S = \mathbf{r}_u \times \mathbf{r}_v \) and its scalar magnitude denoted by \( S = |S| \). Such notations assist in the implementation of these expressions.

The first and second partial derivatives of \( S \) are:

\[
S_u = \mathbf{r}_{uu} \times \mathbf{r}_v + \mathbf{r}_u \times \mathbf{r}_{uv} \quad (A.10)
\]

\[
S_v = \mathbf{r}_{uv} \times \mathbf{r}_v + \mathbf{r}_u \times \mathbf{r}_{vv} \quad (A.11)
\]

\[
S_{uu} = \mathbf{r}_{uuu} \times \mathbf{r}_v + 2\mathbf{r}_{uu} \times \mathbf{r}_{uv} + \mathbf{r}_u \times \mathbf{r}_{uuu} \quad (A.12)
\]

\[
S_{uv} = \mathbf{r}_{uvu} \times \mathbf{r}_v + \mathbf{r}_{uy} \times \mathbf{r}_{uv} + \mathbf{r}_u \times \mathbf{r}_{uvv} \quad (A.13)
\]

\[
S_{vv} = \mathbf{r}_{uv} \times \mathbf{r}_v + 2\mathbf{r}_{uv} \times \mathbf{r}_{vv} + \mathbf{r}_u \times \mathbf{r}_{vvv} \quad (A.14)
\]

The first and second partial derivatives of the scalar magnitude \( S \) are:

\[
S_u = \frac{S \cdot S_u}{S} \quad (A.15)
\]

\[
S_v = \frac{S \cdot S_v}{S} \quad (A.16)
\]

\[
S_{uu} = \frac{S \cdot S_{uu} + S_u \cdot S_u - S_u^2}{S} \quad (A.17)
\]

\[
S_{uv} = \frac{S \cdot S_{uv} + S_u \cdot S_v - S_u S_v}{S} \quad (A.18)
\]

\[
S_{vv} = \frac{S \cdot S_{vv} + S_v \cdot S_v - S_v^2}{S} \quad (A.19)
\]

The unit surface normal and its first and second partial derivatives can then be expressed in terms of equations (A.10) - (A.19) as:
\[ n = \frac{S}{\bar{S}} \]  
\[ n_u = \frac{SS_u - S_u S}{S^2} \]  
\[ n_v = \frac{SS_v - S_v S}{S^2} \]  
\[ n_{uu} = \frac{S^2S_{uu} - 2SS_u S_u + (2S_u^2 - SS_{uu})S}{S^3} \]  
\[ n_{uv} = \frac{S^2S_{uv} - SS_v S_u - SS_u S_v + (2S_u S_v - SS_{uv})S}{S^3} \]  
\[ n_{vv} = \frac{S^2S_{vv} - 2SS_v S_v + (2S_v^2 - SS_{vv})S}{S^3} \]  

A.3 Partial Derivatives of \( r_a/\|r_a\| \)

The derivatives of the unit vector \( r_a/\|r_a\| \) (where \( a = u \) or \( v \) and \( r_a = \|r_a\| \)) of a parametric surface are required in the evaluation of the Jacobian and Hessian of the objective functions needed for computation of turning points and singular points. These derivatives will be presented next for a parametric surface \( r(u,v) \). We employ the symbol \( P = r_a/\|r_a\| \).

The first and second partial derivatives of \( P \) are:

\[ P_u = \frac{r_{au}}{r_a} - \left( \frac{r_{au} \cdot r_a}{r_a^2} \right) \frac{r_a}{r_a} \]  
\[ (A.26) \]

\[ P_v = \frac{r_{av}}{r_a} - \left( \frac{r_{av} \cdot r_a}{r_a^2} \right) \frac{r_a}{r_a} \]  
\[ (A.27) \]
\[ P_{uu} = \frac{r_{au}}{r_a} - \left( \frac{r_{au} \cdot r_a}{r_a^2} \right) \left( \frac{r_{au}}{r_a} + P_u \right) - \frac{r_{au} \cdot r_a + r_{au} \cdot r_{au}}{r_a^2} - 2 \frac{r_{au} \cdot r_a}{r_a^2} \frac{r_a}{r_a} \]  \hspace{1cm} (A.28)

\[ P_{uv} = \frac{r_{au}}{r_a} - \left( \frac{r_{av} \cdot r_a}{r_a^2} \right) \left( \frac{r_{au}}{r_a} + r_{av} \cdot r_{av} r_{av} \cdot r_{av} \right) - 2 \frac{r_{au} \cdot r_a}{r_a^2} \frac{r_a}{r_a} \]  \hspace{1cm} (A.29)

\[ P_{vv} = \frac{r_{av}}{r_a} - \left( \frac{r_{av} \cdot r_a}{r_a^2} \right) \left( \frac{r_{av}}{r_a} + P_v \right) - \frac{r_{av} \cdot r_a + r_{av} \cdot r_{av}}{r_a^2} - 2 \frac{r_{av} \cdot r_a}{r_a^2} \frac{r_a}{r_a} \]  \hspace{1cm} (A.30)

A.4 Partial Derivatives of \(|r - q|^2\)

The first and second partial derivatives of the distance of two points on the two surfaces \(r(u,v)\) and \(q(s,t)\) is given below for use in the evaluation of the Jacobian and Hessian of the objective functions needed for significant point computation. If we denote \(A(u,v,s,t) = |r - q|^2\), we have the following partial derivatives, forming the elements of the Jacobian of this function:

\[ A_u = 2 \ r_u \cdot (r - q) \]  \hspace{1cm} (A.31)

\[ A_v = 2 \ r_v \cdot (r - q) \]  \hspace{1cm} (A.32)

\[ A_s = -2 \ q_s \cdot (r - q) \]  \hspace{1cm} (A.33)

\[ A_t = -2 \ q_t \cdot (r - q) \]  \hspace{1cm} (A.34)

The following partial derivatives form the elements of the Hessian matrix of this function:
\[ A_{uu} = 2 \mathbf{r}_{uu} \cdot (\mathbf{r} - \mathbf{q}) + 2 \mathbf{r}_u \cdot \mathbf{r}_u \quad (A.35) \]
\[ A_{uv} = 2 \mathbf{r}_{uv} \cdot (\mathbf{r} - \mathbf{q}) + 2 \mathbf{r}_u \cdot \mathbf{r}_v \quad (A.36) \]
\[ A_{us} = -2 \mathbf{r}_u \cdot \mathbf{q}_s \quad (A.37) \]
\[ A_{ut} = -2 \mathbf{r}_u \cdot \mathbf{q}_t \quad (A.38) \]
\[ A_{vv} = 2 \mathbf{r}_{vv} \cdot (\mathbf{r} - \mathbf{q}) + 2 \mathbf{r}_v \cdot \mathbf{r}_v \quad (A.39) \]
\[ A_{vs} = -2 \mathbf{r}_v \cdot \mathbf{q}_s \quad (A.40) \]
\[ A_{vt} = -2 \mathbf{r}_v \cdot \mathbf{q}_t \quad (A.41) \]
\[ A_{ss} = -2 \mathbf{q}_{ss} \cdot (\mathbf{r} - \mathbf{q}) + 2 \mathbf{q}_s \cdot \mathbf{q}_s \quad (A.42) \]
\[ A_{st} = -2 \mathbf{q}_{st} \cdot (\mathbf{r} - \mathbf{q}) + 2 \mathbf{q}_s \cdot \mathbf{q}_t \quad (A.43) \]
\[ A_{tt} = -2 \mathbf{q}_{tt} \cdot (\mathbf{r} - \mathbf{q}) + 2 \mathbf{q}_t \cdot \mathbf{q}_t \quad (A.44) \]

**A.5 Partial Derivatives of Turning and Singular Point Objective Function**

The simpler version of the objective function used for the computation of the turning and singular points of an intersection curve is as follows:

\[ \text{Minimize} \quad ||\mathbf{r} - \mathbf{q}||^2 + \left( \frac{\mathbf{r}_v}{||\mathbf{r}_v||} \cdot \mathbf{n}_2 \right)^2 \text{ for } u\text{-turning points} \quad (A.45) \]

\[ \text{Minimize} \quad ||\mathbf{r} - \mathbf{q}||^2 + \left( \frac{\mathbf{r}_u}{||\mathbf{r}_u||} \cdot \mathbf{n}_2 \right)^2 \text{ for } v\text{-turning points} \quad (A.46) \]

\[ \text{Minimize} \quad ||\mathbf{r} - \mathbf{q}||^2 + \left( \frac{\mathbf{r}_v}{||\mathbf{r}_v||} \cdot \mathbf{n}_2 \right)^2 + \left( \frac{\mathbf{r}_u}{||\mathbf{r}_u||} \cdot \mathbf{n}_2 \right)^2 \text{ for singular points} \quad (A.47) \]

where \( \mathbf{n}_2 \) is the unit normal vector on the second parametric surface \( \mathbf{q}(s,t) \). The first and second partial derivatives of the first term in the above expressions were given in the previous section. The first and second partial derivatives of \( \mathbf{n}_2 \) were given in the second section of this Appendix and the first and second partial derivatives of \( \mathbf{r}_u/||\mathbf{r}_u|| \) and \( \mathbf{r}_v/||\mathbf{r}_v|| \) were given in the third section of this Appendix. Using this information the Jacobian and
Hessian matrix of the complete objective function can be assembled. To complete the illustration, the first and second partial derivatives of the second term in the $u$-turning point objective function will be given below. If we denote by $R = r_{\gamma}/\|r_{\gamma}\|$, $B = R \cdot n_2$ and $F = B^2$, we have the following first partial derivatives:

\[ F_u = 2B B_u = 2B (R_u \cdot n_2) \]  
(A.48)

\[ F_v = 2B B_v = 2B (R_v \cdot n_2) \]  
(A.49)

\[ F_s = 2B B_s = 2B (R \cdot n_{2,u}) \]  
(A.50)

\[ F_t = 2B B_t = 2B (R \cdot n_{2,v}) \]  
(A.51)

and the following second partial derivatives:

\[ F_{uu} = 2B u^2 + 2B (R_{uu} \cdot n_2) \]  
(A.52)

\[ F_{uv} = 2B u B_v + 2B (R_{uv} \cdot n_2) \]  
(A.53)

\[ F_{us} = 2B u B_s + 2B (R_u \cdot n_{2,s}) \]  
(A.54)

\[ F_{ut} = 2B u B_t + 2B (R_u \cdot n_{2,t}) \]  
(A.55)

\[ F_{vv} = 2B v^2 + 2B (R_{vv} \cdot n_2) \]  
(A.56)

\[ F_{vs} = 2B v B_s + 2B (R_v \cdot n_{2,s}) \]  
(A.57)

\[ F_{vt} = 2B v B_t + 2B (R_v \cdot n_{2,t}) \]  
(A.58)

\[ F_{ss} = 2B s^2 + 2B (R \cdot n_{2,ss}) \]  
(A.59)

\[ F_{st} = 2B s B_t + 2B (R \cdot n_{2,st}) \]  
(A.60)

\[ F_{tt} = 2B t^2 + 2B (R \cdot n_{2,tt}) \]  
(A.61)

By adding the $A$ (from section 4 of this Appendix) and $F$ derivatives as shown above the complete Jacobian and Hessian of the $u$-turning point objective function can be assembled. As an example if $G(u,v,s,t)$ is the objective function for the $u$ - turning points, the Jacobian of $G$ is a matrix with elements given by:

\[ G_a = A_a + F_a, \text{ where } a = u, v, s, t \]  
(A.62)

and the Hessian of $G$ is a matrix with elements given by:
\[ G_{ab} = A_{ab} + F_{ab}, \text{ where } a, b = u, v, s, t \]  \hspace{1cm} (A.63)

The Jacobian and Hessian of the objective function for the \( v \) turning points and the singular points can be similarly assembled.

**A.6 Partial Derivatives of Collinear Normal Point Objective Function**

The objective function used for the computation of the collinear normal points of two surfaces is as follows:

\[
\text{Minimize} \quad \|r - q\|^2 - [(r - q) \cdot n_1]^2 + 1 - (n_1 \cdot n_2)^2 \]  \hspace{1cm} (A.64)

where \( n_1 \) and \( n_2 \) are the unit normal vectors on the first and second parametric surfaces \( r(u,v) \) and \( q(s,t) \). The first and second partial derivatives of the first term in the above expression were given in the fourth section of this appendix (function \( A \) and its derivatives). To assist in the computation of the Jacobian and Hessian matrix of the above objective function, the first and second partial derivatives of term \( C = (r - q) \cdot n_1 \) in the above expression are given below.

\[
C_u = r_u \cdot n_1 + (r - q) \cdot n_{1,u} \]  \hspace{1cm} (A.65)

\[
C_v = r_v \cdot n_1 + (r - q) \cdot n_{1,v} \]  \hspace{1cm} (A.66)

\[
C_s = -q_s \cdot n_1 \]  \hspace{1cm} (A.67)

\[
C_t = -q_t \cdot n_1 \]  \hspace{1cm} (A.68)

\[
C_{uu} = r_{uu} \cdot n_1 + 2 r_u \cdot n_{1,u} + (r - q) \cdot n_{1,uu} \]  \hspace{1cm} (A.69)

\[
C_{uv} = r_{uv} \cdot n_1 + r_u \cdot n_{1,v} + r_v \cdot n_{1,u} + (r - q) \cdot n_{1,uv} \]  \hspace{1cm} (A.70)

\[
C_{us} = -q_s \cdot n_{1,u} \]  \hspace{1cm} (A.71)

\[
C_{ut} = -q_t \cdot n_{1,u} \]  \hspace{1cm} (A.72)

\[
C_{vv} = r_{vv} \cdot n_1 + 2 r_v \cdot n_{1,v} + (r - q) \cdot n_{1,vv} \]  \hspace{1cm} (A.73)
\[ C_{ys} = -q_s \cdot n_{1,v} \]  
\[ C_{yl} = -q_l \cdot n_{1,v} \]  
\[ C_{sl} = -q_{sl} \cdot n_1 \]  
\[ C_{tl} = -q_{tl} \cdot n_1 \]  
\[ C_{ss} = \frac{\partial}{\partial t} n_1 \]  
\[ C_{st} = \frac{\partial}{\partial s} n_1 \]  
\[ C_{tt} = \frac{\partial}{\partial t} n_1 \]  
(A.74)  
(A.75)  
(A.76)  
(A.77)  
(A.78)  

In addition, the first and second partial derivatives of term \( D = n_1 \cdot n_2 \) in the above objective function are given below.

\[ D_u = n_{1,u} \cdot n_2 \]  
\[ D_v = n_{1,v} \cdot n_2 \]  
\[ D_s = n_1 \cdot n_{2,s} \]  
\[ D_t = n_1 \cdot n_{2,t} \]  
\[ D_{uu} = n_{1,uu} \cdot n_2 \]  
\[ D_{uv} = n_{1,uv} \cdot n_2 \]  
\[ D_{us} = n_{1,u} \cdot n_{2,s} \]  
\[ D_{ut} = n_{1,u} \cdot n_{2,t} \]  
\[ D_{vv} = n_{1,vv} \cdot n_2 \]  
\[ D_{vs} = n_{1,v} \cdot n_{2,s} \]  
\[ D_{vt} = n_{1,v} \cdot n_{2,t} \]  
\[ D_{ss} = n_1 \cdot n_{2,ss} \]  
\[ D_{st} = n_1 \cdot n_{2,st} \]  
\[ D_{tt} = n_1 \cdot n_{2,tt} \]  
(A.79)  
(A.80)  
(A.81)  
(A.82)  
(A.83)  
(A.84)  
(A.85)  
(A.86)  
(A.87)  
(A.88)  
(A.89)  
(A.90)  
(A.91)  
(A.92)  

The first and second partial derivatives of \( n_1 \) and \( n_2 \) required in the above expressions were given in the second section of this appendix.
The last step is to assemble all the elements in the definition of the Jacobian and Hessian of the objective function given in equation (A.64). If we denote the objective function with \( G(u,v,s,t) \), we have

\[
G = A - C^2 + 1 - D^2
\]  

(A.93)

The elements of the Jacobian of \( G \) are then defined by:

\[
G_a = A_a - 2C C_a - 2 D D_a, \text{ where } a = u, v, s, t
\]  

(A.94)

and the elements of the Hessian of \( G \) are then defined by:

\[
G_{ab} = A_{ab} - 2C_a C_b - 2 C C_{ab} - 2 D_a D_b - 2 D D_{ab},
\]  

where \( a, b = u, v, s, t \)  

(A.95)
Appendix B

DERIVATIVES OF THE DISTANCE FUNCTION AND TENSORIAL DIFFERENTIAL EQUATIONS

In this appendix, we develop the tensorial differential equations describing iso-distance curves on two surfaces and the intersection curve between two surfaces and we determine the first and second partial parametric derivatives of the oriented distance function between two surfaces. The tensorial differential equations describing iso-distance curves on the two surfaces are obtained using the differential equations describing the orthogonal projection of a curve on a surface and the differential equations describing an iso-distance curve of a surface from another surface.

B.1 Tensorial Differential Equations

The orthogonal projection of a point $R$ onto a $C^1$ continuous surface $q(s,t)$ is the set of points $Q$ of the surface such that the vector $R - q$ is normal to the surface at any point $q \in Q$ [Pegna 89]. For an orthogonal projection of a point of surface $r(u,v)$ onto surface $q(s,t)$ the following needs to be valid

\[ (r(u,v) - q(s,t)) \cdot q_s = 0 \quad (B.1) \]

\[ (r(u,v) - q(s,t)) \cdot q_t = 0 \quad (B.2) \]

When point $R$ describes a curve $\gamma(w)$ on surface $r(u,v)$, then its projection on another surface in general describes another curve on the second surface, and equations (B.1) and (B.2) must be valid at all points in this curve and consequently:

\[ \frac{d}{dw}((r(u,v) - q(s,t)) \cdot q_s) = 0 \quad (B.3) \]

\[ \frac{d}{dw}((r(u,v) - q(s,t)) \cdot q_t) = 0 \quad (B.4) \]
This projection curve is regular (ie. exhibits no self-intersections) provided no point of \( \gamma(w) \) is a principal center of curvature of the second surface [Pegna 89].

Expanding the differentiation in the above equations and using the chain rule, we obtain the differential equations describing the projection curve of \( \gamma(w) \) in the parametric domain of surface \( q(s,t) \).

\[
(r_{u\frac{du}{dw}} + r_{v\frac{dv}{dw}} - q_s\frac{ds}{dw} - q_t\frac{dt}{dw}) \cdot \mathbf{q}_s + (r - q) \cdot (q_{ss\frac{ds}{dw}} + q_{st\frac{dt}{dw}}) = 0 \\
(r_{u\frac{du}{dw}} + r_{v\frac{dv}{dw}} - q_s\frac{ds}{dw} - q_t\frac{dt}{dw}) \cdot \mathbf{q}_t + (r - q) \cdot (q_{ts\frac{ds}{dw}} + q_{tt\frac{dt}{dw}}) = 0
\]  

which becomes

\[
\begin{bmatrix}
\frac{ds}{dw} \\
\frac{dt}{dw}
\end{bmatrix} = [K_{ij}]^{-1} \begin{bmatrix} G_{ab} \\
\frac{du}{dw} \frac{dv}{dw}
\end{bmatrix}
\]  

\( [K_{ij}] \) is the orthogonal projective tensor of surface \( q(s,t) \) given by

\[
K_{ij} = \mathbf{q}_i \cdot \mathbf{q}_j - (r-q) \cdot q_{ij} \quad \text{with} \quad i,j = s,t
\]

a function of the first and second fundamental forms of surface \( q(s,t) \). \( [G_{ab}] \) is given by

\[
G_{ab} = r_b \cdot q_a \quad \text{with} \quad a = s,t \quad \text{and} \quad b = u,v
\]

Equation (B.7) assumes the knowledge of curve \( \gamma(w) \) on surface \( r(u,v) \). We are interested in the intersection curve of the two surfaces, which is an iso-distance curve of the distance function \( \phi(u,v) \) defined in Section 3.2. Provided \( \phi_u \) and \( \phi_v \) do not vanish simultaneously, this can be expressed as the solution of

\[
\frac{d\phi}{dw} = \phi_u\frac{du}{dw} + \phi_v\frac{dv}{dw} = 0
\]  

The general solution of equation (B.10) is

\[
\begin{bmatrix}
\frac{du}{dw} \\
\frac{dv}{dw}
\end{bmatrix} = [\alpha \phi_v, -\alpha \phi_u]
\]

If parameter \( w \) corresponds to arc length in the parameter space of surface \( r(u,v) \), then parameter \( \alpha \) is given by
\[ \alpha = \frac{1}{\sqrt{\phi_u^2 + \phi_v^2}} \tag{B.12} \]

Equations (B.7) and (B.11) are a four-dimensional first order differential equation system describing a curve on the second surface which is equidistant to curve \( \gamma(w) \) on the first surface. The intersection curve between the two surfaces is a special case of this differential equation system. In this case, the second term in equation (B.8) (the contribution of the second fundamental form of surface \( q(s,t) \)) is zero, since for an intersection point \( r - q = 0 \). The final tensorial differential equations describing the transversal intersection curve of two surfaces are given by:

\[
\frac{du}{dw} = \frac{\phi_v}{\sqrt{\phi_u^2 + \phi_v^2}} \tag{B.13}
\]
\[
\frac{dv}{dw} = \frac{-\phi_u}{\sqrt{\phi_u^2 + \phi_v^2}} \tag{B.14}
\]
\[
\left[ \frac{ds}{dw}, \frac{dr}{dw} \right]^T = [q_i \cdot q_j]^{-1} \left[ r_b \cdot q_a \right] \left[ \frac{du}{dw}, \frac{dv}{dw} \right]^T \tag{B.15}
\]

with \( b = u, v \) and \( i, j, a = s, t \)

B.2 Partial Derivatives of the Oriented Distance Function

The oriented distance function from a point on the parameter space of surface \( r(u,v) \) to a point \( Q \) on surface \( q(s,t) \) is defined as

\[
\phi(u,v) = n_q[Q(r(u,v)) - q(r(u,v))] \tag{B.16}
\]

with \( n_q \) the normal vector on surface \( q(s,t) \). When \( \phi(u,v) \) is a proper distance function,

\[
(r-q) \cdot q_s = 0 \text{ and } (r-q) \cdot q_t = 0 \tag{B.17}
\]

The first partial derivatives of equation (B.16) for such cases are given by
\[ \phi_u = n_q \cdot r_u \]  \hspace{1cm} (B.18)

\[ \phi_v = n_q \cdot r_v \]  \hspace{1cm} (B.19)

The derivation of equation (B.18) is shown below.

\[ \phi_u = n_q \cdot (r_u - q_u) + n_{q,u} \cdot (r - q) \Rightarrow \]  \hspace{1cm} (B.20)

Using the chain rule and the property of the normal to a surface \( n_q \cdot q_s = 0 \) and \( n_q \cdot q_t = 0 \), the term \( n_q \cdot q_u \) equals 0. Using the chain rule, equation (B.17) and the normal vector property \( n_{q,s} \cdot n_q = 0 \) and \( n_{q,t} \cdot n_q = 0 \), we find that the term \( n_{q,u} \cdot (r-q) \) of equation (B.20) is also zero and equation (B.18) is valid. A similar argument holds for equation (B.19).

The second partial derivatives of equation (B.16) are computed using the tensorial differential equations describing the orthogonal projection of a curve on a surface. They are given by

\[ \phi_{uu} = n_{q,u} \cdot r_u + n_q \cdot r_{uu} \]  \hspace{1cm} (B.21)

\[ \phi_{uv} = \phi_{vu} = n_{q,v} \cdot r_u + n_q \cdot r_{uv} \]  \hspace{1cm} (B.22)

\[ \phi_{vv} = n_{q,v} \cdot r_v + n_q \cdot r_{vv} \]  \hspace{1cm} (B.23)

where \( n_{q,u} \) and \( n_{q,v} \) are given by

\[ n_{q,u} = [n_{q,s}, \ n_{q,t}] \cdot [K_{ij}]^{-1} [r_u \cdot q_s, \ r_u \cdot q_t]^T \]  \hspace{1cm} (B.24)

\[ n_{q,v} = [n_{q,s}, \ n_{q,t}] \cdot [K_{ij}]^{-1} [r_v \cdot q_s, \ r_v \cdot q_t]^T \]  \hspace{1cm} (B.25)

where \([K_{ij}]\) is defined in equation (B.8) and is the orthogonal projective tensor of surface \( q(s,t) \). To obtain equations (B.24) and (B.25) the chain rule is used to get

\[ n_{q,u} = n_{q,s} \frac{ds}{du} + n_{q,t} \frac{dt}{du} \]  \hspace{1cm} (B.26)

\[ n_{q,v} = n_{q,s} \frac{ds}{dv} + n_{q,t} \frac{dt}{dv} \]  \hspace{1cm} (B.27)

If curve \( \gamma(w) \) is a \( v \) isoparameter curve on surface \( r(u,v) \), then \( w = u \) is substituted in
equation (B.5). If curve $\gamma(w)$ is a $u$ isoparameter curve on surface $r(u,v)$, then $w = v$ is substituted in equation (B.6). These new equations are then used to solve for $\frac{dr}{dw}, \frac{dt}{dw}, \frac{dr}{dv}$ and $\frac{dr}{dv}$ which are substituted in equations (B.26) and (B.27) to obtain equations (B.24) and (B.25).
Appendix C

DISTANCE OF A POINT FROM A PARAMETRIC SURFACE

In this Appendix, we develop the methodology used to determine the minimum distance of a point from a parametric surface defined over a rectangular parametric domain. Figure C-1 illustrates the problem.

Figure C-1: Distance of a Point From a Surface

Constrained minimization techniques are used to solve this problem. The objective function to minimize is given by

\[ \text{Minimize } F(s,t) = \|P - q(s,t)\|^2. \] (C.1)

This is subject to bounds in the variables \( s \) and \( t \in [0,1] \).

For the solution of this minimization problem, an initial approximation to the minimum distance point is required. There are two techniques used to provide this initial approximation. The first technique exploits the property of B-splines, that their polyhedral net provides an approximation of their geometry. According to this technique, the control
point of surface \( q(s,t) \), which is the minimum distance from point \( P \) is determined and the node values corresponding to this point are used as the initial approximation to the minimum distance point. The second technique is applicable for general parametric surfaces for which the notion of a control polyhedron is not applicable. This technique evaluates a lattice of points on the parametric surface and then determines the point on the lattice which is closest to point \( P \). The parameter values of this point are used as initial approximations to the minimization problem. There is no guarantee that either of these techniques will provide a good initial approximation to the minimum distance point, especially if the point of the first surface is near the cut locus of the second surface. In this case the initial approximation may be closer to a local minimum of the distance between the point and the surface.

For a robust and efficient solution of this minimization problem, the Jacobian and Hessian of the objective function \( F(s,t) \) are used at each iteration step as was also explained in Section 5.3.1. The Jacobian of this objective function has elements given by:

\[
F_s(s,t) = -2 (P - q) \cdot q_s
\]  
\[
F_t(s,t) = -2 (P - q) \cdot q_t
\]  

(C.2)  
(C.3)

The Hessian matrix of this objective function has elements given by:

\[
F_{ss}(s,t) = -2 (P - q) \cdot q_{ss} + 2 q_s \cdot q_s
\]  
\[
F_{st}(s,t) = -2 (P - q) \cdot q_{st} + 2 q_t \cdot q_s
\]  
\[
F_{tt}(s,t) = -2 (P - q) \cdot q_{tt} + 2 q_t \cdot q_t
\]  

(C.4)  
(C.5)  
(C.6)

This numerical method solves the general minimum distance problem of a point from a surface. In this work, a minimum distance point is only considered "proper", if the vector \( P-q \) is collinear with the normal at point \( q \). Improper minimum distance points usually occur on one of the boundary curves of the surface and are the result of the boundedness of the domain of the surface definition. They may also occur if the point is on the cut locus of
the surface and this may lead to inaccuracy in the minimum distance computation (computation of a local minimum, instead of the global minimum). The ideas developed in this thesis for the computation of the critical points of the vector field of the distance function between two surfaces, can be readily used for computation of all extrema of the distance function between a point and a surface, in order to find the global minimum distance. This problem was addressed and a method for its solution was described in sections 3.2 and 5.2.2 as the determination of all extrema of a height function of two variables, such as equation (C.1).
Appendix D

COMPUTATION OF SURFACE AREA ENCLOSED BY BOUNDING CONES AND RECTANGULAR PYRAMIDS ON THE UNIT SPHERE

In this Appendix, we present the methodology used to compute the surface area on the unit sphere enclosed by the rectangular pyramids and the cones described in Chapter 4, which bound the first partial derivatives and the normal vectors of a parametric surface. This surface area provides a metric for comparison of the tightness of the two alternative formulations. The region on the unit sphere which is enclosed by a rectangular pyramid or cone, also bounds the vector directions which are bounded by the pyramid or cone. To compute the surface area of such a region spherical trigonometry is used [Beyer 82].

Figure D-1 presents a unit sphere and half of the region enclosed by a cone with half angle \( q \), base radius \( a \) and height \( 1-h \). The surface area of this region is given by [Beyer 82]

\[ S = 2 \pi h = \pi (a^2 + h^2) = 4 \pi \sin^2(q/2) \quad (D.1) \]

For our computations, we need the ratio of the areas enclosed by both parts of the cone as compared to the total area of the unit sphere. For the bounding cone this is given by

\[ S_{ratio,cone} = 2 \, \sin^2(q/2) \quad (D.2) \]

For the computation of the area of the region enclosed by a rectangular pyramid, the trigonometric results for the computation of the area of an oblique spherical triangle are needed. A spherical triangle is constructed from the intersection of three major circles of the sphere. Figure D-2 presents a spherical triangle on a unit sphere with sides of length \( a \), \( b \), \( c \) and with angles \( A \), \( B \), \( C \) opposite to these sides. Using the half-angle formulas for an oblique spherical triangle we can determine \( A \), \( B \), \( C \) if we know \( a \), \( b \), \( c \) [Beyer 82]

\[ \tan \frac{1}{2} A = \frac{k}{\sin(s - a)}, \quad \tan \frac{1}{2} B = \frac{k}{\sin(s - b)}, \quad \tan \frac{1}{2} C = \frac{k}{\sin(s - c)} \quad (D.3) \]

where
**Figure D-1:** Definition of a Bounding Cone on a Unit Sphere

\[ k^2 = \frac{\sin(s - a) \sin(s - b) \sin(s - c)}{\sin s} \quad \text{and} \quad (D.4) \]

\[ s = \frac{1}{2} (a + b + c). \quad (D.5) \]

The area enclosed by a spherical triangle with angles A, B, C in a unit sphere is given by [Beyer 82]

\[ S = A + B + C - \pi \quad (D.6) \]

All the sides of a rectangular pyramid belong to major circles of the sphere and as a result any three sides of the rectangular pyramid form a spherical triangle on the sphere. As a result the area of the region enclosed by a rectangular pyramid on the unit sphere is the sum of the areas of two identical spherical triangles (formed by two adjacent sides of the pyramid and the diagonal plane of the pyramid). The size of each side of the spherical triangle is equal to the angle formed by two of the edges of the rectangular pyramid. As a result if \( C_0, C_1, C_2 \) are three of the edges of the rectangular pyramid (unit vectors), then the sides of the spherical triangle defined by these vectors are given by

\[ a = \text{Arcos}(C_0 \cdot C_1), \quad b = \text{Arcos}(C_0 \cdot C_2), \quad c = \text{Arcos}(C_1 \cdot C_2) \quad (D.7) \]
Using equations (D.3) to (D.7) the area of the region enclosed by a rectangular pyramid in the unit sphere can be computed. For our computations, we need the ratio of the areas enclosed by both parts of the pyramid as compared to the total area of the unit sphere. For the bounding rectangular pyramid this is given by

\[ S_{\text{ratio,pyramid}} = \frac{A + B + C - \pi}{\pi} \]  

(D.8)

Equations (D.2) and (D.8) were used in Chapter 4, to construct Table 4-I and compare the tightness of the two formulations.
References

[Abhyankar 86a] Abhyankar, S. S., Bajaj, C.
Automatic Rational Parametrization of Curves and Surfaces I: Conics and Conicoids.
Report No: CSD-TR-583, Computer Sciences Department, Purdue University, March, 1986.

[Abhyankar 86b] Abhyankar, S. S., Bajaj, C.
Automatic Parametrization of Rational Curves and Surfaces III:
Algebraic Plane Curves.
Report No. CSD-TR-619, Department of Computer Sciences, Purdue University, August, 1986.

[Arner 87] Arner P. R.
Another Look at Surface / Surface Intersection.

[Arnon 83] Arnon, D. S.
Topologically Reliable Display of Algebraic Curves.

[Bajaj 87] Bajaj, C., Hoffmann, C. M., Hopcroft, J.
Tracing Planar Algebraic Curves.
Report No. CSD-TR-637, Department of Computer Sciences, Purdue University, September, 1987.

[Bajaj 88] Bajaj, C. L., Hoffmann, C. M., Hopcroft, J. E., Lynch, R. E.
Tracing Surface Intersections.

[Barnhill 87] Barnhill, R. E., Farin, G., Jordan, M., Piper, B. R.
Surface/Surface Intersection.

[Barnhill 89] Barnhill R. E., Kersey S. N.
A Marching Method for Parametric Surface / Surface Intersection.

CRC Standard Mathematical Tables.

[Blum 73] Blum H.
Biological Shape and Visual Science (Part I).

[Boehm 80] Boehm W.
Inserting New Knots Into B-Spline Curves.
[Chen 86] Chen Y. J.
Computational Problems in Computer-Aided Design and Manufacturing of Ruled and Parametric Surfaces.

[Chen 87] Chen Y. J., Ravani B.
Offset Surface Generation and Contouring in Computer-Aided Design.

[Chen 88] Chen J. J., Ozsoy T. M.
Predictor-Corrector Type of Intersection Algorithm for C^2 Parametric Surfaces.

[Cheng 88] Cheng, K.-P.
Using Plane Vector Fields to Obtain All the Intersection Curves of Two General Surfaces.

[Chiang 89a] Chiang, L. E., Wilde D. J.
Solving Quadratic Surface Intersections Using a General Cylinder as an Auxiliary Surface.

[Chiang 89b] Chiang, L. E., Wilde D. J.
A Method to Find Parametric Bicubic Surface Intersections.

[Chuang 88] Chuang, J. H., Hoffmann, C. M.
On Local Implicit Approximation and its Applications.
Report No: CSD-TR-812, Computer Sciences Department, Purdue University, September, 1988.

[Dahlquist 74] Dahlquist, G. and Bjorck, A.
Numerical Methods.

[De Boor 72] De Boor, C.
On Calculating with B-Splines.

[deMontaudouin 86]
De Montaudouin, Y., Tiller, W., Vold, H.
Applications of Power Series in Computational Geometry.
[deMontaudouin 89]
De Montaudouin, Y.
Cross Product of Cones of Revolution.

[Dokken 85]
Dokken, T.
Finding Intersections of B-Spline Represented Geometries Using Recursive Subdivision Techniques.

[Farouki 86a]
Farouki, R. T.
The Characterization of Parametric Surface Sections.

[Farouki 86b]
Farouki, R. T.
The Approximation of Non-Degenerate Offset Surfaces.

[Farouki 87]
Farouki, R. T., Rajan, V. T.
On the Numerical Condition of Polynomials in Bernstein Form.

[Farouki 89]
Farouki, R. T., Neff C. A.
Algebraic Properties of Plane Offset Curves.
to appear.

[Faux 81]
Faux, I. D. and Pratt, M. J.
*Computational Geometry for Design and Manufacture.*

[Gear 71]
Gear C. W.
*Numerical Initial Value Problems in Ordinary Differential Equations.*

[Geisow 83]
Geisow, A.
*Surface Interrogations.*

[Gill 74]
Gill, P. E., Murray, W.
Newton-Type Methods for Unconstrained and Linearly Constrained Optimization.

[Gill 81]
Gill P.E., Murray W., Wright A.
*Practical Optimization.*
*B-Spline Curves and Surfaces.*

[Gursoy 89] Gursoy H. N.
*Shape Interrogation by Medial Axis Transform for Automated Analysis.*

*Modern Numerical Methods for Ordinary Differential Equations.*

[Hoffmann 87] Hoffmann, C. M., Lynch, R.E.
Following Space Curves Numerically.

[Hoffmann 89] Hoffmann C. M.
*Geometric and Solid Modeling: An Introduction.*

Implementation of a Divide-and-Conquer Method for Intersection of Parametric Surfaces.

[Johnstone 89] Johnstone J. K., Goodrich M. T.
A Localized Method for Intersecting Plane Algebraic Curve Segments.

*Plane Vector Fields.*

[Kriezis 89] Kriezis, G. A., Patrikalakis, N. M.
Interrogation of Rational Polynomial Surface Intersections.

[Kriezis 90] Kriezis, G. A., Prakash, P. V., Patrikalakis, N. M.
A Method for Intersecting Algebraic Surfaces With Rational Polynomial Patches.
To appear.

[Lane 80] Lane, J. M., Riesenfeld, R. F.
A Theoretical Development for the Computer Display and Generation of Piecewise Polynomial Surfaces.
[Lane 81] Lane, J. M., Riesenfeld, R. F.
Bounds on a Polynomial.

[Lasser 86] Lasser D.
Intersection of Parametric Surfaces in the Bernstein-Bezier
Representation.

[Levin 79] Levin, J. Z.
Mathematical Models for Determining the Intersections of Quadric
Surfaces.

[Lyche 85] Lyche, T., Cohen, E., Morken, K.

Solutions of Tangential Surface and Curve Intersections.

[Miller 87] Miller, J. R.
Geometric Approaches to Nonplanar Quadric Surface Intersection
Curves.

[Mortenson 85] Mortenson, M. E.
*Geometric Modeling.*

[Mudur 84] Mudur S. P., Koparkar P.A.
Interval Methods for Processing Geometric Objects.

[NAG 89] *NAG Fortran Library Manual*

[Ortega 70] Ortega, J. M., Rheinboldt, W. C.
*Iterative Solution of Nonlinear Equations in Several Variables.*

[Patrikalakis 87] Patrikalakis, N. M. and Prakash, P. V.
*Computation of Algebraic Polynomial Parametric Surface Intersections.*

[Patrikalakis 88] Patrikalakis, N. M. and Kriezis, G. A.
*Piecewise Continuous Algebraic Surfaces in Terms of B-Splines.*
[Patrikalakis 89] Patrikalakis, N. M. and Krieps, G. A.
Representation of Piecewise Continuous Algebraic Surfaces in Terms of B-Splines.

[Patrikalakis 90a] Patrikalakis, N. M., Prakash, P. V.
Surface Intersections for Geometric Modeling.

[Patrikalakis 90b] Patrikalakis, N. M.
Shape Interrogation.

[Patrikalakis 90c] Patrikalakis N. M., Bardis, L.
Localization of Rational B-Spline Surfaces.
*Design Laboratory Memorandum 90-1*, February, 1990.

[Pegna 89] Pegna J.
Interactive Design of Curvature Continuous Fairing Surfaces.

[Pegna 90] Pegna J., Wolter F. - E.
Designing and Mapping Trimming Curves on Surfaces Using Orthogonal Projection.

[Peng 84] Peng Q. S.
An Algorithm for Finding the Intersection Lines Between Two B-Spline Surfaces.

[Petersen 84] Petersen, C. S.
Adaptive Contouring of Three-Dimensional Surfaces.

[Piegl 89] Piegl, L.
Geometric Method of Intersecting Natural Quadrics Represented in Trimmed Surface Form.

[Prakash 88a] Prakash, P. V.
*Computation of Surface-Surface Intersections for Geometric Modeling.*
[Prakash 88b] Prakash, P. V., Patrikalakis, N. M.
*Surface-to-Surface Intersections for Geometric Modeling.*

[Pratt 86] Pratt, M. J.
Reliability of Solid Modelling Systems.

[Requicha 82] Requicha, A. A. G., Voelcker, H. B.

[Rossignac 87] Rossignac, J. R., Requicha, A. A. G.
Piecewise-Circular Curves for Geometric Modeling.

[Sarraga 83] Sarraga, R. F.
Algebraic Methods for Intersections of Quadric Surfaces in GMSOLID.

[Sarraga 84] Sarraga R.F., Waters W.C.
Free-Form Surfaces in GMSOLID: Goals and Issues.

[Sederberg 84a] Sederberg, T. W.
Planar Piecewise Algebraic Curves.

[Sederberg 84b] Sederberg T. W., Anderson D. C., Goldman R. N.
Implicit Representation of Parametric Curves and Surfaces.

[Sederberg 85] Sederberg, T. W.
Piecewise Algebraic Surface Patches.

[Sederberg 88a] Sederberg, T. W., Meyers, R. J.
Loop Detection in Surface Patch Intersections.

[Sederberg 88b] Sederberg, T. W., Wang, X.
Rational Hodographs.

[Sederberg 88c] Sederberg, T. W.
An Algorithm for Algebraic Curve Intersection.
Sederberg, T. W., Nishita, T.
Direct Approximation of Surface Patch Intersection Curves.  

Sederberg T. W., Zhao J., Zundel A. K.
Approximate Parametrization of Algebraic Curves.  

Sederberg, T. W., Christiansen, H. N., Katz, S.
An Improved Test for Closed Loops in Surface Intersections.  

Sederberg, T. W., White, S. C., Zundel, A. K.
Fat Arcs: A Bounding Region with Cubic Convergence.  

Sinha, P., Klassen, E., Wang, K. K.
Exploiting Topological and Geometric Properties for Selective Subdivision.  

Smith, B., Rinaudot, G. R., Reed, K. A., Wright, T.

Solomon, B. J.
*Surface Intersection for Solid Modelling.*  

Thomas, S. W.
*Modeling Volumes Bounded by B-Spline Surfaces.*  
PhD thesis, Department of Computer Science, University of Utah, Salt Lake City, Utah, June, 1984.

Tiller, W.
Rational B-Splines for Curve and Surface Representation.  

Varady, T.
Surface-Surface Intersections for Double-Quadratic Parametric Patches in a Solid Modeller.  

Waggenspack W. N., Anderson D. C.
Piecewise Parametric Approximations for Algebraic Curves.  
*Cut Loci in Bordered and Unbordered Riemannian Manifolds.*