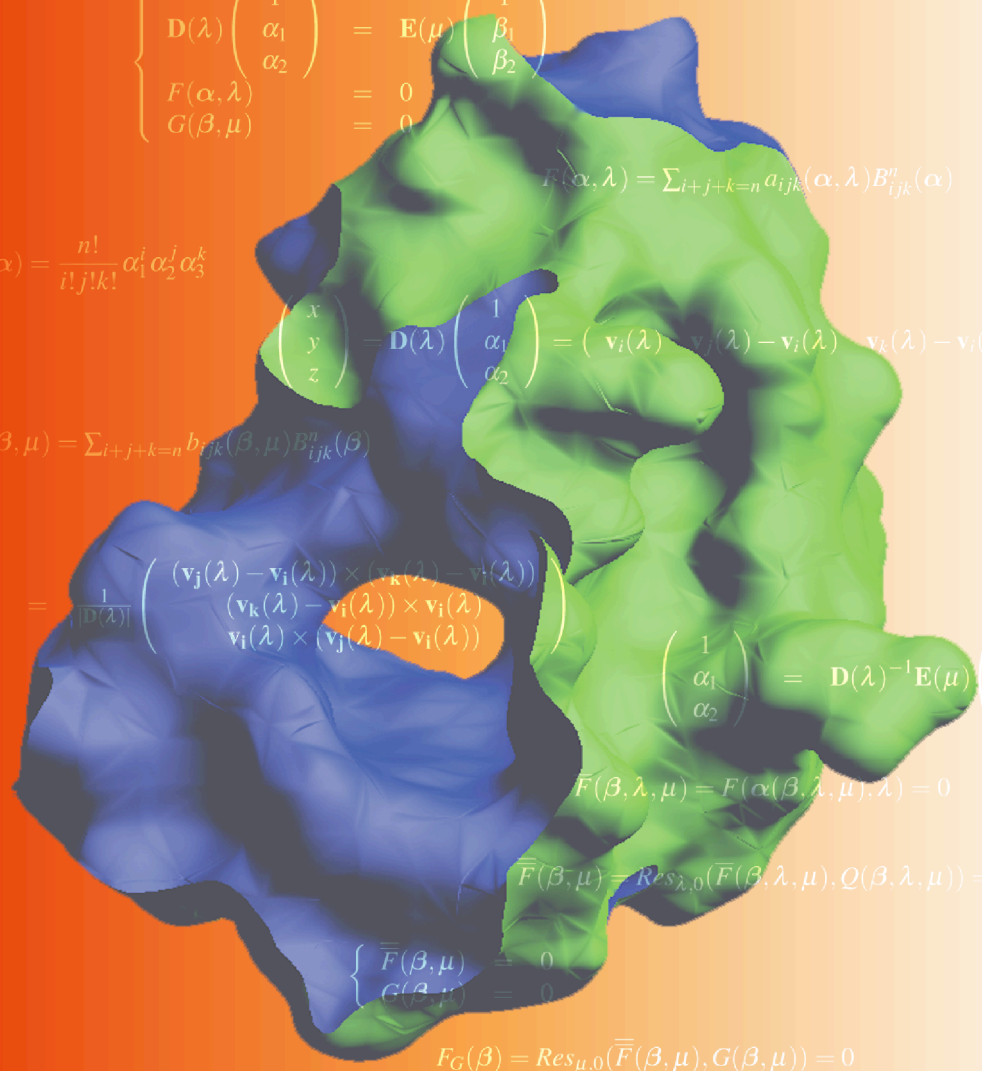
ROMA TRE
UNIVERSITÀ DEGLI STUDI

Tesi Dottorato SICS

Bio-medical Symbolic Modeling with Algebraic Patches



Simone Portuesi

Roma Tre University
SICS Doctorate School

Bio-medical Symbolic Modeling with Algebraic Patches

Simone Portuesi

Bio-medical Symbolic Modeling with Algebraic Patches

A thesis presented by
Simone Portuesi

in partial fulfillment of the requirements for the degree of
Dottore di Ricerca (Doctor of Research)

Roma Tre University
Faculty of Engineering
SICS Doctorate School

2008

DOTTORANDO (DOCTORATE CANDIDATE):
Simone Portuesi

TUTOR/RELATORE (PH.D. ADVISOR):
Prof. Alberto Paoluzzi

COORDINATORE DI DOTTORATO (PH.D. COORDINATOR):
Prof. Fabio Carassiti

Dedicated to my father

Abstract

The mathematical description of geometry is of paramount importance in the modeling of systems of all kinds. Standard geometric modeling describes curved objects through parametric functions as the image of compact domains. Alternatively, as in algebraic geometry, one may describe curved geometry as the zero-set of polynomials. Geometric modeling of biological systems highlight certain persistent and open problems more effectively addressed using algebraic geometry.

In this thesis a framework for computer-based geometric representation based on algebraic geometry is introduced. Several algebraic representation schemes known as A-splines and A-patches are detailed as a description of geometry, by using a piecewise continuous gluing of algebraic curves and surfaces. The application of A-splines and A-patches to biological modeling is discussed in the context of protein molecular interface modeling. The rationale is to present an algebraic representation of bio-modeling under an unified point of view. This framework provides a suitable background for the main contribution of this thesis: the formulation and implementation of algorithms for Boolean operations (union, intersection, difference, etc.) on the algebra of curved polyhedra whose boundary is triangulated with A-patches. Boolean operations on curved geometry are yet an open obstinate research problem and its exact solution is only definable within the domain of algebraic geometry. The exact formulation is here used as basis for a geometrically approximate yet topologically accurate solution, closed in the geometric domain of A-patches. The prototype implementation has been applied to pairs of molecular models of ligand proteins in docking configuration. To date, the computational use of algebraic geometry is still experimental and is far from being a major component of current systems. This thesis shows an evidence that representation techniques derived from algebraic geometry have strong potential in bio-medical modeling, still needing much further research and engineering.

Acknowledgments

I'd like to thank Fabio Carassiti for making the SICS doctorate school possible; Alberto Paoluzzi for his patience during the duration of the doctorate; Chandrajit Bajaj for introducing me to new ways to do geometric modelling; all my colleagues of the Doctorate school with whom I shared this experience; all the students with whom I shared the PLM Laboratory; all the people that helped me at my visit at ICES in Austin, Texas; and, of course, my family and friends for their support in all these years.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Computer Aided Design and Engineering is very effective with the design, modeling and simulation of physical systems with a very large number of components and fine granularity. Its application to natural systems has been found less satisfying especially when these systems have a degree of intra- and inter- mutual dependance. The mathematical description of geometry is of paramount importance in the modeling of these systems. In standard geometric modeling, curved geometry is represented by parametric functions, that is, a mathematical description on how to generate a certain point set. In mathematics one may also describe geometry as the geometric loci of points satisfying a set of equations. This is an important and established branch of mathematics known as algebraic geometry. Algebraic geometry has been successfully applied to the geometric modeling and visualization of biological systems. Algebraic geometry is very useful in the geometric and topological reconstruction from bio-medical acquired images. An algebraic geometric description is also more natural in the geometric description of biological molecules. To date, techniques and methods from algebraic geometry have been proven advantageous in both troublesome geometric reconstructions and certain open geometric modeling problems. In this chapter the algebraic geometry representation schemes are introduced highlighting their rich mathematical structure, on why they are apt to the application to biomedical sciences and engineering, and their distinctive mathematical properties and constructions useful for geometric modeling.

1.1 Algebraic Finite Elements

In Geometric Modeling [Man88, Hof89, Pao03], geometry is represented as a piecewise gluing (with continuity constraints) of geometric elements. A two dimensional curved surface element is called "patch". Each element is described mathematically, and traditionally, for tractability and computational efficiency, by parametric functions. An

m-dimensional curved geometry embedded in an n-dimensional one is represented as the image of a compact subset U of $\mathbb{R}^m$ by vector valued function $f : \mathbb{R}^m \to \mathbb{R}^n$ (usually polynomial or rational). On the other hand it is possible to represent each element implicitly as an algebraic variety, the ideal generated by the zero set of one or more polynomials $e : \mathbb{R}^n \to \mathbb{R}$. The image $\{\mathbf{p} = f(u)\}_{u \in U \subseteq \mathbb{R}}$ of a single n-dimensional parametric function $f : \mathbb{R} \to \mathbb{R}^n$ will be called a *parametric element* and geometrically describes a curve (a trajectory) in the n-dimensional space. A single polynomial function in a n-dimentional space $e : \mathbb{R}^n \to \mathbb{R}$ describes a scalar field. The zero-set $\{e(\mathbf{p}) = 0\}_{\mathbf{p} \in \mathbb{R}^n}$ is called an *algebraic hyper-element* and geometrically describes a $(n-1)$-dimensional hyper-surface in a n-dimensional space. Parametric and algebraic representation are dual to each other, parametric representation is generative while algebraic representation is co-generative: the former describes a process on how the geometry is generated; the later express a constraint in equational form that the geometry must satisfy.

Geometric representation using piecewise parametric rational functions is a well established technique [Hof89, Far01, Pao03], used in all major modeling software. There are general, computationally efficient, and mutually convertible parametric representations of arbitrary degree and dimensionality, moreover control data usually have an intuitive geometric meaning [PT96, Gol02]. Computer geometric representation using algebraic elements is a less explored field [Baj93, Baj90]. The study of algebraic varieties is an important and established branch of mathematics known as algebraic geometry [Abh90]. The geometric domain of algebraic varieties has a richer mathematical structure and is a geometric superset of parametric images of polynomial, rational, and irrational functions. An algebraic n-dimensional hyper-element has $\binom{d+n}{n} - 1$ degrees of freedom, where d is the degree. Compare to polynomial parametric or rational parametric element with $dn+1$ and $(d+1)n+1$ degrees of freedom respectively. It is to be noted that a d-degree n-dimensional polynomial parametric element may always be converted to a dn degree algebraic variety. By power-raising (to eliminate roots), distributing the denominator and computing the resultant (to eliminate the parameter) it is always possible given an irrational parametric function to compute a set of equations that the geometry will satisfy. Vice versa given a set of equation, in general one cannot formulate the geometry parametrically, though it is possible to define the set of parameterizable algebraic equations [AB88] and there are techniques to piecewise approximate an algebraic variety through parametric functions [BX97]. Algebraic geometry has a richer and intrinsic topological properties. An algebraic variety F may be disconnected, self intersect at singular points $\nabla(F)(\mathbf{p}) = \mathbf{0}$, have x_i-extreme points $\partial F(\mathbf{p})/\partial x_i = 0$, and other second derivative $\mathscr{H}(F(\mathbf{p}))$ properties (curvature, flexes, saddles, etc). Due to its richer mathematical structure algebraic geometry is regarded as complicated and arcane, Moreover it is less regular and more fragmentary then the simpler parametric geometry. The task engineering such results into representation schemes, modeling techniques and operative algorithms software is yet in a developing phase.

With the term *Algebraic Finite Elements* one wants to cover all description of geometry

as a piecewise gluing of compact algebraically defined discrete elements, that is, each element geometry is the zero set of one or more functions $f_l(\mathbf{x}) = 0,\ l \in [1..n-m]$, where $\mathbf{x}$ is a generic point of an n-dimensional space. The functions $f_l(\mathbf{x})$ are usually polynomial, but sometimes may be formulated in rational or irrational form to highlight certain parameters, in this case they may (by power-raising and denominator distribution) be converted to polynomials. While it is possible to model a general closed geometry of arbitrary genus using a single implicit surface. On one hand, the geometry of such global surface is difficult to specify, control and polygonize. On the other hand, the defining functions will be of high degree.

Why is *low* degree important? The *geometric degree* of an algebraic surface is the maximum number of intersections between the surface and a line, counting complex, infinite and multiple intersections. It is a measure of the "waviness" of the surface. This geometric degree is the same as the degree of the defining polynomial f of the algebraic surface in the implicit definition, but may be as high as n^2 for a parametrically defined surface with rational functions G_i of degree n. The geometric degree of an algebraic space curve is the maximum number of intersections between the curve and a plane, counting complex, infinite and multiple intersections. A well known theorem of algebraic geometry (Bezout's theorem) states that the geometric degree of an algebraic intersection curve of two algebraic surfaces may be as large as the product of the geometric degrees of the two surfaces [SR49]. The use of low degree surface patches to construct models of physical objects thus results in faster computations for subsequent geometric model manipulation operations such as computer graphics display, animation, and physical object simulations, since the time complexity of these manipulations is a direct function of the degree of the involved curves and surfaces. Furthermore, the number of singularities (sources of numerical ill-conditioning) of a curve of geometric degree m may be as high as m^2 [Wal78]. Keeping the degree low of the curves and surfaces thus leads to potentially more robust numerical computations.

In an abstract definition, an algebraic finite element is a smooth nonsingular piece of an algebraic variety delimited by a closed boundary. An *A-spline* (Algebraic Spline) is a curve segment in the plane defined by a single equation $f(\mathbf{x}) = f(x,y) = 0$, delimited by two points on the curve. An *A-patch* is a smooth nonsingular surface in a 3-dimensional space defined by a single equation $f(\mathbf{x}) = f(x,y,z) = 0$. The boundary must be a closed cycle of curve segments on the surface. Recursively one may define an n-dimensional finite element as an algebraic hyper-element delimited by a closed boundary of $(n-1)$-dimensional finite elements. Other varieties (e.g. a curve in space a surface in a m-dimensional space) may be defined either supplying more equations, or, by supplying a rational mapping. These two approaches are equivalent as an algebraic variety may always be formulated as a single equation and a rational mapping (by variable elimination).

Imposing continuity and bounds is more difficult in algebraic geometry thus making the gluing of algebraic elements problematic. When defining an algebraic finite element

representation of geometry, there are several constraints one need or wishes to impose:

Regularity The first set of constraint involve the "sanity" of the algebraic variety enclosed by the boundary. Algebraic varieties may be disconnected, non-continuous, have singularities and be multiple sheeted.

Continuity Gluing the various algebraic elements with a desired continuity imposes inter-patch constraints.

Control One, usually, wants to control the geometric element with control data that have an intuitive geometric meaning: interpolate and/or approximate points; impose a certain derivative information on certain geometric places; etc.

Though the abstract definition of piecewise geometric representation through algebraic elements is elegant and general. Due to these constraints, the abstract definition is not useful for practical purposes. Instead one focuses on sub-classes of A-patches defined by particular construction schemes and polynomials defined in an apropriate polinomial base. These computer based representations of geometry are problem dependent, of fixed degree and of fixed dimensionality. Conversions between representations tend to be ad-hoc and sometimes approximate. A typology of such representations are the "A-patches"[Dah89, BCX94, BCX95, BCX95, BX99b, BXHN02, BCX97, Guo92], smooth algebraic surface patch families, defined using a fixed degree trivariate polynomial within a compact polyhedron domain (also called the patch scaffold). Simple A-patches use a tetrahedrons, cube, or a triangular and quadrangular prism scaffolds. Common 2-dimensional scaffolds are triangles (2-simpexes), rectangles, parallelepipeds, and prisms (general 4-gon). In 3 dimensions tetrahedra, cubes, pyramids and prisms are used. The polyhedron is defined generatively from geometric data and thus are defined by low-degree (linear, bilinear, or trilinear) parametric functions. Moreover the scaffold supplies a local coordinate system on which the algebraic element is defined. The algebraic element is usually defined in Berstein-Bézier (B.B.) form in barycentric, tensor or mixed basis, because of the intuitivety of the control data and, more importantly, due to the diminishing variation property.

1.2 Bio-medical Modeling

Algebraic patches and other methods from algebraic geometry are particularly useful for geometric modeling and visualization of biological systems with applications in the biomedical sciences and engineering. Philosophically speaking, biological systems at all levels (eg. molecular surfaces, cell walls, biological tissue) have intrinsically curved geometry and this geometry has usually a predominant co-generative nature, as it tends to adapt to physical and external constraints. Algebraic geometry richer topological

structure is a resource when dealing with complex structure and to discriminate inherent topological characteristics from topological errors introduced by approximation.

Methods from algebraic geometry are apt to reconstruct geometry and topology from acquired data, eg. electron microscopy (EM), electron tomography (ET), Magnetic resonance (MRI), X-ray computed tomography (CT), ultrasound, X-raycrystallography (X-ray), or nuclear magnetic resonance spectroscopy (NMR) [CS04, LB02, BDST04, BLMP97, BDST04, BCCSX05, ZXB07, ZXB06]. In extreme simplification, imaging data may be seen as an uniform sampling of a real (single component) valued function $v = I(\mathbf{x})$. One wants to reconstruct a geometry to elucidate a certain structure. The image is processed using various image manipulation techniques to filter noise and enhance contrast. With methods derived from *Morse theory* one can build a hierarchical tree of critical points, effectively discerning the topology of all level sets and permitting active contouring methods. Furthermore one may subdivide the domain in a non uniform mesh and, by resampling, build for each subdivision a local low-degree algebraic variety. Effectively one constructs a geometric structure to the previously unstructured sampling. Models using A-patches built using this reconstruction activity will be presented as examples in this thesis. However the actual techniques are very specific and an interested reader may refer to the aforementioned literature.

Algebraic varieties are natural in the interface surface modeling of biological molecules. In this case the objective is to build a piecewise low-degree local approximation of a level-set of the electron density function [ZnBS05, KOB$^+$04, BYA03]. Geometric Models are used to approximate the real-world objects. Locating each molecule and modeling the ensemble of an arrangement of atoms is known as *Explicit Modeling* a.k.a. *Labeled Embedded Graph* (*"LEG/Bone/Skeleton"* or *"Ball-Stick"*) Model. Modeling to analyze density or average distribution of certain species concentration using analytical functions is known as *Implicit Modeling* a.k.a. *Interface Model* a.k.a. *Boundary Representation* (*"Skin"* or *"Spatial Occupancy"*) Model. These (geometric) models are useful not only to visualize, but to do extensive computations. For example, level sets give the subdomain in which the protein/ion density' is greater than certain threshold. This yields an analytical function, which facilitates in visualizing using splines/patches. The conversion of an Explicit Model to an Implicit Model is known as *Mean Field Approximation*. The appropriate model is chosen depending on the context. The interface between a protein and a water molecule is captured with an interface model. The interface model separates interface of one molecule from another. This model is a continuum geometric model.The input data for the explicit model is obtained from the *Protein Data Bank (PDB)*, a repository for 3-D structural data of proteins and nucleic acids. This data, typically obtained by *X-ray crystallography* or *NMR spectroscopy*, is submitted by biologists and biochemists from around the world, is released into the public domain, and can be accessed for free.

1.3 Symbolic Operations

Geometric modeling is not only the construction and representation of geometry but also the ability to operate and compute on the geometry. Algebraic varieties are symbolically closed under several geometric operations such as intersections, offset, revolution, blending and convolution:

Intersection The intersection of two varieties is well defined in algebraic geometry as it is simply the ideal generated by the defining equations. The intersection of two n-dimensional hyper-elements $e_1(\mathbf{x}) = 0$ of degree d_1 and $e_2(\mathbf{x}) = 0$ of degree d_2 is a $n-1$ dimensional element embedded in a n dimensional space of degree $d_1 d_2$ defined by the ideal generated by the equations:

$$\begin{cases} e_1(\mathbf{x}) = 0 \\ e_2(\mathbf{x}) = 0 \end{cases}$$

With algebraic methods one may "move" between the equations in the ideal: resultant, the elimination of a variable in a system of equations; and; Gröbner Bases, a "canonical" basis of the ideal. In particular an $n-1$ element embedded in a n dimensional space may be defined by a linear transformation $\mathbf{y} = \mathbf{T}\mathbf{x}$, ($\mathbf{y} = (y_1, \ldots, y_n)$), an $n-1$ hyper-element $e((y') = 0$, and a mapping $y_n = f(\mathbf{p}')$. The linear transformation is needed to avoid degenerate cases, without loss of generality one may consider the identity case (e.g. $\mathbf{T} = \mathbf{I}$, $\mathbf{x} = \mathbf{y}$). In this case the intersection of two hyper-elements $e_1(\mathbf{x}) = 0$ and $e_2(\mathbf{x}) = 0$ may be reduced to:

$$\begin{cases} e(\mathbf{x}) & = & Res_{x_n,0}(e_1(\mathbf{x}), e_2(\mathbf{x})) = 0 \\ x_n & = & Res_{x_n,1}(e_1(\mathbf{x}), e_2(\mathbf{x})) \end{cases}$$

Where $Res_{v,n}(e_1, e_2)$ is the n resultant on variable v between p_1 and p_2. That is, the polynomial obtained by the elimination of the variable v until the n term in the system of equations $e_1 = 0$ and $e_2 = 0$.

Lofting Given a set of curves, each represented by the intersection of two implicit surfaces i.e., $C_{i_0 \leq i \leq n} : (f_i, g_i)$ where f_i and g_i are surfaces in irreducible implicit form, the surface S which G^k interpolates C_i is known as the G^k *lofted surface* of C_i . The surface S is defined as

$$\alpha_0 f_0 + \beta_0 g_0^{k+1} = \alpha_1 f_1 + \beta_1 g_1^{k+1} = \ldots = \alpha_n f_n + \beta_n g_n^{k+1}$$

where α_i and β_i are polynomials of degree $\leq k$. Among the two implicit surfaces to represent a curve C_i , one typically represents the lower degree surface as g_i whereas the other one as f_i to obtain the computational efficiency. This definition can be trivially extended to higher dimensions. Note that if each $C_i \in \mathbb{R}^{n-1}$ the lofted surface is in $\mathbb{R}^n$.

Revolution Given a cyclic boundary in the xz-plane defined by $f(x,z) = 0$ and a planar curve in xy-plane defined by $g(x,y) = 0$, the surface $g(f(x,z),y) = 0$ obtained in revolving $g(x,y)$ around y-axis along the boundary of $f(x,z)$ is known as the surface of revolution of $g(x,y)$ around y-axis w.r.t.$f(x,z)$. This definition can be trivially extended to higher dimensions. Note that if each $C \in \mathbb{R}^{n-1}$ then the surface of revolution is in R^n.

Offset The convolution of an n-dimensional hyper-element $e(\mathbf{x}) = 0$ with a n-sphere creates a volume. The inner surface of the volume is said the inner offset and the outer surface of the volume is said outer offset. One may equivalently roll the n-sphere on both sides of the n-dimensional hyper-element.

In particular the possibility of calculating the intersection of two algebraic varieties is the fundamental operation to implement boolean operations on geometries described by algebraic finite elements. Boolean operators [LTH86, Mas93, PSR89, PBCF93] are the set-theoretical operations (Union, Intersection, Difference, containment, etc) between the point sets described by two geometric models. Boolean operation between geometrical models have the critical step of computing the intersection of two elements. The intersection of two varieties is well defined in algebraic geometry as it is simply the ideal generated by the union of the defining equations. However the exact boolean operator is often not closed inside the domain of fixed degree A-patches. The reason is simply that the intersection of two algebraic surfaces of degree d (say cubic) is in general, a space curve of degree d^2. It is possible however to build a piecewise low-degree topologically correct approximate, in case of tetrahedral and prismatic patches a piecewise linear tracing of the intersection curve[BX97]. This tracing is the starting point of the splitting of each geometric elements along the approximate intersection and the classification of each generated sub-element as either belonging to one and/or the other model. The final step is to assemble the result depending on the wanted operation.

1.4 Guide to the Reader

Topics typical in an undergraduate course in geometric modeling are required for the comprehension of this thesis though there has been an effort to recall them if needed. In particular it is required at a least prior exposition to[Pao03]:

- linear algebra, vector spaces, and affine spaces;
- representation of geometric space in both tensor and barycentric coordinate systems;
- the convection to denote variables as: scalar in lowercase italic $(a, \lambda, x, \ldots)$, vectors and points in lower case italic bold $(\mathbf{v}, \mathbf{p}, \boldsymbol{\alpha}, \ldots)$, matrix in upper case bold

($\mathbf{A}, \mathbf{T}, \ldots$), and operators and notation are quietly overloaded (when apropriate) to vectors and matrixes, e.g. $\begin{pmatrix} \mathbf{p_1} - \mathbf{q} & \mathbf{p_2} - \mathbf{q} & \alpha\mathbf{v} \end{pmatrix}$ to denote a row of vectors and thus a matrix (actually a 1-1 tensor);

- representation of polynomial and rational functions in power, Berstein-Bézier and B-spline basis;
- parametric representation of curves, surfaces, and higher dimensional curved geometry;
- notation is quietly overloaded for vector polynomials, e.g. $\sum_{i=0}^{n} \mathbf{p_i} x^i : \mathbb{R} \to \mathbb{R}^n$;
- polynomial tensor product and transfinite blending;
- fundamentals of geometric representation.

An introduction on algebraic geometry certainly would have been useful. However due to time and space considerations it has been preferred to introduce the needed concepts as they are needed. Furthermore, literature in algebraic geometry satisfying both mathematical soundness and operatively effectiveness under the engineering point of view is rare. However, in appendix A, several examples of application of symbolical algebraic geometry operations in the context of the PLaSM [Pao03, Pa] language and the GANITH [BR90b, Por07b] algebraic geometry toolkit are exposed. In this thesis a thorough full background on algebraic finite elements, A-splines, and A-patches is presented, it is a very specific field in geometric modeling and it is highly unlikely that a reader will have any prior knowledge.

This chapter 1 has: introduced bio-medical modeling using algebraically defined geometric elements; given an abstract definition of algebraic finite elements; discussed the hurdles and advantages of an implicit algebraic definition of geometry in respect to the more common explicit parametric representation; and presented the problematics for regularity, continuity and control for a more operative definition. To date no singular all comprehensive operative definition exists but a spectrum of representation schemes. In chapter 2 and 3, a thorough overview of these schemes is given and the relevant literature cited, for the two and three dimensional case respectively. Original sources of these chapters are Prof. Chandrajit Bajaj course in Geometric Modeling and Visualization [Baj07, Baj] as well his articles on A-Spines [BX99a, XBC00a, XBC00b, XBC00c] and A-patches [BCX95, BCX97, BX99b, XHB01, BX01, XBE02, ZXB07]. These three chapters are fundamental for the comprehension of this thesis.

In chapter 4 a series of exercise and examples on algebraic varieties and finite algebraic elements are presented. These examples were worked by the author and Prof. Na Lei (Inst. of Mathematics, Jilin University, China) for the post graduate course in Geometric Modeling and Visualization held by professor Chandrajit Bajaj at the University of Texas at Austin [Baj07]. The rationale is to provide a good overview on the possible operations

on algebraic finite elements in practical and by example manner, however, it does not constitute a requisite reading.

The construction algorithm for the construction of molecular models using A-patches is sketched in chapter 5. A similar algorithm is implemented in TexMol, a molecular visualization software developed at CVC (Center for Computational Visualization) of the ICES (Institute of Computational Engineering and Sciences) at the University of Texas at Austin. Original source of this chapter is an handout from Prof. Chandrajit Bajaj course in Geometric Modeling and Visualization [Baj07]. This chapter is useful as an introduction on the models, however it is not strictly a necessary reading.

Then in chapters 6, 7, and 8 three types of algebraic finite elements will be studied in detail and operational results will be given. Chapter 6, presents a detailed description including the calculation of the defining polynomial coefficients for arbitrary degree polynomial interpolating G^k continuous A-splines on triangular domain. Content of this chapter has been extracted and constitute a reduced form of [BX99a]. Chapter 7, presents a detailed description, an algorithm for the scaffold construction, and the calculation of the coefficients for cubic polynomial interpolating A-patches in tetrahedral domain. Content of this chapter has been extracted and constitute a reduced form of [BCX95]. Chapter 8 explains the construction from a triangulation and the control polynomial coefficients for simple cubic A-patches in a prismatic domain of both triangular and quadrangular base. Sources of this chapter are from both [ZXB07] and [BPP$^+$08] as well as other unpublished materials from Chandrajit Bajaj, Na Lei, Wenqi Zhao and the author. These three chapter give very specific information and are intended for a reader interested in operational result, however, they are not needed for an "intuitive" understanding of Algebraic Finite Elements. The techniques presented in the rest of the thesis may be adapted to any type of algebraic finite elements, however they will be specifically targeted at the case for cubic interpolating prismatic patches. This means that only chapter 8 is actually useful while chapters 6 and 7 are optional aimed at an interested reader.

Chapter 9, explains and discusses the new methods and algorithms for boolean operations on geometry described by an A-patch boundary representation. It is the principal contribution of the thesis. The results of the application of these methods and this algorithm to both simple geometry and to complex molecular models will be shown. This chapter is an expanded and detailed version of [BPP$^+$08]. Needless to say that is a required reading!

In the concluding chapter 10 a critical assessment on current status of algebraic biomodeling is made. The current weak and strong characteristics of the boolean algorithm and implementation is discussed as well as its possible future improvement and application.

In appendix A the current status on the integration of of the Ganith[BR90b, Por07b] algebraic geometry library in the PLaSM [Pao03, Pa] language is described. The rationale

is to further develop this work by porting and integrating in PLaSM both the visualization techniques of Ganith and the A-spline/A-patches techniques described in this thesis including the boolean operators. This would result in a sophisticated environment and programming language to manipulate symbolically and numerically both parametric and algebraic geometry.

This thesis does not include other works of the author during his Ph.D. course such as: optimization of the PLaSM[Pao03, Pa] geometric functional language for the version 5.1 of the interpreter; study and research in the field of implicit parallelism and application to the geometric mapping in the PLaSM language[Por07a]; application to the field of VR visualization and modeling of critical infrastructures[ABG+07].

Chapter 2

Piecewise Algebraic Curve Segments (A-Splines)

The possible applications of A-splines range from the traditional interactive design of curves in computer graphics, to topologically correct piecewise approximation of higher degree algebraic curves, and most importantly, in the field of biological modeling, to the fitting curves for image reconstruction. A-splines are the first step in the study of algebraic elements as they are the simplest kind of algebraic elements possible. Many concepts are more easily introduced and explained with a specific bi-dimentional example in mind. Their defining equation is in the form of $f(x,y) = 0$ thus one may easily visualize the graph $[x, y, w = f(x,y)]$ of the the function $f(x,y)$ and the intersection with the $w = 0$ plane. It is easier to visualize problems linked to non-singularity, single-sheetness, continuity and interpolation. For example in figure (2.1) shows the graph of a function in B.B. form over a triangle and its controlling coefficients. This chapter constitute a reasoned synthesis of some current articles on A-splines [BX99a, XBC00a, XBC00b, XBC00c].

2.1 Notation for Scaffold and Algebraic Equation

A-splines are defined in a compact polyhedron in particular trilaterals (triangles) and quadrilaterals. Let $\mathbf{p_i} = (x_i, y_i) \in \mathbb{R}^2,\ i \in [1..k]$, then $[\mathbf{p_1} \dots \mathbf{p_n}]$ denotes he *closed convex hull*. That is, $[\mathbf{p_1} \dots \mathbf{p_n}] = \{\mathbf{p} = \sum_{i=0}^{k} \alpha_i \mathbf{p_i},\ 0 \le \alpha_i \le 1,\ \sum_{i=0}^{k} \alpha_i = 1\}$. If $k = 3$ and $\mathbf{p_1}$, $\mathbf{p_2}$ and $\mathbf{p_3}$ are affine independent, then $[\mathbf{p_1}\mathbf{p_2}\mathbf{p_3}]$ is a triangle and $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \alpha_3)^T$ are known as barycentric coordinates which relate to the Cartesian coordinates $(x,y)^T$ by

$$\mathbf{p} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} = \begin{pmatrix} \mathbf{p_1} & \mathbf{p_2} & \mathbf{p_3} \\ 1 & 1 & 1 \end{pmatrix} \boldsymbol{\alpha} \tag{2.1}$$

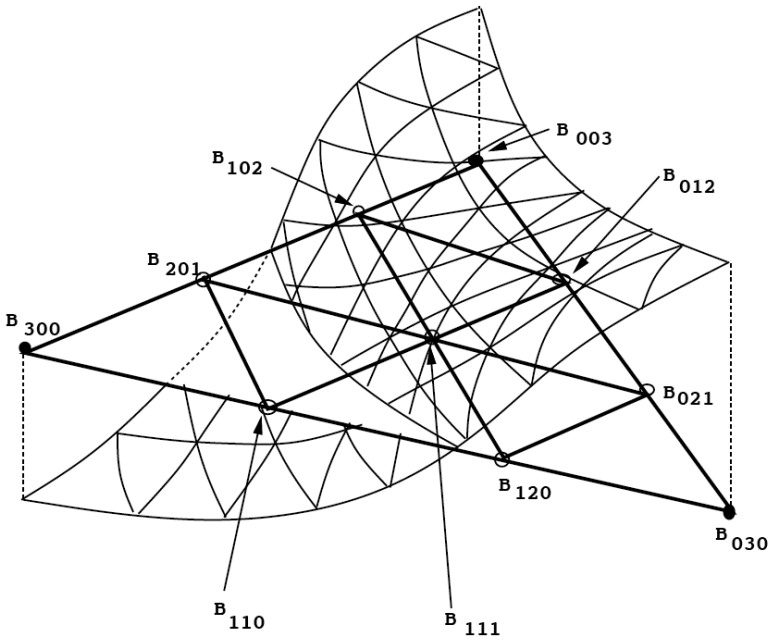


Figure 2.1: Graph of a function in B.B. form over a triangle [Baj].

with $|\boldsymbol{\alpha}| = \alpha_1 + \alpha_2 + \alpha_3 = 1$. Using this constraint one may write the coordinate transformation equations (2.1) in three equivalent ways using $\alpha_1 = 1 - \alpha_2 + \alpha_3$, $\alpha_1 = 1 - \alpha_2 + \alpha_3$, and as $\alpha_3 = 1 - \alpha_1 - \alpha_2$ (i.e. the last case):

$$\begin{aligned} \mathbf{p} &= \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_1 - x_3 & x_2 - x_3 & x_3 \\ y_1 - y_3 & y_2 - y_3 & y_3 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{p_1} - \mathbf{p_3} & \mathbf{p_2} - \mathbf{p_3} & \mathbf{p_3} \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ 1 \end{pmatrix} \end{aligned}$$

Relation (2.1) may be easily inverted as:

$$\boldsymbol{\alpha} = (\alpha_1 \alpha_2 \alpha_3)^T = \frac{\left(\begin{vmatrix} \mathbf{p} & \mathbf{p_2} & \mathbf{p_3} \\ 1 & 1 & 1 \end{vmatrix} \begin{vmatrix} \mathbf{p_1} & \mathbf{p} & \mathbf{p_3} \\ 1 & 1 & 1 \end{vmatrix} \begin{vmatrix} \mathbf{p_1} & \mathbf{p_2} & \mathbf{p} \\ 1 & 1 & 1 \end{vmatrix} \right)^T}{\begin{vmatrix} \mathbf{p_1} & \mathbf{p_2} & \mathbf{p_3} \\ 1 & 1 & 1 \end{vmatrix}} \tag{2.2}$$

or equivalently as

$$\begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} = \frac{1}{\begin{vmatrix} \mathbf{p_1} & \mathbf{p_2} & \mathbf{p_3} \\ 1 & 1 & 1 \end{vmatrix}} \begin{pmatrix} y_2 - y_3 & x_3 - x_2 \\ y_3 - y_1 & x_1 - x_3 \end{pmatrix} \begin{pmatrix} x - x_3 \\ y - y_3 \end{pmatrix}$$

On a triangle and using barycentric coordinates an algebraic curve may be defined by the zero contour of a polynomial in Berstein Bézier (B.B.) form.

$$F_n(\boldsymbol{\alpha}) = \sum_{i+j+k=n} \beta_{ijk} B^n_{ijk}(\boldsymbol{\alpha}), \;\; with\; B^n_{ijk}(\boldsymbol{\alpha}) = \frac{n!}{i!j!k!} \alpha_1^i \alpha_2^j \alpha_3^k \tag{2.3}$$

The points where the influence of the β_{ijk} coefficents is maximum may be plotted on the triangle, see figure (2.2 left) for a cubic.

If $k = 4$ and any three of the $\mathbf{p_i}$, $(i = 1, \ldots, 4)$ are affine independent then $[\mathbf{p_1 p_2 p_3 p_4}]$ is a quadrilateral. One may map the unit square $[0,1] \times [0,1]$ in the uv-plane to the quadrilateral $[\mathbf{p_1 p_2 p_3 p_4}]$ in global coordinates by

$$\begin{aligned} \mathbf{p} &= \mathbf{p_1}(1-u)(1-v) + \mathbf{p_2}(1-u)v + \mathbf{p_3}u(1-v) + \mathbf{p_4}uv \\ &= (\mathbf{p_1} + \mathbf{p_4} - \mathbf{p_2} - \mathbf{p_3})uv + (\mathbf{p_3} - \mathbf{p_1})u + (\mathbf{p_2} - \mathbf{p_1})v + \mathbf{p_1} \end{aligned} \tag{2.4}$$

If $\mathbf{p_1} + \mathbf{p_4} = \mathbf{p_2} + \mathbf{p_4}$, i.e. $[\mathbf{p_1 p_2 p_3 p_4}]$ is a parallelogram, then (2.4) is linear and inversion is easy, otherwise the inverse involves solving a quadratic equation. It is common to specify the quadrilateral in the skew case by two points ($\mathbf{p_1}$ and $\mathbf{p_2}$) and two normals ($\mathbf{n_1}$ and $\mathbf{n_2}$) because one wants to interpolate these points with this normal. In this case it is called prism and one writes:

$$\mathbf{p} = \mathbf{p_1}(v)(1-u) + \mathbf{p_2}(v)u, \;\; with\; \mathbf{p_i}(v) = \mathbf{p_i} + \mathbf{n_i}v$$

On a quadrilateral, the algebraic curve is defined by the zero contour of (see figure 2.2 right) of a polynomial in B.B. form in tensor form of degree (n,m).

$$G_{mn}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \beta_{ij} B^m_i(u) B^n_j(v), \;\; with\; B^n_i(s) = \frac{n!}{i!(n-i)!} s^i (1-s)^{n-i} \tag{2.5}$$

2.2 Sufficient Conditions for Regularity

The primary drawback for the widespread use of the implicit algebraic curves is that the real curve may have singularities (e.g. cuspidal cubic) and may be disconnected (hyperbola) in a given region of the curve. In most application one surely wants the curve segment to be connected in all the scaffold and not singular (e.g. $\nabla f(\mathbf{p}) \neq \mathbf{0}$) in

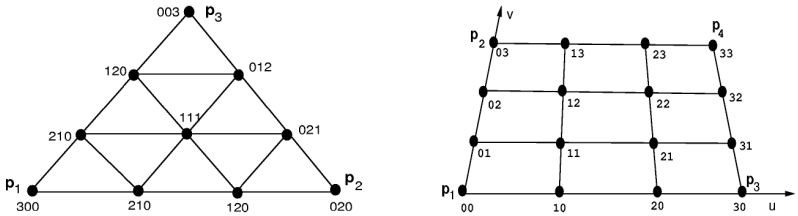
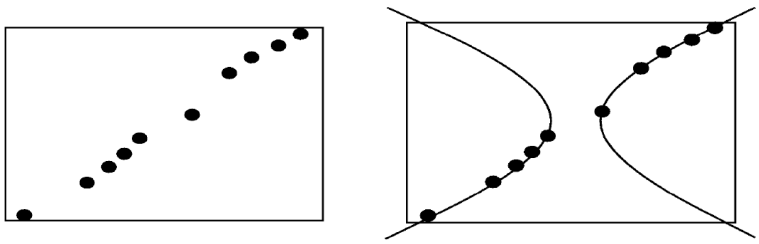


Figure 2.2: B.B. form coefficients for: (left) cubic over a triangle (right) bicubic over a quadrilateral. [XBC00a]



(a) (b)

Figure 2.3: Curve fitting: (a) points; (b) curve. [XBC00a]

the interior. For example, if we fit a cluster of points as shown in figure (2.3a) with a quadratic, one will have the result as shown in figure (2.3b), if no additional conditions are imposed on the curve.

In attempts to overcome these difficulties, Sedeberg in [SAG85] set conditions on the coefficients of the BB-form of an implicitly defined bivariate polynomial on a triangle in such a way that if the coefficients on the lines that are parallel to one side, say L, of the triangle all increase (or decrease) monotonically in the same direction, then any line parallel to L will intersect the algebraic curve segment at most once (see Fig. 2.4 left). In [SZZ88], Sederberg, Zhao and Zundel give another similar set of conditions which guarantees the single-sheeted property of their PAC (piecewise algebraic curves) by requiring that the Bézier coefficients $\beta_{i0} \geq 0$, that $\beta_{0i}, \beta_{m-1,i} \leq 0$, and that the directional derivative of PAC with respect to any direction $s = \alpha u$ be nonzero within the triangle domain. Papers of Paluszny and Patterson [PP92], [PP93] construct G^1 and G^2 continuous cubic algebraic splines by using the cubic $F(\alpha_1, \alpha_2, \alpha_3) = \beta_{201}\alpha_1^2\alpha_3 + \beta_{102}\alpha_1\alpha_3^2 - \beta_{120}\alpha_1\alpha_2^2 - \beta_{021}\alpha_2^2\alpha_3 + \beta_{111}\alpha_1\alpha_2\alpha_3$ with $\beta_{201} > 0$, $\beta_{102} > 0$, $\beta_{021} > 0$, and $(\alpha_1, \alpha_2, \alpha_3)$ being barycentric coordinates (see Fig. 2.4 right). All the above characterizations dealing with B.B. triangles have been generalized by Bajaj and

Xu ([BX99a] and chapter 6) in which the coefficients of the B.B. form have a one-time sign change. For the B.B. form on the quadrilateral, a characterization for the single-sheeted purpose is given by Patrikalakis and Kriezis [PK89] and is similar to Sederberg's [SZZ88]. In particular, if the coefficients increase or decrease monotonically in the x or y direction, then any line that is parallel to the x or y axis will intersect the curve at most once. Bajaj and Xu [XBC00a][XBC00b][XBC00c] themselves extend and unify the approach for both the triangle and the quadrilateral with the introduction of discriminating families. In the later approach the polynomial may be expressed by a rational equation obtaining a higher degree of continuity with less coefficients then a degree elevation. Moreover in [BX99a], chapter 6 and [XBC00a][XBC00b][XBC00c], the *single-sheeted property* is obtained. Using the single sheeted property one can easily evaluate the algebraic curve at a point and thus having a dual implicit and explicit representation.

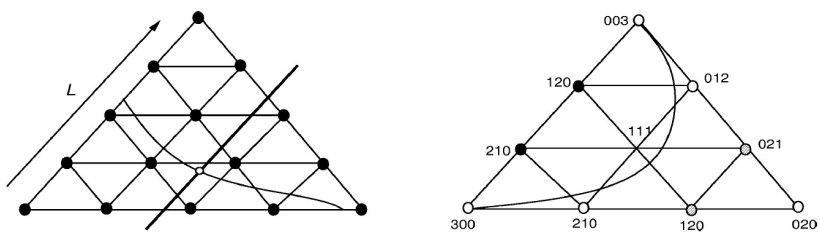


Figure 2.4: Left: the coefficients on the lines, that parallel to L, increase. Right: Cubic coefficients, the real dots are positive, shaded are negative, empty are zero. [XBC00a]

Consider the classical one variable C^1 function (it is of course defining a smooth curve) $y = g(x),\ x \in [a,b]$. The smootheness of the curve $f(x,y) = y - g(x) = 0$ can be tested by considering if every straight line $x = \alpha,\ \alpha \in [a,b]$, intersects the curve only once. The essential point behind this observation is that if each line in the set $\{x = \alpha :\ \alpha \in [a,b]\}$ intersects the curve only once, then the curve is regular. That is, the family of these lines can be used to judge the regularity of a curve. In [BX99a] and chapter 6, the lines

$$\left\{\alpha(t) = (1-t)(\beta, 1-\beta, 0)^T + t(0,0,1)^T, t \in [0,1] : \beta \in (0,1)\right\}$$

are used. The conclusion obtained is that under certain conditions on the coefficients of a bivariate polynomial $F_n(\boldsymbol{\alpha})$, each line in this family will intersect the curve $F_n(\boldsymbol{\alpha}) = 0$ only once in the triangle and the curve is regular. The concept may be easily extended with the general definition of *discriminating family*.

Definition 2.2.1. (From [XBC00a]) For a given triangle or quadrilateral R, let R_1 and R_2 be two closed pieces of boundary of R with $R_1 \cap R_2 = 0$ (see Fig. 2.5). Let $D =$

$\{A_s(x,y) = \gamma(x,y) - s\delta(x,y) = 0 : s \in [0,1]\}$ be an algebraic curve family with s as a parameter and $\delta(x,y) > 0$ on $R \setminus \{R_1, R_2\}$ such that:

1. Each curve in D passes through R_1 and R_2;
2. Each curve in D is regular in the interior of R;
3. For all $\mathbf{p} \in R \setminus \{R_1, R_2\}$, there exists one and only one $s \in [0,1]$ such that $A_s(\mathbf{p}) = 0$.

Then we say D is a *discriminating family* on R, denoted by $D(R, R_1, R_2)$.

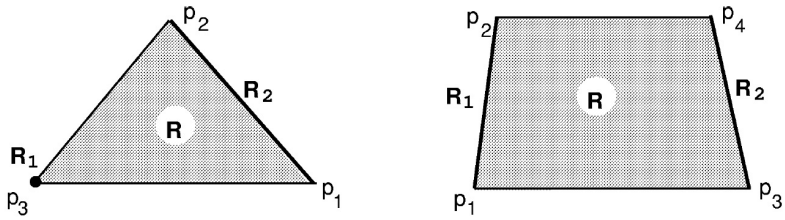


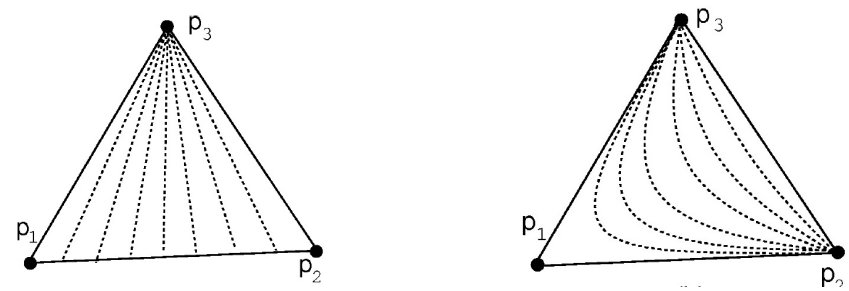
Figure 2.5: Closed pieces R_1 and R_2 of the boundaries of a triangle and a quadrilateral [XBC00a].

Intuitively a discriminating family is a family of non intersecting curves from R_1 to R_2 covering R. In figure (2.6) and (2.7) the following examples of discriminating families are shown:

$$D_1([\mathbf{p_1p_2p_3}], \mathbf{p_3}, [\mathbf{p_1p_2}]) = \{\alpha_2 - s(\alpha_1 + \alpha_2) = 0; s \in [0,1]\}$$
$$D_2([\mathbf{p_1p_2p_3}], \mathbf{p_2}, \mathbf{p_3}) = \left\{(1-s)\alpha_2\alpha_3 - s\alpha_1^2 = 0; s \in [0,1]\right\}$$
$$D_3([\mathbf{p_1p_2p_3p_4}], [\mathbf{p_1p_2}], [\mathbf{p_3p_4}]) = \{s - v; s \in [0,1]\}$$
$$D_4([\mathbf{p_1p_2p_3p_4}], \mathbf{p_1}, \mathbf{p_4}) = \{(1-s)u(1-v) - s(1-u)v = 0; s \in [0,1]\}$$

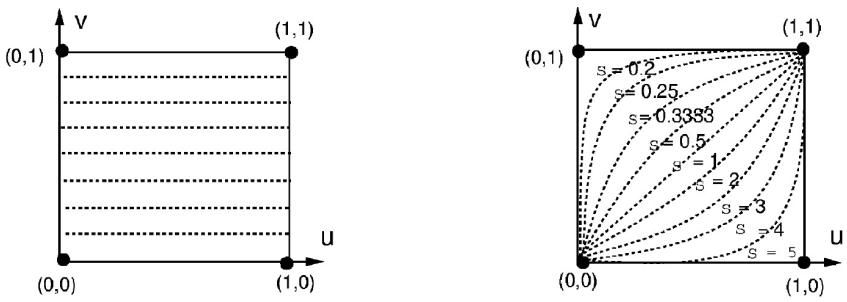
Definition 2.2.2. (from [XBC00a]) For a given discriminating family $D(R, R_1, R_2)$, let $f(x,y)$ be a bivariate polynomial (or a C_1 continuous function on $R \setminus \{R_1, R_2\}$). If the curve $f(x,y) = 0$ intersects with each curve in $D(R, R_1, R_2)$ exactly once in the interior of R, we say the curve $f = 0$ is regular with respect to $D(R, R_1, R_2)$ (concisely stated as being $D(R, R_1, R_2)$-regular).

The next question raised is for given an equation $F = 0$ on a triangle or a quadrilateral and a given specific discriminating family D, which condition it imposes on the curve so that it may be judged regular? In [XBC00a] the details of these kind of proofs for polynomials in B.B. form for the example discriminating family are detailed. In chapter 6 the case for line in a triangle (e.g. D_1) and F polynomial of arbitrary degree in B.B. form will be worked out.



(a) (b)

Figure 2.6: Discriminating families: (a) linear D_1; (b) quadratic D_2. [XBC00a]



(a) (b)

Figure 2.7: Discriminating families: (a) linear D_3 ; (b) hyperbolic D_4. [XBC00a]

2.3 Curve Control and Continuity

Once the conditions for regularity has been defined, the next logical step is to assign the remaining degrees of freedom of the coefficients in order to control the curve itself.

Interpolation/Approximation Control the curve by supplying data of geometric meaning. For example, given a set of points $\{\mathbf{p_i}\}$ one wants to build the piecewise algebraic curve segments in order to interpolate (e.g. $f_i(\mathbf{p_i}) = 0$) or approximate (e.g. $\exists \mathbf{p} : f_i(\mathbf{p}) = 0 \wedge \mathbf{p} - \mathbf{p_i} \leq \varepsilon$) said points. One wants also to control first derivative data ∇f, this is done in many ways for example by: (1) such as assigning the normals $\{\mathbf{n_i}\}$ at the point explicitly (e.g. $\nabla f(\mathbf{p_i}) = \mathbf{n_i}$), (2) supplying a tangent $\frac{d\mathbf{g}}{dt}|_{\bar{t}} = (t_x, t_y)^T$ of a parametric curve $\mathbf{g}(t)$ at the point $\mathbf{p_i}$ (e.g. $\mathbf{g}(\bar{t}) = \mathbf{p_i}$) and (3) require the curve to be tangent to a segment (e.g. tangent to $\mathbf{q_i} - \mathbf{p} : f_i(\mathbf{p}) = 0$.

Higher order data are rarely given explicitly as the hessian matrixes at the points $\{\mathscr{H}(f)(\mathbf{p_i})\}$

$$\mathscr{H}(f) = \begin{pmatrix} \frac{\partial f}{\partial^2 x} & \frac{\partial f}{\partial x \partial y} \\ \frac{\partial f}{\partial y \partial x} & \frac{\partial f}{\partial^2 y} \end{pmatrix}$$

but by either supplying the second derivative $\frac{d^2\mathbf{g}}{dt^2}|_{\bar{t}} = (c_x, c_y)^T$ of a parametric curve $\mathbf{g}(t)$ at the point $\mathbf{g}(\bar{t}) = \mathbf{p_i}$ or other curvature data (e.g. the radii of the osculating spheroids).

Continuity One wants that two consecutive segments, say f_i and f_{i+1} to meet at a common border point $\mathbf{p_{i+1}}$ (e.g. $f_i(\mathbf{p_{i+1}}) = f_{i+1}(\mathbf{p_{i+1}}) = 0$. One may require algebraic C^1 continuity $\nabla f_i(\mathbf{p_{i+1}}) = \nabla f_{i+1}(\mathbf{p_{i+1}})$. C^2 or higher C^k algebraic continuity is rare, one instead requires *geometric continuity* G^k, also known as *rescaling continuity*. Two algebraic curves $f(x,y) = 0$ and $g(x,y) = 0$ meet with G^k-continuity on a point $\mathbf{p}$ if and only if there exists functions $\alpha(x,y)$ and $\beta(x,y)$ such that all derivatives upto order k of $\alpha f - \beta g$ equals to zero in $\mathbf{p}$.

Shape Control Another set of constrains or control data may be defined for finer shape control. A common shape control constraint is that each segment is convex, that is, does not contain inflection points. Moreover one often requires the curve to approximate a lower degree curve (e.g. quadratic precision for a cubic curve). Shape control data are usually coefficients that control the curve using weighted least squares approximation from additional points and normals, generated locally for each scaffold.

Geometric control data, similarly to the parametric curve splines, are given in terms of a polygonal chain whose points the spline must interpolate or approximate and whose some of edges describe the tangent of the curve. A polygonal chain is an ordered sequence of polygonal line segments, where any three adjacent points are not collinear. Several geometry processing tasks generate polygonal chains for shape representation in 2D. Examples include shape or font design, fitting from 'noisy' data, image contouring, snakes [KWT88] and level set methods [Set96]. Here some of them that have some attached error or uncertainty are mentioned [XBC00c].

Noisy vertex data The vertex data (position) come from a multi-sampling process with possible error. The error bound ε is known in advance. Fig. 2.8 shows such a case. The white circles are the repeatedly sampled points, the black dots are approximations of the sampled points. The approximation of the point can be computed as the center of gravity or center of bounding circular fits. The polygonal chain is obtained by connecting these black dots. The spline curve to be constructed interpolates the vertices of the polygonal chain. Hence the error around each vertex is bounded by ε .

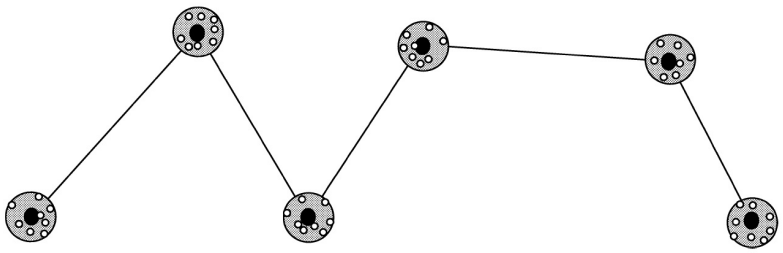


Figure 2.8: Polygonal chain extracted from over-sampled points [XBC00c].

Noisy curve data Suppose a curve is sampled within some ε error band around the curve. The sampled point sequence $\{\mathbf{v_i}\}$ could be dense. To produce a polygonal chain to these points, we use a 'strip pasting' technique. Choose the strip width to be no less than 2ε . Then use the minimal number of strips to cover the sample points (see Fig. 2.9). The vertices of the polygonal chain are the intersection points of two mid-lines of adjacent strips. A computational method for obtaining the minimal number strips can be found in [BNK93]. A greedy method to obtain a minimal "strip pasting" uses an adaptive piecewise linear least square fitting, starting from one end of the data.

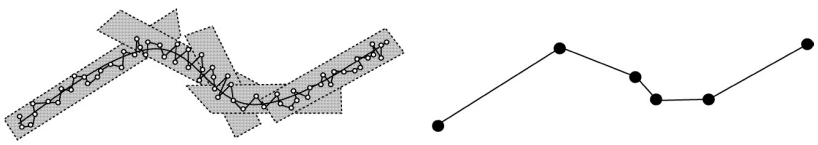


Figure 2.9: Polygonal chain from noisy curve data and using adaptive "strip pasting": The white circles are original sampled points with error, and the black dots are the vertices of an extracted polygonal chain [XBC00c].

Contour from an image A 2D image can be treated as a piecewise C^0 bilinear function interpolating the intensity values at each pixel. A linear isocontour of the function is a polygonal chain. Of course, such a polygonal chain may be quite dense, hence a simplification step is often used to obtain coarser or multiresolution representations. Fig. 2.10 shows an image and an isocontour with two simplified polygonal chains. The simplification method is established based on geometric error (Euclidean distance) control, that is, a point is removed if the distance of the point to the line, that interpolates its two neighbor points, is less than a given ε. Hence all original points are within an ε-neighborhood of the simplified polygonal chain.

The two simplified polygonal chains in Fig. 2.10 are obtained by taking ε to be 0.05 and 0.25, respectively.

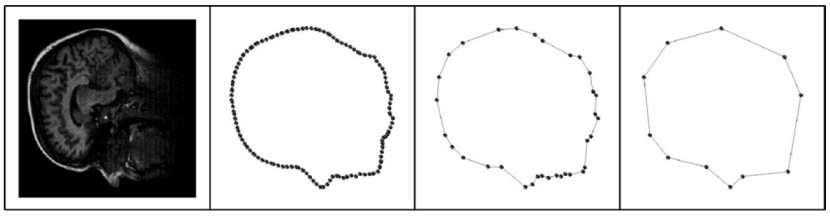
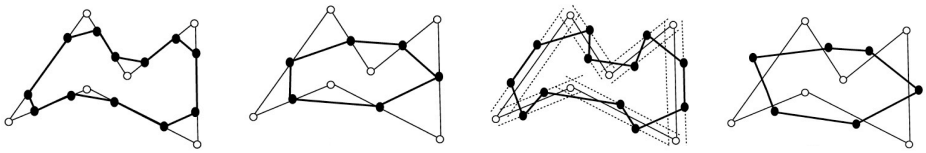


Figure 2.10: From an image to polygonal chains [XBC00c].

Polygonal chain One polygonal chain can be produced from another polygonal chain by subdivision or corner cutting. Fig. 2.11 shows four polygonal chains obtained by corner cutting with cutting ratios 0.25 and 0.5, respectively and subdivision. When the cutting ratio is 0.5, then each edge of the new polygonal chain is convex if the tangents at the vertices are taken to be the original edges. Smooth approximations of these polygonal chains are suitable for triangular A-splines ([BX99a] and chapter 6). The vertices of the polygonal chain (c) are located away from the original edge by a specified distance δ. We call this an "offset corner cutting" scheme. For the same purpose, an interpolatory subdivision scheme (see, e.g. [War95]) could also be employed (see Fig. 2.11(d)) such as the 4-point rule with mask (-1/16, 9/16, 9/16, -1/16) (see [DGL87]).



(a) (b) (c) (d)

Figure 2.11: Polygonal chains (of black vertices) produced from polygonal chains (of white vertices) [XBC00c]: (a) corner cut with cutting ratio 0.25; (b) corner cut with cutting ratio 0.5 yielding a convex polygon; (c) offset corner cut with cutting ratio 0.25; (d) interpolatory subdivision.

Control data are given in a global coordinates system. To build the piecewise algebraic curves spline one may use both the scaffold construction and the coefficients of the curve

in the local coordinate system. In the former case the scaffold is built according the control data and the curve's control coefficients are assigned for a certain behavior in the scaffold. In the later case the control data is converted in the local coordinate system and the coefficients are then calculated to interpolate/approximate this "localized" control data.

In the case of triangular splines ([BX99a] and chapter 6) the control polygonal chain is a sequence of consecutive polygon segments denoted by $\{\widehat{\mathbf{p_i v_i p_{i+1}}}\}_{i=0}^{m}$. The $\mathbf{p_i}$, $\mathbf{v_i}$ and $\mathbf{p_{i+1}}$ must be three affine independent points in the xy-plane. One consider the two line segments $[\mathbf{p_i v_i}]$ and $[\mathbf{v_i p_{i+1}}]$ as a segment of a polygon, denoted by $\mathbf{p_i}, \widehat{\mathbf{v_i}, \mathbf{p_{i+1}}}$. We shall consider $\mathbf{v_i}$ as a controller and $\mathbf{p_i}$ and $\mathbf{p_{i+1}}$ as interpolation points. A polygon $\{\widehat{\mathbf{p_i v_i p_{i+1}}}\}_{i=0}^{m}$ is said to be type G^1 (see figure 2.12) if

$$(\mathbf{v_1} - \mathbf{p_{i+1}}) = \alpha_i(\mathbf{v_{i+1}} - \mathbf{p_{i+1}})$$

One way to construct the polygon from sampled points and normals is to set $\mathbf{p_i}$ and $\mathbf{p_{i+1}}$ as interpolation points and $\mathbf{v_i}$ as the intersection point between the (extended) tangents $\mathbf{t_i}$ and $\mathbf{t_{i+1}}$ at $\mathbf{p_i}$ and $\mathbf{p_{i+1}}$. As it'll be shown, this guarantees G^1 continuity with the imposed normals. In free form A-spline drawing a trivial choice is to ask the user to insert a vertex per edge in the polygon. In figure 2.13 shows the polygon $\mathbf{p_0}, \widehat{\mathbf{v_0}, \mathbf{p_1}}$, the built scaffold, and the B.B. coefficients for a polynomial $f(\boldsymbol{\alpha})$ of degree n.

The coefficients of the curve are given in order to interpolate the two points $\mathbf{p_1}$ and $\mathbf{p_2}$ at the base of the triangle and here tangent to the sides $[\mathbf{p_1 p_3}]$ and $[\mathbf{p_3 p_2}]$ respectively. By building the triangular scaffold on these points, these conditions translate in assigning the b_{n00} and b_{0n0} to zero for the interpolation of $\mathbf{p_1}$ and $\mathbf{p_2}$ respectively. Tangency to $[\mathbf{p_1 p_3}]$ and $[\mathbf{p_3 p_2}]$ is obtained by requiring $b_{(n-1)01} = 0$ and $b_{0(n-1)1} = 0$. Furthermore assigning $b_{(n-i)0i} = 0$ (and $b_{0(n-i)i} = 0$) for $i \in [0,k]$ will force the curve to be tangent in $\mathbf{p_1}$ to $[\mathbf{p_1 p_3}]$ (and in $\mathbf{p_2}$ to $[\mathbf{p_3 p_2}]$) with multiplicity $k-1$ (e.g. k=2 the point is a flex point). Higher G^k continuity and control is given in terms of parameters expressed in the local coordinate system.

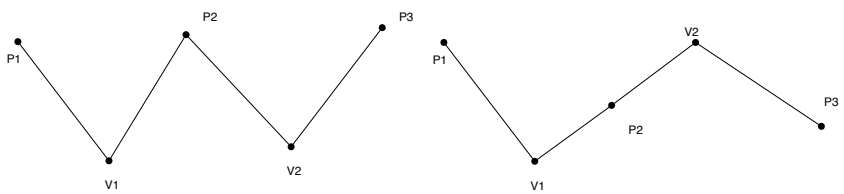


Figure 2.12: A (left) C^0 and (right) C^1 polygon. [BX99a]

In [XBC00b] a different approach is used in order to unify under a common framework the treatment of regularity and curve control in both trilateral and quadrilateral A-splines. The framework involves the following steps (see Fig. 2.14):

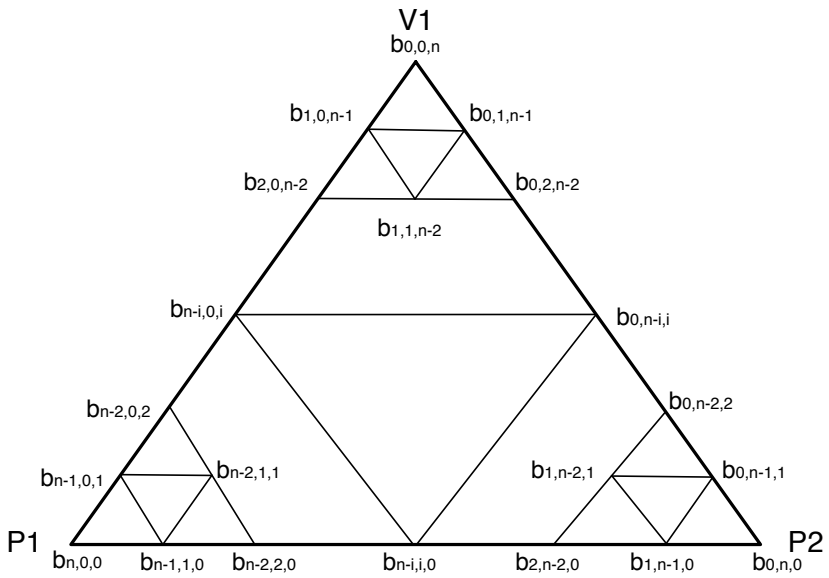


Figure 2.13: Bezier coefficients of the curve over $\widehat{\mathbf{p_1}, \mathbf{v_1}, \mathbf{p_2}}$.

1. Build a transform M that converts the problem on triangle or quadrilateral in the xy-plane into a classical rational interpolation or approximation problem on the strip $[0,1] \times [-\infty, \infty]$ in the st-plane.
2. Solve the interpolation/approximation problem in the common st-plane and assign the controlling coefficients.
3. Transform the solution in the st-plane back to the xy-plane by the inverse transform M^{-1}, to get a solution of the original problem.

There are plenty of methods that could be used for solving the interpolation and approximation problem, the main task in these steps is to build the transform M.

Using the definition of *discriminating family* (2.2.1) the "orthogonal" concept of *traversal family* is introduced.

Definition 2.3.1. (from [XBC00b]) Let $D(R, R_1, R_2)$ be a given discriminating family, and

$$T(R, R_1', R_2') = \{B_t(x,y) = \mu(x,y) - t\nu(x,y) = 0,\ t \in (-\infty, \infty)\}$$

be an algebraic curve family with t being a linear parameter, $\mu(x,y)$ and $\nu(x,y)$ are bivariate polynomials and $\nu(x,y) > 0$ on $R \setminus \{R_1, R_2\}$ and R_1' and R_2' being two open (no end points) pieces of the boundary ∂R of R (see Fig. 2.15). If

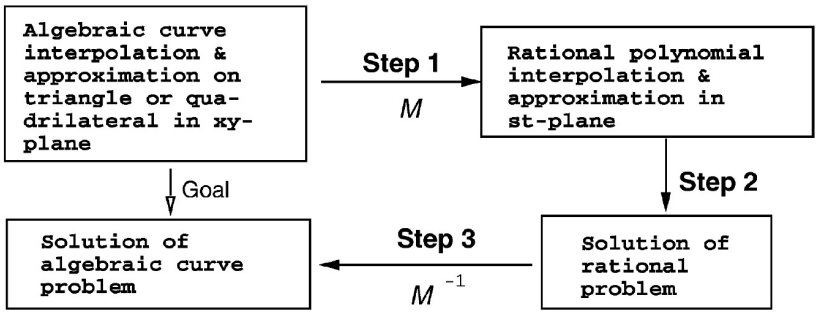


Figure 2.14: The pipeline of solving the problem of interpolation and approximation by D-regular algebraic curves [XBC00b].

1. $\partial R \setminus (R_1 \cup R_2) = R'_1 \cup R'_2$ and $R'_1 \cap R'_2 = 0$;
2. Each curve in T passes through R'_1 and R'_2;
3. Each curve in T is $D(R, R_1, R_2)$-regular;
4. For $\forall \mathbf{p} \in R \setminus \{R_1, R_2\}$, there exists one and only one $t \in (-\infty, \infty)$ such that $B_t(\mathbf{p}) = 0$

Then $T(R, R'_1, R'_2)$ is a traversal family of $D(R, R_1, R_2)$.

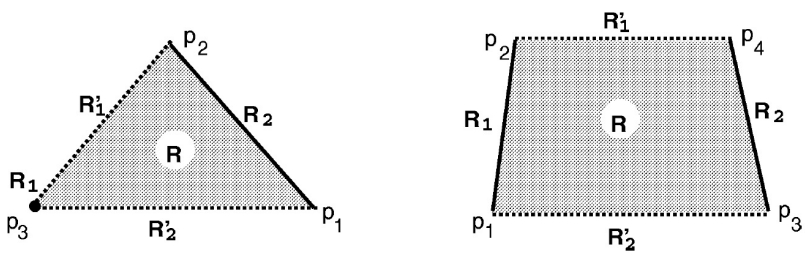


Figure 2.15: R_1 , R_2 , R'_1, R'_2 for a triangle and a quadrilateral [XBC00b].

Intuitively a traversal family is a family of curves that are "orthogonal" to the ones of a discriminating family. To further comprehend this concept in figures (2.16) and (2.17) show the traversal families T_1, T_2, T_3, and T_4 for the previously introduced discriminating

families D_1, D_2, D_3, and D_4 (see Fig. (2.6) and (2.7)

$$
\begin{aligned}
&T_1\left([\mathbf{p_1p_2p_3}],(\mathbf{p_1p_3}),(\mathbf{p_2p_3})\right)\\
&=\left\{B_1^2(\alpha_3)t-B_2^2(\alpha_3)+B_0^3(\alpha_3)=0:\ t\in(-\infty,\infty)\right\}\\
&T_2\left([\mathbf{p_1p_2p_3}],(\mathbf{p_1p_2}),(\mathbf{p_1p_3})\right)\\
&=\left\{(\alpha_1^2+\alpha_2\alpha_3)t-\alpha_3^2+\alpha_2^2=0:\ t\in(-\infty,\infty)\right\}\\
&T_3\left([\mathbf{p_1p_2p_3p_4}],(\mathbf{p_1p_3}),(\mathbf{p_2p_4})\right)\\
&=\left\{B_1^2(u)t-B_2^2(u)+B_0^2(u)=0:\ t\in(-\infty,\infty)\right\}\\
&T_4\left([\mathbf{p_1p_2p_3p_4}],(\mathbf{p_1},\mathbf{p_2})\cup[\mathbf{p_2p_4}),(\mathbf{p_1},\mathbf{p_3}]\cup[\mathbf{p_3p_4})\right)\\
&=\{[u(1-v)+(1-u)v]t-(1-u-v)=0:\ t\in(-\infty,\infty)\}
\end{aligned}
$$

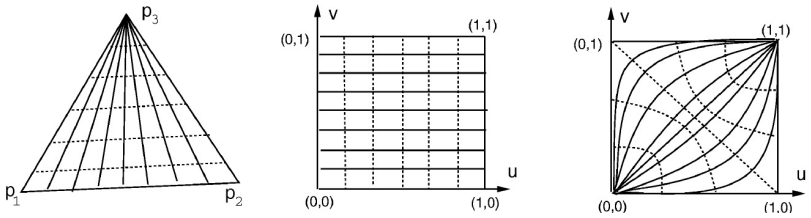


Figure 2.16: Discriminating families (real lines) and their transversal families (dotted lines) [XBC00b].

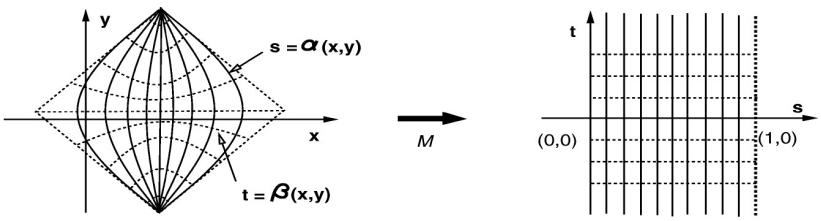


Figure 2.17: A discriminating family and its transversal family define (s,t)-coordinate system [XBC00b].

Given a discriminating family $D=\{A_s(x,y)=\gamma(x,y)-s\delta(x,y)=0:s\in[0,1]\}$ and its traversal family $T(R,R_1',R_2')=\{B_t(x,y)=\mu(x,y)-t\nu(x,y)=0,\ t\in(-\infty,\infty)\}$, one has in fact a map between $R\setminus\{R_1,R_2\}$ in the xy-plane and the strip $[0,1]\times[-\infty,\infty]$ in the st-plane (see Fig. 2.17). Since s and t are linear parameters in $A_s(x,y)=0$ and $B_t(x,y)=0$,

respectively, they can be written as:

$$M(D,T): \begin{cases} s &= \alpha(x,y) &:= \frac{\gamma(x,y)}{\delta(x,y)} \\ t &= \beta(x,y) &:= \frac{\mu(x,y)}{\nu(x,y)} \end{cases}$$

where α and β are well defined rational functions on $R \setminus \{R_1, R_2\}$. Often $M(D_i, T_i)$ is denoted as M_i. For completeness and as an example, the explicit forms of the transforms and their inverses for the pairs of families shown so far are given [XBC00b].

$$M_1: \begin{cases} s = \alpha_2/(\alpha_1 + \alpha_2), \\ t = [B_2^2(\alpha_3) - B_0^2(\alpha_3)]/B_1^2(\alpha_3), \end{cases}$$

$$M_1^{-1}: \begin{cases} \alpha_1(s,t) = (1-s)[1 - 1/(\sqrt{1+t^2} + 1 - t)], \\ \alpha_2(s,t) = s[1 - 1/(\sqrt{1+t^2} + 1 - t)], \\ \alpha_3(s,t) = 1/(\sqrt{1+t^2} + 1 - t). \end{cases}$$

$$M_3: \begin{cases} s = v, \\ t = [B_2^2(u) - B_0^2(u)]/B_1^2(u), \end{cases}$$

$$M_3^{-1}: \begin{cases} u(s,t) = 1/(\sqrt{1+t^2} + 1 - t), \\ v(s,t) = s. \end{cases}$$

$$M_4: \begin{cases} s = [u(1-v)]/[u(1-v) + (1-u)v], \\ t = (1-u-v)/[u(1-v) + (1-u)v], \end{cases}$$

$$M_4^{-1}: \begin{cases} u(s,t) = 2s/[t + 2s + \sqrt{t^2 + 4s(1-s)}], \\ v(s,t) = 2(1-s)/[t + 2(1-s) + \sqrt{t^2 + 4s(1-s)}], \end{cases}$$

For the actual solution of the curve control an interested reader may consult [XBC00b]. Hence in [XBC00c] parallelogram (Fig. 2.18) and rectangle (Fig. 2.19) G^1 and G^2 A-spline (Fig. 2.20) are introduced that interpolate a given polygon chain of points $\{\mathbf{p_i}\}$ and tangents $\{\mathbf{r_i}\}$.

Though never formalized in any article it is easy to adapt the sides of a prism A-patches [BX99b, BX01, ZXB07, BPP+08] (see Chapter 8) to prism A-splines. The advantage of this adaptation is that prism A-splines can be adapted for curves in 3D space or higher by simply using n-dimentional points and normals. Prism A-splines come in two forms: simple (single) and shell (double) A-splines:

Simple prism A-splines The control data is a sequence of points $\{\mathbf{v_i}\}$ and normals $\{\mathbf{n_i}\}$. As already mentioned, the scaffold is built for each spline segment $[\mathbf{v_i v_{i+1}}]$ by building the quadrilateral along the (extended) normals $\{\mathbf{n_i}\}$ and $\{\mathbf{n_{i+1}}\}$ by tensor product of two linear interpolations. The coordinate transformation is ex-

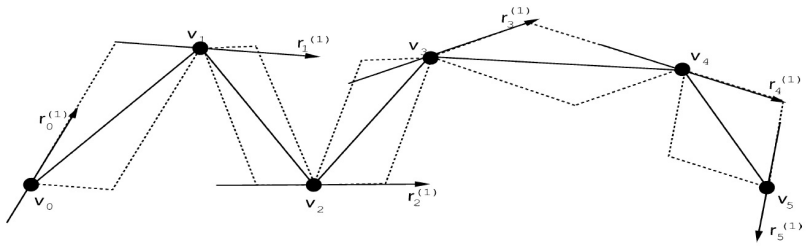


Figure 2.18: Parallelogram chain [XBC00c].

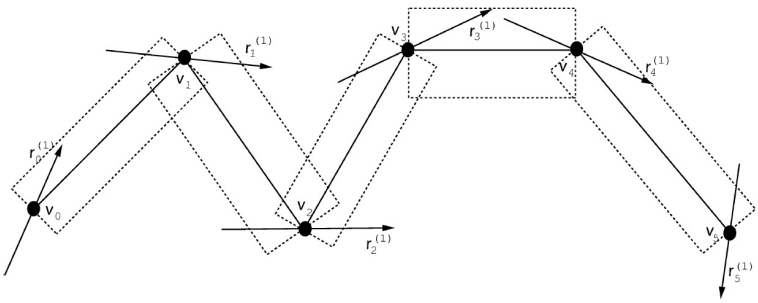


Figure 2.19: Rectangular chain. The width of the rectangle for edge $[\mathbf{v_{i-1}}\mathbf{v_i}]$ is $2\varepsilon_i$ [XBC00c].

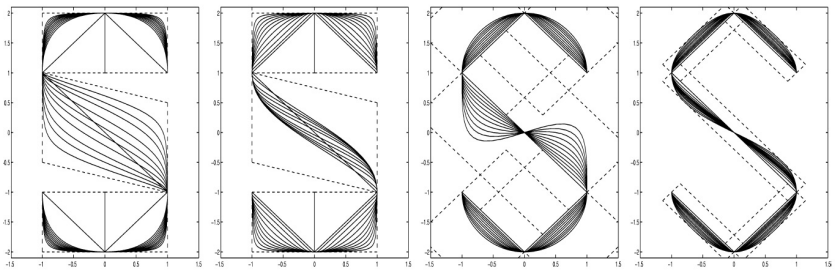
pressed as (see Fig. 2.21(a)):

$$\begin{aligned}
\mathbf{p} &= \mathbf{v_i}(\lambda)(1-\alpha_2)+\mathbf{v_{i+1}}(\lambda)\alpha_2 && \textit{with } \mathbf{v_i}(\lambda)=\mathbf{v_i}+\mathbf{n_i}\lambda \\
&= \mathbf{v_i}+(\mathbf{v_{i+1}}-\mathbf{v_i})\alpha_2+\lambda\left[\mathbf{n_i}+(\mathbf{n_{i+1}}-\mathbf{n_i})\alpha_2\right] \\
&= \begin{pmatrix} 1 & \lambda \end{pmatrix}\begin{pmatrix} \mathbf{v_i} & \mathbf{v_{i+1}} \\ \mathbf{n_i} & \mathbf{n_{i+1}} \end{pmatrix}\begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} && \textit{with } \alpha_1+\alpha_2=1
\end{aligned}$$

The algebraic curve is expressed similarly as a B.B. transfinite blending of polynomials $a_{ij}(\lambda)$ (in other words the B.B. coefficients are themselves polynomials)

$$\begin{aligned}
F_i(\alpha_1,\alpha_2,\lambda) &= \textstyle\sum_{i+j=n} a_{ij}(\lambda)B^n_{ij}(\alpha_1,\alpha_2) \\
F_i(\alpha_2,\lambda) &= \textstyle\sum_{i=0}^{n} a_i(\lambda)B^n_i(\alpha_2)
\end{aligned}$$

with $\alpha_1+\alpha_2=1$ as barycentric coordinates, $a_i(\lambda)=a_{i(n-i)}(\lambda)$ the polynomial B.B. coefficients, and $B^n_i(\alpha_2)=B^n_{i(n-i)}(1-\alpha_2,\alpha_2)$ the univariate B.B. basis polynomials. The single sheeted and regularity is imposed by declaring the curve to



(a) (b) (c) (d)

Figure 2.20: (a) G^1 a-spline families on parallelograms. (b) G^2 a-spline families on parallelograms. (c) G^1 a-spline families on rectangles with $\varepsilon = 1.0$. (d) G^1 a-spline families on rectangles with $\varepsilon = 0.2$ [XBC00c].

be the minimal $|\lambda|$ such that $F(\alpha_1, \alpha_2, \lambda) = 0$. In the cubic it is easy to prove that the curve will interpolate local point $(\alpha_1 = 1, \alpha_2 = 0, \lambda = 0)$ (i.e. global point $\mathbf{v_i}$) if $a_{30}(\lambda) = \lambda$ and local point $(\alpha_1 = 0, \alpha_2 = 1, \lambda = 0)$ (i.e. global point $\mathbf{v_{i+1}}$) if $a_{03}(\lambda) = \lambda$. The normal $\mathbf{n_i}$ is imposed in $\mathbf{p_i}$ by

$$a_{21}(\lambda) = \lambda + \frac{1}{3}\mathbf{n_i} \cdot (\mathbf{v_{i+1}} - \mathbf{v_i})$$

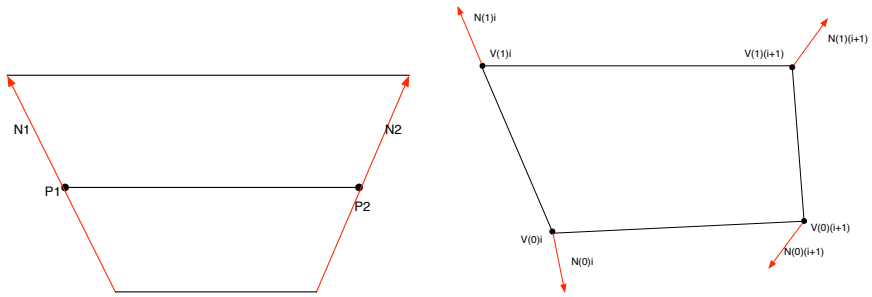
Symmetrically one sets $a_{12}(\lambda)$ for $\mathbf{n_{i+1}}$.

Shell prism A-splines The objective is to define two curves containing a strip in the plane (a "shell"). The control data is specified by two sequences of matched points $\{\mathbf{v_i^{(0)}}\}$ and $\{\mathbf{v_i^{(1)}}\}$ with attached normals $\{\mathbf{n_i^{(0)}}\}$ and $\{\mathbf{n_i^{(1)}}\}$. The scaffold is built using just the point data by building the quadrilateral $[\mathbf{v_i^{(0)}}\mathbf{v_{i+1}^{(0)}}\mathbf{v_{i+1}^{(1)}}\mathbf{v_i^{(1)}}]$ (see Fig. 2.21(b)):

$$\begin{aligned} \mathbf{p} &= \mathbf{v_i}(\lambda)(1-\alpha_2) + \mathbf{v_{i+1}}(\lambda)\alpha_2 \\ &= \begin{pmatrix} 1-\lambda & \lambda \end{pmatrix} \begin{pmatrix} \mathbf{v_i^{(0)}} & \mathbf{v_{i+1}^{(0)}} \\ \mathbf{v_i^{(1)}} & \mathbf{v_{i+1}^{(1)}} \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} \end{aligned}$$

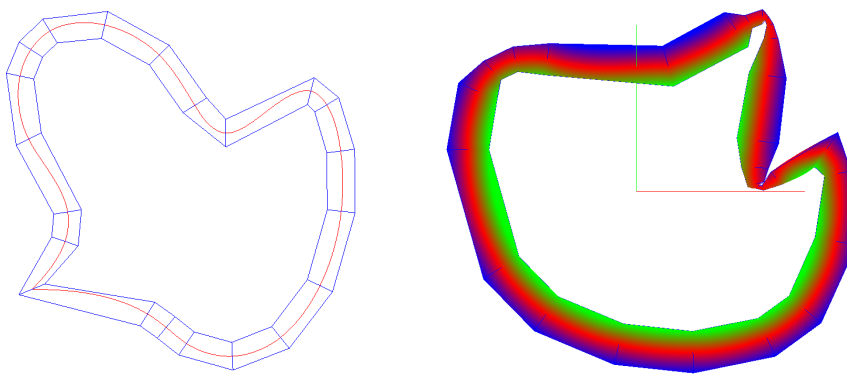
with $\mathbf{v_l}(\lambda) = \mathbf{v_l^{(0)}}(1-\lambda) + \mathbf{v_l^{(1)}}\lambda$ and $\alpha_1 + \alpha_2 = 1$. The two curves will be definined by the (± 1)-set of one function $F(\alpha_2, \lambda)$. The curve equation is built by transfinite blending of two Hermitte interpolations such that:

$$\begin{array}{llll} F(\mathbf{v_i^{(0)}}) = -1 & F(\mathbf{v_i^{(1)}}) = 1 & F(\mathbf{v_{i+1}^{(0)}}) = -1 & F(\mathbf{v_{i+1}^{(1)}}) = -1 \\ \nabla F(\mathbf{v_i^{(0)}}) = \mathbf{n_i^{(0)}} & \nabla F(\mathbf{v_i^{(1)}}) = \mathbf{n_i^{(1)}} & \nabla F(\mathbf{v_{i+1}^{(0)}}) = \mathbf{n_{i+1}^{(0)}} & \nabla F(\mathbf{v_{i+1}^{(1)}}) = \mathbf{n_{i+1}^{(1)}} \end{array}$$



(a) (b)

Figure 2.21: Prism scaffold: (a) simple (b) shell



(a) (b)

Figure 2.22: Prism A-spline: (a) A-spline and scaffold; (b) the function $F(\alpha_2, \lambda)$ as colored level-sets: red=0, green=-1, blue=1, and shaded in between

Chapter 3

Piecewise Algebraic Surface Patches (A-Patches)

Like A-splines, the possible applications of A-patches range from the traditional interactive design of surfaces in computer graphics, to topologically correct piecewise approximation of higher degree algebraic surfaces, and most importantly in the field of biological modeling to the fitting surfaces for image reconstruction. The defining equation of an A-patch is in the form $f(x,y,z) = 0$. Where f is a polynomial in any polynomial appropriate basis. To visualize the function graph $(x,y,z,f(x,y,z))$ one possible technique is to imagine a scalar field as a spectrum of shades in the 3D space and the c-set surface (the intersection of $w = c$, usually $c = 0$) as the iso-colored surfaces of the color representing c. This chapter constitute a reasoned synthesis of some current articles on A-patches [Baj, BCX95, BCX97, BX99b, XHB01, BX01, XBE02, ZXB07].

In one of the first definitions, A-patches [BI92b] were defined with the equation in global coordinates and provided, like in the abstract definition, three delimiting curves. In the *curvlinear-mesh* based scheme, Bajaj and Ihm [BI92b] construct low-degree implicit polynomial spline surfaces by interpolating a mesh of curves in space using the techniques of [Baj92, BI92a, BIW93]. They consider an arbitrary spatial triangulation T (i.e., T is a set of tetrahedra) consisting of vertices in $\mathbb{R}^3$ (or more generally a simplicial polyhedron P when the triangulation is closed) with, possibly, normal vectors at the vertex points. Their algorithm constructs a G^1 continuous mesh of real implicit polynomial surface patches over T or P . The scheme is local (each patch has independent free parameters) and there is no local splitting. The algorithm first converts the given triangulation or polyhedron into a curvilinear wireframe with at most cubic parametric curves which G^1 interpolates all the vertices. The curvilinear wireframe is then meshed to produce a single implicit surface patch of degree at most 7 for each triangular face of P. If the triangulation is convex then the degree is at most 5. Furthermore, the G^1 in-

terpolation scheme is local in that each triangular surface patch has independent degrees of freedom which may be used to provide local shape control. Extra free parameters may be adjusted and the shape of the patch controlled by using weighted least squares approximation from additional points and normals, generated locally for each triangular patch.

3.1 Notation for Scaffold and Algebraic equation

More recently A-patches have been defined as an equation $f = 0$ over a simple compact polyhedron (see figure 3.1). The local coordinates of the polyhedron are then mapped parametrically to the global coordinate system and thus the zero-set surface is mapped in space. The patch is thus delimited by the sides of the polyhedron in global coordinates. There are many elements over which we can define the patches, the simplest compact domains one considers are (see Fig 3.1 and 3.2):

Cube (tensor domain) (see Fig 3.1(a)) The local coordinates $(u,v,w)^T$ are defined over the interval $[0,1]$ $(u \in [0,1], v \in [0,1], w \in [0,1])$ (cube domain). This yield a tensor product Bernstein-Bezier coordinate system for trivariate polynomials. The mapping to global coordinate is defined as trilinear vector tensor form

$$\begin{aligned}
\mathbf{p} &= \textstyle\sum_{i=0}^{1}\sum_{j=0}^{1}\sum_{k=0}^{1} \mathbf{p_{ijk}} B_i^1(u) B_j^1(v) B_k^1(w) \\
&= \mathbf{p_{000}}(1-u)(1-v)(1-w) + \mathbf{p_{100}} u(1-v)(1-w) \\
&+ \mathbf{p_{010}}(1-u)v(1-w) + \mathbf{p_{110}} uv(1-w) \\
&+ \mathbf{p_{001}}(1-u)(1-v)w + \mathbf{p_{101}} u(1-v)w \\
&+ \mathbf{p_{011}}(1-u)vw + \mathbf{p_{111}} uvw
\end{aligned}$$

where $\mathbf{p_{ijk}}$ are eight points in 3D space. The function f is a polynomial over the cube $(u,v,w) \in [0,1] \times [0,1] \times [0,1]$ usually defined in B.B. tensor form

$$f(u,v,w) = \sum_{i=0}^{l}\sum_{j=0}^{m}\sum_{k=0}^{n} a_{ijk} B_i^l(u) B_j^m(v) B_k^n(w)$$

where a_{ijk} are scalar coefficients and $B_i^n(t) = \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i}$ are the univariate berstein basis polynomials (see Fig 3.2(a)).

Tetrahedron (see Fig 3.1(b)) The local coordinates $(\boldsymbol{\alpha} = \alpha_1, \alpha_2, \alpha_3, \alpha_4)^T$ are defined with the condition: $0 \le \alpha_i \le 1$ and $|\boldsymbol{\alpha}| = \sum_i^4 \alpha_i = 1$, with the last equation one of the α_i may always be eliminated as $\alpha_i = 1 - \left(\sum_{j \neq i} \alpha_j\right)$. This yields a barycentric coordinate system for trivariate polynomials. The mapping to global coordinate is

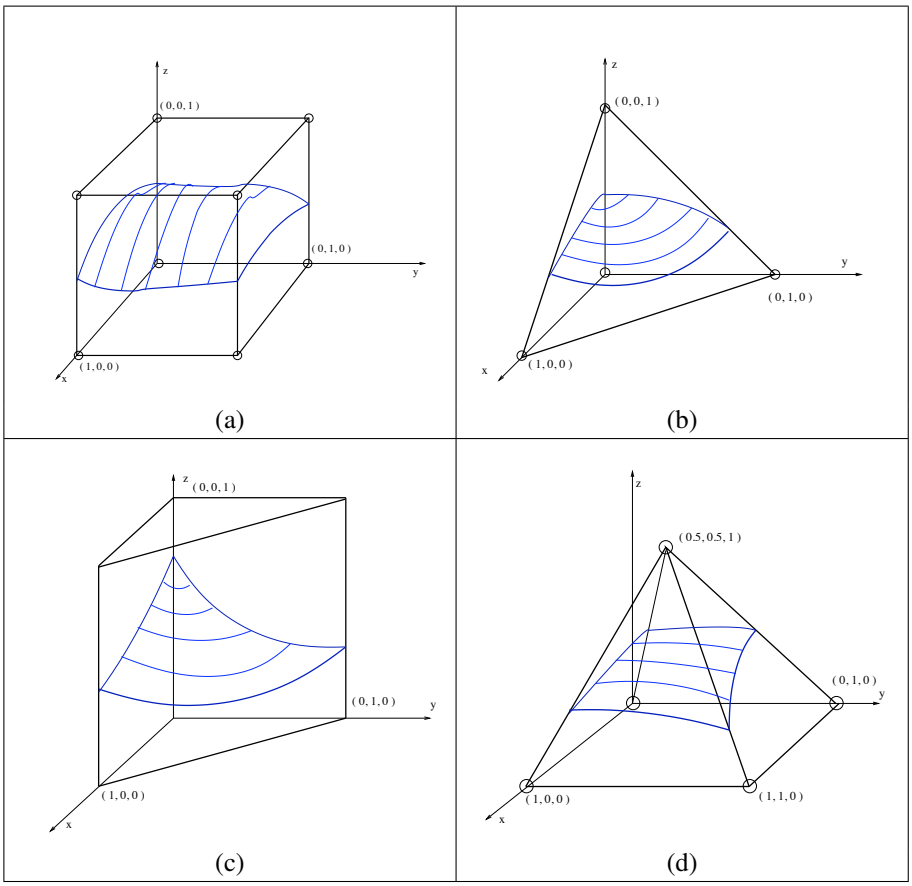


Figure 3.1: A-patch defined within different domain elements[Baj]: (a) A-patch within a cube, which is tensor in 3 dimensions; (b) A-patch within a tetrahedra, which is in barycentric domain; (c) A-patch within a triangular prism; (d) A-patch within a square pyramid.

defined as linear vector barycentric form

$$
\begin{aligned}
\mathbf{p} &= \textstyle\sum_i^4 \mathbf{p_i}\alpha_i \\
\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} &= \begin{pmatrix} p_{0x} & p_{1x} & p_{2x} & p_{3x} \\ p_{0y} & p_{1y} & p_{2y} & p_{3y} \\ p_{0z} & p_{1z} & p_{2z} & p_{3z} \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{pmatrix}
\end{aligned}
$$

where $\mathbf{p_i}$ are four affine independent points in 3D space. The function f is a polynomial over the simplex $0 \leq \alpha_i \leq 1$ and $|\boldsymbol{\alpha}| = \sum_i^4 \alpha_i = 1$ usually defined in B.B. barycentric form

$$f(\boldsymbol{\alpha}) = \sum_{i+j+k+l=n} a_{ijkl} B^n_{ijkl}(\boldsymbol{\alpha})$$

where a_{ijkl} are scalar coefficients and $B^n_{ijkl}(\boldsymbol{\alpha}) = \frac{n!}{i!j!k!l!}\alpha_1^i\alpha_2^j\alpha_3^k\alpha_4^l$ are the trivariate barycentric Berstein basis polynomials (see Fig 3.2(b)).

Triangular prism (see Fig 3.1(c)) The local coordinates $(\alpha = (\alpha_1, \alpha_2, \alpha_3), w)^T$ are now defined as follows: w is defined over the interval $[0,1]$ ($w \in [0,1]$) and $\alpha_1, \alpha_2, \alpha_3$ range over $0 \leq \alpha_i \leq 1$ and $|\boldsymbol{\alpha}| = \sum_i^3 \alpha_i = 1$, with the last equation one of the α_i may always be eliminated as $\alpha_i = 1 - \left(\sum_{j \neq i} \alpha_j\right)$. This yields a mixed coordinate system and the mapping to the global coordinates is defined as a tensor product between a barycentric coordinates and the single coordinate w and it is bilinear

$$\begin{aligned} \mathbf{p} &= \textstyle\sum_{i=1}^3 \mathbf{p_i^{(0)}} \alpha_i (1-w) + \sum_{i=1}^3 \mathbf{p_i^{(1)}} \alpha_i w \\ \begin{pmatrix} x \\ y \\ z \end{pmatrix} &= \begin{pmatrix} (1-w) & w \end{pmatrix} \begin{pmatrix} \mathbf{p_1^{(0)}} & \mathbf{p_2^{(0)}} & \mathbf{p_3^{(0)}} \\ \mathbf{p_1^{(1)}} & \mathbf{p_2^{(1)}} & \mathbf{p_3^{(1)}} \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} \end{aligned}$$

where $\mathbf{p_i^{(1)}}$ are six points. The function f is a polynomial over the prism $w \in [0,1]$, $0 \leq \alpha_i \leq 1$ and $|\boldsymbol{\alpha}| = \sum_i^3 \alpha_i = 1$ and may be expressed in mixed B.B. form

$$f(\boldsymbol{\alpha}, w) = \sum_{i+j+k=n} \sum_{l=0}^{m} a_{ijk}^{(l)} B_l^m(w) B^n_{ijk}(\boldsymbol{\alpha})$$

where $a_{ijk}^{(l)}$ are scalar coefficients, $B_l^m(w) = \frac{n!}{l!(n-l)!} w^i (1-w)^{m-l}$ are the univariate berstein basis polynomials, and $B^n_{ijk}(\boldsymbol{\alpha}) = \frac{n!}{i!j!k!}\alpha_1^i\alpha_2^j\alpha_3^k$ are the bivariate barycentric Berstein basis polynomials (see Fig 3.2(c)).

Square pyramid (see Fig 3.1(d)) The local coordinates $(u, v, \alpha)^T$ are defined u and v to satisfy $u \in [0,1]$ and $w \in [0,1]$, while α satisfies the condition $0 \leq u + v + \alpha \leq 1$. This yields a mixed coordinate system and the mapping to the global coordinates is defined as a barycentric interpolation of a tensor domain and a point q and it is trilinear.

$$\mathbf{p} = \alpha \left[\sum_{i=0}^{1} \sum_{j=0}^{1} \mathbf{p_{ij}} B_i^1(u) B_j^1(v) \right] + (1-\alpha)\mathbf{q}$$

where $\mathbf{p_{ij}}$ define a quadrilateral and $\mathbf{q}$ is the vertex of the pyramid. The function f is a polynomial over the pyramid and may be expressed in mixed B.B. form

$$f(u, v, \alpha_1, \alpha_2) = \sum_{i+j=n} \sum_{k=0}^{j} \sum_{l=0}^{j} a_{ijkl} B^n_{ij}(\alpha_1, \alpha_2) B_k^j(u) B_l^j(v)$$

where $a^{(l)}_{ijkl}$ (also as $a_{i(n-i)kl}$ as $i+j=n$) are scalar coefficients, $B^n_i(t) = \frac{n!}{i!(n-i)!}t^i(1-t)^{n-i}$ is the univariate berstein basis polynomial, and $B^n_{ij}(\alpha_1,\alpha_2)$ is the univariate barycentric Berstein basis polynomial, and as usual one may rewrite i.e. as $\alpha_2 = \alpha$ and $\alpha_1 = 1-\alpha$. Thus equivalently (as in figure 3.2(d)):

$$f(u,v,\alpha) = \sum_{i=0}^{n}\sum_{k=0}^{n-i}\sum_{l=0}^{n-i} a_{ikl}B^n_i(\alpha)B^i_k(u)B^i_l(v)$$

3.2 Regularity and Continuity

Similarly to the case of triangular A-splines, regularity of tetrahedral A-patches may be obtained by enforcing a sign conditions on the coefficients of the polynomial in B.B. form. This technique is employed in cubic C^1 tetrahedral A-patches (see [BCX95] and chapter 7) and potentially applied to A-patches with different scaffold. Under these constraints an A-patch is guaranteed to be smooth (non-singular) and functional. Unlike A-splines, tetrahedral A-patches regularity come into two flavors. In a *three-sided* A-patch any line segment passing through an apex (vertex) of the tetrahedron and its opposite face intersects the surface patch at most once. In a *four-sided* A-patch any line segment connecting two points on opposite edges intersects the patch at most once. The reason for this distinction will be explained shortly. One may notice that the main reference for cubic A-patches [BCX95] predates the formalization of general degree A-splines [BX99a]. In fact a general solution for arbitrary degree A-patches doesn't exist to date and in the case of C^2 tetrahedral quintic patches [BCX97] the sign conditions are too strong and have been relaxed.

In C^1 rational tetrahedral A-patches [XHB01] a technique similar to the discriminating and traversal families [XBC00a, XBC00b, XBC00c] is employed to define a reduced form low degree rational function, in this case the requirement of a single-sheeted in the tetrahedron has been dropped in favor of separating a single sheeted surface piece within each tetrahedron. The low-degree rational form may be converted to a polynomial, albeit of higher degree, however, it may be evaluated at a point without requiring root finding (e.g. may be transformed in a rational form where one α_i is linear). Rational C^1 A-patches has been defined also on pyramid scaffold (quadrilateral base) [XBE02]. A similar technique is employed in prism A-patches [BX99b, BX01, ZXB07, BPP$^+$08] (see Chapter 8) in this case the surface with minimal height (w.r.t the base) is guaranteed to be non-singular in the polyhedron. However, in prism A-pathces root finding is still necessary in order to generate points.

Enforcing continuity in A-patches is complicated by the fact it has to be enforced on the boundary curves as well as points.

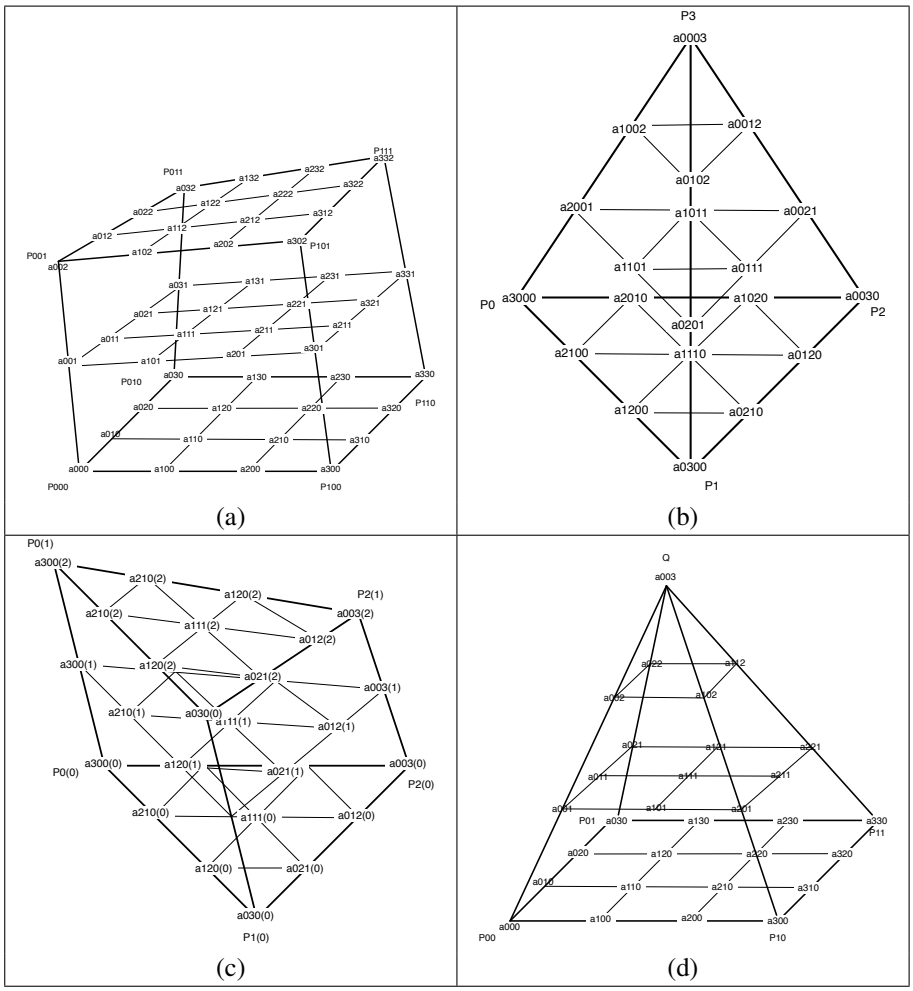


Figure 3.2: A-patch B.B. coefficients for cubics: (a) cuboid (b) tetrahedron (c) triangular prism (d) square pyramid.

Definition 3.2.1. Two algebraic surfaces $f(x,y,z) = 0$ and $g(x,y,z) = 0$ meet with G^k-continuity (rescaling continutiy) along a curve C if and only if there exists functions $\alpha(x,y,z)$ and $\beta(x,y,z)$ such that all derivatives upto order k of $\alpha f - \beta g$ equals zero at all points along C , see for e.g.,[GW91].

For the specific case of A-patches, G^1 continuity between adjacent A-patches reduces to linear equality (coplanarity of the coefficients in a 4D space) conditions between coeficients on common faces and their neighborhoods (see [Baj92, XB95, XHB01, XBE02] and chapter 7)

3.3 Surface Control

Control data for A-patches is usually supplied through a *linear surface triangulation*, a $2-complex$ boundary for a 3-dimentional solid. The triangulation may be obtained by: (1) user input or generative modeling (2) point data of 3D scans (3) iso-surfaces of volumetric images. To understand exactly what is meant by *linear surface triangulation* some definitions are needed [Pao03].

Definition 3.3.1 (*Convex and affine hull*)**.** Le $\{\mathbf{p_1},\ldots,\mathbf{p_d}\}\in\mathbb{R}^n$. Then the *convex hull* of these points is defined by $[\mathbf{p_1},\ldots,\mathbf{p_d}]=\{\mathbf{p}\in\mathbb{R}^3\ :\ \mathbf{p}=\sum_{i=1}^{d}\alpha_i\mathbf{p_i},\ \alpha_i\geq 0\ ,\ \sum_{i=1}^{j}=1\}$, and the *affine hull* is defined by $<\mathbf{p_1},\ldots,\mathbf{p_d}>=\{\mathbf{p}\in\mathbb{R}^3\ :\ \mathbf{p}=\sum_{i=1}^{d}\alpha_i\mathbf{p_i},\ \sum_{i=1}^{j}=1\}$. The interior of the convex hull is denoted $(\mathbf{p_1},\ldots,\mathbf{p_d})=\{\mathbf{p}\in\mathbb{R}^3\ :\ \mathbf{p}=\sum_{i=1}^{d}\alpha_i\mathbf{p_i},\ \alpha_i\geq 0\ ,\ \sum_{i=1}^{d}=1\}$.

Definition 3.3.2 (*d-Simplex*)**.** A *d-simplex* $\sigma_d\subset\mathbb{E}^n$, $(0\leq d\leq n)$ may be defined as the convex hull of $d+1$ affine independent points. A d-simplex may be seen as a d-dimensional triangle: a 0-simplex is a point, a 1 simplex is a segment, a 2-simplex is a triangle, a 3 simplex is a tetrahedron, and so on.

Definition 3.3.3 (*Triangulation and Simplicial d-complex*)**.** A set Σ of d-simplices is called a *triangulation*. A *simplicial complex*, often simply called a complex is a triangulation that verifies the following conditions:

1. if $\sigma\in\Sigma$, then any face of σ belongs to Σ;
2. if $\sigma,\tau\in\Sigma$, then either $\sigma\cap\tau=0$, or $\sigma\cap\tau$ is a face of both σ and τ.

Sometimes one allows the set Σ to contain elements that are not pure d-simplexes but convex hulls of m-points $(m>d)$ of which $d+1$ are affine independent and in this case it is often called *polyhedral d-complex*. Notice that this is a practical representation distinction and as far the mathematical definition goes a polyhedral complex is always isomorphic to simplicial d-complex (the convex hulls of m-points $(m>d)$ may always be split in simplexes).

Definition 3.3.4 (*Polyhedron*)**.** A d-polyhedron [Req77] is any compact set $P_d\subset\mathbb{E}^n$ that allows at least one pair (K,h), where K is a m-complex, and $h:|K|\rightarrow P$ is a homeomorphism, where the support space of K, denoted as $|K|$, is the point-set union of simplices $\sigma\in K$.

An m-polyhedron is said *linear* if h is the identity function; it is said *regular* if the complex K associated with it is pure, i.e. if each simplex is a face of some m-simplex. Two simplices σ_1 and σ_2 in a complex K are s-adjacent if they have a common s-face; they are s-connected if a sequence of simplices in K exists, beginning with σ_1 and ending with σ_2 , such that any two successive terms of the sequence are s-adjacent. If any $(d-1)$-simplex in Σ is a face of exactly two d-simplices, then Σ is said to be closed; otherwise it is said to be open.

Definition 3.3.5 (*Surface*)**.** We call *surface* any 2-polyhedron which supports a triangulation (K,h) such that:

1. any 1-simplex in K is a face of at most two 2-simplices in K;
2. for each 0-simplex $v \in K$, the 2-simplicies of K for which v is a face that can be circularly ordered so that any consecutive pair is 1-adjacent.

Such conditions impose that a surface should be a *manifold* object, where the neighbourhood of each point is homeomorphic to the open disc. If the first condition is strictly verified, i.e. if each 1-simplex in K is a face of exactly two 2-simplices in K, then the surface is *closed*; otherwise it is *open*. If the surface is closed and orientable, then it is the boundary of a solid.

Definition 3.3.6 (*Boundary*)**.** The *boundary* ∂P_d of a polyhedron P_d is the geometric carrier of the $(d-1)$ complex whose $(d-1)$-simplices are faces of exactly one d-simplex in a complex that triangulates P_d.

Definition 3.3.7 (*Shell*)**.** A maximal 1-connected component of a closed surface is called a *shell* of the surface.

In tetrahedral (see [BCX95, BCX97, XHB01] and chapter 7) and triangular prism patches (see [ZXB07, BPP$^+$08] and chapter 8) one is given a list of points $P = \{\mathbf{p_1}, \ldots, \mathbf{p_k}\} \in \mathbb{R}^n$ and a surface triangulation T of these points. In case of mixed or multisided schemes (see [XBE02, BX01] and chapter 8) the "triangulation" is allowed to contain quadrilaterals. In case of shell surfaces [BX99b, BX01] a pair of matched triangulations $[T^{(0)}, T^{(1)}]$ is given. Furthermore additional data is given for each point such as set of normals $N = \{\mathbf{n_1}, \ldots, \mathbf{n_k}\} \in \mathbb{R}^n$ for C^1 interpolation. In [BCX97] C^2 interpolation is provided by suplying the hessian matrix $\mathscr{H}$ at each point of the triangulation.

$$\mathscr{H}(f) = \begin{pmatrix} \frac{\partial f}{\partial^2 x} & \frac{\partial f}{\partial x \partial y} & \frac{\partial f}{\partial x \partial z} \\ \frac{\partial f}{\partial y \partial x} & \frac{\partial f}{\partial^2 y} & \frac{\partial f}{\partial y \partial z} \\ \frac{\partial f}{\partial z \partial x} & \frac{\partial f}{\partial z \partial y} & \frac{\partial f}{\partial^2 z} \end{pmatrix}$$

The objective is to construct a mesh (piecewise gluing) of low degree algebraic surfaces (an A-patch) such the composite surface (as A-patch set) is single sheeted, interpolates/aproximates the points, and has a desired continuity. That is to build a topologically equivalent non-linear surface triangulation defined by algebraic finite elements.

To do so a simplicial hull of T denoted by Σ is built to supply a scaffold for each patch. Different configurations of vertex "normals" for edges and faces of T are categorized as "convex" and "non-convex":

Edge (Figure 3.3) Let $[\mathbf{p_i p_j}]$ be an edge of T. If $(\mathbf{p_j} - \mathbf{p_i})^T \mathbf{n_i} (\mathbf{p_i} - \mathbf{p_j})^T \mathbf{n_j} \geq 0$ and at least one of $(\mathbf{p_j} - \mathbf{p_i})^T \mathbf{n_i}$ and $(\mathbf{p_j} - \mathbf{p_i})^T \mathbf{n_j}$ is positive, then the edge $[\mathbf{p_j p_i})]$ is said *negative convex*. If both are zero then we say it is *zero convex*. A *positive convex* edge is similarly defined. If $(\mathbf{p_j} - \mathbf{p_i})^T \mathbf{n_i} (\mathbf{p_i} - \mathbf{p_j})^T \mathbf{n_j} < 0$, then we say the edge is *non convex*.

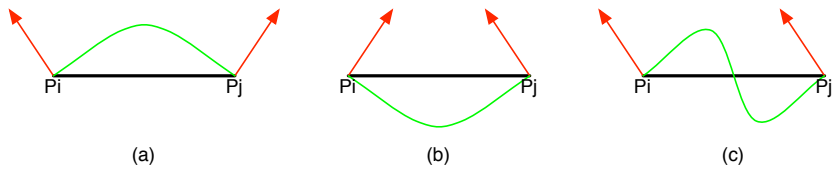


Figure 3.3: Edge convexity: (a) positive convex; (b) negative convex; (c) non-convex.

Face (Figure 3.4) Let $[\mathbf{p_i p_j p_k}]$ be a face of T. If its three edges are nonnegative (positive or zero) convex and at least one of them is positive convex, then we say the face $[\mathbf{p_i p_j p_k}]$ is positive convex. If all of the three edges are zero convex, then we label the face as zero convex. A negative convex face is similarly defined. All of the other cases $[\mathbf{p_i p_j p_k}]$ are labeled as nonconvex.

The edges and faces together with their normals are thus tagged as "convex" and "non-convex". In polynomial tetrahedral patches and shell prism patches not all triangulations are apt to build the A-patch set. In this case offending triangles in the triangulation have to be split. Then along the (new) triangulation T:

Face In tetrahedral A-patches for every face $[\mathbf{p_i p_j p_k}]$ of T, a *face tetrahedra* $[\mathbf{p_i p_j p_k q}]$ are built by adding a point q over and/or under the face (see figure 3.5a). If the face is positive convex the point is added over the face, if negative complex the point is added below the face, and both points are added for the non convex face. For the actual location of the added point $\mathbf{q}$ refer to the relevant chapter (7) or aforementioned articles. The patch built on face tetrahedra will be *three sided*. In mixed and multisided schemes, pyramidal A-patches are built analogously if the T contains quadrilaterals.

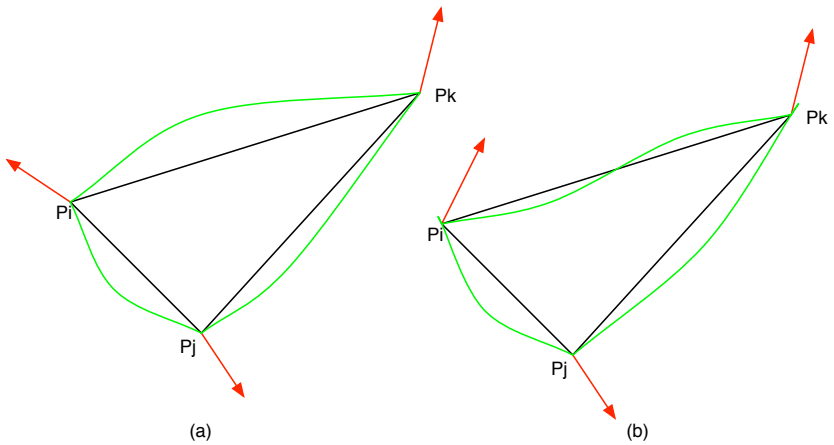


Figure 3.4: Face convexity: (a) convex; (b) non-convex.

In prism A-patches (see figure 3.5b) only one prism is built for each face. If the prism A-patch is simple, then for each face $[\mathbf{p_i p_j p_k}]$ the prism built will be

$$[(\mathbf{p_i}-\mathbf{n_i})(\mathbf{p_j}-\mathbf{n_j})(\mathbf{p_k}-\mathbf{n_k})(\mathbf{p_i}+\mathbf{n_i})(\mathbf{p_j}+\mathbf{n_j})(\mathbf{p_k}+\mathbf{n_k})].$$

In shell prism for every pair of matched faces $[\mathbf{p_i^{(0)}p_j^{(0)}p_k^{(0)}}]$ and $[\mathbf{p_i^{(1)}p_j^{(1)}p_k^{(1)}}]$ the following prism will be built

$$[\mathbf{p_i^{(0)}p_j^{(0)}p_k^{(0)}p_i^{(1)}p_j^{(1)}p_k^{(1)}}].$$

In mixed and multisided schemes a quadrilateral prism is built if the "triangulation" T contains quadrilaterals.

Edge Let $[\mathbf{p_i p_j p_k}]$ and $[\mathbf{p'_i p_j p_k}]$ be two adjacent faces in the triangulation T with edge $[\mathbf{p_j p_k}]$. In all prism A-patches no additional complex is built for edges. The quadrilateral $[\mathbf{p_j^{(0)}p_k^{(0)}p_j^{(1)}p_k^{(1)}}]$ will contain the prism A-spline segment of the boundary of the resulting curved triangulation. In tetrahedral schemes, two face tetrahedral will only meet at the common edge $[\mathbf{p_j p_k}]$ thus a joining simplicial hull (a "join") has to be built. If both faces are convex then only one "join" is needed, if one is non convex then two "joins" are needed (one above and one below the triangulation). In polynomial tetrahedral patches each "join" consists of two *egde tetrahedra* $[\mathbf{p_j p_k q q''}]$ and $[\mathbf{p_j p_k q' q''}]$ by adding a point $\mathbf{q''}$ (Fig 3.6a). Again for the actual location of the added point $\mathbf{q''}$ refer to the relevant chapter (7) or aforementioned articles. In rational tetrahedral patches only one tetrahedra

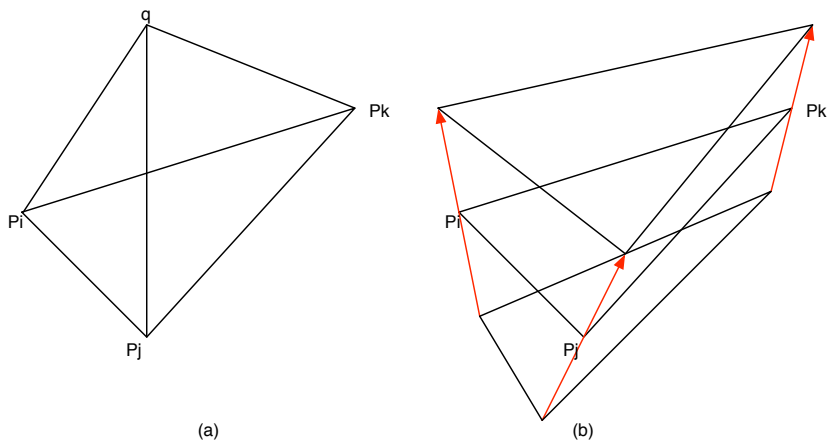


Figure 3.5: Face polyhedron: (a) tetrahedra; (b) prism.

$[\mathbf{p_j p_k q q'}]$ per "join" is needed (Fig 3.6b). In mixed and multisided schemes pyramidal A-patches nothing changes as the edge $[\mathbf{p_j p_k}]$ may be in common between two quadrilaterals/triangles. The patch built on an edge tetrahedra will be *four sided*.

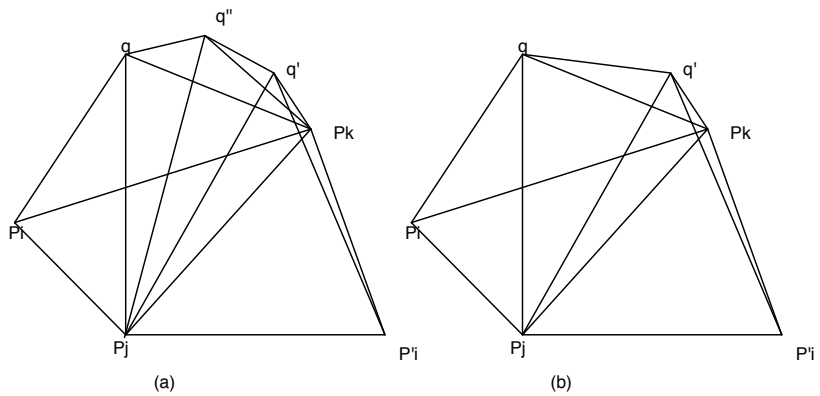


Figure 3.6: Join of tetrahedra: (a) polynomial A-patch; (b) rational A-patch.

After the scaffolding simplicial hull is built, the coefficients of the algebraic equation are assigned for interpolation/aproximation, continuity and finer curve control. If the coefficients a_{n000}, a_{0n00}, a_{00n0}, and a_{00n0} are zero then the patch will interpolate the

corresponding point on the scaffold, if all four are zero then the curve will be singular. If, say, $a_{(n-k)k00} = 0$ (or any other "segment" of coefficients) then the surface will be tangent to the edge (i.e. $[\mathbf{p_i}\mathbf{p_j}]$) at the point (i.e $\mathbf{p_i}$) k times, if $k = n$ then the surface will interpolate all the edge (i.e. $[\mathbf{p_i}\mathbf{p_j}]$). If $a_{(n-i-k)ik0} = 0$, for $i \in [0,n]$ and $k < n$ then the surface will interpolate the edge (i.e. $\mathbf{p_i}\mathbf{p_j}$) and here tangent to the face (i.e. $[\mathbf{p_i}\mathbf{p_j}\mathbf{p_k}]$) k times. The actual assignment will be studied in the next chapters for cubic polynomial tetrahedral A-patches case (chapter 7), prism cubic A-patches (chapter 8). For all remaining schemes please refer to the provided literature.

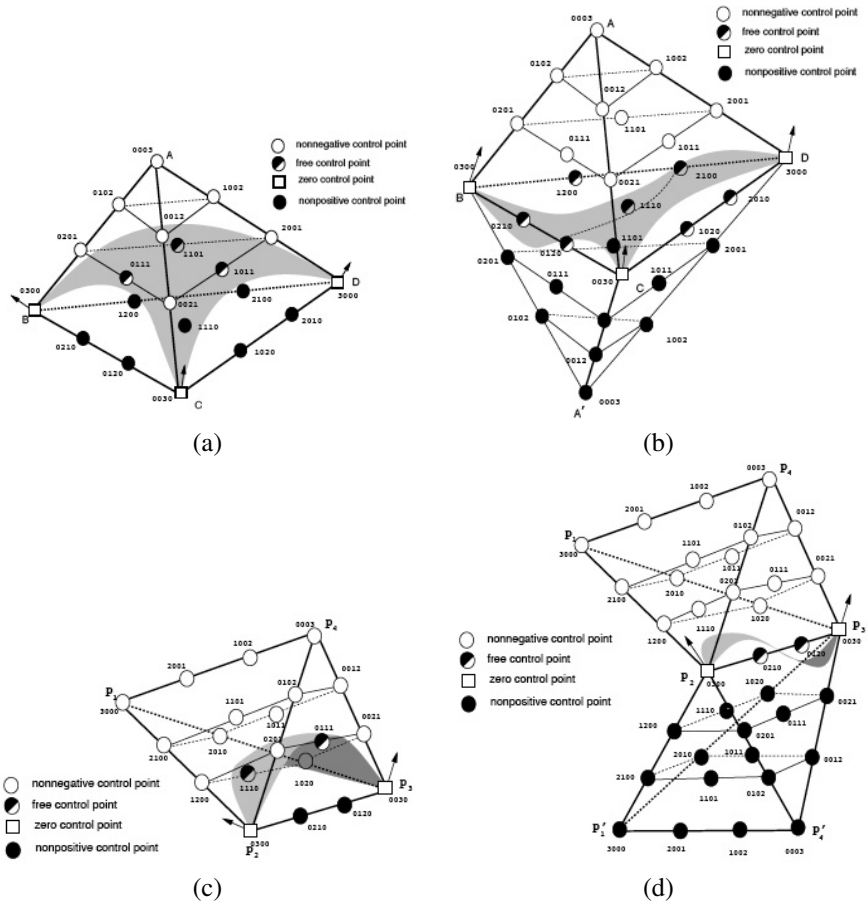


Figure 3.7: A-patches defined by a single change of coefficients in preferred directions[Baj]. (a) A three sided A-patch interpolating at points **B**,**C**,**D**. Convex vertex normals. (b) A three sided A-patch defined over a double stack of tetrahedra for a case of non-convex vertex normals. (c) A four sided A-patch in case of convex vertex normals. (d) A four sided A-patch defined over a double stack of tetrahedra for a case of non-convex vertex normals.

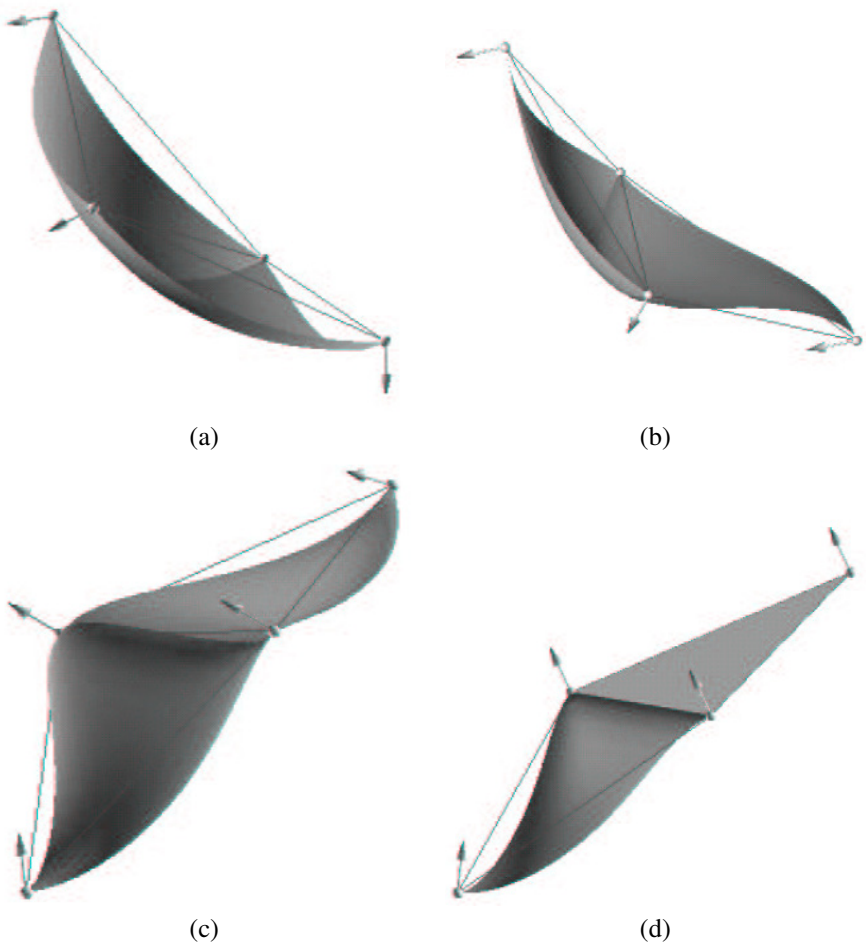


Figure 3.8: Two prism patches with different join configurations[Baj]: (a) two convex patches; (b) convex patch and a non-convex patch; (c) two non-convex patches; (d) zero-convex (triangle) patch and a non-convex patch.

Chapter 4

Operations on algebraic finite elements

In this chapter a series of exercise and examples on algebraic varieties and finite algebraic elements are presented. These examples were worked by the author and Prof. Na Lei (Inst. of Mathematics, Jilin University, China) for the post graduate course in Geometric Modeling and Visualization held by professor Chandrajit Bajaj at the University of Texas at Austin. The rationale is to provide a good overview on the possible operations on algebraic finite elements in practical and by example manner. The definition of algebraic finite element used is quite free. The point is, to define the characteristics that an algebraic finite element scheme should have in order to solve a problem or define an algorithm for a certain operation. They are long away from being complete applications, however, they highlight well the problematics, sketch quite well the general solution and have an excellent explanatory value. The examples on the computation of the surface of an union of spheres and of the offset surfaces of said union of spheres introduce and provide a "human executable" operative detail (vs "computer executable") on the algorithm presented in Chapter 5 to build surface representation of molecular surfaces. Similarly, all the examples on the intersection between algebraic varieties and their use to build an algebraic finite elements description by assembly of these varieties, constitute an introduction and, again, a "human executable" detail to the algorithm for boolean operations provided in Chapter 9.

4.1 Intersection and Assembly

Surface to Space Curve Intersection

An algebraic space curve curve and can be represented by the intersection of two algebraic surfaces. An algebraic curve segment is also provided with two boundary vertices (points on the curve), and a direction indicator for the starting and ending point. Com-

pute the intersection of the surface $S : x^2+y^2+z^2-1=0$ and the space curve segment $C : (x^2+y^2-z=0, x=0)$ with starting vertex $(0,2,4)$ and ending vertex $(0,-2,4)$.

Surface S: $x^2+y^2+z^2-1=0$, a sphere with center $C=(0,0,0)$ and radius $r=1$).

Curve C: $(x^2+y^2-z=0, x=0)$ (plane $x=0 \cap$ conicoid along z axis)

First of all the curve is easily parameterizable by first parameterizing the plane $x=0$ as $(x=0, y=s, z=t)$ and then substituting these functions in the conicoid equation: $(t=s^2)$ and then back in the plane equation:

$$\begin{aligned} x &= 0 \\ y &= s \\ z &= s^2 \end{aligned}$$

Points $v_1=(0,2,4)$ and $v_2=(0,-2,4)$ are interpolated for $s=2$ and $s=-2$. Deriving the plane curve in s: $(\frac{dx}{ds}=0, \frac{dy}{ds}=1, \frac{dz}{ds}=2s)$ the tangent $t_1=(0,1,-4)$ is satisfied for $s=2$ and $ds=-1$. This means the segment defined by s decreasing from v_1 to v_2 . Thus the segment is defined in $s \in [2 \ldots -2]$ passing through 0. One could reparameterize these functions (e.g. $s' \in [0 \ldots 1]$ and $s=-4s'+2$) but it is not needed.

To intersect one substitutes the parametric functions in the surface equation S having $s^2+s^4-1=0$, solving one has $s^2=t=\frac{-1\pm\sqrt{5}}{2}$, excluding imaginary points one has $s=\pm\sqrt{\frac{-1+\sqrt{5}}{2}}$ (inside the range of s). The intesection points are (figure 4.1):

$$\begin{aligned} x &= 0 \\ y &= \pm\sqrt{\tfrac{-1+\sqrt{5}}{2}} \simeq \pm 0.7861 \\ z &= \tfrac{-1+\sqrt{5}}{2} \simeq 0.6180 \end{aligned}$$

Surface to Surface Intersection

A surface patch is a surface with a closed boundary. An algebraic surface patch can be represented by a single polynomial equation for the surface and a closed cycle of curve segments on the surface. Compute the intersection of the spherical surface patch [Surface: $x^2+y^2+z^2-1=0$, Curve segment: $x^2+y^2-1=0$ with a cycle of vertices $(1,0,0)$, $(0,-1,0)$, $(-1,0,0)$, $(0,1,0)$ defining the cycle of ordered curve segments], with (a) the plane $y=z$, and, with (b) the surface $y^2+z^2-1=0$.

Surface patch S is described by equation: $x^2+y^2+z^2-1=0$, a sphere of center $C=(0,0,0)$ and radius $r=1$; it is delimited, by $S_c : x^2+y^2-1=0$, a circumference centered in C and with radius r on xy plane, and, by the series of ordered vertices

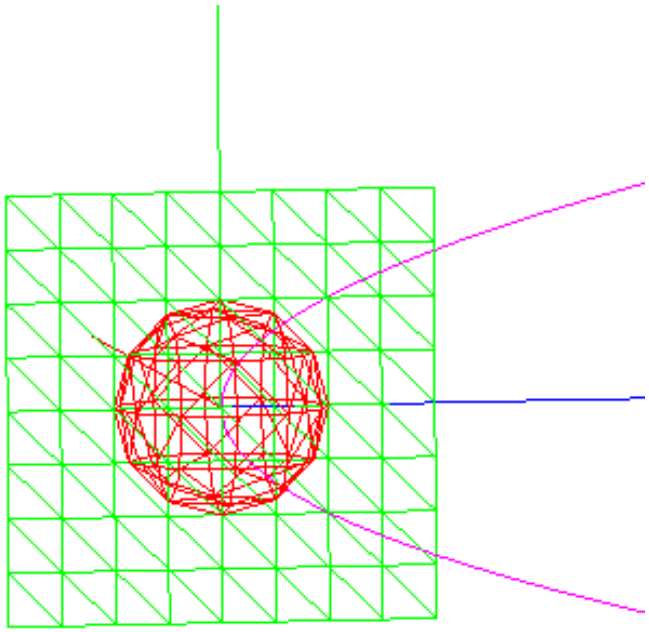


Figure 4.1: Intersection between sphere S and curve C. Image generated by the author using GANITH [BR90a].

$(1,0,0)$,$(0,-1,0)$, $(-1,0,0)$, $(0,1,0)$. Plane S_a:y=z is the plane bisecting $y-z$ axis and $S_b : x^2+z^2-1=0$ is a cylinder centered in O, with radius r lying on xz plane, and expanding along y. By elimination between S and S_c one obtains $z=0$. This means that this equation and any one of the equations S_c or S describes the patch delimiting curve. It means also that the curve lies on the $z=0$ plane. By the ordering of the vertex one obtains the patch is the half sphere pointing towards negative z axis.

(a) $SP \cap S_a$**:** The curve equations describing the intersection is already fully described by S and S_a. However if one wants to compute another surface, containing the curve, which intersected with either S_a or S will describe the curve, one may use the $y=z$ equation to obtain either: $Res_{0,z}(S,S_a) : x^2+2y^2-1=0$ or $Res_{0,y}(S,S_a) : x^2+2z^2-1=0$. The may be useful to find another vertex on curve segment for example when $x=0$ one has $z=y=\frac{1}{\sqrt{2}}=\frac{\sqrt{2}}{2}=sin(\pi/4)=cos(\pi/4)$.

To compute the extreme vertex of the curve segment one must check the intersection $S \cap S_c \cap S_a$. Alternatively one may intersect $S_a, y=z$, and S. Obtaining

$(x = \pm 1, y = 0, z = 0)$

To sum up $SP \cap S_a$ is the curve segment described by the intersections of any two of the following equations S_a, S, $Res_{0,z}(S, S_a)$,$Res_{0,y}(S, S_a)$. A conventional choice is:

$$\begin{array}{ll} Res_{0,z}(S, S_a): & x^2 + 2y^2 - 1 = 0 \\ Res_{1,z}(S, S_a): & x - z = 0 \end{array}$$

The delimiting points are $(1,0,0)$ and $(-1,0,0)$ and the tangent at $(1,0,0)$ is $(0, -\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2})$.

(b) $SP \cap S_b$**:** The curve equations describing the intersection is already fully described by S and S_b. However in this case it is much better to eliminate x and y, thus obtaining $z = 0$:

$$\begin{array}{ccccccccc} x^2 & + & y^2 & + & z^2 & - & 1 & = & 0 \\ x^2 & + & y^2 & & & - & 1 & = & 0 \\ \hline & & & & z^2 & & & = & 0 \end{array}$$

To compute the extreme vertex of the curve segment on must check the intersection $S \cap S_c \cap S_b$. Alternatively one may intersect $z = 0, y = z$, and S. Obtaining $(x = \pm 1, y = 0, z = 0)$

To sum up the intersection curve is described by any two of S_b, S, $z = 0$. The delimiting points are $(1,0,0)$ and $(-1,0,0)$ and the tangent at $(1,0,0)$ is $(0,0,-1)$.

Union of Spheres

Consider an arrangement of spheres of varying radii in $\mathbb{R}^3$ (e.g. atoms of a molecule) as described by a 4-tuples (center-coordinates, radius). Given an arrangement of four spheres

$$[c = (0,0,0), r = 1], [c = (0,0,1), r = 0.75], [c = (0,1,0), r = 0.75], [c = (1,0,0), r = 0.25]$$

compute a boundary representation of the union of the arrangement (e.g. spacial description of the molecule). [Hint: All pairwise spheres and triple sphere intersections must be computed].

The equations of the spheres (see Figure 4.2)

$$\begin{array}{ll} S_1 = [c = (0,0,0), r = 1] & S_2 = [c = (0,0,1), r = 0.75] \\ S_3 = [c = (0,1,0), r = 0.75] & S_4 = [c = (1,0,0), r = 0.25] \end{array}$$

are:

$$\begin{array}{lll} S_1: & x^2 + y^2 + z^2 - 1 & = x^2 + y^2 + z^2 - 1 & = 0 \\ S_2: & x^2 + y^2 + (z-1)^2 - 3/4 & = x^2 + y^2 + z^2 - 2z + 7/16 & = 0 \\ S_3: & x^2 + (y-1)^2 + z^2 - 3/4 & = x^2 + y^2 + z^2 - 2y + 7/16 & = 0 \\ S_4: & (x-1)^2 + y^2 + z^2 - 1/4 & = x^2 + y^2 + z^2 - 2x + 3/4 & = 0 \end{array}$$

$C_{12} = S_1 \cap S_2$:

$$\begin{array}{ccccccccccc} x^2 & + & y^2 & + & z^2 & & & - & 1 & = & 0 \\ x^2 & + & y^2 & + & z^2 & - & 2z & + & 7/16 & = & 0 \\ \hline & & & & & & 2z & - & 23/16 & = & 0 \end{array}$$

Thus the intersection plane is $z = 23/32$ and substituting in S_1 the intersection circle is $C_{12} : x^2 + y^2 + 529/1024 - 1 = x^2 + y^2 - 495/1024 = 0$

$C_{13} = S_1 \cap S_3$:

$$\begin{array}{ccccccccccc} x^2 & + & y^2 & + & z^2 & & & - & 1 & = & 0 \\ x^2 & + & y^2 & + & z^2 & - & 2y & + & 7/16 & = & 0 \\ \hline & & & & & & 2y & - & 23/16 & = & 0 \end{array}$$

Thus the intersection plane is $y = 23/32$ and substituting in S_2 the intersection circle is $C_{13} : x^2 + z^2 - 495/1024 = 0$

$C_{14} = S_1 \cap S_4$:

$$\begin{array}{ccccccccccc} x^2 & + & y^2 & + & z^2 & & & - & 1 & = & 0 \\ x^2 & + & y^2 & + & z^2 & - & 2x & + & 3/4 & = & 0 \\ \hline & & & & & & 2x & - & 7/4 & = & 0 \end{array}$$

Thus the intersection plane is $x = 7/8$ and substituting in S_3 the intersection circle is $C_{14} : 49/64 + y^2 + z^2 - 1 = y^2 + z^2 - 15/6 = 0$

$C_{23} = S_2 \cap S_3$:

$$\begin{array}{ccccccccccc} x^2 & + & y^2 & + & z^2 & - & 2z & + & 7/16 & = & 0 \\ x^2 & + & y^2 & + & z^2 & - & 2y & + & 7/16 & = & 0 \\ \hline & & & & & & 2y-2z & - & 0 & = & 0 \end{array}$$

Thus the intersection plane is $z = y$ and substituting in either S_2 or S_3 the intersection circle is $C_{23} : 2y^2 - 2y + x^2 + 7/16$

$C_{24} = S_2 \cap S_4$:

$$\begin{array}{ccccccccccc} x^2 & + & y^2 & + & z^2 & - & 2z & + & 7/16 & = & 0 \\ x^2 & + & y^2 & + & z^2 & - & 2x & + & 3/4 & = & 0 \\ \hline & & & & & & 2x-2z & - & \frac{5}{16} & = & 0 \end{array}$$

Thus the intersection plane is $x - z - \frac{5}{32} = 0$ ($z = x - 5/32$) and substituting in S_4 one has $2x^2 + y^2 - 37x/16 + 793/1024$. Factoring the polynomial in the variable x and checking the discriminant one has $(-\frac{37}{16})^2 - 8(y^2 + 793/1024) \simeq -8y^2 + 5.347 - 6.195$ and is always negative for any y. The intersection is empty, $C_{24} = 0$.

$C_{34} = S_3 \cap S_4$**:**

$$\begin{array}{ccccccccccc} x^2 & + & y^2 & + & z^2 & - & 2y & + & 7/16 & = & 0 \\ x^2 & + & y^2 & + & z^2 & - & 2x & + & 3/4 & = & 0 \\ \hline & & & & & & 2x-2y & - & \frac{5}{16} & = & 0 \end{array}$$

Thus the intersection plane is $x - y + \frac{5}{32} = 0$ ($y = x - 5/32$) and substituting in S_4 one has $2x^2 + z^2 - 37x/16 + 793/1024$. The discriminant is always negative and the intersection is empty, $C_{34} = 0$.

$\{P_{123}\} = C_{12} \cap C_{23} = S_1 \cap (y = 23/32) \cap (z = 23/32) = \varnothing$**:**
Substituting in $S_1 : X^2 + 529/1024 + 529/1024 - 1$ one has only complex solutions.

$\{P_{234}\} = C_{23} \cap C_{34} = \varnothing \;\; (C_{34} = \varnothing)$

$\{P_{134}\} = C_{13} \cap C_{34} = \varnothing \;\; (C_{34} = \varnothing)$

$\{P_{124}\} = C_{12} \cap C_{24} = \varnothing \;\; (C_{24} = \varnothing)$

To sum up the intersection may be described by the following surface patches. Tangent vectors, giving the direction on the curves, are sometimes given up to positive constants (eg: give the general direction):

1. Surface S_2 delimited by the curve segments consisting of
   a) C_{12} curve, delimited by $p_0 = (0, -3/4, 23/32)$, $p_1 = (0, 3/4, 23/32)$, $v_0 = (1, 0, 0)$
   b) C_{12} curve, delimited by $p_0 = (0, 3/4, 23/32)$, $p_1 = (0, -3/4, 23/32)$, $v_0 = (-1, 0, 0)$
2. Surface S_3 delimited by the curve segments consisting of
   a) C_{13} curve, delimited by $p_0 = (-3/4, 23/32, 0)$, $p_1 = (3/4, 23/32, 0)$, $v_0 = (0, 0, 1)$
   b) C_{13} curve, delimited by $p_0 = (3/4, 23/32, 0)$, $p_1 = (-3/4, 23/32, 0)$, $v_0 = (0, 0, -1)$
3. Surface S_4 delimited by the curve segments consisting of
   a) C_{14} curve, delimited by $p_0 = (7/8, -1/4, 0)$, $p_1 = (7/8, 1/4, 0)$, $v_0 = (0, 0, 1)$
   b) C_{14} curve, delimited by $p_0 = (7/8, 1/4, 0)$, $p_1 = (7/8, -1/4, 0)$, $v_0 = (0, 0, -1)$

4. Surface S_1 delimited by the curve segments :

   a) $x^2+y^2-1 \cap z=0$, $p_0=(7/8,1/4,0)$, $p_1=(3/4,23/32,0)$, $v_0=(cos(\alpha),sin(\alpha),0)$ $\alpha \in (\pi/2,\pi)$
   b) C_{12} , delimited by $p_0=(3/4,23/32,0)$, $p_1=(0,23/32,3/4)$, $v_0=(0,0,1)$
   c) $y^2+z^2-1 \cap x=0$, $p_0=(0,23/32,3/4)$, $p_1=(0,3/4,23/32)$, $v_0=(0,-y,+z)$
   d) C_{13}, delimited by $p_0=(0,3/4,23/32)$, $p_1=(0,-3/4,23/32)$, $v_0=(+x,-y,0)$
   e) $y^2+z^2-1 \cap x=0$, $p_0=(0,-3/4,23/32)$, $p_1=(0,-1,0)$, $v_0=(0,-y,-z)$
   f) $x^2+y^2-1 \cap z=0$, $p_0=(0,-1,0)$, $p_1=(7/8,-1/4,0)$, $v_0=(-1,0,0)$
   g) C_{14}, delimited by $p_0=(7/8,-1/4,0)$,$p_1=(7/8,1/4,0)$, $v_0=(0,0,1)$

5. Surface S_1 delimited by similar curve segments for the other part of the surface:

An automatic computational method will easily compute the various border surface and curves. The arrangement of the various parts in a new patch will require a degree of combinatory logic. In chapter 5 a solution for molecular models will be presented.

4.2 Surface of Revolution

Given a cyclic boundary in the xz-plane defined by $f(x,z)=0$ and a planar curve in xy-plane defined by $g(x,y)=0$, the surface $g(f(x,z),y)=0$ obtained in revolving $g(x,y)$ around y-axis along the boundary of $f(x,z)$ is known as the surface of revolution of $g(x,y)$ around y-axis w.r.t.$f(x,z)$. This definition can be trivially extended to higher dimensions. Note that if each $C \in \mathbb{R}^{n-1}$ then the surface of revolution is in R^n.

A-patch from Revolution of Quadratic A-spline

Consider a C^1-interpolatory quadratic A-spline, D, defined in the $x=0$ plane (i.e., Y-Z plane) with none of the vertices $\overrightarrow{P_0},\overrightarrow{P_1},\ldots,\overrightarrow{P_n}$ incident on the z-axis.

1. Describe a square pyramidal A-patch data structure that represents the spline surface of revolution generated when D is revolved about the Z-axis.

2. What is the degree of the spline surface?

3. What property of the A-spline would yield a lower degree spline surface of revolution?

4. Convert the square-pyramidal representation to a tetrahedral A-patch representation.

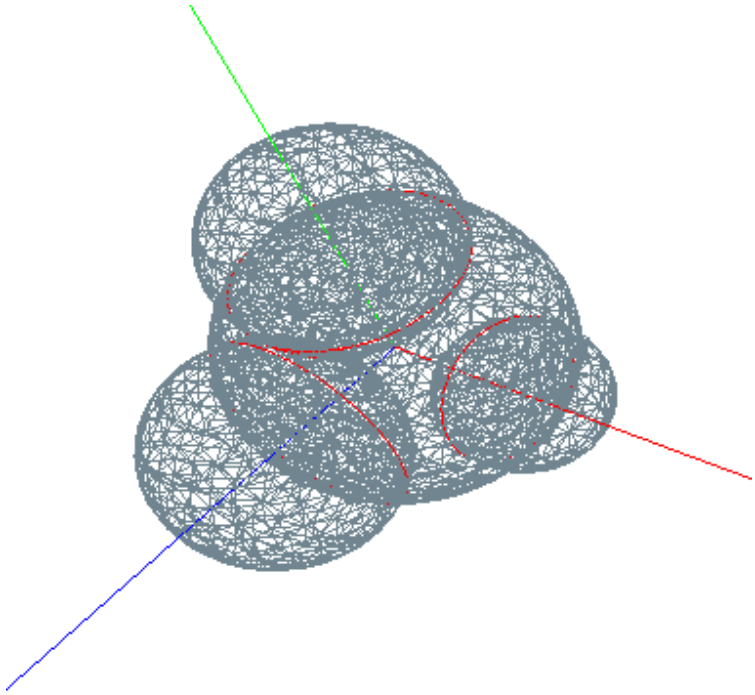


Figure 4.2: Spheres S_1, S_2, S_3, S_4 and their intersections. Image generated by the author using GANITH [BR90a].

The surface of revolution of an algebraic plane curve $f(y,z) = 0$ about the Z axis, can be described by the implicit equation algebraic given by $f(r,z) = f(\sqrt{x^2+y^2},z) = 0$, since the circle of revolution is $x^2+y^2 = r^2$, and $r = y$, when $x = 0$ and r doesn't change. The single square root can be easily removed by separating all terms involving the square-root on one-side of the equation, and the rest of the terms on the other, and then powering both sides without changing the resulting zero-set of the surface equation. If the plane curve is of degree d, and has y-terms with only even exponents ,y^{2i}, $0 \leq i \leq d$, then the square root is already eliminated, and the resulting degree of the surface of revolution is not doubled. A plane curve whose equation has y-terms with only even exponents means it is even in y, alas symmetric to the z axis.

The spline curve in the $Y-Z$ plane, may be constructed, using any kind of A-spline. For example, let the spline be described by a rectangular quadratic A-spline, the equation of

each segment will be:

$$f(\alpha_1,\alpha_2)=\sum_{i=0}^{2}\sum_{j=0}^{2} b_{ij}\begin{pmatrix} n \\ i \end{pmatrix}\alpha_1^i(1-\alpha_1)^{2-i}\begin{pmatrix} n \\ j \end{pmatrix}\alpha_2^j(1-\alpha_2)^{2-j}=0$$

Where the $\alpha_{\_}$ are the local coordinates in the scaffolding rectangle (p_0,p_1,p_2,p_3) of the segment, related to the (r,z) coordinates by:

$$\begin{bmatrix} r \\ z \\ 1 \end{bmatrix}=\begin{bmatrix} (p_{1,r}-p_{2,r}) & (p_{1,r}-p_{3,r}) & p_{1,r} \\ (p_{1,z}-p_{2,z}) & (p_{1,z}-p_{3,z}) & p_{1,z} \\ 1 & 1 & 1 \end{bmatrix}\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ 1 \end{bmatrix}$$

Inverting said relation one has:

$$\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ 1 \end{bmatrix}=\begin{bmatrix} (p_{1,r}-p_{2,r}) & (p_{1,r}-p_{3,r}) & p_{1,r} \\ (p_{1,z}-p_{2,z}) & (p_{1,z}-p_{3,z}) & p_{1,z} \\ 1 & 1 & 1 \end{bmatrix}\begin{bmatrix} r \\ z \\ 1 \end{bmatrix}$$

Substituting the $\alpha_{\_}=\alpha(r,z)$ in f one has a form $f(r,z)=0$, thus effectively transforming the equation in power basis. Alternatively one could have set up the basis transformation matrix. Applying the rotation one has the surface $f(r,y,z)=f(x,y,z)=0$ in the current coordinate system.

At this point one has to define a new A-patch scaffold and transform the surface equation into the new scaffold coordinate system. An octree partition of the space will be used. Let $v_0=(y_0,z_0)$ and $v_1=(y_1,z_1)$ be the original extreme points of the curve segment. Let $p_y=(y_y,z_y)$ and $p_z=(y_z,z_z)$ be the eventually x and y extreme points of the curve segment (being quadratic there are a maximum of one, in general these will be a set of points). The planes partitioning the octree will be (where $\_\in\{0,1,y,z\}$:

$$\begin{aligned} x&=y_{\_} \\ y&=y_{\_} \\ z&=z_{\_} \end{aligned}$$

At this point one has to convert back into tensorial form for the rectangular patches:

$$f(\alpha_1,\alpha_2)=$$
$$\textstyle\sum_{i=0}^{2}\sum_{j=0}^{2}\sum_{k=0}^{2} b_{ij}\begin{pmatrix} n \\ i \end{pmatrix}\alpha_1^i(1-\alpha_1)^{2-i}\begin{pmatrix} n \\ j \end{pmatrix}\alpha_2^j(1-\alpha_2)^{2-j}\begin{pmatrix} n \\ k \end{pmatrix}\alpha_3^k(1-\alpha_3)^{2-k}$$

from the coordinate conversion matrix. Let the generic cuboid with corner $q=(x_q,y_q,z_q)$ and side lengths (a,b,c), the coordinate transformation is :

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}=\begin{bmatrix} a & 0 & 0 & x_q \\ 0 & b & 0 & y_q \\ 0 & 0 & c & z_q \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ 1 \end{bmatrix}$$

To build the square pyramidal A-patch one may split the cuboid in four pyramids at the barycentric point $b = (x_b, y_b, z_b) = (x_q + a/2, y_q + b/2, z_q + b/2)$.
The mixed tensor barycentric form for the pyramidal patch is:

$$f(\alpha_1, \alpha_2) = \sum_{i=0}^{2}\sum_{j=0}^{2}\sum_{k=0}^{2} b_{ij} \begin{pmatrix} n \\ i \end{pmatrix} \alpha_1^i (1-\alpha_1)^{2-i} \begin{pmatrix} n \\ j \end{pmatrix} \alpha_2^j (1-\alpha_2)^{2-j} \begin{pmatrix} n \\ k \end{pmatrix} \alpha_3^k = 0$$

Where coordinates conversion depend slightly on which pyramid, for example in the bottom one:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} a & 0 & x_b & x_q \\ 0 & b & y_b & y_q \\ 0 & 0 & z_b & z_q \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ 1 \end{bmatrix}$$

In any case the resulting surface will be bi-quadratic.

4.3 Curve Lofting

Given a set of curves, each represented by the intersection of two implicit surfaces i.e., $C_{i_0 \le i \le n} : (f_i, g_i)$ where f_i and g_i are surfaces in irreducible implicit form, the surface S which G^k interpolates C_i is known as the G^k *lofted surface* of C_i . The surface S is defined as

$$\alpha_0 f_0 + \beta_0 g_0^{k+1} = \alpha_1 f_1 + \beta_1 g_1^{k+1} = \ldots = \alpha_n f_n + \beta_n g_n^{k+1}$$

where α_i and β_i are polynomials of degree $\le k$. Among the two implicit surfaces to represent a curve C_i , one typically represents the lower degree surface as g_i whereas the other one as f_i to obtain the computational efficiency. This definition can be trivially extended to higher dimensions. Note that if each $C_i \in \mathbb{R}^{n-1}$ the lofted surface is in $\mathbb{R}^n$.

Interpolation of Two Circles

Consider two circles in R^3, of radii 1 and 1, lying on the $x = 1$ and $y = 4$ planes, and with their centers on the x and y axis respectively.

(a) Compute A-spline representation of each circle.

(b) Compute a joining surface that interpolates the circles and contains the origin. Give your answer as an A-patch representation.

(c) Is your solution the lowest degree algebraic surface and with the fewest number of A-patches?

(a) One constructs a scaffold TS around the circle in the following way. Consider the unit circle C_1 whose centre is located at $x = 1$. Choose the bivariate triangle scaffold around each quarter circle defined w.r.t. y and z axes such that the $\angle a_{020}a_{200}a_{002} = \frac{\pi}{2}$ and the scaffold is coplanar with C_1. As a circle is quadratic, there are 3 (including the vertices) equi-spaced control points along each side of TS.

One associates weights to points $a_{020}, a_{011}, a_{002}, a_{101}, a_{200}, a_{110}$. Since C_1 need to interpolate a_{020} and a_{002}, we set $w_{020} = w_{002} = 0$. Since the line segments $a_{020}a_{110}$ and $a_{002}a_{101}$ are tangents to C_1, we set $w_{110} = w_{101} = 0$. Consider the line segment $a_{200}a_{011}$ and let the point of intersection of the circle embedded in the triangle with this line segment be p. From the basic geometry one has, $|\frac{pa_{200}}{pa_{011}}| = \frac{w_{011}}{w_{200}} \Rightarrow |\frac{\sqrt{2}-1}{1-\frac{1}{\sqrt{2}}}| = \frac{w_{011}}{w_{200}} \Rightarrow |\frac{1}{\sqrt{2}}| = \frac{w_{011}}{w_{200}}$. Choose $w_{011} = -1$ and $w_{200} = \sqrt{2}$ to satisfy this equality. This scaffold is guaranteed to yield the one quarter of the circle. The scaffolds for circle quarters in the other quadrants can be constructed symmetrically. For one of these scaffolds, say $a'_{020}a'_{002}a'_{200}$, distinct from TS, since $\angle a_{020}a_{200}a_{002} = \angle a'_{020}a'_{200}a'_{002} = \frac{\pi}{2}$, $a'_{200}a'_{020}$ is collinear with either $a_{200}a_{002}$ or $a_{020}a_{200}$. This helps in keeping the C_1 continuity of the circle segments at the joining points of the generated A-splines.

The A-spline representation of the circle C_2 located in $y = 4$ is symmetric to above except that the scaffolds are constructed in the xz-plane.

(b) The circle C_1 can be represented as an intersection of two implicit surface equations, $f_0 : y^2 + z^2 - 1 = 0$ and $g_0 : x - 1 = 0$. The circle C_2 can be represented as an intersection of two implicit surface equations, $f_1 : x^2 + z^2 - 1 = 0$ and $g_1 : y - 4 = 0$. Then the lofted surface S which interpolates (i.e., with C^0 continutity) the circles C_1 and C_2 is given by, $\alpha_0 f_0 + \beta_0 g_0^{0+1} = \alpha_1 f_1 + \beta_1 g_1^{0+1} \Rightarrow \alpha_0(y^2 + z^2 - 1) + \beta_0(x - 1) = \alpha_1(x^2 + z^2 - 1) + \beta_1(y - 4)$, where $\alpha_0, \beta_0, \alpha_1, \beta_1$ are chosen to be constants to have the lowest degree algebraic surface. Take any four points on the circles C_1 and C_2 to find these constants, ex. $(1,4,0), (0,4,1), (1,1,0), (1,0,1)$. The additional constraint is given by the fact that the origin $(0,0,0)$ need to be interior to the surface. This can be imposed by $\frac{\partial S}{\partial y} < 0$ at $(1,4,0)$.

For example, to define A-patches in tensor domain, consider the smallest cuboid H enclosing the surface S such that S with the planes containing C_1 and C_2 together contain the origin. This can be precisely defined with the intersection of planes on which C_1 and C_2 reside. One partitions H with a set R of hexahedrons such that any line segment joining vertices of any chosen hexahedron in R does not intersect S more than once, and, $|R|$ is the smallest among all possible such sets. For example, this can be achieved by partitioning H with $xz-$ and $yz-$planes, such that to obtain $|R| = 4$. Since S is quadratic, each side of any hexahedron $r \in R$ consists of three control points (including the ver-

tices). Similar to part (a), one associates weights to these control points to construct A-patch within each such r. To maintain C^2 continuity, between adjacent A-patches, one imposes additional constraints.

(c) The equation representing the family of surfaces which G^k continuously interpolates the curves C_1 and C_2 is $\alpha_0 f_0 + \beta_0 g_0^{k+1} = \alpha_1 f_1 + \beta_1 g_1^{k+1}$. Since one is asked to interpolate C_1 and C_2, $k = 0$ has been chosen, which in turn yielded the lowest degree algebraic surface. Since S is of lowest degree and $|R|$ is of smallest size, this solution is the one with the minimum number of A-patches.

4.4 Surface Offset

Offset of a Quadratic Tetrahedral Patch

Consider the normal r-offset surfaces Q_{outer} and Q_{inner} of an algebraic surface patch P inside a tetrahedron, where Q_{outer} is the offset in the positive surface normal direction and Q_{inner} is the offset in the negative surface normal direction by r. If patch P is defined by a quadratic trivariate polynomial equation, give the equation of the Q_{outer} and Q_{inner} surfaces and the patch boundaries within a r-offset (or r-scaled) tetrahedron.

Say, patch P be defined by

$$P : F(x,y,z) = 0 \tag{4.1}$$

then the normal of P at point (x,y,z) is

$$\vec{n} \quad = (n_x, n_y, n_z)^T \quad = (\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z})^T$$

Denote

$$|\,\vec{n}\,| = \sqrt{n_x^2 + n_y^2 + n_z^2}.$$

Let (x', y', z') be the point on Q_{outer}, then one has

$$\begin{cases} x' = x + r\dfrac{n_x}{|\,\vec{n}\,|} \\ y' = y + r\dfrac{n_y}{|\,\vec{n}\,|} \\ z' = z + r\dfrac{n_z}{|\,\vec{n}\,|} \end{cases} \tag{4.2}$$

Eliminate x, y, z from (4.2) and (4.1) to get an equation about x', y', z'. It is the equation of Q_{outer}.

Let $(\tilde{x},\tilde{y},\tilde{z})$ be the point on Q_{inner}, then

$$\begin{cases} \tilde{x} = x - r\dfrac{n_x}{|\vec{n}|} \\ \tilde{y} = y - r\dfrac{n_y}{|\vec{n}|} \\ \tilde{z} = z - r\dfrac{n_z}{|\vec{n}|} \end{cases} \tag{4.3}$$

Eliminate x,y,z from (4.3) and (4.1) to get an equation about $\tilde{x},\tilde{y},\tilde{z}$. It is the equation of Q_{inner}.

Now an example on how to get the equation about x',y',z' is shown.

Let $F(x,y,z) = x^2+y^2+z^2-1$, then

$$\vec{n} \quad = (n_x,n_y,n_z)^T \quad = (\frac{\partial F}{\partial x},\frac{\partial F}{\partial y},\frac{\partial F}{\partial z})^T \quad = (2x,2y,2z)^T$$

$$|\vec{n}| = \sqrt{n_x^2+n_y^2+n_z^2} = 2\sqrt{x^2+y^2+z^2} = 2.$$

Then

$$\begin{cases} x' = x + r\dfrac{n_x}{|\vec{n}|} = x + r\dfrac{2x}{2} = (1+r)x \\ y' = y + r\dfrac{n_y}{|\vec{n}|} = y + r\dfrac{2y}{2} = (1+r)y \\ z' = z + r\dfrac{n_z}{|\vec{n}|} = z + r\dfrac{2z}{2} = (1+r)z \end{cases} \tag{4.4}$$

So

$$x = x'/(1+r), y = y'/(1+r), z = z'/(1+r).$$

Substitute is to $F(x,y,z) = x^2+y^2+z^1-1 = 0$, one gets

$$F'(x',y',z') = (x'^2+y'2+z'^2)/(1+r)^2-1 = 0$$

which is the equation of Q_{outer}.

Similarly we can get the equation of Q_{inner}:

$$\tilde{F}(\tilde{x},\tilde{y},\tilde{z}) = (\tilde{x}^2+\tilde{y}^2+\tilde{z}^2)/(1-r)^2-1 = 0$$

Let the patch P be defined in the tetrahedron $\mathscr{T}:(p_1,p_2,p_3,p_4)$, where $p_1 = (x_1,y_1,z_1), p_2 = (x_2,y_2,z_2), p_3 = (x_3,y_3,z_3), p_4 = (x_4,y_4,z_4)$. Notice that p_1,p_2,p_3 are on the surface Q_{outer}, then

$$x_1^2+y_1^2+z_1^2 = (1+r)^2, x_2^2+y_2^2+z_2^2 = (1+r)^2, x_3^2+y_3^2+z_3^2 = (1+r)^2.$$

The relationship of the barycentric coordinates $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ and (x, y, z) is:

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = M \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{pmatrix}$$

Substitute the above equation into $F'(x', y', z') = 0$ and one gets the equation about the barycentric coordinates

$$\begin{aligned} G'(\alpha_1, \alpha_2, \alpha_3, \alpha_4) &= \left({x_1}^2 + {z_1}^2 + {y_1}^2\right) {\alpha_1}^2 + (2\, y_2\, y_1 + 2\, x_2\, x_1 + 2\, z_2\, z_1)\, \alpha_1\, \alpha_2 \\ &\quad + (2\,(y_3 + y_4)\, y_1 + 2\,(x_3 + x_4)\, x_1 + 2\,(z_3 + z_4)\, z_1)\, \alpha_3\, \alpha_1 \\ &\quad + (2\,(x_3 + x_4)\, x_2 + 2\,(z_3 + z_4)\, z_2 + 2\,(y_3 + y_4)\, y_2)\, \alpha_3\, \alpha_2 \\ &\quad + \left((x_3 + x_4)^2 + (z_3 + z_4)^2 + (y_3 + y_4)^2\right) {\alpha_3}^2 - (1 + r)^2 \\ &\quad + \left({x_2}^2 + {z_2}^2 + {y_2}^2\right) {\alpha_2}^2 \\ &= (1 + r)^2\, {\alpha_1}^2 + (2\, y_2\, y_1 + 2\, x_2\, x_1 + 2\, z_2\, z_1)\, \alpha_1\, \alpha_2 \\ &\quad + (2\,(y_3 + y_4)\, y_1 + 2\,(x_3 + x_4)\, x_1 + 2\,(z_3 + z_4)\, z_1)\, \alpha_3\, \alpha_1 \\ &\quad + (2\,(x_3 + x_4)\, x_2 + 2\,(z_3 + z_4)\, z_2 + 2\,(y_3 + y_4)\, y_2)\, \alpha_3\, \alpha_2 \\ &\quad + \left((1 + r)^2 + 2 x_3 x_4 + 2 z_3 z_4 + 2 y_3 y_4 + x_4^2 + y_4^2 + z_4^2\right) {\alpha_3}^2 \\ &\quad - (1 + r)^2 + (1 + r)^2\, {\alpha_2}^2 \end{aligned}$$

In many applications not only the surface equation must be offset but also the boundary curve and boundary point. When offsetting surface patches that are not C^1 continuous the newly created surfaces will join the offset surfaces. The offset of a curve is obtained by moving a sphere on the curve and it is a surface (a "toroid"). Equivalently one moves a circumference on a plane perpendicular to the tangent of the curve. A curve in space is defined as the intersection of two surfaces. The boundary curves in A-patches are the intersection of the surface equation and the planes containing the faces of the scaffolding tetrahedron. The offset of a tetrahedral patch and the offset of its boundary curves and points is shown in brown in Figure 4.3, and the constructing elements are:

Surface: $f(x, y, z) = 0$ and boundary curves on the tetrahedron (in dark green).

Delimiting planes: $P : ax + by + cz + d = 0$ (in blue outline).

Normal to surfaces: $\overrightarrow{n_f} := \nabla f = (f_x, f_y, f_z)^T$ and $\hat{n_f} = \frac{\nabla f}{|\nabla f|}$ (shown in red at curve segment delimiting points).

Normal to plane: $\overrightarrow{n_P} = (a, b, c)$ and $\hat{n}_P = \frac{(a,b,c)}{\sqrt{a^2+b^2+c^2}}$.

Tangent to boundary curve: $\hat{t} = \hat{n_f} \times \hat{n_p}$. For any α, β one has $(\alpha \hat{n_f} + \beta \hat{n_P}) \cdot \hat{t} = 0$. The circumference will lay on the plane spanned by these two vectors. In particular the binormal is $\hat{b} = \hat{n_f} \times \hat{t}$ (shown in green at delimiting point)

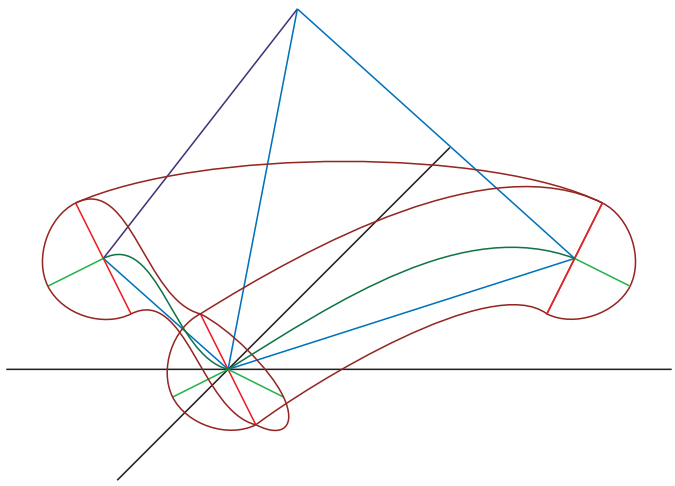


Figure 4.3: Offset of a tetrahedral A-patch. Image created by the author for [Baj07].

To build the scaffold (see figure 4.4) for the offset surface and the offset of the delimiting curve and points for simplicity one will focus on the original A-patch interpolating three points on the tetrahedron. The most natural scaffold is a prismatic scaffold, then one may add a barycentric point in each prism to divide it in 6 tetrahedrons. The main surface positive offset surface Q_{outer} scaffold is build by taking the normal versor $\hat{n}$ at the boundary points and multiply it by two times the offset radius $2r$. Join the other extreme of $2r\hat{n}$ obtaining a prism skew in one direction. The offset surface will pass at the boundary edge exactly at $r\hat{n}$. Similarly one builds the prism for Q_{inner} by inverting the normals (red in figure). The scaffold offset of the boundary curve is obtained by taking the binormal versor $\hat{b} = \hat{n_f} \times \hat{t}$, multiplying it by $2r$ and applying at the original boundary point and the boundary point at the end of $2r\hat{n}$. (green in figure) The scaffold for the offset of the delimiting point is obtained by building an hexahedron on the extended points on the bi-normals. (purple in figure)

Offset of Union of Spheres

Given a union M of n spheres (simple geometric model of a molecule), give an efficient algorithm to generate the r-offset M_r^+ and M_r^- models of M where again M_r^+ is the outer offset and M_r^- the inner offset. What is the relationship of the inner r-offset $(M_r^+)_r^-$ of M_r^+ with M? Provide an algorithm to generate a model of $(M_r^+)_r^-$, that is, the inner offset

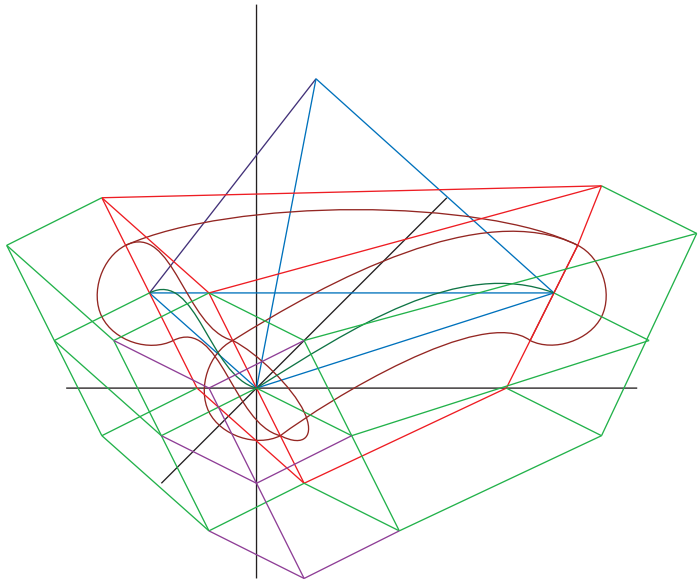


Figure 4.4: Scaffold construction for an A-patches description of tetrahedral A- patch offset. Image created by the author for [Baj07].

of the outer offset of the original union of spheres.

A surface patch decomposition M of a union of n spheres centered at different points $(p_0 = (x_0, y_0, x_0), p_1 = (x_1, y_1, x_1), \ldots p_{n-1} = (x_{n-1}, y_{n-1}, x_{n-1}))$ and with different radii $(r_0, r_1, \ldots, r_{n-1})$ is composed of patches whose components are:

1. Spherical surfaces: $S_i : (x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 - r_i^2 = 0$
2. Boundary curves: these are defined implicitly as the intersection of the spheres $C_{ij} : S_i \cap S_j$ if not empty, but are usually simplified to a plane P_{ij} and a circumference c_{ij} on that plane.
   a) Without loss of generality consider the coordinate system centered in p_i center of the first sphere and the x axis pointing to p_j center of the second sphere. Let d_{ij} be the distance between the two centers. vector.
   b) One has:

$$\begin{array}{c} x^2 + y^2 + z^2 - r_i^2 = 0 \\ (x - d_{ij})^2 + y^2 + z^2 - r_j^2 = 0 \\ \hline -2d_{ij}x + d_{ij}^2 + (r_j^2 - r_i^2) = 0 \end{array}$$

c) Thus the plane (in the local system) is

$$x = h'_{ij} = \frac{d_{ij}^2 + (r_j^2 - r_i^2)}{2d_{ij}}$$

d) Substituting the plane equation in the sphere equation one has the equation of the circumference and its radius:

$$y^2 + z^2 - r_i j^2 = 0$$
$$r'^2_{ij} = r_i^2 - h'^2_{ij} = r_i^2 - \left(\frac{d_{ij}^2 + r_j^2 - r_i^2}{2d_{ij}}\right)^2$$

3. Intersection points of three spheres is a point $p_{ijk} = S_i \cap S_j \cap S_k$, equivalently one may easily intersect the planes of the boundary curves: $p_{ijk} = P_i \cap P_j \cap P_k$.

Once the intersection planes, circumferences and points are calculated one may assemble the surface patch for M (see Figure 4.5).

To calculate an offset one has to r offset each component of the surface patch in the space [BK88][BK90]:

1. The r offset of spherical surface is very simple. The normal to a sphere is always pointing in the direction of the radius, thus, the offset is just the original surface equation with the the added radius:

$$(S_r^+)_i : (x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 - (r + r_i)^2 = 0$$

2. The r offset of a boundary circumference C_{ij} will become torii $T_{C_{ij}}$ with outer radius the same as the boundary circumference r'_{ij} and the inner radius the offset r. It may obtained by rotating on the x' axis the circle $(x' - h'_{ij})^2 + (y' - r'_{ij})^2 - r^2 = 0$ of radius r centered at (h'_{ij}, r'_{ij}):

$$(x - h'_{ij})^2 + (\sqrt{y^2 + z^2} - r')^2 - r^2 = 0$$
$$((x - h'_{ij})^2 + y^2 + z^2 - r^2)^2 + 4h'^2_{ij}(z^2 + y^2)$$

3. The r offset of a boundary point will be a sphere centered at the point p_{ijk} and radius r: $S_{p_{ijk}}$

When calculating the outer offset of the union of spheres M_r^+ the offset of the spherical surfaces will dominate in respect to the torii around the intersection circumference and the sphere centered on the intersection points (See Figure 4.6).

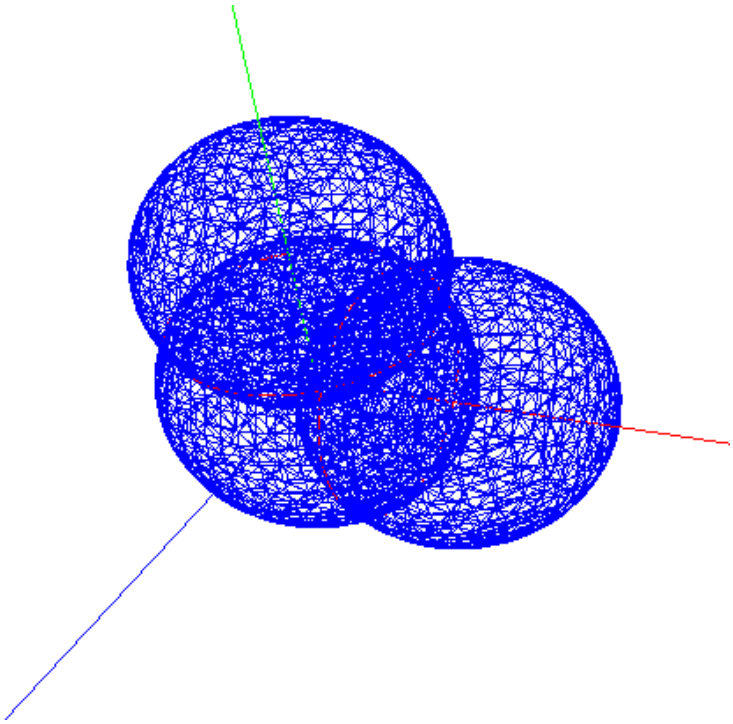


Figure 4.5: Three spheres and intersections. Image generated using GANITH [BR90a].

At this point one has to compute all the components of the outer offset M_r^+. The boundary circles of the surface patch of M_r^+ not only have changed but new intersections may be created. Operating similarly to the original patch, the intersection plane will be:

$$x' = h''_{ij} = \frac{d_{ij}^2 + ((r_j + r)^2 - (r_i + r)^2)}{2d_{ij}}$$

The new intersection circumference $(C_r^+)_{ij}$ will have radius:

$$r''_{ij} = (r_i + r)^2 - h''^2_{ij} = (r_i + r)^2 - \left(\frac{d_{ij}^2 + (r_j + r)^2 - (r_i + r)^2}{2d_{ij}}\right)^2$$

Similarly the intersection points must be recalculated as $(p_r^+)_{ijk}$.

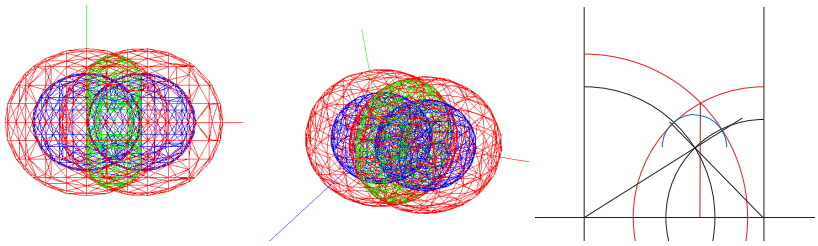


Figure 4.6: Outer offset components of two spheres. Images created by the author for [Baj07], first two using GANITH [BR90a].

When calculating the inner offset $(M_r^+)_r^-$, the inner surface of the torii $T_{(C_r^+)_{ij}}$ created by the boundary circles (r''_{ij}, h''_{ij}) of the outer offset surface M_r^+ will not be hidden but have the role of "smoothing" out the spheres (See Figure 4.7).

The boundary between a spherical surfaces $((S_r^+)_r^-)_i$ and the joining torus $T_{(C_r^+)_{ij}}$ will be a circumference (See Figure 4.8). One may calculate the containing plane and its radius

$$\hat{r_{ij}} = \frac{r_i r''_{ij}}{r_i + r}$$

$$x = \hat{h_{ij}} = \frac{r_i h''_{ij}}{r_i + r}$$

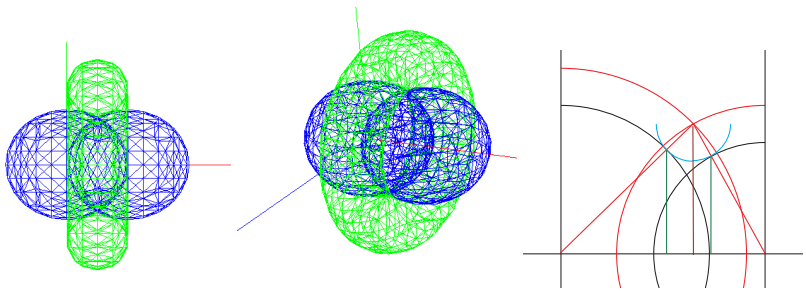


Figure 4.7: Inner offset components of two spheres. Images created by the author for [Baj07], first two using GANITH [BR90a].

Similarly the spheres at the intersection points $(p_r^+)_{ijk}$ will join smoothly the torii (See Figure 4.9).

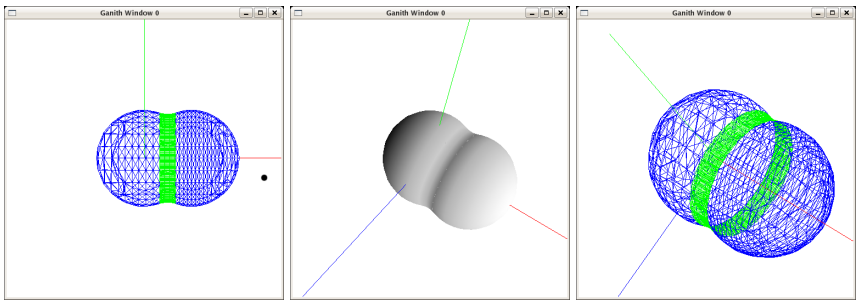


Figure 4.8: Inner offset of two spheres. Images created by the author for [Baj07] using GANITH [BR90a].

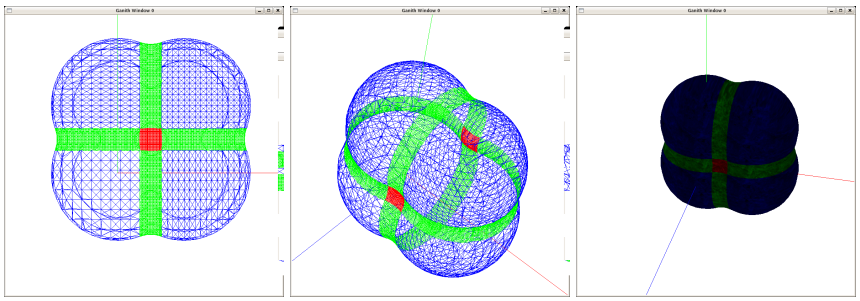


Figure 4.9: Inner offset of four spheres. Images created by the author for [Baj07] using GANITH [BR90a].

Chapter 5

Protein Modeling

In Chapter 4, two examples have been shown on how to compute the union of spheres and then their inner offset of the outer offset. These examples give insight on how to build a molecular model made of spheres, each sphere of radius equal to the Van de Wall (*VDW*) radius. Then on this model the inner offset surface of the outer offset surface of the original model is computed to build the Lee Richard (*L-R*) model of the solvent contact surface. The input data is the *LEG* (Labelled Embedded Graph) representation of a molecule, which is simply an annotated graph representation of the chemical structure of the molecule, in which each node represents an atom and each edge a chemical bond. Each atom may be annotated by its symbol and the *VDW* radius, each edge may be annotated by the length of the corresponding chemical bond and possibly a dihedral angle, and each pair of consecutive edges by a bond angle. Original source of this chapter is an handout from Prof. Chandrajit Bajaj course in Geometric Modeling and Visualization [Baj07].

Geometric Models are used to approximate real-world objects. Locating each molecule and modeling the ensemble of an arrangement of atoms is known as *Explicit Modeling* a.k.a. *Labeled Embedded Graph* (*"LEG/Bone/Skeleton"* or *"Ball-Stick"*) Model. Modeling to analyze density or average distribution of certain species concentration using analytical functions is known as *Implicit Modeling* a.k.a. *Interface Model* a.k.a. *Boundary Representation* (*"Skin"* or *"Spatial Occupancy"*) Model. These (geometric) models are useful not only to visualize, but to do extensive computations. For example, level sets give the subdomain in which the protein/ion density' is greater than certain threshold. This yields an analytical function, which facilitates in visualizing using splines/patches. The conversion of an Explicit Model to an Implicit Model is known as *Mean Field Approximation*. The appropriate model is chosen depending on the context.

Labeled Embedded Graph Model

The *LEG* model captures secondary, tertiary, quaternary structures from the input primary structure. This model is a discrete/combinatorial model. In this model, the geometric molecule of the structure is represented as a graph, $G(V,E,F)$. Here the set V represents the set of discrete points (vertices) on the surface/volume such that the points in V are interpolated by the geometric model. The set E represents the set of edges/curves joining any two vertices belonging to V . The set F represents the two dimensional regions with boundaries, wherein the boundaries belong to E. Note that the faces F require to be precisely (unambiguously) defined, especially whenever E comprise curves.

The *Protein Data Bank (PDB)* is a repository for 3-D structural data of proteins and nucleic acids. This data, typically obtained by *X-ray crystallography* or *NMR spectroscopy*, is submitted by biologists and biochemists from around the world, is released into the public domain, and can be accessed for free. The database stores information about the exact location of all atoms in a large biomolecule, type of each atom, and the error bound associated with the location information. Typically, the information about the primary structure of a protein (or a nucleic acid, or a saccharine) is obtained from PDB, and, then the geometric models are constructed.

The X-ray crystallography is used to determine the 3-D structure of biological entities. It determines the arrangement of atoms within a crystal from the manner in which a beam of X-rays are scattered/diffracted from the electrons within the crystal. The method produces a three-dimensional picture of the density of electrons within the crystal, from which the mean atomic positions, their chemical bonds, their disorder and several other information is derived in terms of the intensity (or grayscale) of pixels of the volumetric image. As the derivatives exist at these pixels, we can fit splines (ex. bilinear) with C^1-continuity and visualize the 3-D structure. The computational and experimental difficulties are the major bottlenecks for the X-ray crystallography. The Fourier transform decomposes a function into a continuous spectrum of its frequency components, and the inverse transform synthesizes a function from its spectrum of frequency components. These transforms are used in the reconstruction of the model from the image ex. to filter the noise in the X-ray crystallography. A 3D-image Fourier transform can be determined from a sequence of 2D-image Fourier transforms, whose inverse in turn produces the image in 3D-space.

The nuclear magnetic resonance (NMR) refers to a family of scientific methods that exploit nuclear resonance to study molecules. The NMR spectroscopy is used in understanding the protein and nucleic acid structure and function at an atomic resolution. This technique is applicable to a wide variety of samples, both in the solution and the solid state; however, the technique is limited to small structures.

Interface Model

The H_2O, Na^+, Cl^-, Ca^{2+} are ions imperative for the survival of animals and they typically exist in the native environment. For example, Na^+, Cl^- exist because *NaCl* dissolves easily as we increase the temperature. The solvation is the attraction and association of molecules of a solvent with molecules or ions of a solute. As ions dissolve in a solvent they spread out and become surrounded by solvent molecules. The bigger the ion, the more solvent molecules are able to surround it and the more it becomes solvated. The interface between a protein and a water molecule is captured with an interface model (Fig. 5.1). The interface model separates interface of one molecule from another. This model is a continuum geometric model. The convolution of a sphere representing a water molecule W with the boundary of the union of the arrangement of soluble molecules, known as *CPK*, results in the *outer envelope* a.k.a. *Solvent Accessible Surface (SAS)* a.k.a *+ve offset* of W w.r.t. CPK, which is the locus of the W around CPK. This envelope is known as outer envelope, because it is exterior to CPK. Similarly, *inner envelope* a.k.a. *Solvent Excluded/Contact Surface (SES/SCS)* a.k.a. *-ve offset* of W w.r.t. CPK is the envelope interior to CPK, wherein it is obtained by rolling W along the CPK. The envelope CPK is known as *van der Walls* surface (VDW) (this is because spheres of soluble molecules are approximated with van der Walls radii). The model of molecules which does not consider the environment in which those molecules are embedded is known as the Face Occupancy Model a.k.a. CPK model, whereas the solvent environment is modeled with the *Lee-Richard (LR) model* (Fig. 5.2). Offsets of interface model may contain singularities. For example, two water molecules in a solvent environment may protrude between two molecules of a solvent to cause a singularity.

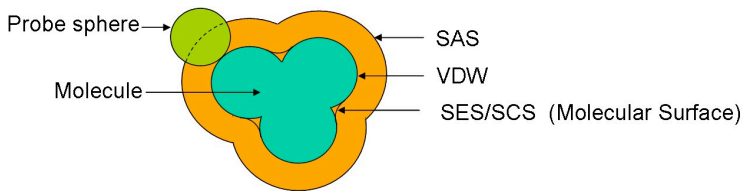


Figure 5.1: Molecules in solvent [Baj07].

The *Hard-Sphere kernel* represents each atom by a sphere. This model considers that the electron density at a point **p** is unity as long as **p** is within the sphere defined by the union of electron clouds of that atom. Instead of representing the probability of finding an electron within a unit sphere as one as in Hard-Sphere kernel, the Gaussian kernels and cardinal B-spline kernels model the real-world situations more aptly. For a molecule

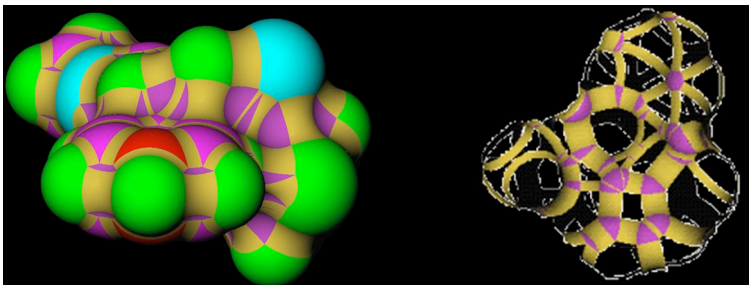


Figure 5.2: Lee Richard (LR) Model [Baj07].

with M atoms, we can define a synthetic electron density function as

$$f(\mathbf{x}) = \sum_{i=1}^{M} G_i(\mathbf{x}),\ \mathbf{x} \in \mathbb{R}^3$$

Molecular surface for linear decay kernels is

$$f(\mathbf{x}) = \sum_{i=1}^{M} e^{\beta + \frac{\beta}{r_i^2}(\mathbf{x}-\mathbf{x_i})^2} \delta(\mathbf{x}-\mathbf{x_i})$$

Molecular surface for quadratic decay kernels is

$$f(\mathbf{x}) = \sum_{i=1}^{M} e^{\beta(r_i + |\mathbf{x}=\mathbf{x_i}|)} \delta(\mathbf{x}-\mathbf{x_i})$$

Note that in these kernel functions, when $\mathbf{x} = \mathbf{x_i}$, the influence of other atoms is nullified. Radial basis spline kernel is obtained by the surface of revolution of a smooth B-spline curve such that. the x-axis is a tangent to that curve at $x = 1$. Radial basis spline kernel has advantages over the Gaussian kernel because of its decay pattern i.e., the Gaussian kernel meets x-axis at infinity whereas the radial basis spline meets x-axis at a finite distance. These models are particularly useful in modeling the electron density functions, which has natural decay (See Fig. 5.3). The surface of an LR model of SES/SCS consists of convex spherical, toroidal, and concave spherical patches, which are represented with either A-patches or NURBS.

Summary

Given a *LEG* representation of a protein P, this chapter will, in the respective sections:

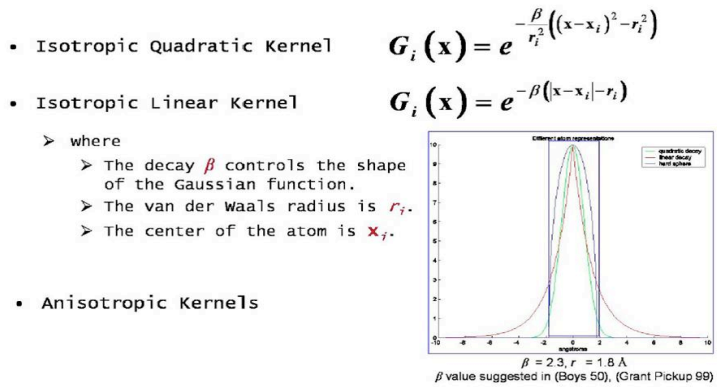


Figure 5.3: Hard-Sphere/Gaussian/Radial basis spline kernels [Baj07].

(1) Key mathematical properties of a molecule.

(2) Describe an algorithm to compute the *VDW* (union-of-hard-spheres) surface of *P* .

(3) Describe how to construct the *L-R* molecular surface of the protein *P*. To do so an algorithm is needed to detect all solvent exposed atoms of *P* and to detect where two or three of these exposed atoms intersect. To ensure the quality of the model a method is needed to detect where if at all, the *L-R* surface, self intersects.

The overall algorithm will have a time complexity of $O(n\log(n))$ time, where n is the number of atoms in the protein. A similar algorithm is implemented in TexMol, a molecular visualization software developed at CVC (Center for Computational Visualization) of the ICES (Institute of Computational Engineering and Sciences) at the University of Texas at Austin. This is the software used to create the surface models using prism A-patches of Chapter 8 on which the boolean operators will be tested in Chapter 9.

5.1 Properties

A couple of properties of a molecule (described in [HO94]) will be exploited to design efficient algorithms for manipulating the "union-of-sphere" model of the molecule (e.g., for computing the molecular surface). In the worst case, the arrangement defined by n balls in 3-space (i.e., the subdivision of 3-space into cells of dimensions 0, 1, 2, and 3, defined by the balls) may have $O(n^3)$ combinatorial complexity, the boundary defined

by their union may have complexity $O(n^2)$. However, the balls defining the atoms in the "union-of-sphere" model of a molecule have the following two properties which allows for more efficient and simpler algorithms for manipulating them:

- The centers of two balls cannot get too close to each other.
- The range of radii of the balls is fairly restricted (e.g., see Table 5.1 below).

Table 5.1: Radii of balls used to represent different types of atoms [HO94].

C	Cal	H	N	O	P	S
1.52 $\mathring{A}$	3.48 $\mathring{A}$	0.70 $\mathring{A}$	1.36 $\mathring{A}$	1.28 $\mathring{A}$	2.18 $\mathring{A}$	2.10 $\mathring{A}$

The following theorem, proved in [HO94], gives a couple of useful consequences of the two properties listed above.

Theorem 5.1.1. *(Theorem 2.1 in [HO94]). Let $M = \{B_1, ..., B_n\}$ be a collection of n balls in 3-space with radii $r_1, ..., r_n$ and centers at $c_1, ..., c_n$. Let $r_{min} = min_i\{r_i\}$ and let $r_{max} = max_i\{r_i\}$. Also let $S = \{S_1, ..., S_n\}$ be the collection of spheres such that S_i is the boundary surface of B_i. If there are positive constants k, ρ such that $\frac{r_{max}}{r_{min}} < k$ and for each B_i the ball with radius ρr_i and concentric with B_i does not contain the center of any other ball in M (besides c_i), then:*

(i) For each $B_i \in M$, the maximum number of balls in M that intersect it is bounded by a constant.

(ii) The maximum combinatorial complexity of the boundary of the union of the balls in M is $O(n)$.

Table 5.2 lists the values of k, ρ and the maximum and average number of balls intersecting any given ball in various molecules [HO94]. As the table shows, k is quite small and ρ is closer to 1 than 0, resulting in a small number of intersections per ball.

Given a "union-of-ball" representation of a molecule, Theorem 5.1.1 can now be used to design an efficient data structure that can answer intersection queries with either a point or with a ball whose radius is bounded by a r_{max} .

Table 5.2: Values of k, ρ and the maximum and average number of balls intersecting a single ball in various molecules [HO94].

molecule	k	ρ	maximum number of balls intersecting a given ball	avgerage number of balls intersecting a given ball
caffine	2.17	0.71	10	4.5
acetyl	3.11	0.67	16	5.4
crambin	1.64	0.78	10	5.5
felix	1.64	0.81	9	4.9
SuperOxide Dismutase	1.95	0.76	16	5.5

An Efficient Intersection Query Data Structure (from [HO94]). Let M be the set of n balls as defined in Theorem 5.1.1. We subdivide the entire 3-space into axis-parallel cubes of size $2r_{max} \times 2r_{max} 2r_{max}$ each. For each $B \in M$, we compute the grid cubes that B intersects. Let C be the set of non-empty grid cubes. Since each ball can intersect at most 8 grid cubes, the size of C is bounded by $O(n)$. Also observe that according to Theorem 5.1.1, each cube can be intersected by at most a constant number of balls. The cubes in C are stored in a balanced binary search tree ordered lexicographically by the bottom-left-front vertices of the cubes. With each cube we store the list of $O(1)$ balls of M that intersects it.

Now given a query ball Q, one computes all (at most 8) grid cubes it intersects, and search for each of these cubes in the binary search tree. For each such cube that exists in the search tree, we check the balls stored in it for intersection with Q. Each search will take $O(\log(n))$ time, and the total number of balls tested will be $O(1)$. Hence, one has the following theorem.

Theorem 5.1.2. *(Theorem 3.1 in [HO94]). Given a collection M of n balls as defined in Theorem 5.1.1, one can construct a data structure using $O(n)$ space and $O(n\log(n))$ preprocessing time, to answer intersection queries for balls whose radii are not greater than r_{max}, in time $O(\log(n))$.*

5.2 *VDW* Molecular Surface

One first convert the *LEG* representation of P to the "union-of-spheres" representation, and then compute its boundary surface.

LEG to "Union-of-Spheres" Conversion.

For each ball B_i in the union we need to compute its center c_i and radius r_i. The r_i value is simply the van der Waals (*VDW*) radius of the atom, and can be obtained from various sources (e.g., [GG99], see also Table 5.1). The *LEG* representation itself might be annotated with the *VDW* radius of each atom, However, since *VDW* radii are not standardized, values obtained from different sources might differ slightly. The c_i values can be computed easily using the internal coordinates (i.e., bond lengths, bond angles and dihedral angles) specified in the *LEG* representation. For example, we can choose the N atom on an arbitrary peptide plane (see Figure 5.4) of the protein, and put the atom (i.e., its center) at the origin. The C_α atom connected to the N atom is placed at distance $1.45\mathring{A}$ from the origin along the positive x-axis. The H atom connected to the N atom is then placed on the xy plane using the bond length $N-H = 1\mathring{A}$ and the bond angle $C_\alpha - N - H = 118.2^\circ$. After the peptide plane containing these three atoms is fixed, it is straight-forward to compute the coordinates of the remaining atom centers using the given internal coordinates.

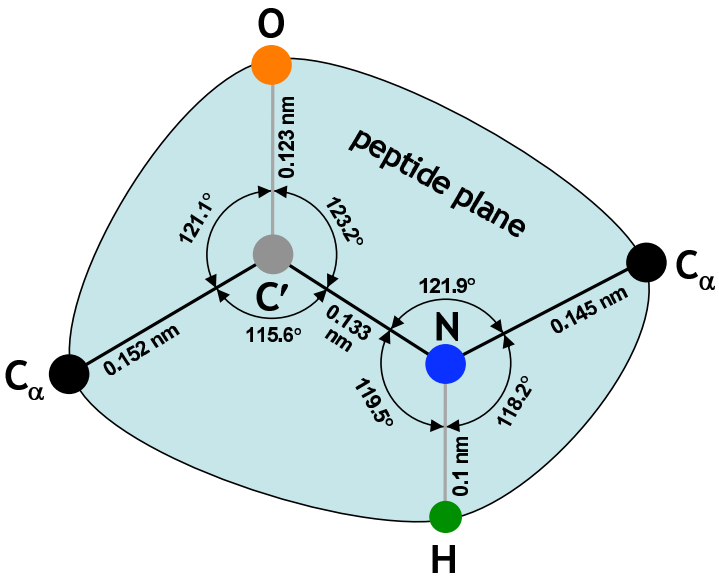


Figure 5.4: A peptide plane with all bond lengths and bond angles shown [GG99].

Computing the *VDW* Surface [HO94].

Given a collection M of balls as defined in Theorem 5.1.1, the algorithm proceeds in the following three steps:

(1) For each $B \in M$, compute the balls in $M \setminus \{B\}$ intersecting it.

(2) Using the information generated in step 1, compute the (potentially null) contribution of each ball $B \in M$ to the union boundary.

(3) Transform the local information generated in step 2 into global structures describing the required connected component of the union boundary.

Each step is described in more details below.

Step 1 The intersection query data structure described earlier is used (Theorem 5.1.2). Each intersection query takes $O(\log(n))$ time, and hence the total cost of this step is $O(n \log n)$.

Step 2 Let B_i be a ball, and one wants to compute its contribution to the union boundary. It is know from Theorem 5.1.1 that at most a constant number of other balls intersect B_i. Let B_j be such a ball, if any. Consider the following three cases:

(i) If B_j fully contains B_i, stop processing B_i as it cannot contribute to the union boundary.

(ii) If B_i fully contains B_j , simply ignore B_j.

(iii) If neither of the two cases above holds, compute the intersection between S_i and S_j, which is a circle C_{ij} on S_i (and S_j). This circle splits S_i into two parts: one part is completely contained within S_j and hence cannot contribute to the union boundary, and the other part which is called the free part of S, may actually appear on the union boundary.

After the process above is repeated for every ball intersecting B_i, one gets a collection of circles on S_i. These circles form a 2D arrangement A_i on S_i. A face of A_i belongs to the union boundary iff it is on the free part defined by each such circle. Since the number of such circles is $O(1)$, A_i can be computed in $O(1)$ time using brute force. Within the same bound one can mark each face on A_i as free or not free. A free face is guaranteed to appear on the union boundary. Since the above procedure for B_i takes $O(1)$ time, the total time complexity of this step is $O(n)$.

Step 3: In this step, the outer connected component of the union boundary of M is represented using a graph data structure. In order to do so, each arrangement A_i is augmented slightly as follows. If A_i is the whole sphere S_i, it is split into two parts using some circle. Next, if a boundary component of A_i is a simple circle C, then C is split into two arcs by adding two new vertices. If C belongs to two arrangements A_i and A_j, the same two vertices are added to both arrangements. Finally, if a free face of A_i contains holes, the face is split into (sub)faces by adding extra arcs so that none of (sub)faces contains any holes in it. All these additions are made canonical by fixing a direction d, and adding all extra edges along great circles that are intersections of S_i with planes parallel to the direction d. This step can easily be performed in $O(n)$ time, and after this step the union boundary will consist of only simple faces that are bounded by at least two edges, and each edge will be shared by exactly two faces (assuming general position). Now one computes the *VDW* surface (i.e., the outer union boundary) of M and store it as a graph $G = (V,E)$ (which is initially empty) as follows. First, find the ball $B_{max} \in M$ with the point having the largest z-coordinate. Let f_{max} be the face of B_{max} that contains this point. Then clearly f_{max} belongs to the outer union boundary of M. Now starting from this face f_{max} traverse the entire connected component containing f_{max} using depth-first search. Each free face f encountered during this traversal will be made a vertex $v_f \in V$, and each arc shared by two such faces f_1 and f_2 will be made an edge $(v_{f_1}, v_{f_2}) \in E$. Every time one encounters a free face f which has not been visited before, one must determine its boundary (i.e., the edges bounding this face), which can be done in $O(1)$ time. For each such edge e of f, one can find the face f' that shares e in $O(1)$ time. If f' is a free face and has not been visited before, recursively visit f'. After one has visited each free face reachable from f_{max} once, G contains the *VDW* surface of M . Clearly, this traversal takes $O(1)$ time.

Hence, the following theorem follows.

Theorem 5.2.1. *(Theorem 4.1 in [HO94]). The* VDW *surface of the union of a connected collection of balls as defined in Theorem 5.1.1 can be computed in $O(n\log(n))$ time and $O(n)$ space.*

5.3 *L-R* Molecular Surface

Solvent Exposed Atoms

Increase the radius of each atom in P by r_s , where r_s is the radius of a solvent atom. Clearly, the collection of these enlarged atoms still satisfies the requirements of Theorem 5.1.1, and the theorem holds. Hence, one can find all atoms in this collection that contribute to the outer union boundary in $O(n\log n)$ time and $O(n)$ space using the same algorithm as in previous section. This gives an algorithm to construct the intersection query data structure described earlier for the set of enlarged atoms in P . If P contains n

atoms, this construction takes $O(n \log n)$ time and uses $O(n)$ space. The algorithm also identifies all solvent exposed atoms. Now, for any ball B_i representing a solvent exposed atom in P , one can find the set T_i of $O(1)$ other solvent exposed atoms that intersects it in $O(\log(n))$ time. Since the size of the set $Ti \cup \{B_i\}$ is bounded by a constant, one can detect in $O(1)$ time where two or three of the atoms in this set intersect. One can identify all such intersections in $O(n \log(n))$ time and $O(n)$ space by using the same process as above for each solvent exposed atoms in P .

L-R Surface

The *L-R* surface of a molecule M with respect to a solvent atom B of radius r is the inner envelope of the region described by B rolling on the *VDW* surface B of M in all possible directions [BLMP97]. This surface can be decomposed into a collection of three kinds of patches: convex spherical, toroidal and concave spherical (see Figure 5.5).

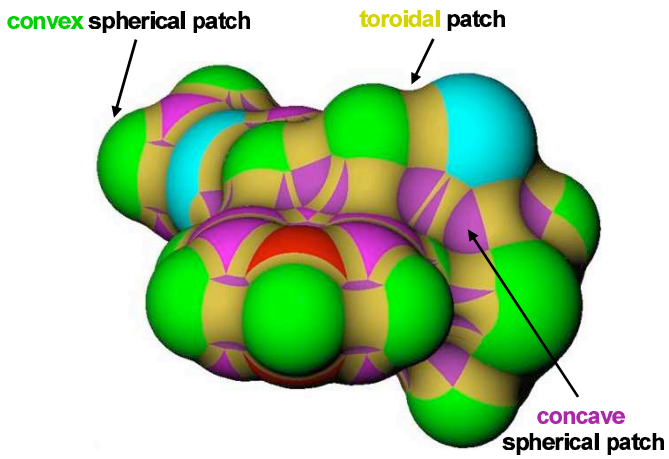


Figure 5.5: 3D image showing the decomposition of the *L-R* surface into three different kinds of patches: convex spherical, toroidal and concave spherical [Baj07].

Convex Spherical Patches. A convex spherical patch is formed when the rolling solvent atom B is in contact with only one atom $B_i \in M$, and it is the maximal connected set of points on the spherical surface S_i of B_i that B touches in this manner. In order to find the extent of the spherical patch on S_i that belongs to the *L-R* surface, simply increase the radius of all balls within distance $2_{rmax} + r$ of B_i

by r (by Theorem 5.1.1 there are only a constant number of them), and use the method in step 2 of previous section.

Toroidal Patches. A toroidal patch is formed when the rolling solvent atom B (of radius r) is in touch with the spherical surfaces S_1 and S_2 of two solute atoms. First one needs to increase the radius of S_1 by r and compute its intersection circle l_2 with S_2 , and similarly increase the radius of S_2 by r and compute its intersection l_1 with S_1. Then, by moving B along the intersection of S_1 and S_2 , it will keep in touch with the two spheres along l_1 and l_2, respectively, and the inward facing arc of B will sweep a torus (see figure 5.6). Other atoms intersecting with S_1 and S_2 may split this torus into several toroidal patches.

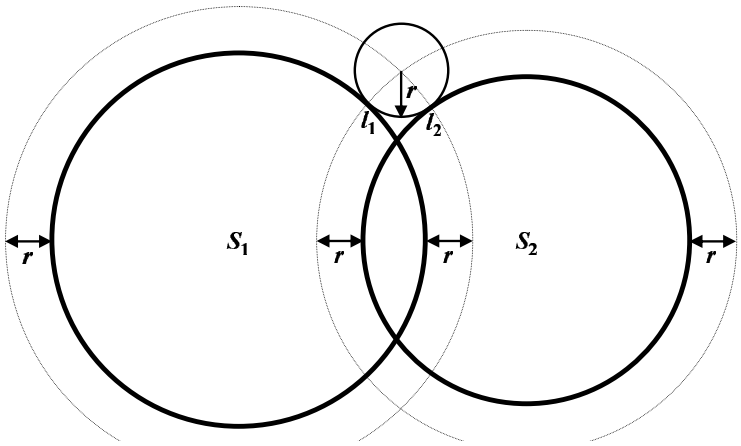


Figure 5.6: The solvent atom of radius r sweeps a torus if it is moved in such a way that it is always in touch with the spheres S_1 and S_2 . The line l_1 (l_1) along which it keeps in touch with S_1 (resp. S_2) can be found by increasing the radius of S_2 (resp. S_1) by r and computing its intersection with S_1 (resp. S_2) [Baj07].

Concave Spherical Patches. A concave spherical (triangular) patch is formed when the rolling solvent atom B simultaneously touches three atoms. The three contact points define a spherical triangle on the surface S of B whose edges are arcs of great circles on S. This triangle is a concave spherical patch on the *L-R* surface.

Self Intersection

To ensure the quality of the surface model one must know how the three different types of patches (i.e., convex spherical, toroidal and concave spherical) on an *L-R* surface can

intersect one another, and how to detect them.

Convex Spherical Patches. It has been shown in [BLMP97] (see Lemma 3) that the convex spherical patches cannot intersect any other part of the *L-R* surface.

Toroidal Patches. It has been shown in [BLMP97] that two different toroidal patches cannot intersect each other (see Lemma 4 in [BLMP97]), and also that a toroidal patch cannot intersect another convex/concave spherical patch (see Lemma 5 in [BLMP97]).

A toroidal can only intersect itself. As shown in Figure 5.7, a toroidal patch intersects itself when it can be constructed as a rotational surface of an arc of a circle around an axis that intersects the arc.

Concave Spherical Patches. It has been shown in [BLMP97] that a concave spherical patch cannot intersect itself (see Lemma 6 in [BLMP97]), or another concave patch (see Lemma 7 in [BLMP97]), or a toroidal patch (see Lemma 5 in [BLMP97]).

As shown in Figure 5.8 two distinct concave patches can intersect each other. Since each concave patch is a part of a sphere, we can detect this type of intersections easily by solving sphere-sphere intersection problems.

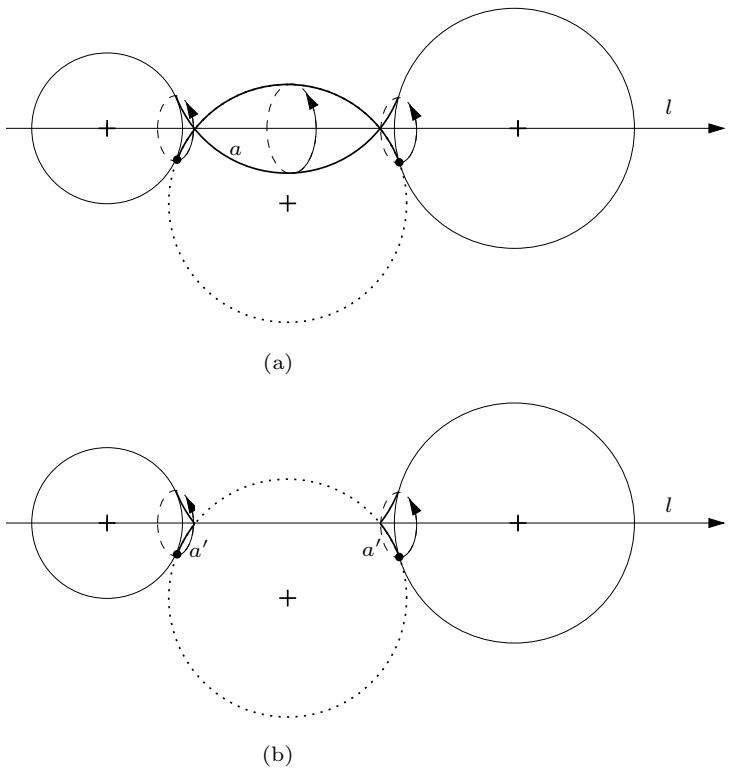


Figure 5.7: (a) The arc a rotating around the axes l describes a self intersection portion of torus. (b) The arc a' rotating around the axes l describes portion of torus with no self intersection [Baj07].

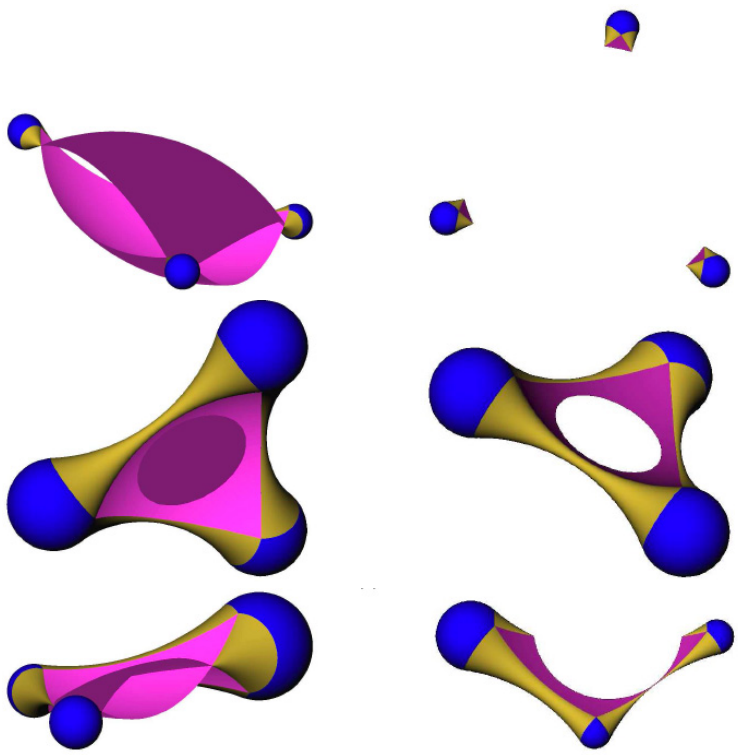


Figure 5.8: Three possible self-intersecting *L-R* surfaces for different radii of the solvent and molecule atoms. On the left the self-intersecting *L-R* surfaces are shown. On the right the corresponding solvent contact surfaces are shown (without self-intersections) [Baj07].

Chapter 6

G^k-continuous A-Splines in a Triangular Domain

In this chapter a detailed description including the calculation of the defining polynomial coefficients for arbitrary degree polynomial interpolating A-splines on triangular domain are given. Content of this chapter has been extracted and constitute a reduced form of [BX99a]. The chapter is structured in section: (1) main definitions are recalled and the notation homogenized; (2) sufficient conditions for regularity and conditions for C^0 and C^1 interpolation are given; (3) general conditions are worked out for G^{2n-3} continuity of degree n A-splines; (4) as an example the case of G^3 continuity for cubic A-splines is fully developed.

6.1 Notation

Let $f(x,y)$ be a bivariate polynomial of degree n with real coefficients, and $\mathbf{p_1}, \mathbf{p_2}, \mathbf{v_1}$ be three affine independent points in the xy-plane. Then the transform

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \mathbf{p_1} & \mathbf{p_2} & \mathbf{p_3} \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} \tag{6.1}$$

maps $f(x,y)$ into its barycentric form $F(\alpha_1,\alpha_2,\alpha_3) = f(x,y)$ on the triangle $[\mathbf{p_1}\mathbf{p_2}\mathbf{v_1}]$, where $0 \le \alpha_i \le 1$ and $\alpha_1+\alpha_2+\alpha_3 = 1$. In the barycentric coordinate system, $F(\alpha_1,\alpha_2,\alpha_3)$ can be expressed in B.B. form [Far01, Gol02] (see figure 6.1(right))

$$F(\alpha_1,\alpha_2,\alpha_3) = \sum_{i+j+k=n} b_{ijk} B_{ijk}(\alpha_1,\alpha_2,\alpha_3), \tag{6.2}$$

where

$$B_{ijk}(\alpha_1,\alpha_2,\alpha_3) = \frac{n!}{i!j!k!}\alpha_1^i\alpha_2^j\alpha_3^k \tag{6.3}$$

Let $\mathbf{p_1}$, $\mathbf{v_1}$ and $\mathbf{p_2}$ be three affine independent points in the xy-plane (see figure 6.1(left)) Then we consider the two line segments $[\mathbf{p_1v_1}]$ and $[\mathbf{v_1p_2}]$ as a segment of a polygon, denoted by $\widehat{\mathbf{p_1,v_1,p_2}}$. We shall consider $\mathbf{v_1}$ as a controller and $\mathbf{p_1}$ and $\mathbf{p_2}$ as interpolation points. An arbitrary polygon chain consists of a sequence of consecutive polygon segments denoted by $\{\widehat{\mathbf{p_iv_iv_{i+1}}}\}_{i=0}^m$. A polygon $\{\widehat{\mathbf{p_iv_iv_{i+1}}}\}_{i=0}^m$ is said to be type G^1 (see figure 6.1(left)) if

$$(\mathbf{v_1}-\mathbf{p_{i+1}}) = \alpha_i(\mathbf{v_{i+1}}-\mathbf{p_{i+1}})$$

One way to construct the polygon from sampled points and normals is to set $\mathbf{p_1}$ and $\mathbf{p_2}$ as interpolation points and $\mathbf{v_1}$ as the intersection point between the (extended) tangents $\mathbf{t_1}$ and $\mathbf{t_2}$ at $\mathbf{p_1}$ and $\mathbf{p_2}$. As it'll be shown this will guarantee G^1 continuity with the imposed tangents. In free form A-spline drawing a trivial choice is to ask the user to insert a vertex per edge in the polygon.

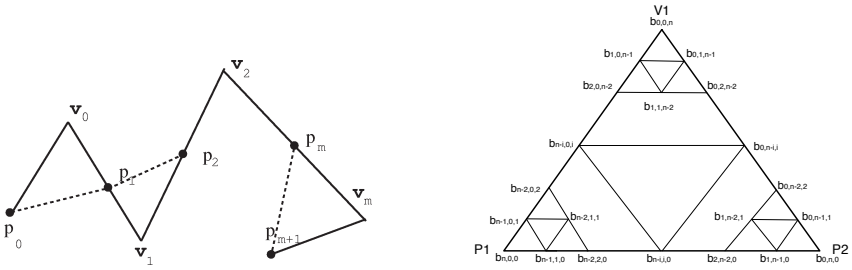


Figure 6.1: Left: a G^1 polygon [BX99a]. Right: Bézier coefficients.

6.2 Regularity, Single Sheet-ness, G^0 and G^1 Continuity

Let $F(\alpha_1,\alpha_2,\alpha_3)$ be defined as (6.2) on the triangle $[\mathbf{p_1p_2v_1}]$. Since there is a constant multiplier to the equation $F(\alpha_1,\alpha_2,\alpha_3)=0$, one may assume $b_{00n}=-1$ if $b_{00n}\neq 0$. Theorems in this section are from [BX99a].

Theorem 6.2.1. *For the given polynomial $F(\alpha_1,\alpha_2,\alpha_3)$ defined as* (6.2)*, if there exists an integer K $(0\le K\le n)$ such that (see figure 6.1(right))*

$$\begin{array}{lll} b_{ijk}\ge 0 & for & j\in[0,\ldots,n-k]; k\in\{0,\ldots,K-1\}, \\ b_{ijk}\le 0 & for & j\in[0,\ldots,n-k]; k\in\{K+1,\ldots,n\} \end{array} \tag{6.4}$$

and

$$\sum_{j=0}^{n} b_{n-j,j,0} > 0 \ , \ \sum_{j=0}^{n-k} b_{n-j-k,j,0} < 0$$

for at least one k ($K < k \leq n$), then for any β that $0 < \beta < 1$ the straight line

$$(\alpha_1, \alpha_2, \alpha_3)(t) = (1-t)(\beta, 1-\beta, 0) + t(0,0,1) \tag{6.5}$$

that passes through v_1 and $\beta\mathbf{p_1} + (1-\beta\mathbf{p_2})$, intersects the curve $F(\alpha_1, \alpha_2, \alpha_3) = 0$ one and only one time in the interior of the triangle $[\mathbf{p_1}\mathbf{p_2}\mathbf{v_1}]$.

Proof of this theorem is a simple application of the variation diminishing property polynomials in B.B. form. [Far01]. Detailed proof of this and other theorems in this chapter may be found in [BX99a]. This theorem guarantees that there is one and only one curve segment of $F(\alpha_1, \alpha_2, \alpha_3) = 0$ within the triangle. Notice that substituting (6.5) into $F(\alpha_1, \alpha_2, \alpha_3) = 0$ one has an expression $F(t, \beta) = 0$. For each given β (a coordinate on the base $[\mathbf{p_1}\mathbf{p_2}]$ of the triangle $[\mathbf{p_1}\mathbf{p_2}\mathbf{v_1}]$), one may solve the equation $F_\beta(t) = 0$, and compute the corresponding point on the curve.

Furthermore, if $b_{n00} = 0, b_{0n0} = 0, b_{00n} = 0$ then the curve will pass through $\mathbf{p_1}, \mathbf{p_2}, \mathbf{v_1}$ respectively. We are interested in interpolating $\mathbf{p_1}$ and $\mathbf{p_2}$ while one sets conventionally $b_{00n} = -1$. The next theorem further describe the properties on the boundary of the triangle and the smoothness of the curve $F(\alpha_1, \alpha_2, \alpha_3) = 0$.

Theorem 6.2.2. *Let $F(\alpha_1, \alpha_2, \alpha_3)$ be defined as in theorem 6.2.1, then*

i The curve $F(\alpha_1, \alpha_2, \alpha_3) = 0$ is smooth in the interior of the triangle $[\mathbf{p_1}\mathbf{p_2}\mathbf{v_1}]$.

ii If one further assumes $b_{n-k,0,k} = 0$ for $k \in \{0, \ldots, K\}$, $b_{n-(K-1),0,K+1} < 0$, and $b_{n-1,1,0} > 0$, then the curve in the triangle passes through $\mathbf{p_1}$, tangent with the line $\alpha_2 = 0$ with multiplicity $K+1$ at $\mathbf{p_1}$ and no other intersection for $\alpha_2 = 0$ for $\alpha_1 > 0$ and $\alpha_3 > 0$. Similarly if $b_{0,n-k,k} = 0$ for $k \in \{0, \ldots, K\}$, $b_{0,n-(K+1),K+1} < 0$, and $b_{1,n-1,0} > 0$, then the curve passes through $\mathbf{p_2}$, tangent with the line $\alpha_1 = 0$ with multiplicity $K+1$ at $\mathbf{p_2}$ and no other intersection for $\alpha_1 = 0$ for $\alpha_2 > 0$ and $\alpha_3 > 0$.

iii If $b_{n,0,0} = b_{n-1,0,1} = b_{n-1,1,0} = 0$, then $\mathbf{p_1}$ is a singular point of the curve. Similarly, if $b_{0,n,0} = b_{1,n-1,0} = b_{0,n-1,1} = 0$, then $\mathbf{p_2}$ is a singular point of the curve.

In the general case for $n \geq 4$ the convexity of the curve is an open problem. However in the cubic case with the curve segment tangent to the sides of the triangle at p_1 and p_2 (i.e. a G^1 spline as in theorem 6.2.2), then it is convex.

Theorem 6.2.3. *The cubic A-spline defined as 6.2.2(ii) has no inflection point inside its reference triangle.*

6.3 G^k Continuity

In this section it will be assumed that the A-spline control data is given as a polygon $\{\widehat{\mathbf{p_i v_i v_{i+1}}}\}_{i=0}^{m}$ in the plane in order to achieve G^1 continuity. The G^k continuity is achieved by: (1) computing the first k terms of the local power series expansion at the join points p_i; (2) determining the coefficients b_{ijk} such that F=0 has the same first k terms of the local power series at the join points. This will give a reoccurrence formula to compute the coefficients b_{ijk} from the coefficients of power series. The algorithm presented in this section is from [BX99a].

The coefficients of the power series may be in turn be given as a discrete data or computed from a parametric or implicit curve to be approximated. Often, these, have to be converted from the global coordinate system to the barycentric coordinates in the triangle.

Consider a two segment A-spline curve

$$F_l(\alpha_1,\alpha_2,\alpha_3) = \sum_{i+j+k=n} b_{ijk}^{(l)} B_{ijk}(\alpha_1,\alpha_2,\alpha_3) = \tilde{b}_{ijk}^{(l)} \alpha_1^i, \alpha_2^j \alpha_3^k = 0$$

On triangles $[\mathbf{p_1^{(l)} p_2^{(l)} v_1^{(l)}}]$ for $l = 1,2$ with $\mathbf{p_1^{(1)}} = \mathbf{p_2^{(2)}}$ as join point (see figure 6.2), where $\tilde{b}_{ijk}^{(l)} = \frac{n!}{i!j!k!} b_{ijk}^{(l)}$.

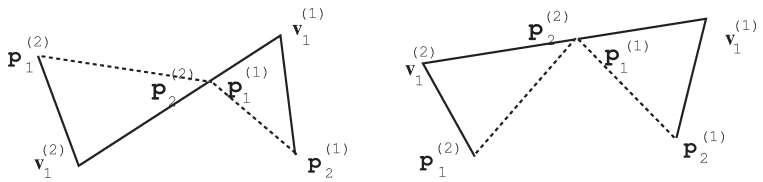


Figure 6.2: The two different cases of G^1 join polygon segments [BX99a].

Assuming $b_{n-1,1,0}^{(1)} > 0$ and $b_{1,n-1,0}^{(2)} > 0$, then the curves are regular in $\mathbf{p_1^{(1)}}$. In triangle $[\mathbf{p_1^{(1)} p_2^{(1)} v_1^{(1)}}]$ it is required that the A-spline to pass through $\mathbf{p_1^{(1)}}$ and here tangent with the line $[\mathbf{p_1^{(1)} v_1^{(1)}}]$. Therefore, the curve $F_1(1-\alpha_2-\alpha_3,\alpha_2,\alpha_3) = 0$ can be represented as a power series at $\mathbf{p_1^{(1)}}$

$$\alpha_2(\alpha_3) = \sum_{i=0}^{\infty} a_i^{(1)} \alpha_3^i \tag{6.6}$$

with $a_0^{(1)} = 0$. Similarly, $F_2(1-\alpha_2-\alpha_3,\alpha_2,\alpha_3) = 0$ at $\mathbf{p_2^{(2)}}$

$$\alpha_1(\alpha_3) = \sum_{i=0}^{\infty} a_i^{(2)} \alpha_3^i \tag{6.7}$$

with $a_0^{(2)} = 0$.

It follows from theorem 6.2.2 that the curve $F_1 = 0$ is tangent with $[\mathbf{p_1^{(1)}}\mathbf{v_1^{(1)}}]$ $n-2$ times at $\mathbf{p_1^{(1)}}$ if and only if

$$b_{n-k,0,k}^{(1)} = 0, \quad for\ k = 0,1,\ldots,n-2 \tag{6.8}$$

Symmetrically, $F_2 = 0$ is tangent with $[\mathbf{p_2^{(2)}}\mathbf{v_1^{(2)}}]$ $n-2$ times at $\mathbf{p_2^{(2)}}$ if and only if

$$b_{0,n-k,k}^{(2)} = 0, \quad for\ k = 0,1,\ldots,n-2 \tag{6.9}$$

Assuming (6.8) and (6.9) hold, the power series (6.6) and (6.7) become

$$\alpha_2(\alpha_3) = \sum_{i=n-1}^{\infty} a_i^{(1)} \alpha_3^i \tag{6.10}$$

$$\alpha_1(\alpha_3) = \sum_{i=n-1}^{\infty} a_i^{(2)} \alpha_3^i \tag{6.11}$$

Substitute (6.6) and (6.7) in $F_1(1-\alpha_2-\alpha_3,\alpha_2,\alpha_3) = 0$ and $F_2(1-\alpha_2-\alpha_3,\alpha_2,\alpha_3) = 0$ respectively

$$F_1(1-\alpha_2(\alpha_3)-\alpha_3,\alpha_2(\alpha_3),\alpha_3) = \sum_{i=n-1}^{\infty} g_i^{(1)} \alpha_3^i = 0 \tag{6.12}$$

$$F_2(\alpha_1(\alpha_3),1-\alpha_1(\alpha_3)-\alpha_3,\alpha_3) = \sum_{i=n-1}^{\infty} g_i^{(2)} \alpha_3^i = 0 \tag{6.13}$$

From $g_i^{(1)} = 0$ for $i = n-1,\ldots,2n-3$, one derives:

$$\begin{array}{lcl} \tilde{b}_{n-1,1,0}^{(1)} & = & -\frac{\tilde{b}_{1,0,n-1}^{(1)}}{a_{n-1}^{(1)}} \\ \cdots & = & \cdots \\ \tilde{b}_{n-i-2,1,i+1}^{(1)} & = & -\frac{\sum_{j=0}^{i} \tilde{b}_{n-1-j,1,j}^{(1)} \sum_{l=0}^{n-1-j} (-1)^l C_{n-1-j}^l a_{n+1-l-j}^{(1)}}{a_{n-1}^{(1)}} \end{array} \tag{6.14}$$

for $i = 1,2,\ldots,n-3$, where $C_n^k = \frac{n!}{k!(n-k)!}$, $a_j^{(1)} = 0$ if $j < n-1$. That is the coefficients determined by 6.14 will lead to the curve $F_1(\alpha_1,\alpha_2,\alpha_3) = 0$ matching the power series 6.10 up to the first $2n-3$ terms. It is noted that, each of the formulas 6.14 determines one of the coefficients bs, and introduces one of the coefficients as. Among all the coefficients bs, there is one degree of freedom. Since $b_{n-1,1,0}^{(1)} > 0$ and $b_{1,0,n-1}^{(1)} < 0$ implies $a_{n-1}^{(1)} > 0$. The correct sign of $b_{n-1-k,1,k}^{(1)}$ can be obtained by giving $a_{n+k-1}^{(1)}$ properly.

Similarly for the curve $F_2 = 0$ at $\mathbf{p_2^{(2)}}$ one has for $i = 1, 2, \ldots, 2n-3$

$$\begin{array}{lcl} \tilde{b}^{(2)}_{1,n-1,0} & = & -\frac{\tilde{b}^{(2)}_{0,1,n-1}}{a^{(2)}_{n-1}} \\ \cdots & = & \cdots \\ \tilde{b}^{(2)}_{1,n-i-2,i+1} & = & -\frac{\sum_{j=0}^{i} \tilde{b}^{(2)}_{1,n-1-j,j} \sum_{l=0}^{n-1-j} (-1)^l C^l_{n-1-j} a^{(1)}_{n+1-l-j}}{a^{(2)}_{n-1}} \end{array} \tag{6.15}$$

If we further assume (the reason for this will be shown shortly)

$$a^{(1)}_{n-1} = b^{(1)}_{1,0,n-1} = 0 \textit{ and } a^{(2)}_{n-1} = b^{(2)}_{0,1,n-1} = 0$$

Then, with a similar development, one has for $i = 1, 2, \ldots, n-2$

$$\begin{array}{lcl} \tilde{b}^{(1)}_{n-1,1,0} & = & -\frac{\tilde{b}^{(1)}_{0,0,n}}{a^{(1)}_{n}} \\ \cdots & = & \cdots \\ \tilde{b}^{(1)}_{n-i-1,1,i} & = & -\frac{\sum_{j=0}^{i-1} \tilde{b}^{(1)}_{n-1-j,1,j} \sum_{l=0}^{n-1-j} (-1)^l C^l_{n-1-j} a^{(1)}_{n+1-l-j}}{a^{(1)}_{n}} \end{array} \tag{6.16}$$

$$\begin{array}{lcl} \tilde{b}^{(2)}_{1,n-1,0} & = & -\frac{\tilde{b}^{(2)}_{0,0,n}}{a^{(2)}_{n}} \\ \cdots & = & \cdots \\ \tilde{b}^{(2)}_{1,n-i-1,i} & = & -\frac{\sum_{j=0}^{i} \tilde{b}^{(2)}_{1,n-1-j,j} \sum_{l=0}^{n-1-j} (-1)^l C^l_{n-1-j} a^{(1)}_{n+1-l-j}}{a^{(2)}_{n}} \end{array} \tag{6.17}$$

Formulas 6.16 and 6.17 match the power series up to the first $2n-2$ terms. If we only fit the first $2n-3$ terms, $b^{(l)}_{1,1,n-2}$ could be free. For the G^3 fitting with cubics in next section, it is chosen to be zero.

Now it is explained why both the cases of $a^{(l)}_{n-1} > 0$ and $a^{(l)}_{n-1} = 0$ has been considered. Let $[\mathbf{p_1^{(1)}}\mathbf{p_2^{(1)}}\mathbf{v_1^{(1)}}]$ and $[\mathbf{p_1^{(2)}}\mathbf{p_2^{(2)}}\mathbf{v_1^{(2)}}]$ two segments of the polygon. If they join at $\mathbf{p_1^{(1)}} = \mathbf{p_2^{(2)}}$, then there are two join configurations (see figure 6.2): *nonconvex* and *convex* join. In the nonconvex join $\mathbf{p_2^{(1)}}$ and $\mathbf{p_1^{(2)}}$ lie on different sides of the line $[\mathbf{v_1^{(1)}}\mathbf{v_2^{(2)}}]$, while in the convex join, $\mathbf{p_2^{(1)}}$ and $\mathbf{p_1^{(2)}}$ lie on the same side of the line $[\mathbf{v_1^{(1)}}\mathbf{v_2^{(2)}}]$. Since the A-splines are always contained within the triangles considered, if $\mathbf{p_1^{(1)}}$ is of a nonconvex join, then the curve will be tangent with the line $[\mathbf{v_1^{(1)}}\mathbf{v_2^{(2)}}]$ an odd number of times, otherwise, it will be tangent with the line $[\mathbf{v_1^{(1)}}\mathbf{v_2^{(2)}}]$ an even number of times. Therefore,

- If $\mathbf{p_1^{(1)}}$ is of a nonconvex join and n is an even number, then $a^{(l)}_{n-1} > 0$; vice versa if n is an odd number $a^{(l)}_{n-1} = 0$.

- If $\mathbf{p_1^{(1)}}$ is of a nonconvex join and n is an even number, then $a_{n-1}^{(l)} = 0$; vice versa if n is an odd number $a_{n-1}^{(l)} > 0$.

Theorem 6.3.1. *The degree n A-spline can achieve G^{2n-3} continuity by fitting locally the given derivative data at the join points. If n is an odd/even number and all the join point are nonconvex/convex then G^{2n-2} continuity is achievable.*

Notice that if $n > 3$ the coefficients b_{ijk} are free for $i > 1$ and $j > 1$. These $((n-2)(n-3)/2)$ degrees of freedom can be used for a finer control of the curve.

6.4 Cubic G^3 A-splines

For clarity, consider as an applied example the cubic case ($n = 3$) [BX99a]. According to theorem 6.3.1 we should be able to achieve G^3 continuity. In this case the B.B. form of $F(\alpha_1, \alpha_2, \alpha_3)$

$$F(\alpha_1, \alpha_2, \alpha_3) = \sum_{i+j+k=3} b_{ijk} B_{ijk}(\alpha_1, \alpha_2, \alpha_3),$$

will have $\binom{n+2}{2} = (n+2)(n+1)/2$ coefficients (see figure 6.3). Since there is a constant

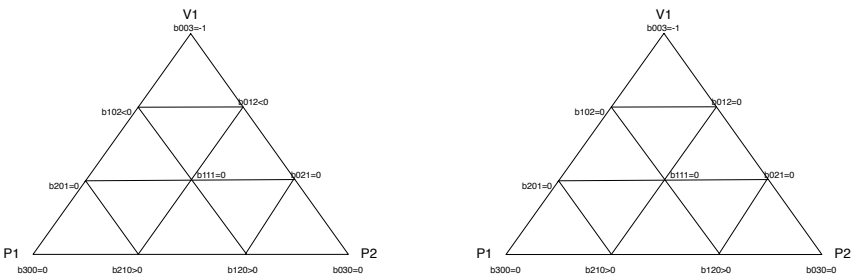


Figure 6.3: Cubic Beziér coefficients (a) $a_2^{(1)} > 0$ (b) $a_2^{(1)} = 0$. [BX99a]

multiplier to the equation $F(\alpha_1, \alpha_2, \alpha_3) = 0$ one sets conventionally $b_{003} = -1$. According to theorem 6.2.2, imposing G^0 and G^1 continuity will have $b_{201} = 0$ and $b_{102} = 0$ (and also $a_1^{(1)} = 0$). Then depending on the join type at $\mathbf{p_1}$ one has:

$\mathbf{p_1}$ non-convex join (see figure 6.3b) One has $a_2^{(1)} = 0$, as $\mathbf{p_1}$ is a flex point and the second derivative must be zero, furthermore this imposes $b_{102} = 0$. According to 6.16 one has

$$\tilde{b}_{210}^{(1)} = \frac{1}{a_3^{(1)}}, \qquad \textit{with} \quad a_3^{(1)} > 0$$
$$\tilde{b}_{111}^{(1)} = -\frac{a_4^{(1)} - 2a_3^{(1)}}{[a_3^{(1)}]^2}$$

where $a_3^{(1)}$ is the third derivative at the point and must be greater then zero in order for the curve to be inside the triangle.The coefficient $b_{111}^{(1)}$ is free but shared by both join points $\mathbf{p_1}$ and $\mathbf{p_2}$. If both are non-convex it can be used to achieve G^4 continuity, otherwise it is set to $b_{111}^{(1)} = 0$. Symmetrically the join point at $\mathbf{p_2}$ determines the coefficients b_{012} and b_{120}.

$\mathbf{p_1}$ convex join (see figure 6.3a) in this case $a_2^{(1)} > 0$ in order for the curve to be inside the triangle. According to 6.14

$$\tilde{b}_{210}^{(1)} = \frac{\tilde{b}_{102}^{(1)}}{a_3^{(1)}}$$
$$\tilde{b}_{111}^{(1)} = -\frac{\tilde{b}_{102}^{(1)}[a_3^{(1)} - a_2^{(1)}] + a_2^{(1)}}{[a_3^{(1)}]^2}$$

and with the convection $b_{111}^{(1)} = 0$ it follows that

$$\tilde{b}_{102}^{(1)} = -\frac{a_2^{(1)}}{a_3^{(1)} - a_2^{(1)}}$$

Symmetrically the join point at $\mathbf{p_2}$ determines the coefficients b_{012} and b_{120}.

6.5 Examples

G^k continuous triangular A-splines are implemented in the Ganith Algebraic Toolkit. In this software the user may interactively define the control polygon or load it from file. Quadratic, cubic and quartic splines are supported with varying degrees of continuity. The user may define the eventual additional degrees of freedom for finer control. It allows the user to draw A-spline curves, sweep or rotate an A-spline curve, subgroup the curve data by A–splines to lead to smooth piecewise approximation, smooth blending A–spline curves, and interpolate quadratic curves by surfaces.

In figure 6.4 the same closed control polygon is used to generate: (a) G^1 quadratic, (b) G^2 cubic, (c) G^3 cubic and (d) G^5 quartic A-splines. As an application of A-splines in Fig. 6.5, a stack of iso-contour slices reconstructions of a human head from volume MRI (Magnetic Resonance Imaging) data, is built using G^3 cubic A-splines. Fig. 6.6 shows G^3 cubic A-splines approximations of degree six and degree four algebraic plane curves:

$$(a) \quad (x^2+y^2)^3 - 4x^2y^2 = 0 \qquad (b) \quad x^3 - 3x - (1/9)(y^4 - 12y^2 + 18)$$

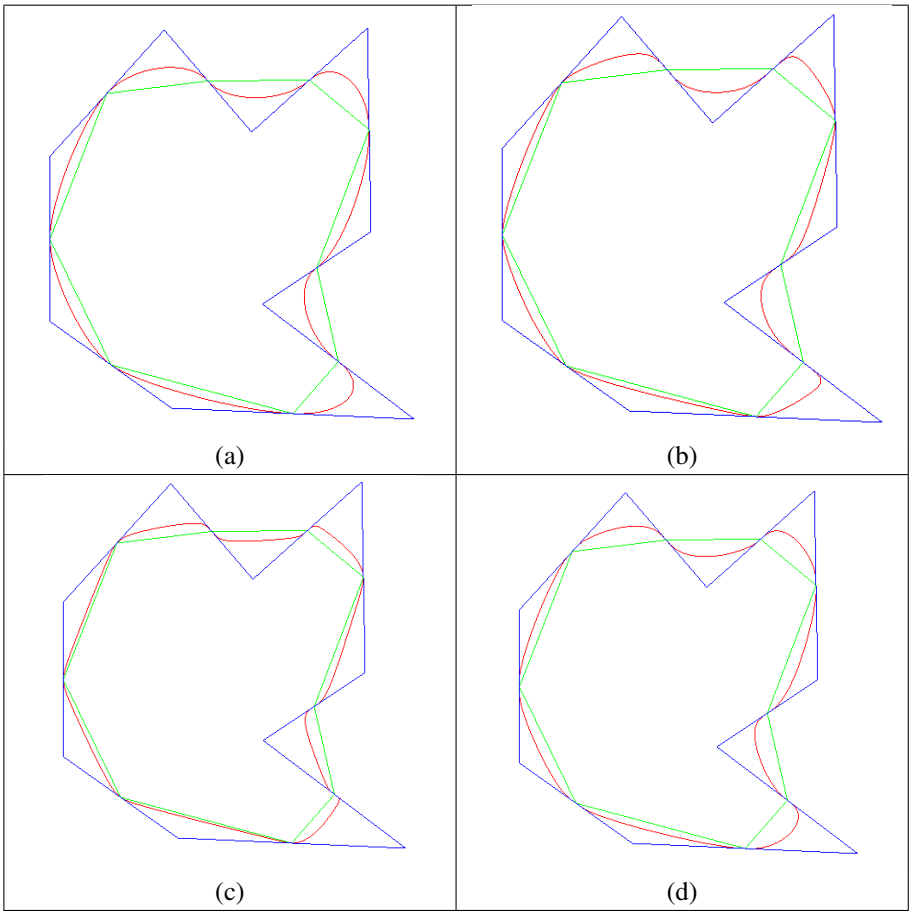


Figure 6.4: A-spline on a closed control polygon: (a) G^1 quadratic, (b) G^2 cubic, (c) G^3 cubic and (d) G^5 quartic. Images created using the A-Spline module of GANITH [BR90a]

Figure 6.5: G^3 cubic A-splines approximation of a stack of Magnetic Resonance Imaging volumetric cross-sectional data [BX99a].

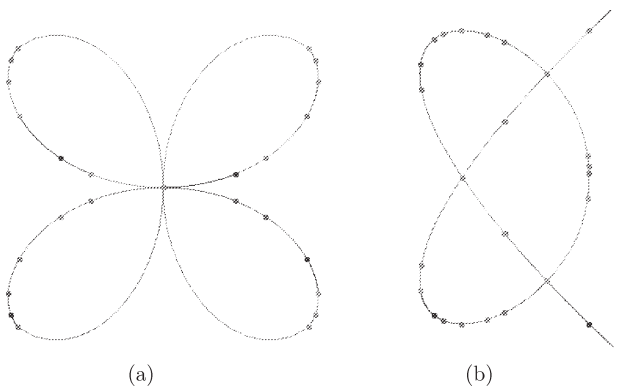


Figure 6.6: A-spline approximation of implicit algebraic curves: (a) $(x^2+y^2)^3-4x^2y^2=0$ and (b) $x^3-3x-(1/9)(y^4-12y^2+18)$. The curve segments between consecutive vertices (dots) are all cubic degree and with G^3 continuity at the vertices [BX99a].

Chapter 7

Tetrahedral C^1 Cubic A-Patches

In this chapter a detailed description including the calculation of the defining polynomial coefficients of cubic polynomial interpolating A-patches in tetrahedral domain are given. Content of this chapter has been extracted and constitute a reduced form of [BCX95].The chapter is structured in section: (1) main definitions are recalled, the notation homogenized and some fundamental facts introduced; (2) three and four-sided patches are introduced, sufficient conditions for regularity given, and main properties introduced; (3) it is shown the construction from a surface triangulation of the simplicial hull, providing the supporting scaffolds for each patch; (4) general conditions on the coefficients of each patch are worked out for C^1 continuity.

7.1 Notation

Convex and affine hulls Let $\{\mathbf{p_1},\ldots,\mathbf{p_j}\} \in \mathbb{R}^3$ with $j < 4$. Then the *convex hull* of these points is defined by $[\mathbf{p_1},\ldots,\mathbf{p_j}] = \{\mathbf{p} \in \mathbb{R}^3 \ : \ \mathbf{p} = \sum_{i=1}^{j} \alpha_i \mathbf{p_i},\ \alpha_i \geq 0\ ,\ \sum_{i=1}^{j} = 1\}$, and the *affine hull* is defined by $< \mathbf{p_1},\ldots,\mathbf{p_j} >= \{\mathbf{p} \in \mathbb{R}^3 \ : \ \mathbf{p} = \sum_{i=1}^{j} \alpha_i \mathbf{p_i},\ \sum_{i=1}^{j} = 1\}$. The interior of the convex hull is denoted $(\mathbf{p_1},\ldots,\mathbf{p_j}) = \{\mathbf{p} \in \mathbb{R}^3 \ : \ \mathbf{p} = \sum_{i=1}^{j} \alpha_i \mathbf{p_i},\ \alpha_i \geq 0\ ,\ \sum_{i=1}^{j} = 1\}$.

Barycentric coordinates Let $\mathbf{p_1},\mathbf{p_2},\mathbf{p_3},\mathbf{p_4} \in \mathbb{R}^3$ be affine independent. Then the tetrahedron with vertices $\mathbf{p_1},\mathbf{p_2},\mathbf{p_3}$, and $\mathbf{p_4}$ is V = $V = [\mathbf{p_1},\mathbf{p_2},\mathbf{p_3},\mathbf{p_4}]$. For any $\mathbf{p} = \sum_{i=1}^{4} \alpha_i \mathbf{p_i}$, $\boldsymbol{\alpha} = [\alpha_1,\alpha_2,\alpha_3,\alpha_4]^T$ is the barycentric coordinates of $\mathbf{p}$. Let $\mathbf{p} = (x,y,z)^T$ and $\mathbf{p_i} = (x_i,y_i,z_i)^T$ then the barycentric coordinates relate to the Cartesian coordinates via the following re-

lation:

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{pmatrix} \tag{7.1}$$

Berstein-Beziér (BB) form Any polynomial $f(\mathbf{p})$ of degree n can be expressed as BB form over V as

$$f(\mathbf{p}) = \sum_{|\boldsymbol{\lambda}|=n} b_{\boldsymbol{\lambda}} B^n_{\boldsymbol{\lambda}}(\boldsymbol{\alpha}),\ \boldsymbol{\lambda} \in \mathbb{Z}^4_+$$

where

$$B^n_{\boldsymbol{\lambda}}(\boldsymbol{\alpha}) = \frac{n!}{\lambda_1!\lambda_2!\lambda_3!\lambda_4} \alpha_1^{\lambda_1} \alpha_2^{\lambda_2} \alpha_3^{\lambda_3} \alpha_4^{\lambda_4}$$

is the Berstein basis polynomial, $|\boldsymbol{\lambda}| = \sum_{i=1}^4 \lambda_i$ with $\boldsymbol{\lambda} = (\lambda_1\lambda_2\lambda_3\lambda_4)^T \in \mathbb{Z}^4_+$, the $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)^T = \sum_{i=1}^4 \alpha_i e_i$ is a barycentric coordinate of $\mathbf{p}$, $b_{\boldsymbol{\lambda}}$ are the *control points*, $\mathbb{Z}^4_+$ stands for the set of all four-dimentional vectors with nonnegative integer components, and e_i stands for the i-th basis vector of $\mathbb{Z}^4_+$ (e.g. $e_3 = 0010$).

The following facts of polynomials in BB forms will be used. Details and dimonstation may be found in [Far01]. The first is derived from the directional derivative formulas:

Lemma 7.1.1. *If $f(\mathbf{p}) = \sum_{|\boldsymbol{\lambda}|=n} b_{\boldsymbol{\lambda}} B^n_{\boldsymbol{\lambda}}(\boldsymbol{\alpha})$, then*

$$b_{(n-1)e_i+e_j} = b_{ne_i} + \frac{1}{n}(\mathbf{p_j} - \mathbf{p_i})^T \nabla f(\mathbf{p_i}),\ j = 1\ldots4,\ j \neq i$$

where $\nabla f(\mathbf{p}) = [\frac{\partial f(\mathbf{p})}{\partial \alpha_1}, \frac{\partial f(\mathbf{p})}{\partial \alpha_2}, \frac{\partial f(\mathbf{p})}{\partial \alpha_3}]^T$

Lemma 7.1.2. *Let $f(\mathbf{p}) = \sum_{|\boldsymbol{\lambda}|=n} a_{\boldsymbol{\lambda}} B^n_{\boldsymbol{\lambda}}(\boldsymbol{\alpha})$ and $g(\mathbf{p}) = \sum_{|\boldsymbol{\lambda}|=n} b_{\boldsymbol{\lambda}} B^n_{\boldsymbol{\lambda}}(\boldsymbol{\alpha})$ two polynomials defined on two tetrahedra $[\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \mathbf{p_4}]$ and $[\mathbf{q_1}, \mathbf{q_2}, \mathbf{q_3}, \mathbf{q_4}]$. Then*

i f and g are C^0 continuous at the common face $[\mathbf{p_2}\mathbf{p_3}\mathbf{p_4}]$ if and only if (iff)

$$a_{\boldsymbol{\lambda}} = a_{\boldsymbol{\lambda}},\ \ for\ any\ \boldsymbol{\lambda} = 0\lambda_2\lambda_3\lambda_4,\ |\boldsymbol{\lambda}| = n. \tag{7.2}$$

ii f and g are C^1 continuous at the common face $[\mathbf{p_2}\mathbf{p_3}\mathbf{p_4}]$ iff (7.2) holds and

$$b_{1\lambda_2\lambda_3\lambda_4} = \beta_1 a_{1\lambda_2\lambda_3\lambda_4} + \beta_2 a_{0(\lambda_2+1)\lambda_3\lambda_4} + \beta_3 a_{0\lambda_2(\lambda_3+1)\lambda_4} + \beta_4 a_{0\lambda_2\lambda_3(\lambda_4+1)} \tag{7.3}$$

where $\boldsymbol{\beta} = (\beta_1\beta_2\beta_3\beta_4)^T$ are defined by the relation $\mathbf{p'_1} = \beta_1\mathbf{p_1}\beta_2\mathbf{p_2}\beta_3\mathbf{p_3}\beta_4\mathbf{p_4}$, $|\boldsymbol{\beta}| = 1$.

Relation (7.3) will be called coplanar condition [BCX95].

Degree Elevation The polynomial $f(\mathbf{p}) = \sum_{|\boldsymbol{\lambda}|=n} b_{\boldsymbol{\lambda}} B^n_{\boldsymbol{\lambda}}(\boldsymbol{\alpha})$ can be written as one of degree $n+1$: $f(\mathbf{p}) = \sum_{|\boldsymbol{\lambda}|=n+1} (Eb_{\boldsymbol{\lambda}}) B^{n+1}_{\boldsymbol{\lambda}}(\boldsymbol{\alpha})$, where $(Eb_{\boldsymbol{\lambda}}) = \frac{1}{n+1} \sum_{i=1}^4 \lambda_i b_{\lambda - e_i}$.

Variation Diminishing Property Let $y(t) = \sum_{i=0}^n b_i B^n_i(t)$; then $y(t)$ has no more intersections (counting the multiplicities) with any line than does the polygon $\{i/n, b_i\}_i^n$ in $[0,1]$.

Transformation since $\sum_{i=1}^4 \alpha_i = 1$, from (7.1) one has

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_1 - x_4 & x_2 - x_4 & x_3 - x_4 \\ y_1 - y_4 & y_2 - y_4 & y_3 - y_4 \\ z_1 - z_4 & z_2 - z_4 & z_3 - z_4 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} + \begin{pmatrix} x_4 \\ y_4 \\ z_4 \end{pmatrix} = A \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} + \begin{pmatrix} x_4 \\ y_4 \\ z_4 \end{pmatrix} \tag{7.4}$$

Let $f(x,y,z) = g(\alpha_1, \alpha_2, \alpha_3)$. Then it is easy to check that

$$\nabla f(x,y,z) = (A^{-1})^T \nabla g(\alpha_1, \alpha_2, \alpha_3) \tag{7.5}$$

Therefore, the surface $f(x,y,z) = 0$ is smooth (i.e. $\nabla f(x,y,z) \neq 0$) if only if the surface $g(\alpha_1, \alpha_2, \alpha_3) = 0$ is smooth (i.e. $\nabla g(\alpha_1, \alpha_2, \alpha_3) = 0$). This means that the smoothness problem of the surface $f(x,y,z) = 0$ can be treated directly in its barycentric form.

7.2 Sufficient Conditions

Let $F(\boldsymbol{\alpha}) = \sum_{|\boldsymbol{\lambda}|=n} b_{\boldsymbol{\lambda}} B^n_{\boldsymbol{\lambda}}(\boldsymbol{\alpha})$ be a given polynomial of degree n over the 3-dimentional simplex $S = \{\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)^T \in \mathbb{R}^4 : \ \sum_{i=0}^4 \alpha_i = 1, \ \alpha_i \geq 0\}$. The surface patch within the complex is defined by $S_F \subset S : F(\boldsymbol{\alpha}) = 0$ the following two condition on the trivariate BB-form will be used.

Smooth vertices condition For each $i \in \{1 \ldots 4\}$ there is at least one non-zero $b_{\lambda_1 \lambda_2 \lambda_3 \lambda_4}$ for $\lambda_i \geq n-1$.

Smooth edges condition For each pair $(i,j) \in \{1 \ldots 4\} \times \{1 \ldots 4\}, \ i \neq j$, there is at least one non-zero $b_{me_i + (n-m)e_j}$ for $m = \{0 \ldots n\}$, or the polynomials

$$\sum_{m=0}^{n-1} b_{me_i + (n-m-1)e_j + e_k} B^{n-1}_m(t)$$

and

$$\sum_{m=0}^{n-1} b_{me_i + (n-m-1)e_j + e_l} B^{n-1}_m(t)$$

have no common zero in $[0,1]$, for distinct i,j,k,l.

If the surface S_F contains a vertex/edge, then by the formulas of directional derivatives (see [Far01]), it is easy to show that the surface is smooth there if the smooth vertex/edge conditions above are satisfied.

Definition 7.2.1. *Three-sided patch*. Let the surface patch S_F, be smooth on the boundary of the tetrahedron S. If any open line segment $(e_j, \boldsymbol{\alpha}^*)$ with $\boldsymbol{\alpha}^* \in S_j = \{\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)^T : \ \alpha_j = 0, \ \alpha_i > 0, \ \sum_{i \neq j}^4 \alpha_i = 1, \ \alpha_i \geq 0\}$ (e.g. segment from vertex j to the opposing base) intersects S_F at most once (counting multiplicities), then S_F is called a *three sided j-patch patch* (see figure 7.1(a)(b)(c)(d)).

Definition 7.2.2. *Four-sided patch*. Let the surface patch S_F, be smooth on the boundary of the tetrahedron S. Let (i,j,k,l) be a permutation of $(1,2,3,4)$. If any open line segment $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$, with $\boldsymbol{\alpha}^* \in (e_i e_j)$ and $\boldsymbol{\beta}^* \in (e_k e_l)$ (e.g. a segment between opposing edges) intersects S_F at most once (counting multiplicities), then S_F is called a *a four sided $ij-kl$-patch* (see figure 7.1(e)(f)).

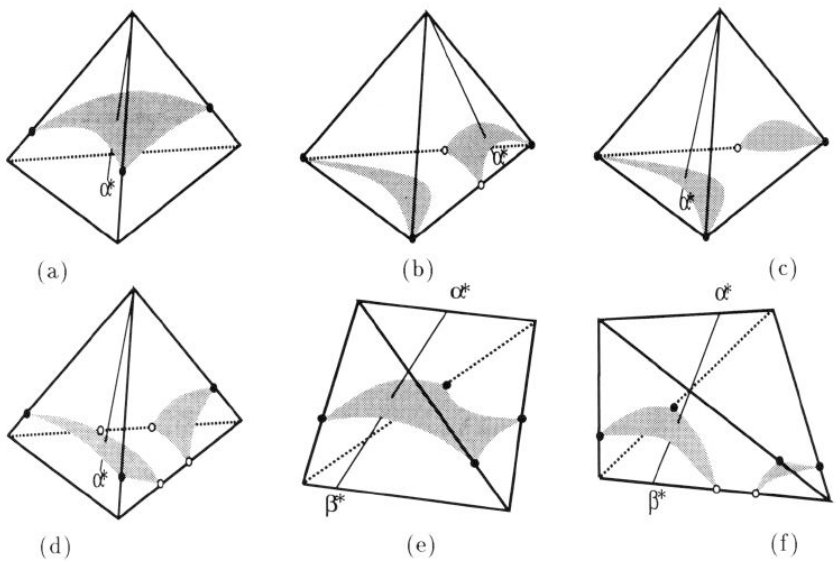


Figure 7.1: Three-sided (a, b, c and d) and four-sided patches (e and f). Some are disconnected. The filled vertices mark the boundaries of the patches [BCX95].

Lemma 7.2.3. *The three-sided j-patch and the four-sided $ij-kl$-patch are smooth (non-singular).*

Theorem 7.2.4. *Let $F(\boldsymbol{\alpha}) = \sum_{|\boldsymbol{\lambda}|=n} b_{\boldsymbol{\lambda}} B^n_{\boldsymbol{\lambda}}(\boldsymbol{\alpha})$ satisfy the smooth vertex and smooth edge conditions and $j \in [1..4]$ be a given integer. If there exists an integer $k \in [0..n]$ such that*

$$\begin{array}{ll} b_{\lambda_1\lambda_2\lambda_3\lambda_4} \geq 0 & \lambda_j \in [0..k-1] \\ b_{\lambda_1\lambda_2\lambda_3\lambda_4} \leq 0 & \lambda_j \in [k+1..n] \end{array}$$

and $\sum_{|\boldsymbol{\lambda}|=n,\ \lambda_j=0} b_{\boldsymbol{\lambda}} > 0$ if $k > 0$, $\sum_{|\boldsymbol{\lambda}|=n,\ \lambda_j=m} b_{\boldsymbol{\lambda}} < 0$ if $k > 0$ for at least one $m \in [k+1..n]$, then S_F is a three-sided j-patch.

Theorem 7.2.5. *Let $F(\boldsymbol{\alpha}) = \sum_{|\boldsymbol{\lambda}|=n} b_{\boldsymbol{\lambda}} B^n_{\boldsymbol{\lambda}}(\boldsymbol{\alpha})$ satisfy the smooth vertex and smooth edge conditions and (i, j, kl) be a permutation of $(1,2,3,4)$. If there exists an integer $k \in [0..n]$ such that*

$$\begin{array}{ll} b_{\lambda_1\lambda_2\lambda_3\lambda_4} \geq 0 & \lambda_i + \lambda_j \in [0..k-1] \\ b_{\lambda_1\lambda_2\lambda_3\lambda_4} > 0 & \lambda_i + \lambda_j \in [k+1..n] \end{array}$$

and $\sum_{|\boldsymbol{\lambda}|=n,\ \lambda_i+\lambda_j=0} b_{\boldsymbol{\lambda}} > 0$ if $k > 0$, $\sum_{|\boldsymbol{\lambda}|=n,\ \lambda_i+\lambda_j=m} b_{\boldsymbol{\lambda}} < 0$ for at least one $m \in [k+1..n]$ then S_F is a four-sided $ij-kl$-patch.

Note that the conditions on the coefficients $b_{\boldsymbol{\lambda}}$ in Theorems 7.2.4 and 7.2.5 are sufficient but not necessary.

Some properties of A-patches

1. For a three sided j-patch, if $b_{\boldsymbol{\lambda}} = 0$ for $\boldsymbol{\lambda} = (n-l)e_m + le_j$, $l \in [0..k]$, $m \neq j$, $k < n$, and $b_{\boldsymbol{\lambda}} \neq 0$ for $\boldsymbol{\lambda} = (n-1)e_m + e_s$, $s \neq j$ then the edge $[e_j e_m]$ is tangent with S_F at e_m with multiplicities k. See figure 7.2(a).

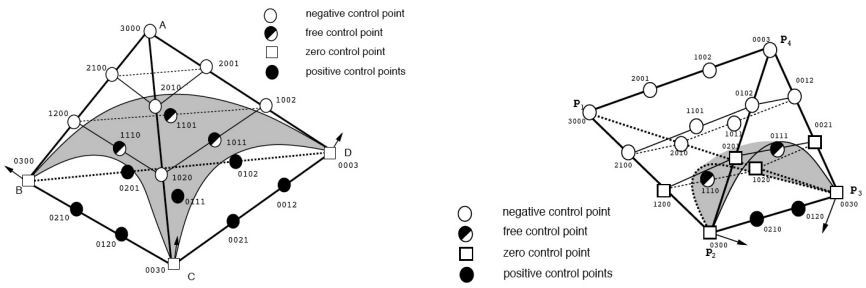
2. For a four sided $ij-kl$-patch, if $b_{\boldsymbol{\lambda}} = 0$ for $\boldsymbol{\lambda} = (n-q_1-b_2)e_k + q_1 e_i + q_2 e_j$, $q_1 + q_2 \in [0..s]$, and $b_{\boldsymbol{\lambda}} \neq 0$ for $\boldsymbol{\lambda} = (n-1)e_k + e_l$, then S_F is tangent s times with face $[e_i e_j e_k]$ at e_k with multiplicities k. See figure 7.2(b).

3. For a three sided j-patch, if $b_{\boldsymbol{\lambda}} = 0$ for $\boldsymbol{\lambda} = (n-m)e_i + me_k$, $m \in [0..n]$ then S_F contains the edge $[e_i e_k]$. If further, $b_{\boldsymbol{\lambda}} = 0$ for $\boldsymbol{\lambda} = (n-m-1)e_i + me_k + e_j$, $m \in [0..n-1]$, then S_F is tangent with face $[e_i e_j e_k]$. See figure 7.3.

4. For a three sided j-patch, any point $\mathbf{p} \in S_F$ can be mapped to a triple $(\alpha_i, \alpha_k, \alpha_l)$, $\alpha_i + \alpha_k + \alpha_l = 1$, $\alpha_i, \alpha_k, \alpha_l \geq 0$ or a point $\boldsymbol{\alpha}^* \in S_j = \{(\alpha_1, \alpha_2, \alpha_3, \alpha_4)^T : \alpha_j = 0\}$, Furthermore, there exists a one to one mapping between S_F and $S'_j = \{\boldsymbol{\alpha}^* : \boldsymbol{\alpha}^* \in S_j, F(\mathbf{e_j})\dot{F}(\boldsymbol{\alpha}^*) \leq 0\}$.

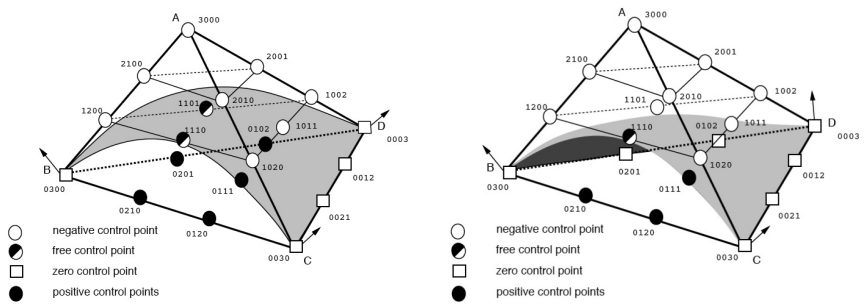
5. For a four-sided $ij-kl$-patch, and point $\mathbf{p} \in S_F$ can be mapped to a tuple (α_i, α_k), $0 \leq \alpha_i \leq 1$, $0 \leq \alpha_k \leq 1$, or two points $\boldsymbol{\alpha}^* \in \{(\mathbf{e_i e_j}) = \{(\alpha_1, \alpha_2, \alpha_3, \alpha_4)^T : \alpha_k = \alpha_l = 0\}$ and $\boldsymbol{\beta}^* \in \{(\mathbf{e_k e_l}) = \{(\alpha_1, \alpha_2, \alpha_3, \alpha_4)^T : \alpha_i = \alpha_j = 0\}$. Furthermore there exists a one-to-one mapping between S_F and $\{(\alpha_i, \beta_j)^T : F(\boldsymbol{\alpha}^*)\dot{F}(\boldsymbol{\beta}^*) \leq 0\}$. If $F(\boldsymbol{\alpha}^*) = 0$, S_F is degenerate and all the points with same α_k collapse into one point.

Hence a three-sided patch can be mapped into a triangular domain while a four-sided patch can be mapped into a quadrilateral domain. This observation gives rise to the terms three-sided patch and four-sided patch. Note that smooth three-sided or smooth four-sided patches are not necessarily connected within a single tetrahedron. Figure 7.1 shows some examples. Subsequent sections detail how a combination of smooth A-patches are pieced together to form a C^1 smooth global surface.



(a) (b)

Figure 7.2: (a) Three-sided patch tangent at $\mathbf{p_1}$,$\mathbf{p_2}$,$\mathbf{p_3}$. (b) Degenerate four-sided patch tangent to face $[\mathbf{p_1}\mathbf{p_2}\mathbf{p_4}]$ at $\mathbf{p_3}$ and to face $[\mathbf{p_1}\mathbf{p_3}\mathbf{p_4}]$ at $\mathbf{p_3}$. [BCX95]



(a) (b)

Figure 7.3: (a) Three-sided patch interpolating edge $[\mathbf{p_1}\mathbf{p_3}]$. (b) Three-sided patch interpolating edges $[\mathbf{p_2}\mathbf{p_3}]$ and$[\mathbf{p_1}\mathbf{p_3}]$. [BCX95]

Note that a four-sided patch may degenerate into a two-sided patch; see Figure 7.2(b). However, there is no need to treat the degenerate patches any different, but consider it to

be a special four-sided patch.

7.3 Scaffold Construction

Given point set $P = \{\mathbf{p_1}, ..., \mathbf{p_k}\} \in \mathbb{R}^3$ and their surface triangulation T, a normal set $N = \{\mathbf{n_1}, ..., \mathbf{n_k}\} \in \mathbb{R}^4$ for P, that is a normal $\mathbf{n_i}$ for each point $\mathbf{p_i}$. One wants to construct a patch set to interpolate points $\mathbf{p_i}$ and therein normal to $\mathbf{n_i}$ for $i \in [1, ..., k]$.

Without loss of generality, we assume that the assigned normals all point to the same side of T. If T is a closed surface triangulation (a simplicial polyhedron), then we assume that the normals all point to the exterior.

Definition 7.3.1. *Convex edge and nonconvex edge*. Let $[\mathbf{p_i}\mathbf{p_j}]$ be an edge of T. If $(\mathbf{p_j} - \mathbf{p_i})^T \mathbf{n_i} (\mathbf{p_i} - \mathbf{p_j})^T \mathbf{n_j} \geq 0$ and at least one of $(\mathbf{p_j} - \mathbf{p_i})^T \mathbf{n_i}$ and $(\mathbf{p_j} - \mathbf{p_i})^T \mathbf{n_j}$ is positive, then the edge $[\mathbf{p_j}\mathbf{p_i})]$ is said *negative convex*. If both are zero then we say it is *zero convex*. A *positive convex* edge is similarly defined. If $(\mathbf{p_j} - \mathbf{p_i})^T \mathbf{n_i} (\mathbf{p_i} - \mathbf{p_j})^T \mathbf{n_j} < 0$, then we say the edge is *non convex*.

Definition 7.3.2. *Convex face and nonconvex face*. Let $[\mathbf{p_i}\mathbf{p_j}\mathbf{p_k}]$ be a face of T. If its three edges are nonnegative (positive or zero) convex and at least one of them is positive convex, then we say the face $[\mathbf{p_i}\mathbf{p_j}\mathbf{p_k}]$ is positive convex. If all of the three edges are zero convex, then we label the face as zero convex. A negative convex face is similarly defined. All of the other cases face $[\mathbf{p_i}\mathbf{p_j}\mathbf{p_k}]$ is labeled as nonconvex.

Note that here we are overloading the term convex to characterize the relations between the normals and edges of faces. We distinguish between convex and non-convex faces in the simplicial hull below, where we build one tetrahedron for convex faces and double tetrahedral for nonconvex faces.

Definition 7.3.3. Simplicial hull. A simplicial hull of T, denoted by Σ, is a collection of non-degenerate tetrahedra that satisfies:

1. Each tetrahedron in Σ has either a single edge of T (then it is called an edge tetrahedron) or a single face of T (then it is called a face tetrahedron).

2. For each face of T, there is (are) only one or two face tetrahedral in Σ if the face is convex or non-convex respectively.

3. Two face tetrahedra that share a common edge do not intersect anywhere else. This condition is referred to as *non intersection*.

4. For each edge, there is (are) only one or two pair(s) of common face sharing edge tetrahedra in Σ if the edge is convex or nonconvex such that the pair(s) fill the region between the two adjacent face tetrahedral in the same side of T.

5. For each vertex, the tangent plane defined by the vertex normal is contained in all of the tetrahedral containing the vertex. This condition is called *tangent plane containment*.

Note that, for a given surface triangulation with normal assignments, T. there may exist infinitely many simplicial hulls, or no simplicial hull may exist. Now a scheme is given for the construction of a simplicial hull for the surface triangulation T and prescribed vertex normal assignment. Exceptional configurations where the construction simplicial hull of T is difficult are enumerated. In this case the original triangle must be split and then construct the simplicial hull for the locally modified triangulation T.

(1) Face tetrahedra construction For each face $F = [\mathbf{p_1 p_2 p_3}]$ of T, let L be a straight line that is perpendicular to the face F and that passes through the center of the inscribed circle of F. Then choose points $\mathbf{p_4}$ and/or $\mathbf{q_4}$ off each side of F to be the furthermost intersection points between L and the tangent planes of the vertices of the face. If F is a non-convex face, two face tetrahedral $[\mathbf{p_1 p_2 p_3 p_4}]$ and $[\mathbf{p_1 p_2 p_3 q_4}]$ are formed (double tetrahedral). If F is positive convex, then $\mathbf{p_4}$ is chosen on the same side as the direction of the normals, and a single face tetrahedron $[\mathbf{p_1 p_2 p_3 p_4}]$ is formed. If F is negative convex, then $\mathbf{q_4}$ is chosen on the opposite side of the normals, and again a single face tetrahedron $[\mathbf{p_1 p_2 p_3 q_4}]$ is formed. Figure 7.4 shows the cases where both faces are convex, and Figure 7.5 shows the case where at least one of the two adjacent faces is non-convex.

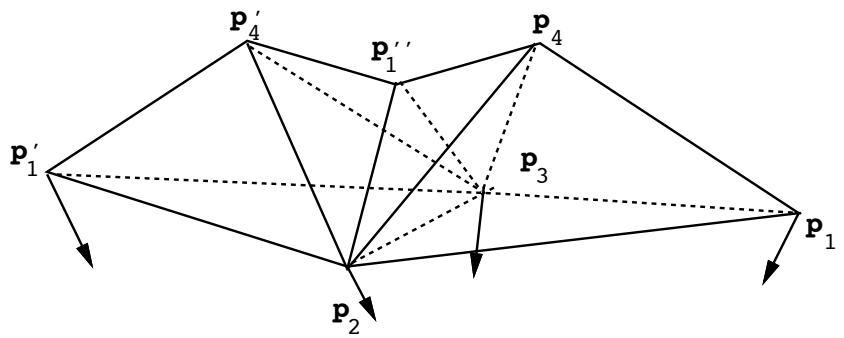


Figure 7.4: The Construction of Tetrahedra for Adjacent Convex/Convex Faces Intersecting Tetrahedra. [BCX95]

A sufficient condition for constructing face tetrahedral with tangent plane containment is that the angle of the assigned normal $\mathbf{n}$, at each vertex $\mathbf{p}$, with each of the surrounding

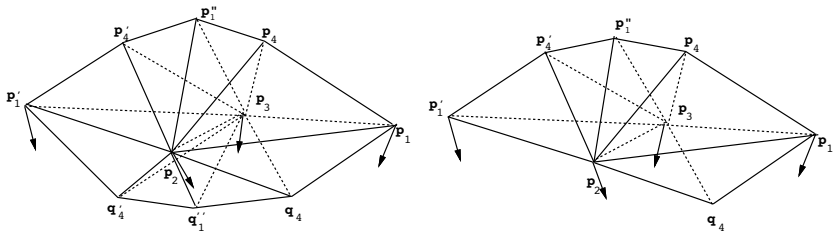


Figure 7.5: The construction of tetrahedra for (left) adjacent non-convex/non-convex faces and (right) convex/non-convex faces. [BCX95]

face's normals is less than $\pi/2$. If this condition is not met, then an exception occurs and the triangle must be split. See Figure 7.6.

A sufficient condition for adjacent face tetrahedral to be nonintersecting is as follows: For two adjacent faces $F = [\mathbf{p_1 p_2 p_3}]$ and $F' = [\mathbf{p'_1 p_2 p_3}]$, the angle between them, denoted as $\angle FF'$, is defined as the outer dihedral angle if the edge between F and F' is negative convex and inner dihedral angle otherwise. For $[\mathbf{p_2 p_3}]$ the common edge between F and F', let $[\mathbf{p_1 p_2 p_3 p_4}]$ and $[\mathbf{p'_1 p_2 p_3 p'_4}]$ be the face tetrahedral, respectively. Then the two tetrahedral are nonintersecting if the angles $\angle[\mathbf{p_4 p_2 p_3}][\mathbf{p_1 p_2 p_3}] < \frac{1}{2}\angle FF'$ and $\angle[\mathbf{p'_4 p_2 p_3}][\mathbf{p'_1 p_2 p_3}] < \frac{1}{2}\angle FF'$. If this condition is not met, then an exception occurs and the triangle must be split. See Figure 7.6.

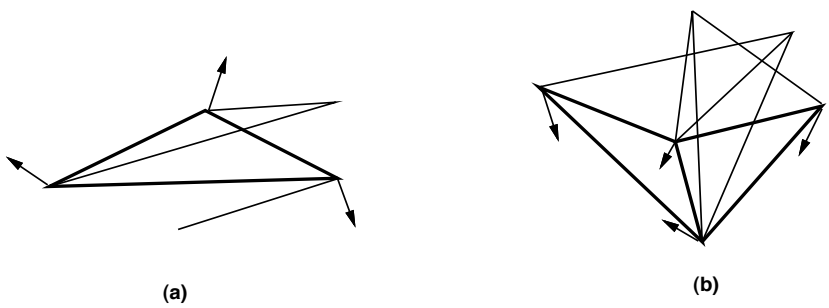


Figure 7.6: Degenerate cases (a)no tangent plane containment.(b) self-intersecting tetrahedra. [BCX95]

(2) Edge tetrahedra construction Let $E = [\mathbf{p_2 p_3}]$ be an edge of T, and let $F = [\mathbf{p_1 p_2 p_3}]$ and $F' = [\mathbf{p'_1 p_2 p_3}]$ be the two adjacent faces. Let $V_+ = [\mathbf{p_1 p_2 p_3 p_4}]$ and/or

$V_- = [\mathbf{p_1 p_2 p_3 q_4}]$, and $V'_+ = [\mathbf{p'_1 p_2 p_3 p'_4}]$ and/or $V'_- = [\mathbf{p'_1 p_2 p_3 p'_4}]$ be the face tetrahedral built for the faces F and F', respectively. Then, if edge E is non-convex, two pairs of tetrahedral need to be constructed. The first pair $[\mathbf{p''_1 p_2 p_3 p_4}]$ and $[\mathbf{p''_1 p_2 p_3 p'_4}]$ is between V_+ and V'_+. The second pair $[\mathbf{q''_1 p_2 p_3 q_4}]$and $[\mathbf{q''_1 p_2 p_3 q'_4}]$ is between V_- and V'_-. Set $\mathbf{p''_1}$ to be on or above $[\mathbf{p_4 p'_4}]$:

$$\mathbf{p''_1} = \frac{1-t}{2}(\mathbf{p_2}+\mathbf{p_3}) + \frac{t}{2}(\mathbf{p_4}+\mathbf{p'_4}),\ t \geq 1$$

so that $\mathbf{p''_i}$ is above plane of T and plane of T'. Similarly, set $\mathbf{q''_i}$ to be on or above $[\mathbf{q_4 q'_4}]$:

$$\mathbf{q''_1} = \frac{1-t}{2}(\mathbf{p_2}+\mathbf{p_3}) + \frac{t}{2}(\mathbf{q_4}+\mathbf{q'_4}),\ t \geq 1$$

so that $\mathbf{q''_i}$ is below plane of T and plane of T'. If edge E is positive/negative convex, only the first/second pair above is needed. If the edge E is zero convex, no tetrahedron is needed here. It should be noted that $\mathbf{p_4}$ and $\mathbf{p'_4}$ ($\mathbf{q_4}$ and $\mathbf{q'_4}$) are always visible.

7.4 Cubic C^1 Continuous A-patches

Having established a simplicial hull Σ for the given surface triangulation T and a set of vertex normals N, we now construct a piecewise C^1 function f on the hull Σ such that

$$f(\mathbf{p_l}) = 0,\ \ \nabla f(\mathbf{p_l}) = \mathbf{n_i},\ \ l \in [1..k] \tag{7.6}$$

and the zero contour of f within Σ forms a C^1 continuous single-sheeted surface with the same topology as T.

The construction of a piecewise C^1 cubic function

The construction of the function f over two adjacent faces of T is divided into the following three cases:

(a) Both faces are nonconvex.

(b) Both faces are convex.

(c) One of them is convex, and the other is nonconvex.

(a) Both faces are nonconvex.

Let $F = [\mathbf{p_1 p_2 p_3}]$ and $F' = [\mathbf{p_1 p_2 p_3'}]$ be two adjacent nonconvex faces. Then we have double tetrahedra $[\mathbf{p_1 p_2 p_3 p_4}]$ and/or $[\mathbf{p_1 p_2 p_3 q_4}]$ for F, and double tetrahedral $[\mathbf{p_1 p_2 p_3 p_4'}]$ and/or $[\mathbf{p_1 p_2 p_3 q_3'}]$ for F' (see figure 7.7). Let

$$\begin{array}{ll} V_1 = [\mathbf{p_1 p_2 p_3 p_4}] & V_2 = [\mathbf{p_1 p_2 p_3 p_4'}] \\ V_1' = [\mathbf{p_1 p_2 p_3 q_4}] & V_2' = [\mathbf{p_1 p_2 p_3 q_4'}] \\ W_1 = [\mathbf{p_1'' p_2 p_3 p_4}] & W_2 = [\mathbf{p_1'' p_2 p_3 p_4'}] \\ W_1' = [\mathbf{p_1'' p_2 p_3 q_4}] & W_2' = [\mathbf{p_1'' p_2 p_3 q_4'}] \end{array}$$

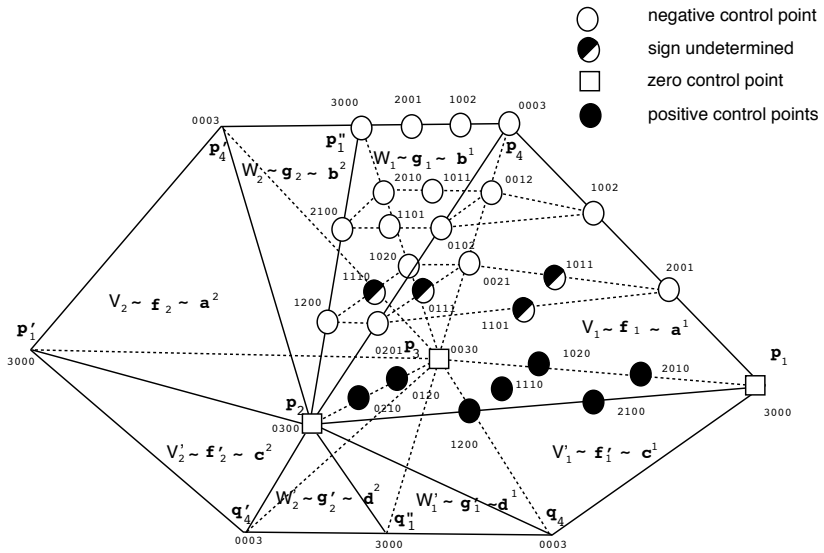


Figure 7.7: Adjacent double faces, tetrahedra, functions, and control points for two nonconvex adjacent faces. [BCX95]

and the cubic polynomials f_i over V_i, g_i over W_i, f_i' over V_i', andg_i' over W_i' be expressed in BB forms with coefficients $a^i_{\boldsymbol{\lambda}}$, $b^i_{\boldsymbol{\lambda}}$, $c^i_{\boldsymbol{\lambda}}$ and $d^i_{\boldsymbol{\lambda}}$ $i = 1,2$ respectively. Now these coefficients are determined.

C^0 **continuity** If two tetrahedral share a common face, we equate the control points of the associated cubic polynomials on the common face (see Lemma 7.1.2):

$$\begin{array}{lll} a^i_{\lambda_1\lambda_2\lambda_3 0} = c^i_{\lambda_1\lambda_2\lambda_3 0} & a^i_{0\lambda_2\lambda_3\lambda_4} = b^i_{0\lambda_2\lambda_3\lambda_4} & c^i_{0\lambda_2\lambda_3\lambda_4} = d^i_{0\lambda_2\lambda_3\lambda_4} \\ b^1_{\lambda_1\lambda_2\lambda_3 0} = b^2_{\lambda_1\lambda_2\lambda_3 0} & d^1_{\lambda_1\lambda_2\lambda_3 0} = d^2_{\lambda_1\lambda_2\lambda_3 0} & \end{array}$$

Interpolation Since zero contours of f_i and f_i' and g_i and g_i' pass through $\mathbf{p_2}$ and $\mathbf{p_3}$, then $a^i_{\boldsymbol{\lambda}} = b^i_{\boldsymbol{\lambda}} = c^i_{\boldsymbol{\lambda}} = d^i_{\boldsymbol{\lambda}} = 0$ for $i \in \{1,2\}$ and $\boldsymbol{\lambda} \in \{0300, 0030\}$.

Normal Condition From (7.6) and (7.1.1) one has for $j \in \{2,3\}$,

$$\begin{array}{ll} a^1_{2e_j+e_1} = \frac{1}{3}(\mathbf{p_1} - \mathbf{p_j})^T \mathbf{n_j}, & a^2_{2e_j+e_1} = \frac{1}{3}(\mathbf{p_1'} - \mathbf{p_j})^T \mathbf{n_j} \\ a^1_{2e_j+e_4} = \frac{1}{3}(\mathbf{p_4} - \mathbf{p_j})^T \mathbf{n_j}, & a^2_{2e_j+e_4} = \frac{1}{3}(\mathbf{p_4'} - \mathbf{p_j})^T \mathbf{n_j} \\ b^1_{2e_j+e_1} = \frac{1}{3}(\mathbf{p_1''} - \mathbf{p_j})^T \mathbf{n_j}, & d^1_{2e_j+e_1} = \frac{1}{3}(\mathbf{q_1''} - \mathbf{p_j})^T \mathbf{n_j} \\ c^1_{2e_j+e_4} = \frac{1}{3}(\mathbf{q_4} - \mathbf{p_j})^T \mathbf{n_j}, & c^2_{2e_j+e_4} = \frac{1}{3}(\mathbf{q_4'} - \mathbf{p_j})^T \mathbf{n_j} \end{array} \tag{7.7}$$

C^1 **Conditions** At present, consider $a^j_{2e_4+e_j}$, $c^j_{2e_4+e_j}$, b^j_{2001}, d^j_{2001} $j \in [1..4]$ as free parameters and determine the other control points:

(1) Interface of $[\mathbf{p_2 p_3 p_4}]$ and $[\mathbf{p_2 p_3 p_4'}]$. Suppose

$$\begin{array}{ll} \mathbf{p_1''} = \beta^1_1 \mathbf{p_1} + \beta^1_2 \mathbf{p_2} + \beta^1_3 \mathbf{p_3} + \beta^1_4 \mathbf{p_4}, & \sum_{i=1}^4 \beta^1_i = 1 \\ \mathbf{p_1''} = \beta^1_1 \mathbf{p_1'} + \beta^1_2 \mathbf{p_2} + \beta^1_3 \mathbf{p_3} + \beta^1_4 \mathbf{p_4'}, & \sum_{i=1}^4 \beta^1_i = 1 \end{array} \tag{7.8}$$

Then, the C^1 conditions require (see Lemma 7.1.2)

$$b^i_{1\lambda_2\lambda_3\lambda_4} = \beta^i_1 a^i_{1\lambda_2\lambda_3\lambda_4} + \beta^i_2 a^i_{0\lambda_2\lambda_3\lambda_4+0100} + \beta^i_3 a^i_{0\lambda_2\lambda_3\lambda_4+0010} + \beta^i_4 a^i_{0\lambda_2\lambda_3\lambda_4+0001} \tag{7.9}$$

For $\lambda_2\lambda_3\lambda_4 \in \{002, 101, 011, 110\}$. Hence b^i_{1002}, b^i_{1101} b^i_{1011} are defined, leaving a^i_{1011} and a^i_{1101} to be determined. Equation 7.9 for $\lambda_2\lambda_3\lambda_4 = 011$ will be treated later.

(2) Interface of $[\mathbf{p_2 p_3 p_1''}]$. Suppose

$$\begin{array}{ll} \mathbf{p_1''} = \mu^1_1 \mathbf{p_1} + \mu^1_2 \mathbf{p_2} + \mu^1_3 \mathbf{p_3} + \mu^1_4 \mathbf{p_4}, & \sum_{i=1}^4 \mu^1_i = 1 \\ \mathbf{p_1''} = \mu^1_1 \mathbf{p_1'} + \mu^1_2 \mathbf{p_2} + \mu^1_3 \mathbf{p_3} + \mu^1_4 \mathbf{p_4'}, & \sum_{i=1}^4 \mu^1_i = 1 \end{array} \tag{7.10}$$

Then, the C^1 conditions require

$$b^i_{\lambda_1\lambda_2\lambda_3 1} = \mu^i_1 b^1_{\lambda_2\lambda_3\lambda_4 1} + \mu^i_2 b^2_{\lambda_2\lambda_3\lambda_4 1} + \mu^i_3 b^i_{\lambda_2\lambda_3\lambda_4 0+0100} + \mu^i_4 b^i_{\lambda_2\lambda_3\lambda_4 1+0010} \tag{7.11}$$

For $\lambda_2\lambda_3\lambda_4 \in \{200, 110, 101, 011\}$. Hence b^i_{3000}, b^i_{2100} b^i_{2010} are defined. The equation for $\lambda_2\lambda_3\lambda_4 = 011$ will be treated later.

(3) Interface of $[\mathbf{p_2 p_3 q_4}]$, $[\mathbf{p_2 p_3 q_1''}]$, and $[\mathbf{p_2 p_3 q_4'}]$. All control points of g_i' and some of the control points of f_i' can be fixed as f_i and g_i. That is, the relations (7.9)-(7.11)

hold when the quantities a's b's, β's, and μ's are substituted by c's, d's, γ's, and η's, respectively. The two untreated equations left are

$$d^i_{1110} = \gamma^i_1 a^i_{1110} + \gamma^i_2 a^i_{0210} + \gamma^i_3 a^i_{0120} + \gamma^i_4 c^i_{0111} \tag{7.12}$$

$$d^i_{1110} = \eta^i_1 c^1_{1110} + \eta^i_2 c^2_{0111} + \eta^i_3 a^i_{0210} + \eta^i_4 a^i_{0120} \tag{7.13}$$

where the coefficients γ_i and η_i are defined by

$$\begin{array}{ll} \mathbf{q_1''} = \gamma^1_1 \mathbf{p_1} + \gamma^1_2 \mathbf{p_2} + \gamma^1_3 \mathbf{p_3} + \gamma^1_4 \mathbf{q_4}, & \sum_{i=1}^4 \gamma^1_i = 1 \\ \mathbf{q_1''} = \gamma^1_1 \mathbf{p_1'} + \gamma^1_2 \mathbf{p_2} + \gamma^1_3 \mathbf{p_3} + \gamma^1_4 \mathbf{q_4'}, & \sum_{i=1}^4 \gamma^1_i = 1 \\ \mathbf{q_1''} = \eta^1_1 \mathbf{q_4} + \eta^1_2 \mathbf{q_4'} + \eta^1_3 \mathbf{p_2} + \eta^1_4 \mathbf{p_3}, & \sum_{i=1}^4 \eta^1_i = 1 \end{array}$$

(4) Interface of $[\mathbf{p_1 p_2 q_3}]$ and $[\mathbf{p_1' p_2 p_3}]$. Let

$$\begin{array}{ll} \mathbf{q_4} = \alpha^1_1 \mathbf{p_1} + \alpha^1_2 \mathbf{p_2} + \alpha^1_3 \mathbf{p_3} + \alpha^1_4 \mathbf{p_4}, & \sum_{i=1}^4 \alpha^1_i = 1 \\ \mathbf{q_4'} = \alpha^1_1 \mathbf{p_1} + \alpha^1_2 \mathbf{p_2} + \alpha^1_3 \mathbf{p_3} + \alpha^1_4 \mathbf{p_4'}, & \sum_{i=1}^4 \alpha^1_i = 1 \end{array}$$

Then one has

$$c^i_{0112} = \alpha^i_1 c^1_{1110} + \alpha^i_2 c^2_{0210} + \alpha^i_3 a^i_{0120} + \alpha^i_4 a^i_{0111} \tag{7.14}$$

The equations (7.9), (7.11), (7.12), (7.13) and (7.14) must be solved for a minimal number of free parameters.

Equations (7.9), (7.11), (7.12), (7.13) may be combined as:

$$\mu_1 a^1_{0111} + \mu_2 a^2_{0111} + \mu_3 a^i_{0210} + \mu_4 a^i_{0120} = \beta^i_1 a^i_{1110} + \beta^i_2 a^i_{0210} + \beta^i_3 a^i_{0120} + \beta^i_4 a^i_{0111} \tag{7.15}$$

(7.12), (7.13) may be combined as:

$$\eta_1 c^1_{0111} + \eta_2 c^2_{0111} + \eta_3 a^i_{0210} + \eta_4 a^i_{0120} = \gamma^i_2 a^i_{0210} + \gamma^i_3 a^i_{0120} + \gamma^i_4 c^i_{0111} \tag{7.16}$$

Therefore (7.15) and (7.16) together with (7.14) form a linear system with six equations and six unknowns $a^i_{0111}, a^i_{1110}, c^i_{0111}$ for $i \in \{1,2\}$. It is important to point out that this is not an independent system (see Theorem 7.4.1 for the solvability of the system). It has four independent equations and infinitely many solutions. In, fact if one assumes $\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \mathbf{p_1'}$ are not coplanar and then denote

$$\begin{array}{ll} \mathbf{p_4} = \theta^1_1 \mathbf{p_1} + \theta^1_2 \mathbf{p_2} + \theta^1_3 \mathbf{p_3} + \theta^1_4 \mathbf{p_1'}, & \sum_{i=1}^4 \theta^1_i = 1 \\ \mathbf{p_4'} = \theta^2_1 \mathbf{p_1} + \theta^2_2 \mathbf{p_2} + \theta^2_3 \mathbf{p_3} + \theta^3_4 \mathbf{p_1'}, & \sum_{i=1}^4 \theta^1_i = 1 \\ \mathbf{q_4} = \vartheta^1_1 \mathbf{p_1} + \vartheta^1_2 \mathbf{p_2} + \vartheta^1_3 \mathbf{p_3} + \vartheta^1_4 \mathbf{p_1'}, & \sum_{i=1}^4 \vartheta^1_i = 1 \\ \mathbf{q_4'} = \vartheta^2_1 \mathbf{p_1} + \vartheta^2_2 \mathbf{p_2} + \vartheta^2_3 \mathbf{p_3} + \theta^3_4 \mathbf{p_1'}, & \sum_{i=1}^4 \vartheta^1_i = 1 \end{array} \tag{7.17}$$

Then from (7.15) and (7.16) one can derive that

$$a^i_{0111} = \theta^i_1 a^1_{1110} + \theta^i_2 a^1_{0210} + \theta^i_3 a^1_{0120} + \theta^i_4 a^2_{1110} \tag{7.18}$$

$$c^i_{0111} = \vartheta^i_1 a^1_{1110} + \vartheta^i_2 a^1_{0210} + \vartheta^i_3 a^1_{0120} + \vartheta^i_4 a^2_{1110} \tag{7.19}$$

Actually, this means any group of four weights (e.g. a^1_{1110}, a^i_{0210}, and a^2_{1110}) define the same $4-D$ hyperplane in its own barycentric coordinates (e.g., $[\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \mathbf{p'_1}]$. Therefore, besides a^1_{0210} and a^1_{1201} (or c^1_{0210} and c^1_{1201}) there are only 2 degrees of freedom left. One chooses a^i_{1110} (or c^i_{1110}) to be the free parameters. They may be determined by approximating a quadratic (see later or Dahmen and Thamm-Schaar [1993]).

(b) Both faces are convex.

(b1) Both faces are nonnegative (or nonpositive) convex. Following the discussion of (a), the scheme for determining the control points are as before, except for the following:

1. Only half of the control points are needed. That is, we need $a^i_{\boldsymbol{\lambda}}$, $b^i_{\boldsymbol{\lambda}}$ for functions f_i, and g_i, if F and F' are nonnegative convex, or $c^i_{\boldsymbol{\lambda}}$, $d^i_{\boldsymbol{\lambda}}$ for functions f'_i, and g'_i, if F and F' are nonpositive convex.

2. a^i_{1110} (or c^i_{1110}) can be determined freely as in (a). One way to choose them is to make the cubic approximate a quadratic. In particular, $a^i_{1110} = 0$ (or $c^i_{1110} = 0$) if the face is zero convex.

3. Only (7.18) is needed for unknown a^1_{0111} and a^2_{0111} if the edge $[\mathbf{p_2}\mathbf{p_3}]$ is nonnegative convex, or (7.19) for unknowns c^1_{0111} and c^2_{0111} if the edge $[\mathbf{p_2}\mathbf{p_3}]$ is nonpositive convex.

(b2) One positive convex face and one negative convex face. In this case, the common edge must be zero convex. Suppose F is positive convex and F is negative convex. All of the control points are determined as before, except for the following:

1. We only need to construct f_i, g_i and f'_2; that is, $c^1_{\boldsymbol{\lambda}}$ and $d^i_{\boldsymbol{\lambda}}$ are not needed. The functions g_i, and f_2 have no contribution to the surface and are used for smooth transition from f_1 to f'_2.

2. $a^1_{1110} \leq 0$ and $c^2_{1110} \geq 0$ can be determined freely.

3. Only (7.14) and (7.18) are needed for unknowns a^1_{0111}, a^2_{0111}, and c^1_{0111}.

(b3) Both faces are zero convex. This case, in fact, is included in case (bl). The surface is defined directly as the planar faces of the surface triangulation. No function needs to be constructed.

(c) One convex face and one nonconvex face.

Suppose $[\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}]$ is convex and $[\mathbf{p'_1}, \mathbf{p_2}, \mathbf{p_3}]$ is nonconvex, with the following exceptions:

1. The functions f'_i and g'_i and their control points $c^i_{\boldsymbol{\lambda}}$ and $d^i_{\boldsymbol{\lambda}}$ are not needed if F is nonnegative convex. The functions f_i and g_i, and their control points $a^i_{\boldsymbol{\lambda}}$ and $b^i_{\boldsymbol{\lambda}}$ are not needed if F is nonpositive convex.
2. $a^1_{1110} \leq 0$ (or $c^1_{1110} \geq 0$) and a^2_{1110} (c^2_{1110}) can be determined freely. In particular $a^1_{1110} = 0$ (or $c^1_{1110} = 0$) if $[\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}]$ is zero convex.
3. Only (7.14) for $i = 2$ and (7.18) are needed to solve for unknown a^1_{0111}, a^2_{0111}, and c^1_{0111} if edge $[\mathbf{p_2}, \mathbf{p_3}]$ is non negative convex. Otherwise solve (7.14) for $i = 2$ and (7.16) for unknown c^1_{0111}, c^2_{0111}, and a^2_{0111} if edge $[\mathbf{p_2}, \mathbf{p_3}]$ is non positive convex.

Coplanarity of adjacent faces.

In the discussions above, we have assumed that $\mathbf{p_1}, \mathbf{p'_1}\mathbf{p_2}, \mathbf{p_3}$ are affine independent. If this is not the case, then the coefficient matrices of the linear systems (7.15) and (7.16) are singular. However, the system (5. 11)-(5. 13) is still solvable (see Theorem 7.4.1).

The solvability of the related system

Theorem 7.4.1. *Given two affine independent point sets* $(\mathbf{p_2}, \mathbf{p_3}\mathbf{p'_4}, \mathbf{p_4})$ *and* $(\mathbf{p_2}, \mathbf{p_3}\mathbf{q'_4}, \mathbf{q_4})$ *as in figure (7.7).*

(i) The system (7.14)-(7.16) has four independent equations. If $(\mathbf{p_1}, \mathbf{p'_1}\mathbf{p_2}, \mathbf{p_3})$ *are affine independent, then (7.15) and (7.16) are four independent equations for the unknown* a^i_{0111} *and* c^i_{0111} *for* $i = 1, 2$.

(ii) Let

$$\{\mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}, \mathbf{r_4}, \mathbf{r_5}, \mathbf{r_6}\} = \{\mathbf{p_1}, \mathbf{p'_1}, \mathbf{p_4}, \mathbf{p'_4}, \mathbf{q'_4}, \mathbf{q_4}\}$$

and

$$\{x_1, \ldots, x_n\} = \{a^1_{1110}, a^2_{1110}, a^1_{0111}, a^2_{0111}, c^1_{0111}, c^2_{0111}\}$$

For any $1 \leq i < j \leq 6$, *if* $\mathbf{r_1}, \mathbf{r_3}, \mathbf{p_2}, \mathbf{p_3}$ *are affine independent, then*

$$x_k = \phi^k_1 x_i + \phi^k_2 x_j + \phi^k_3 a^1_{0210} + \phi^k_4 a^1_{0120} \tag{7.20}$$

where ϕ_l^k *are defined by* $\mathbf{r_k} = \phi_1^k \mathbf{r_i} + \phi_2^k \mathbf{r_j} + \phi_3^k \mathbf{p_2} + \phi_4^k \mathbf{p_3}$, $\sum_{m=1}^{4} \phi_m^k = 1$.

Proof of this theorem may be found in [BCX95].

Construction of single-sheeted A-patches

Having built C^1 cubics with some free control points, we now illustrate how to determine these free control points such that the zero contours are three-sided or four-sided A-patches (smooth and single sheeted).

We assume (without loss of generality) that all of the normals point to the same side of the surface triangulation T. That is the side on which $\mathbf{p_4}$ and $\mathbf{p_4'}$ lie (see Figure 7.7). Under this assumption, it follows from Definition 7.3.1 and Eq. (7.7) that the control points on the edge, say, a^i_{0210} and a^i_{0210}, on edge $[\mathbf{p_2}, \mathbf{p_3}]$ (see Figure 7.7), are nonnegative if the edge is nonnegative convex, and nonpositive if the edge is nonpositive convex. Now we can divide all of the control points into seven groups, called layers. The 0-th layer consists of the control points that are "on" the faces of T. The 1st layer is next to the 0th layer, and on the same side as the normal direction, followed by the 2nd and 3rd layers. Next to the 0th layer but opposite to the normal is the -1st layer, and then the -2nd and -3rd layers. Now we show that we can set all of the control points on the 2nd and 3rd layer as positive and the control points on the -2nd and -3rd layers as negative.

For the face-tetrahedra, it is always possible to make the 2nd and 3rd layers' control points positive, because these control points are free under the C^0 condition. For the control points on the edge-tetrahedra, it follows from (7.9) that the 2nd and 3rd layers' control points can be positive only if the 2nd layer's control points on the neighbor face-tetrahedra are large enough. This is achieved since β_4^i in (7.9) are positive. Similarly, the control points on the -2nd and -3rd layers can be chosen to be negative. Furthermore, all of these control points can be chosen as large as one needs in absolute value in order to get single-sheeted patches.

Since the control points around the vertices of T are determined by the normals, the smooth vertex condition is obviously satisfied. If the surface contains the edge $[\mathbf{p_2}\mathbf{p_3}]$ (see Figure 7.7), then, since a^i_{1110} (or a^i_{0111}) is freely chosen, the smooth edge condition is easily satisfied. Referring to Figure 7.7, one has proved in the following that the patches constructed over V_1 and W_1 are single sheeted. The other patches are similar:

Proposition 7.4.2. *If the face* $[\mathbf{p_1}\mathbf{p_2}\mathbf{p_3}]$ *is nonnegative convex, then the control points can be determined so that the surface over* V_1 *is a three-sided 4-patch.*

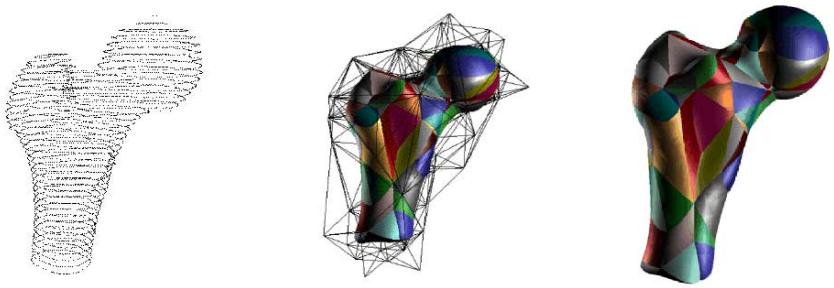
Proposition 7.4.3. *If the edge* $[\mathbf{p_2}\mathbf{p_3}]$ *is nonnegative convex, then the control points can be determined such that the surface over* W_1 *is a four-sided 14-23-patch.*

Theorem 7.4.4. *The global piecewise surface constructed is smooth, connected, and single-sheeted.*

With this, one concludes that the surface is topologically equivalent to the input triangulation.

7.5 Examples

The data for the human femur in Figure 7.8, 9223 points, comes from contouring of a CT scan. The algorithm does not use the fact that the data is arranged in slices. The reconstructed C^1 surface is made by 400 cubic patches.



(a) (b) (c)

Figure 7.8: (a) Data set from a CT scan for the upper part of a human femur. (b) A-patch scaffold construction. (c) Reconstructed object [BCX95].

The engine in Figure 7.9 has been reconstructed from a data set containing 9800 points. The number of patches generated in the approximation phase is 382, with an error equal to 1/100 of the size of the object. Each patch is of degree 3, and is therefore defined by 20 coefficients. At the same time, an approximate C^1 scalar field (pressure form a simulated experiment) over the surface has also been computed. Several techniques can be used to visualize this surface-on-surface data. In Figure 7.9(c) we show iso-pressure regions. With the normal projection method, each point p on the domain surface SD is projected in the direction normal to SD, to a distance proportional to the value $f^F(p)$ of the field at that point. The projected surface is visible in transparency in Figure 7.9(d), with iso-contours of the pressure projected on it. the value of the function at each vertex, the average gradient at vertices and mid-edge points, and the continuity constraint. The

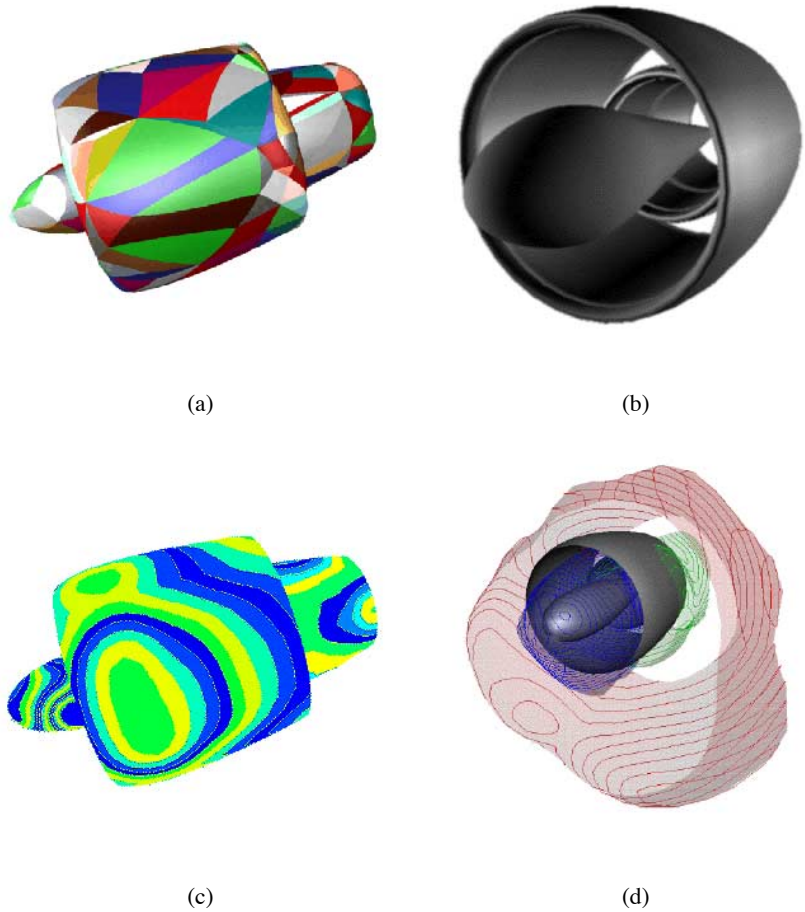


Figure 7.9: A jet engine. (a) C^0 reconstructed domain. Patches are visible in different colors. (b) Reconstructed domain (after C^1 smoothing). (c) Iso-pressure contours and regions of a surface-on-surface pressure function displayed on the surface of the jet engine. (d) The reconstructed engine surface and visualization of the pressure surface function surrounding the jet engine using the normal projection method [BCX95].

Chapter 8

Prism Algebraic Patches

Prism A-patches are low-degree finite elements of algebraic surfaces, with dual implicit and rational parametric representations. In [BX01, ZXB07] the input to the prism A-patch construction is a matched triangulation pair $\mathscr{T} = (\mathscr{T}_0, \mathscr{T}_1)$ (also called a *fat triangulation*) with attached normals at each vertex, which offers a linearization of the inner and outer boundary surfaces of a shell domain. The goal is to reconstruct a smooth fat surface whose bounding surfaces provide approximations of $\mathscr{T}_0$ and $\mathscr{T}_1$, respectively. Additionally mid-surfaces between the boundary surfaces may be also generated.

The matched pair of surface triangulations with normals could be obtained via several methods, including close iso-contours of volume data, point clouds, single surfaces, etc. In this chapter it is described a reduced version in which a single triangulation $\mathscr{T}$ is sufficient, together with external normals $\mathbf{n}_i, \mathbf{n}_{ij}, \mathbf{n}_{ijk}$ attached to either the 0-cells (vertex normals), or to the 1-cells (edge normals, to denote the parallel transport of sharp features), or to the 2-cells (face normals) of $\mathscr{T}$. Sources of this chapter are from both [ZXB07] and [BPP$^+$08] as well as other unpublished materials from Chandrajit Bajaj, Na Lei, Wenqi Zhao and the author. The software was originally developed by Wenqi Zhao [ZXB07] and extended by the author itself [BPP$^+$08].

8.1 Triangular Prism Cubic A-patch

The triangular prism A-patch element is defined within a prism scaffold. For each triangle $[\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k]$ of a triangulation $\mathscr{T}$ of the surface, let

$$\mathbf{v}_\ell(\lambda) = \mathbf{v}_\ell + \lambda \mathbf{n}_\ell, \qquad \ell = i, j, k, \qquad \lambda \in I := [-1, 1],$$

where the unit normals $\mathbf{n}_\ell$ point outward. Then define the prism

$$D_{ijk} := \{\mathbf{p} : \ \mathbf{p} = \alpha_1 \mathbf{v}_i(\lambda) + \alpha_2 \mathbf{v}_j(\lambda) + \alpha_3 \mathbf{v}_k(\lambda), \ \lambda \in I\}$$

where $(\alpha_1, \alpha_2, \alpha_3)$ are the barycentric coordinates of points in $[\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k]$. The coordinate transformation from local (prism's) coordinates to global (world's) reference system can be written, by substituting $\alpha_3 = 1 - \alpha_1 - \alpha_2$, as:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \mathbf{D}(\lambda) \begin{pmatrix} 1 \\ \alpha_1 \\ \alpha_2 \end{pmatrix} = \begin{pmatrix} \mathbf{v}_i(\lambda) & \mathbf{v}_j(\lambda) - \mathbf{v}_i(\lambda) & \mathbf{v}_k(\lambda) - \mathbf{v}_i(\lambda) \end{pmatrix} \begin{pmatrix} 1 \\ \alpha_1 \\ \alpha_2 \end{pmatrix}. \tag{8.1}$$

The map (8.1) is bilinear.

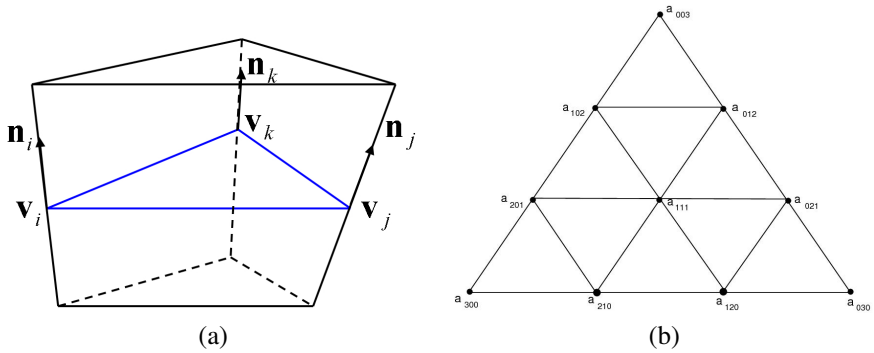


Figure 8.1: (a) A prism D_{ijk} constructed based on the triangle $[\mathbf{v}_i\mathbf{v}_j\mathbf{v}_k]$. (b) The control coefficients of the cubic Bernstein-Bézier basis of function F.

The surface S of a patch is defined as the zero contour of a scalar function field defined as a polynomial in the Benstein-Bézier (BB) form over the prism D_{ijk} :

$$F(\boldsymbol{\alpha}, \lambda) = \sum_{i+j+k=n} a_{ijk}(\lambda) B^n_{ijk}(\boldsymbol{\alpha}) \tag{8.2}$$

where $B^n_{ijk}(\boldsymbol{\alpha})$ is the BB basis

$$B^n_{ijk}(\boldsymbol{\alpha}) = \frac{n!}{i!j!k!} \alpha_1^i \alpha_2^j \alpha_3^k$$

Since S passes through the vertices $v_i,\ v_j,\ v_k$, then

$$a_{300} = a_{030} = a_{003} = \lambda \tag{8.3}$$

To obtain C^1 continuity at the vertices, let $a_{210} - a_{300} = \frac{1}{3}\nabla F(v_i) \cdot (\mathbf{v}_j(\lambda) - \mathbf{v}_i(\lambda))$, where $\nabla F(v_i) = \mathbf{n}_i$. Therefore

$$a_{210} = \lambda + \frac{1}{3}\mathbf{n}_i \cdot (\mathbf{v}_j(\lambda) - \mathbf{v}_i(\lambda)) \tag{8.4}$$

a_{120}, a_{201}, a_{102}, a_{021}, a_{012} are defined similarly.

It is assumed that S is C^1 continuous at the midpoints of the edges of T. Then define a_{111} using the side-vertex scheme [Nie79]:

$$a_{111} = w_1 a_{111}^{(1)} + w_2 a_{111}^{(2)} + w_3 a_{111}^{(3)} \tag{8.5}$$

where

$$w_i = \frac{\alpha_j^2 \alpha_k^2}{\alpha_2^2\alpha_3^2 + \alpha_1^2\alpha_3^2 + \alpha_1^2\alpha_2^2}, \qquad i = 1,2,3;\ i \neq j \neq k$$

Next it is explained how $a_{111}^{(1)}$, $a_{111}^{(2)}$ and $a_{111}^{(3)}$ are defined such that the C^1 continuity is obtained at the midpoint of the edge $\mathbf{v}_j\mathbf{v}_k$, $\mathbf{v}_i\mathbf{v}_k$ and $\mathbf{v}_i\mathbf{v}_j$. Consider the edge $\mathbf{v}_i\mathbf{v}_j$ in the prism D_{ijk}. Recall that any point $\mathbf{p} := (x,y,z)$ in D_{ijk} can be represented by

$$(x,y,z)^\top = \alpha_1 \mathbf{v}_i(\lambda) + \alpha_2 \mathbf{v}_j(\lambda) + \alpha_3 \mathbf{v}_k(\lambda) \tag{8.6}$$

Therefore, after differentiation of both sides of (8.6) with respect to x, y and z respectively, we get

$$I_3 = \begin{pmatrix} \frac{\partial\alpha_1}{\partial x} & \frac{\partial\alpha_2}{\partial x} & \frac{\partial\lambda}{\partial x} \\ \frac{\partial\alpha_1}{\partial y} & \frac{\partial\alpha_2}{\partial y} & \frac{\partial\lambda}{\partial y} \\ \frac{\partial\alpha_1}{\partial z} & \frac{\partial\alpha_2}{\partial z} & \frac{\partial\lambda}{\partial z} \end{pmatrix} \begin{pmatrix} (\mathbf{v}_i(\lambda) - \mathbf{v}_k(\lambda))^\top \\ (\mathbf{v}_j(\lambda) - \mathbf{v}_k(\lambda))^\top \\ (\alpha_1\mathbf{n}_i + \alpha_2\mathbf{n}_j + \alpha_3\mathbf{n}_k)^\top \end{pmatrix} \tag{8.7}$$

where I_3 is a 3×3 unit matrix. Let

$$T = \begin{pmatrix} (\mathbf{v}_i(\lambda) - \mathbf{v}_k(\lambda))^\top \\ (\mathbf{v}_j(\lambda) - \mathbf{v}_k(\lambda))^\top \\ (\alpha_1\mathbf{n}_i + \alpha_2\mathbf{n}_j + \alpha_3\mathbf{n}_k)^\top \end{pmatrix} \tag{8.8}$$

Let $A = \mathbf{v}_i(\lambda) - \mathbf{v}_k(\lambda)$, $B = \mathbf{v}_j(\lambda) - \mathbf{v}_k(\lambda)$ and $C = \alpha_1\mathbf{n}_i + \alpha_2\mathbf{n}_j + \alpha_3\mathbf{n}_k$, then

$$T = \begin{pmatrix} A, & B, & C \end{pmatrix}^\top$$

By (8.7) we have

$$\begin{pmatrix} \frac{\partial\alpha_1}{\partial x} & \frac{\partial\alpha_2}{\partial x} & \frac{\partial\lambda}{\partial x} \\ \frac{\partial\alpha_1}{\partial y} & \frac{\partial\alpha_2}{\partial y} & \frac{\partial\lambda}{\partial y} \\ \frac{\partial\alpha_1}{\partial z} & \frac{\partial\alpha_2}{\partial z} & \frac{\partial\lambda}{\partial z} \end{pmatrix} = T^{-1} = \frac{1}{\det(T)}\,(B \times C,\ C \times A,\ A \times B) \tag{8.9}$$

According to (8.2), at $(\alpha_1,\alpha_2,\alpha_3) = (\frac{1}{2},\frac{1}{2},0)$ (the midpoint of $\mathbf{v}_i\mathbf{v}_j$), one has have

$$\begin{pmatrix} \frac{\partial F}{\partial b_1} \\ \frac{\partial F}{\partial b_2} \\ \frac{\partial F}{\partial \lambda} \end{pmatrix} = \begin{pmatrix} (\mathbf{v}_i(\lambda) - \mathbf{v}_k(\lambda))^\top \\ (\mathbf{v}_j(\lambda) - \mathbf{v}_k(\lambda))^\top \\ (\mathbf{n}_i + \mathbf{n}_j)^\top/2 \end{pmatrix} \left(\frac{\mathbf{n}_i + \mathbf{n}_j}{4}\right) + \begin{pmatrix} \frac{3}{2}(a_{210} - a_{111}) \\ \frac{3}{2}(a_{120} - a_{111}) \\ \frac{1}{2} \end{pmatrix}$$

By (8.5), at $(\alpha_1, \alpha_2, \alpha_3) = (\frac{1}{2}, \frac{1}{2}, 0)$ it results

$$a_{111} = a_{111}^{(3)}$$

Therefore the gradient at $(\frac{1}{2}, \frac{1}{2}, 0)$ is

$$\begin{aligned}\nabla F &= T^{-1}(\frac{\partial F}{\partial \alpha_1}, \frac{\partial F}{\partial \alpha_2}, \frac{\partial F}{\partial \lambda})^\top \\ &= \frac{\mathbf{n}_i + \mathbf{n}_j}{4} + \frac{1}{2\det(T)}[3(a_{210} - a_{111}^{(3)})B \times C \\ &+ 3(a_{120} - a_{111}^{(3)})C \times A + A \times B] \end{aligned} \tag{8.10}$$

Let

$$\begin{aligned}&\mathbf{d}_1(\lambda) = \mathbf{v}_j(\lambda) - \mathbf{v}_i(\lambda) = B - A \\ &\mathbf{d}_2(\alpha_1, \alpha_2, \alpha_3) = \alpha_1 \mathbf{n}_i + \alpha_2 \mathbf{n}_j + \alpha_3 \mathbf{n}_k = C \\ &\mathbf{d}_3(\alpha_1, \alpha_2, \alpha_3, \lambda) = \mathbf{d}_1 \times \mathbf{d}_2 = B \times C + C \times A \end{aligned} \tag{8.11}$$

Define

$$\mathbf{c} = C(\frac{1}{2}, \frac{1}{2}, 0)\mathbf{d}_3(\lambda) = \mathbf{d}_3(\frac{1}{2}, \frac{1}{2}, 0, \lambda) = B \times \mathbf{c} + \mathbf{c} \times A \tag{8.12}$$

Let $\nabla F = \nabla F(\frac{1}{2}, \frac{1}{2}, 0)$. In order to make S be C^1 at $(\frac{1}{2}, \frac{1}{2}, 0)$, one should have $\nabla F \cdot \mathbf{d}_3(\lambda) = 0$. Therefore, by (8.10) and (8.11), we have

$$a_{111}^{(3)} = \frac{\mathbf{d}_3(\lambda)^\top (3a_{210} B \times \mathbf{c} + 3b_{120} \mathbf{c} \times A + A \times B)}{3||\mathbf{d}_3(\lambda)||^2} \tag{8.13}$$

Similarly, one may define $a_{111}^{(1)}$ and $a_{111}^{(2)}$. For the surface evaluation, given the barycentric coordinates of a point $(\alpha_1, \alpha_2, \alpha_3)$ in triangle $[\mathbf{v}_i \mathbf{v}_j \mathbf{v}_k]$, equation $F = 0$ is solved for λ by Newton's method, where F is defined in (8.2). Then the corresponding point on S is

$$(x,\ y,\ z)^\top = \alpha_1 \mathbf{v}_i(\lambda) + \alpha_2 \mathbf{v}_j(\lambda) + \alpha_3 \mathbf{v}_k(\lambda) \tag{8.14}$$

8.2 Quadrangular Prism Cubic A-patch

The quadrangular prism A-patch element is defined within a four sides prism scaffold. For each quadrangle $[\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k, \mathbf{v}_l]$ of $\mathscr{T}$ of the surface, let

$$\mathbf{v}_\ell(\lambda) = \mathbf{v}_\ell + \lambda \mathbf{n}_\ell, \qquad \ell = i, j, k, l, \qquad \lambda \in I := [-1, 1],$$

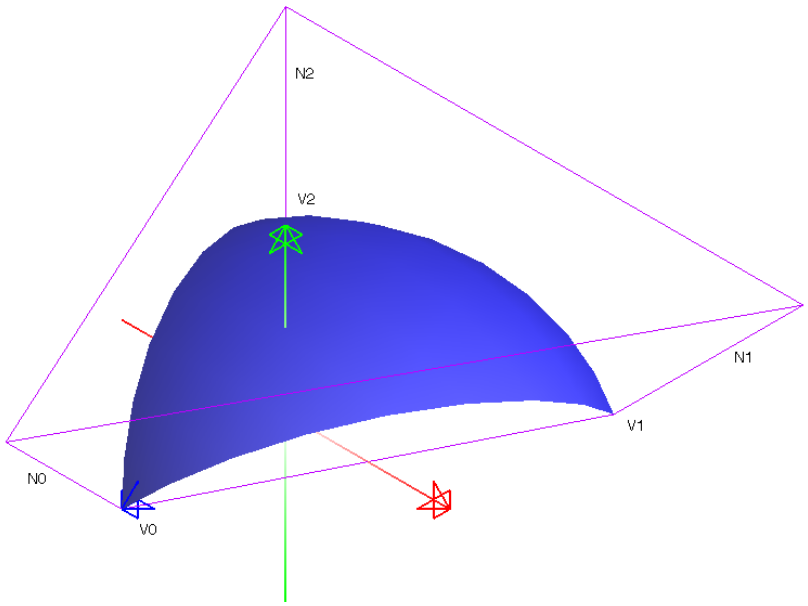


Figure 8.2: The prism A-patch with scaffold. Image generated by the author using the software developed for [BPP+08]

where the unit normals $\mathbf{n}_\ell$ point outward. Then the prism, denoted by D_{ijkl}, for $[\mathbf{v}_i\mathbf{v}_j\mathbf{v}_k\mathbf{v}_l]$ is a volume in $\mathbb{R}^3$ enclosed by the surfaces H_{ij}, H_{jk}, H_{kl}, and H_{li} (see figure 8.5), where H_{sm} is a ruled surface defined by $\mathbf{v}_s$ and $\mathbf{v}_m$ as follows:

$$H_{sm} = \{p : p = b_1\mathbf{v}_s(\lambda) + b_2\mathbf{v}_m(\lambda), \quad b_1 + b_2 = 1, \quad \lambda \in \mathbb{R}\}$$

For any point $p = b_1\mathbf{v}_s(\lambda) + b_2\mathbf{v}_m(\lambda)$ with $b_1 + b_2 = 1$, (b_1, b_2, λ) will be called the $H_{sm}-$coordinate of p. The prism D_{ijkl} as the volume given by

$$\begin{aligned} D_{ijk}(I) = \{p : p \quad &= B_{00}(u,v)\mathbf{v}_i(\lambda) + B_{10}(u,v)\mathbf{v}_j(\lambda) \\ &+ B_{01}(u,v)\mathbf{v}_l(\lambda) + B_{11}(u,v)\mathbf{v}_k(\lambda), \\ &u, v \in [0,1], \lambda \in \mathbb{R}\}, \end{aligned}$$

where $B_{00} = (1-u)(1-v)$, $B_{10} = u(1-v)$, $B_{01} = (1-u)v$, $B_{11} = uv$. One calls (u, v, λ) the local coordinate of p. The equation

$$\begin{aligned} p \quad &= B_{00}(u,v)\mathbf{v}_i(\lambda) + B_{10}(u,v)\mathbf{v}_j(\lambda) \\ &+ B_{01}(u,v)\mathbf{v}_l(\lambda) + B_{11}(u,v)\mathbf{v}_k(\lambda) \end{aligned} \tag{8.15}$$

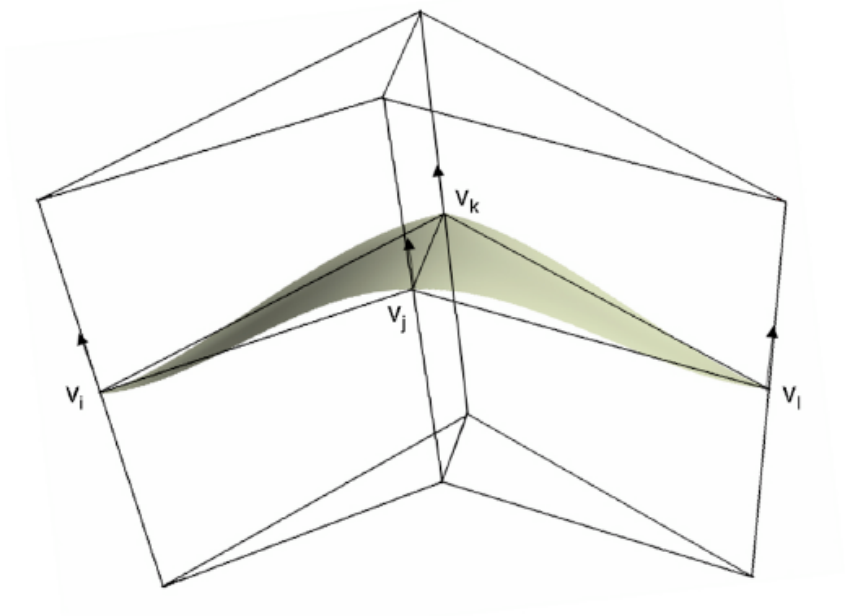


Figure 8.3: Smooth connection of two prism patches [ZXB07].

defines a transform between (u,v,λ) and (x,y,z).

The mapping from the local coordinate to the global coordinate is three-linear in λ,u,v, as it is easy to see by combining explicitly the eight extreme vertices $\mathbf{v}_\ell - \mathbf{n}_\ell, \mathbf{v}_\ell + \mathbf{n}_\ell$ $(\ell = i,j,k,l)$ of D_{ijkl} with three linear BB basis in the variables λ,u,v:

$$\begin{aligned}
\mathbf{p} &= \frac{1}{2}(1+\lambda)(1-u)(1-v)(\mathbf{v}_i+\mathbf{n}_i) + \frac{1}{2}(1-\lambda)(1-u)(1-v)(\mathbf{v}_i-\mathbf{n}_i) \quad (8.16)\\
&+ \frac{1}{2}(1+\lambda)u(1-v)(\mathbf{v}_j+\mathbf{n}_j) + \frac{1}{2}(1-\lambda)u(1-v)(\mathbf{v}_j-\mathbf{n}_j)\\
&+ \frac{1}{2}(1+\lambda)uv(\mathbf{v}_k+\mathbf{n}_k) + \frac{1}{2}(1-\lambda)uv(\mathbf{v}_k-\mathbf{n}_k)\\
&+ \frac{1}{2}(1+\lambda)u(1-v)(\mathbf{v}_l+\mathbf{n}_l) + \frac{1}{2}(1-\lambda)u(1-v)(\mathbf{v}_l-\mathbf{n}_l)
\end{aligned}$$

where $\mathbf{p}\begin{pmatrix} x & y & z \end{pmatrix}^\top$. Let

$$F(\mathbf{v}_i(\lambda)) = \lambda, \qquad \nabla F(\mathbf{v}_i(\lambda)) = N_i.$$

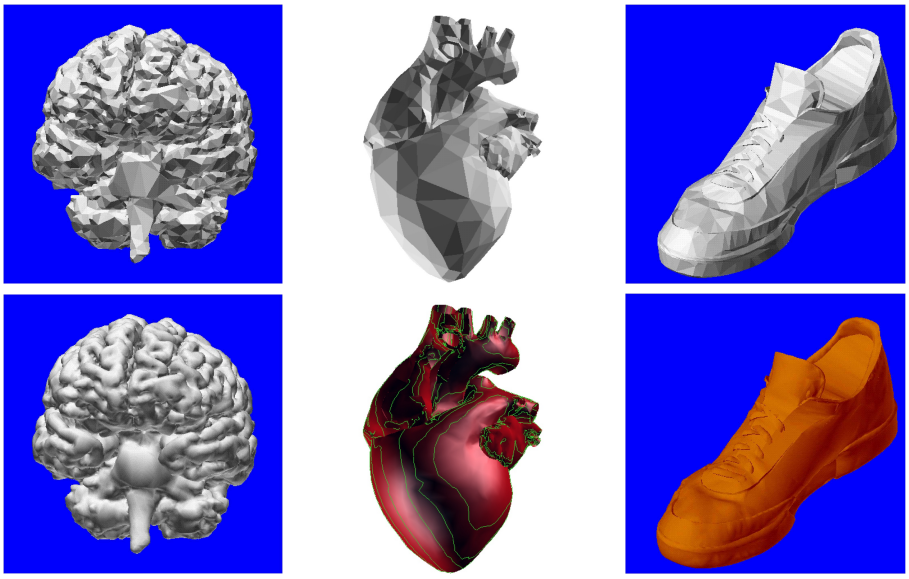


Figure 8.4: (Top) Original triangulated models (Bottom) A-patch model. Images courtesy of CVC at ICES.

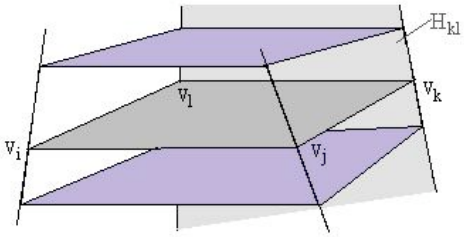


Figure 8.5: A prism D_{ijkl} constructed based on the quadrangular $[\mathbf{v}_i\mathbf{v}_j\mathbf{v}_k\mathbf{v}_l]$ [BX01].

Note $D_{N_i}F = N_i^\top \nabla F$ holds on the edge, where $N_i^\top$ is the transpose of N_i, $D_{N_i}F$ denotes the directional derivative of F in the direction N_i.

Let

$$\begin{aligned} & F_{lm}(t,\lambda) \\ = \; & F(\mathbf{v}_l(\lambda))H_0^3(t) + F(\mathbf{v}_m(\lambda))H_2^3(t) \\ & +[\mathbf{v}_m(\lambda) - \mathbf{v}_l(\lambda)]^\top \nabla F(\mathbf{v}_l(\lambda))H_1^3(t) \\ & +[\mathbf{v}_m(\lambda) - \mathbf{v}_l(\lambda)]^\top \nabla F(\mathbf{v}_m(\lambda))H_3^3(t), \end{aligned} \tag{8.17}$$

with

$$\begin{aligned} H_0^3(t) &= 1 - 3t^2 + 2t^3, & H_1^3(t) &= t - 2t^2 + t^3, \\ H_2^3(t) &= 3t^2 - 2t^3, & H_3^3(t) &= -t^2 + t^3. \end{aligned}$$

Let

$$\begin{aligned} d_1(\lambda) &= \mathbf{v}_m(\lambda) - \mathbf{v}_l(\lambda), \\ d_2(t) &= (1-t)N_l + tN_m, \\ d_3(t,\lambda) &= d_1 \times d_2. \end{aligned}$$

Then we define the gradient $\nabla F_{lm}(t,\lambda)$ by the following conditions:

$$\begin{cases} d_1^\top \nabla F_{lm}(t,\lambda) = \dfrac{\partial F_{lm}(t,\lambda)}{\partial t}, \\ d_2^\top \nabla F_{lm}(t,\lambda) = \dfrac{\partial F_{lm}(t,\lambda)}{\partial \lambda}, \\ d_3^\top \nabla F_{lm}(t,\lambda) = d_3^\top \nabla \check{F}_{lm}(t,\lambda), \end{cases} \tag{8.18}$$

where

$$\nabla \check{F}_{lm}(t,\lambda) = (1-t)\nabla F(\mathbf{v}_l(\lambda)) + t\nabla F(\mathbf{v}_m(\lambda)).$$

Since

$$\|d_3\|^2 [d_1, d_2, d_3]^{-1} = \left[d_1 \|d_2\|^2 - d_2(d_1^\top d_2), d_2 \|d_1\|^2 - d_1(d_1^\top d_2), d_3 \right]^\top,$$

the Equation (8.18) implies

$$\begin{aligned} \nabla F_{lm}(t,\lambda) = \; & \frac{1}{\|d_3\|^2} \{ \left[d_1 \|d_2\|^2 - d_2(d_1^\top d_2) \right] P \\ & + \left[d_2 \|d_1\|^2 - d_1(d_1^\top d_2) \right] Q + d_3 R \}, \end{aligned}$$

where

$$\begin{aligned} P(t,\lambda) &= \frac{\partial F_{lm}(t,\lambda)}{\partial t}, \\ Q(t,\lambda) &= \frac{\partial F_{lm}(t,\lambda)}{\partial \lambda}, \\ R(t,\lambda) &= d_3^\top \nabla \check{F}_{lm}(t,\lambda). \end{aligned}$$

Therefore we have

$$\begin{aligned} F_u(u,v,\lambda) &= H_0^3(u)F_{14}(v,\lambda)+H_1^3(u)d_u(v,\lambda)^\top\nabla F_{14}(v,\lambda) \\ &+ H_2^3(u)F_{23}(v,\lambda)+H_3^3(u)d_u(v,\lambda)^\top\nabla F_{23}(v,\lambda), \end{aligned}$$

$$\begin{aligned} F_v(u,v,\lambda) &= H_0^3(v)F_{12}(u,\lambda)+H_1^3(v)d_v(u,\lambda)^\top\nabla F_{12}(u,\lambda) \\ &+ H_2^3(v)F_{43}(u,\lambda)+H_3^3(v)d_v(u,\lambda)^\top\nabla F_{43}(u,\lambda), \end{aligned}$$

where

$$d_u(v,\lambda)=H_{23}(v,\lambda)-H_{14}(v,\lambda),$$
$$d_v(u,\lambda)=H_{43}(u,\lambda)-H_{12}(u,\lambda).$$

Then we define

$$F^{(\sigma)}(u,v,\lambda) = \begin{cases} F_u(u,v,\lambda) \\ F_v(u,v,\lambda) \\ \dfrac{\omega_u F_u(u,v,\lambda)+\omega_v F_v(u,v,\lambda)}{\omega_u+\omega_v} \end{cases} \qquad u,v\in\{0,1\}, \quad (8.19)$$

where $\omega_u=[(1-v)v]^2,\ \omega_v=[(1-u)u]^2$.
Note that the function $F^{(\sigma)}$ is C^1 within each of the volumes, since the gradient on the faces of the volumes is C^1 and the correction terms is C^1 in the volume.

8.3 Examples

The computation of electrostatic solvation energy (also known as polarization energy) for bio-molecules plays an important role in molecular dynamics simulations. When computing the electro- static solvation energy for bio-molecules, it is crucial to correctly model the molecular surface (MS). A smooth algebraic spline model approximation of the MS is built, based on an decimated triangulation derived from the atomic coordinate information of the bio-molecule, resident in the PDB (Protein data bank). First a triangular prism scaffold P_s covering the PDB structure is constructed, and, then piecewise polynomial Bernstein-Bezier (BB) spline function approximation F within P_s , which are nearly C^1 everywhere, is generated. Approximation error and point sampling convergence bounds may also be computed. An implicit algebraic spline model of the MS which is free of singularity, is extracted as the zero contours of F . Furthermore, this provides a polynomial parameterization of the implicit MS, which allows an efficient point sampling on the MS, and thereby it may be used to simplify the accurate estimation of integrals needed for electro-static solvation energy calculations [ZXB07]. The steps of this construction are shown in figure 8.6 for molecule 1HIA(b) chain B and Y

of the Kallikrein protein. The initial hard-sphere molecular model and the resulting SES model using prism algebraic patches are shown in figure 8.7 and 8.8 for 1CGI(b) human pancreatic secretory trypsin inhibitor (kazal type) variant and 1PPE(b) the Trypsin inhibitor CMT-1.

As an example of prism shell A-patches in figure 8.9 the reconstruction from CT scans of the knee joint is shown, notice how with a single patch both the inner and outer surface of the bone is modeled[BX99b].

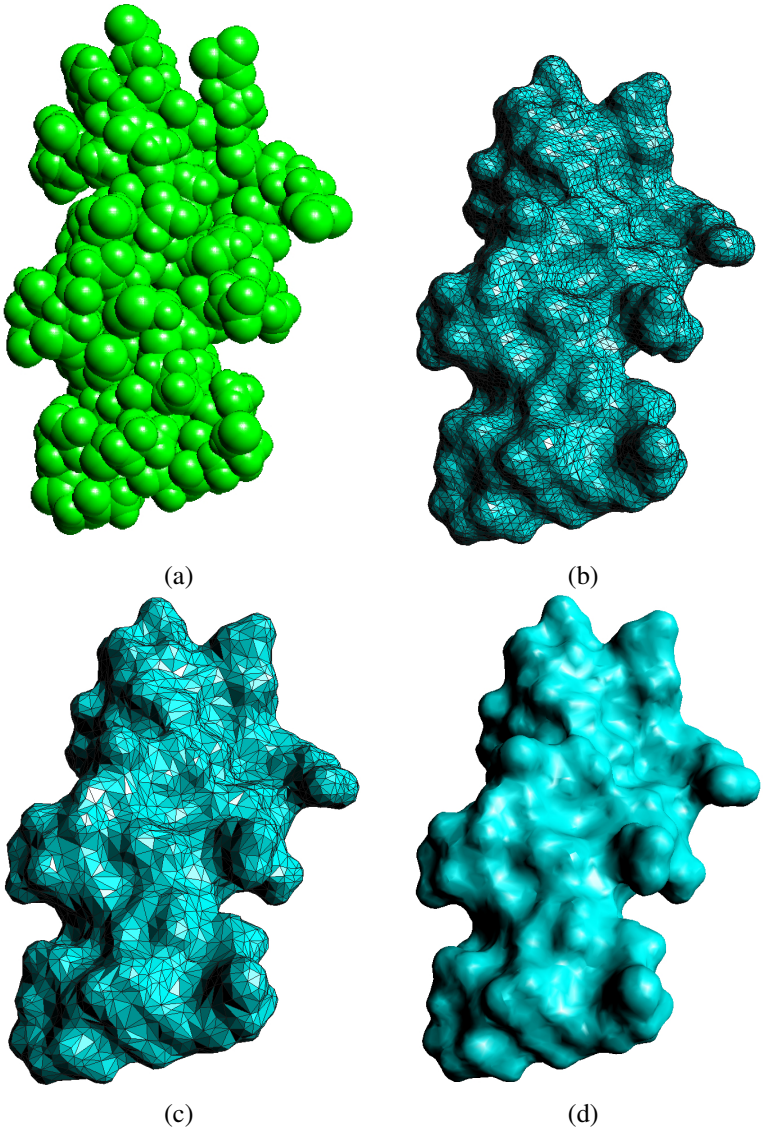
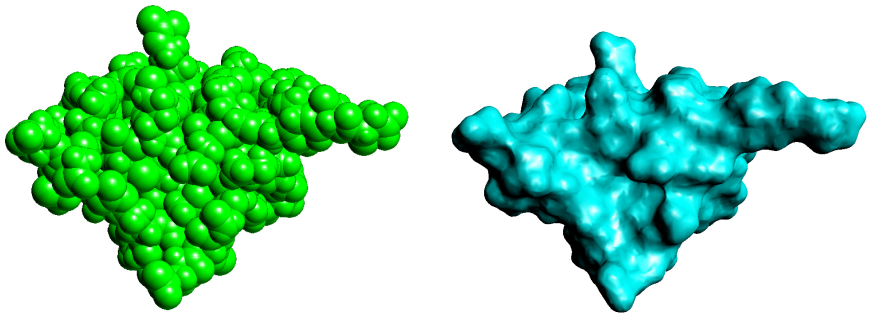
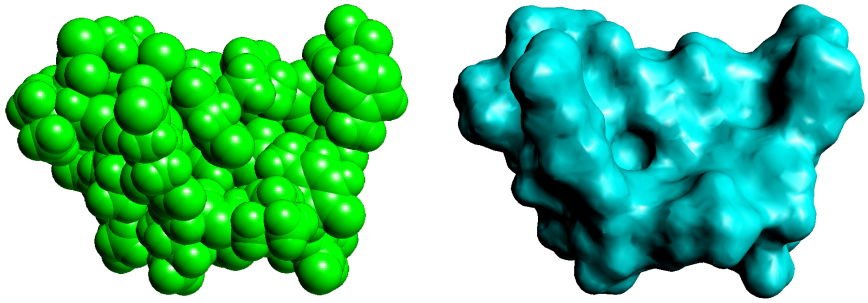


Figure 8.6: Molecular model of a protein(1HIA-B). (a) The van der Waals surface of the atomic structure (693 atoms); (b) The initial triangulation of the solvent excluded surface (SES) (27480 triangles); (c) The decimated triangulation of the SES (7770 triangles); (d) The piecewise algebraic surfaces patches (7770 patches) generated from the decimated triangulation of SES. [ZXB07]



(a) (b)

Figure 8.7: Molecular model of a protein(1CGI-B). (a) The van der Waals surface of the atomic structure; (b) The piecewise algebraic surfaces patches generated from the decimated triangulation of SES. [ZXB07]



(a) (b)

Figure 8.8: Molecular model of a protein(1PPE-B). (a) The van der Waals surface of the atomic structure; (b) The piecewise algebraic surfaces patches generated from the decimated triangulation of SES. [ZXB07]

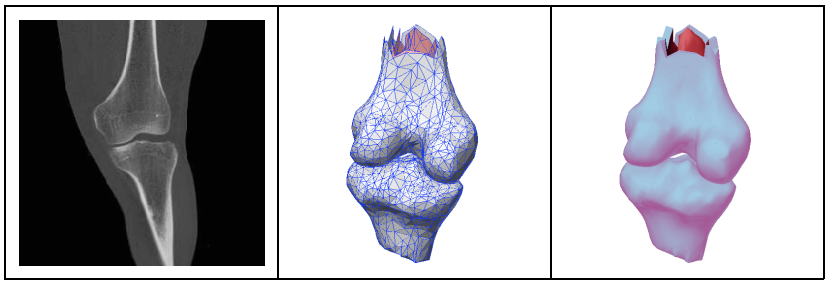


Figure 8.9: Left: CT scan of a human knee. Middle: A pair of extracted C^0 iso-surfaces for the knee skeleton boundaries. Right: Smooth fat surface reconstruction of the knee skeleton [BX01].

Chapter 9

Boolean Operations

Boolean operation are the set-theoretical operations (Union, Intersection, Difference, containment, etc) between the point sets described by two geometric models. Boolean operation between geometrical models have the critical step of computing the intersection of two elements. The intersection of two varieties is well defined in algebraic geometry as it is simply the ideal generated by the defining equations. However the exact boolean operator is often not closed inside the domain of fixed degree A-patches. The reason is simply that the intersection of two algebraic surfaces of degree d (say cubic) is in general, a space curve of degree d^2. It is possible however to build a piecewise low-degree topologically correct approximate, in case of tetrahedral and prismatic patches a piecewise linear tracing of the intersection curve[BX97]. This tracing is the starting point of the splitting of each geometric element along the approximate intersection and the classification of each generated sub-element as either belonging to one and/or the other model. The final step is to assemble the result depending on the wanted operation.

In this chapter a symbolic-numeric algorithm for Boolean operations is discussed, closed in the algebra of curved polyhedra whose boundary is triangulated with algebraic patches (A-patches). This approach uses a linear polyhedron as a first approximation of both the arguments and the result. On each triangle of a boundary representation of such linear approximation, a piecewise cubic algebraic interpolant is built, using a C^1-continuous prism algebraic patch (prism A-patch) that interpolates the three triangle vertices, with given normal vectors. The boundary representation only stores the vertices of the initial triangulation and their external vertex normals. In order to represent also flat and/or sharp local features, the corresponding normal-per-face and/or normal-per-edge may be also given, respectively. The topology is described by storing, for each curved triangle, the two triples of pointers to incident vertices and to adjacent triangles. For each triangle, a scaffolding prism is built, produced by its extreme vertices and normals, which provides a containment volume for the curved interpolating A-patch. When looking

for the result of a regularized Boolean operation, the 0-set of a tri-variate polynomial within each such prism is generated, and intersected with the analogous 0-sets of the other curved polyhedron, when two prisms have non-empty intersection. The intersection curves of the boundaries are traced and used to decompose each boundary into the 3 standard classes of subpatches, denoted *in*, *out* and *on*. While tracing the intersection curves, the locally refined triangulation of intersecting patches is produced, and added to the boundary representation.

This chapter is an expanded and detailed version of [BPP$^+$08]. The prototype software implementing the algorithm described herein has been developed by the author basing it on the prism patch implementation of Wenqi Zhao [ZXB07]. The structure of the chapters is as follows. In Section 9.1 an overall introduction and some useful concepts are recalled and preliminary definitions are given. Prism patches have been introduced in chapter 3 and the formulation of chapter 8 will be used here. In Section 9.2 we give a synthetic introduction to Boolean algorithms between curved triangulated solids. In Section 9.3 we discuss the implemented intersection between prismatic A-patches, with tracing of intersection curves and local triangulation refinement. In the examples and applications Section 9.4 we show the results of our prototype implementation, including the computation of the Boolean union of a docked complex of two proteins containing several thousand prism A-patches.

9.1 Introduction and Background

A-Patches are smooth algebraic surface patch families, defined using a fixed degree trivariate polynomial within a compact polyhedron domain (also called the patch scaffold). Simple A-patches use a tetrahedron, or a cube, or a triangular prism scaffold. An exact union or intersection Boolean operator is often not closed inside the domain of fixed degree A-patches. The reason is simply that the intersection of two algebraic surfaces of degree d (say cubic) is in general, a space curve of degree d^2, (i.e. nine). In this chapter, an efficient solution is provided, using a clever combination of symbolic and geometric methods, for the subproblems below:

(i) The robust computation of the intersection of a pair of algebraic curves, and/or surfaces,

(ii) The decomposition of the explicit union of a pair of A-patches, into a small set of A-patches of the same scaffold type,

The combination of (i) and (ii) provides a union operator capable of reproducing A-patches maintaining the topology of the exact solution. These methods are currently being implemented in the context of the PLaSM Language [Pao03, Pa] and using the Ganith Algebraic Toolkit [BR90b, Por07b]. For the current status on the integration

between these two software please refer to appendix A. The computation of the union of A-patches is a necessary step for the development of Boolean set operations on algebraic finite elements in an integrated software package.

The main assumption that is made here is that the boundary of the solid is triangulated by curved algebraic patches, where each patch is single-sheeted and contained within the prismatic scaffold defined on the triangles of the coarse linear approximation of the solid boundary. This assumption reduces the Boolean problem with curved solids to the much easier and well-known problem with linear polyhedra (see, e.g. [Bra78, LTH86, Man88, Hof89, Mas93]). For the same reason, we may represent quite complex curved solids using the very simple winged-triangle (WT) boundary representation, which does not require the use of Euler operators [EW79]. Such winged representation was introduced in [PSR89], and later generalized to solid decompositions and higher dimensional manifolds in [PBCF93].

Boolean Operators

Before proceeding further some definitions on what boolean algebra and regularized boolean operators are needed.

Boolean Algebra The domain of the representation scheme [Req80] discussed in this paper is the Boolean algebra of solid polyhedra with bounded boundary and where complementation simply exchanges the interior with the exterior. It is easy to see that it verifies all axioms of a Boolean algebra, and that it also has to verify all theorems known for a Boolean algebra.

In particular, we may evaluate the intersection and difference of polyhedra from their union and complements, using the notion that conjunction is dual to disjunction by way of the De Morgan's laws, *i.e.* $\neg(P_1 \cap P_2) = \neg P_1 \cup \neg P_2$ and $\neg(P_1 \cup P_2) = \neg P_1 \cap \neg P_2$. These can also be constructed as definitions of union in terms of intersection and vice versa: $P_1 \cap P_2 = \neg(\neg P_1 \cup \neg P_2)$ and $P_1 \cup P_2 = \neg(\neg P_1 \cap \neg P_2)$. The difference can also be constructed using either union and complement $P_1 - P_2 = \neg(\neg P_1 \cup P_2)$ or intersection and complement $P_1 - P_2 = P_1 \cap \neg P_2$. To complement any boundary representation of a solid is $O(n)$, since it is sufficient to reverse the direction of all the normal vectors to 0- and to 2-cells.

Regularized Operations Also, it is well-known and easy to see (e.g. [Req80]) that a Boolean operation between regular polyhedra may give a non regular result, i.e. a point set with subsets of different dimensions. For this reason the concept of *regularized Boolean* was introduced. If $op \in \{\cup, \cap, -\}$, then the *regularized* op^* is defined as:

$$op^* := \text{clos} \circ \text{int} \circ op$$

so that, for every solid pair of the same dimension P_1 and P_2, it is

$$P_1 \; op^* \; P_2 := \text{clos}\,(\text{int}\,(P_1 \; op \; P_2))$$

Where clos and int respectively denote topological *closure* and *interior*. Therefore, in the following of this paper we only discuss algorithms for *regularized boolean* operations.

In brief, some previous work on boolean operators and trimming of curved free form surfaces is: interactive boundary computation of Boolean combinations of sculptured solids [KGMM97]; netbased modeling [Lin00]; approximate Boolean operations on free-form solids [BKZ01]; ESOLID – a system for exact boundary evaluation [KCF+02]; adaptive trimming of cubic triangular Bézier patches [GMF06].

Scaffold Data Structure

To represent in memory the polyhedral envelope that contains the A-patches which cover the boundary of a curved solid, we use the WT (winged-triangle) [PSR89] representation scheme, i.e. a boundary representation based on vertices and triangles (0- and 2-simplices) of a simplicial complex that triangulates the object boundary. This scheme provides a relational representation with tuples of constant length, that allows multishell polyhedra to be dealt with, regardless of the topological genus of their boundary surfaces. Nonmanifold objects are represented by duplicating some adjacency information. The space of mathematical models represented by the WT scheme is quite extensive. It coincides with the set of regular 3-polyhedra which are possibly: unconnected; unbounded (but with bounded boundary); non-manifold; multishell; with multiply connected faces, i.e. with multiple edge loops.

In particular, the WT representation can be implemented as a table in first normal form, where each tuple, indexed by the t_j triangle, contains (see figure 9.1):

- 3 references to incident vertices;
- 3 references to adjacent triangles.

This scheme is characterized by the design choice of making no use of Euler operators. This choice is allowed by the extreme simplicity of the data structure representing the triangulation of the boundary. If all references to adjacent triangles are valid, then the WT representation automatically fulfills the requirement for closed surfaces, that is, each edge (1-cell) is a face of exactly two triangles (2-cells). Furthermore, if the geometry has more components (shells), each component may be easily extracted in $O(n)$ time, where n is the number of triangles. A WT may be built easily from the common "raw" format for a triangulation, that is, a series of points (with optional normals) and a series of triangles represented by three references to the points. The algorithm to do so is very simple, linear in the number of triangles and may also be used to check if the "raw" format is a valid closed surface triangulation:

1. Build an initially empty hash table associating edges (pair of points) to a triangle.

2. For each processed triangle one checks each edge in the hash table. If it is present then:

   a) fill the current triangle adjacent reference with the triangle in the hash table;

   b) fill the triangle in the hash table adjacent reference with the current triangle;

   c) delete the entry in the hash table (edge used).

   If it is not present then insert the edge and current triangle in the hash table.

3. At the end all references to adjacent triangles must be filled; all points must have been used; and most importantly the hash table must return to the empty state.

Almost all boolean algorithms on geometric representations based on finite elements have two main elements:

- A method to intersect two elements and split into smaller elements along the intersection. Usually this step is highly sophisticated under the mathematical point of view but has a very simple logic.

- An algorithm that guides the method for intersection/split on paired elements of the input geometries and assembles the final result according the wanted operation. This step usually employs very simple mathematic methods but has a complicated combinatorial logic.

The WT supports a very straightforward Boolean algorithm, discussed in the following section 9.2, where the consistency of the boundary simplicial complex is easily maintained. The method for intersecting and splitting two algebraic patches will be discussed in section 9.3.

9.2 Boolean Algorithm for Boundary Representation

Several approaches to the computation of Boolean operators using a boundary representation can be found in the solid modeling literature (see, e.g. [Bra78, Hof89, LTH86, Man88, Mas93]). The very simple Boolean algorithms defined with the winged-triangle (WT) boundary representation, that does not require the use of Euler operators [EW79], can be found in [PSR89].

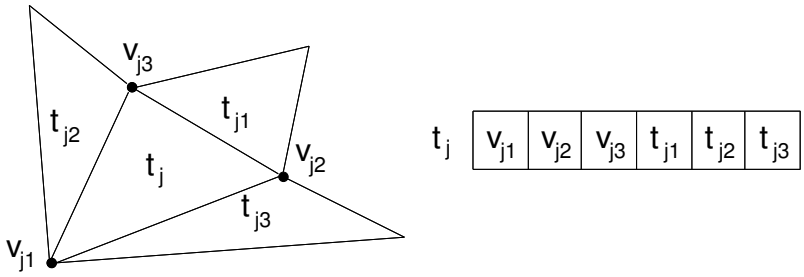


Figure 9.1: Winged edge representation [PSR89].

Outline of the Algorithm Assume one has a method for intersecting two patches, splitting the patches along the intersection, and classify the new patches w.r.t. the intersection. The basic idea of the algorithm is that two shells of the input geometry either intersect or don't intersect. In the former case two shells intersect if at least one pair of patches of the two shells intersect. Then the split and the classification is propagated to all the shell using the patch adjacency information and the shell is split into open surfaces along the intersection. In the later case they either are unrelated or contained one in the other. At this point the algorithm may produce the output for the wanted operation by selecting the appropriately classified surfaces and gluing them.

The main assumption we make is that the boundary of the solid is triangulated by curved algebraic patches, where each patch is single-sheeted and contained within the prismatic scaffold defined by one or two (adjacent) triangles of the linear approximation of the solid boundary. This assumption reduces the Boolean problem with curved solids to the much easier and well-known problem with linear polyhedra.

The algorithm for the regularized set union of two regular polyhedra P_1, and P_2 is composed of two main steps (see [PSR89]). In the first step, each shell of P_1 is intersected with every shell of P_2, where a boundary shell is a maximal connected subset of the boundary. The result of this computation is the space curve where the boundary surfaces of the input polyhedra intersect. In general, this space curve is nonplanar and may be disconnected, i.e. it may contain more than one cycle (connected component). The second step of the union algorithm reconstructs the output boundary $\partial(P_1 \cup P_2)$ by subdividing the input shells in alternate patches along the intersection curve previously determined, and by choosing which subpatches must be collected into the result.

The intersection curve subdivides each intersecting shell into no more than three disjoint open surfaces: the surface *outside*, *inside* and *on the boundary* of the other polyhedron. The part of the boundary of the resulting polyhedron, generated by the intersecting shells, is very easy to set up: the corresponding algorithm is detailed in the paragraph on

partitionable shells. The paragraph on *isolated shells* analyses how to evaluate the part of the boundary of the union where input shells do not intersect. Such isolated shells may or may not appear in the resulting boundary: to discard or to collect them in the Boolean result will depend on their position with respect to the polyhedron to which they do not belong.

Shell Extraction The Boolean operations are closed in the set of polyhedra only if one may regard a polyhedron as unconnected. Even a connected polyhedron may have an unconnected boundary. As an example consider an empty sphere, with an external and an internal boundary shell.

In order to perform a Boolean operation, say $P_1 \cup P_2$, the shells of both arguments must be preliminarly extracted from the data structure used as their computer representation. A very simple recursive algorithm working in linear time may be used with the WT representation, starting a new shell as containing a first boundary triangle, marking it, and extracting recursively the unmarked adjacent triangles. When the extraction stops, a shell (a maximal 1-connected boundary subset) has been extracted. A new shell can then be started from the first non marked triangle in the data structure, if any, and so on, until no more unmarked triangles remain. Let us denote with $\mathscr{S}_1$ and $\mathscr{S}_2$ the sets of shells of P_1 and P_2, respectively.

Shell Intersection Each pair of shells $(s_i, s_j) \in \mathscr{S}_1 \times \mathscr{S}_2$ must be checked for empty intersection, using both global tests, e.g. checking for intersection of the containment boxes or spheres of s_i and s_j, and local tests when the former do not succeed. A local test will check for patch intersection, discussed in the next paragraph.

A shell of P_1 which does not intersect any shell of P_2 is referred to as an *isolated* shell. A shell of P_1 that intersects at least one shell of P_2 is referred to as a *partitionable* shell. Let $\mathscr{I}$ and $\mathscr{P}$ be two sets containing the isolated and partitionable shells of P, respectively, so that $\mathscr{S} = \mathscr{I} \cup \mathscr{P}$. A shell of P_1 moves from the set $\mathscr{I}_1$ (initially set to $\mathscr{S}_1$) to the set $\mathscr{P}_1$ (initially set to $\emptyset$) when one of its patches gives a *valid* intersection with some patch of P_2.

An intersection between two curved triangles (i.e. two A-patches) is said *valid* when there exists points of P_1 that belong to both the *below* and the *above* subspaces of P_2, and viceversa, i.e. there exists points of P_2 that belong to both the *below* and the *above* subspaces of P_1. In other words, an intersection between two patches is valid when they really cross each other, and do not simply touch (i.e. intersect) on a subset of points (topologically: a cell) of whatever dimensionality.

Patch Intersection and Splitting When a valid intersection between two A-patches occurs, the intersection curve is inserted in both input A-patches, by splitting the support

triangles t_1 and t_2—and possibly their adjacent triangles—so as to split the patches into subpatches supported by two new subsets of triangles, pairwise containing a segment of the intersection curve as the common edge. The algebraic formulation of the intersection problem; the numerical tracing of the intersection curve, and the final triangle splitting are discussed in Section 9.3.

Partitionable Shells The intersection curves subdivide the partitionable shells into no more than three sets, which contain the surface patches that are internal, external and on the boundary of the other polyhedron, respectively. Such surfaces must be alternatively chosen to collect the whole boundary of the resulting polyhedron. In the general case, the envelope of the polyhedron $P_1 \cup P_2$ will be composed by:

1. all the boundary patches of P_1 external to P_2 (and vice versa);
2. *some* of the patches laying on the boundary of both input polyhedra. In particular, the only common patches with same orientation of the external normals;
3. *some* of the nonintersecting input shells, that we named isolated patches (see the next paragraph).

Isolated Shells An isolated shell may appear or may not appear in the resulting solid $P_1 \cup P_2$: it will be collected in the final result only if it resides in the exterior of the object to which it does not belong. More precisely: given a shell $s \in \mathscr{I}_1$ (respectively $\mathscr{I}_2$), where $\mathscr{I}_1$ is the set of isolated shells of P_1 (respectively P_2), then $s \subseteq \partial(P_1 \cup P_2)$ if and only if it is external with respect to P_2 (respectively P_1). The problem of classifying a shell versus a nonintersecting polyhedron is reduced to the classification of a single shell point. To solve efficiently this classification problem with respect to every cardinality of the set of patches, some sort of spatial index (say, a BSP-tree) would be very useful.

To sum things up the algorithm for a boolean (i.e. union) operator in pseudocode is:

1. Extract the shell sets S_1 and S_2 of polyhedra P_1 and P_2;
2. For each shell pair $(s_i, s_j) \in S_1 \times S_2$:
   a. If (s_i, s_j) intersect then partition s_i and s_j into (s_i^+, s_i^-), (s_j^-, s_j^+) and s_{ij}^0;
      Else store s_i and s_j into I_1, I_2.
3. Assemble the result R (i.e. glue s_i^+ and s_j^+ along s_{ij}^0);
4. Add each external isolated shell of I_1 and of I_2)to the result R.

Step 2.a may further be detailed as:

2.a. If the bounding boxes or spheres of $(s_i, s_j) \in S_1 \times S_2$ intersect:

i. For each patch pair $(t_k, t_l) \in s_i \times s_j$, if intersect:

i. For all the intersection curves compute an A-spline approximation $t_{kl}{}^0$;

ii. Partition t_k and t_l into A-patch sets $t_k{}^+$, $t_k{}^-$ and $t_l{}^-$, $t_l{}^+$, respectively;

ii. If no intersecting patch pairs are found, then store s_i and s_j into I_1, I_2.

Using the adjacency information:

i. Link all $t_{kl}{}^0$ into loops;

ii. Propagate the partitioning and classification to all non intersecting patches t_k of s_i (and t_l of s_j);

iii. Create the new adjacency information.

9.3 Boolean Operators on an Algebraic Finite Elements

Given two solids, represented by a boundary representation of prismatic A-patches, the Boolean algorithm in section 9.2 has the two critical steps: (1) computation of the intersection of two patch elements; and; (2) split along the intersection into a new patch sets. In this section we describe an approximate solution. The prismatic A-patch intersection may be further decomposed into:

- algebraic formalization of the intersection curve;
- topologically sound piecewise-linear tracing of the intersection curve between two patches;
- construction of new prism (sub)patches along the approximate intersection.

The Boolean algorithm in previous section will then select the appropriate (sub)patches to compute a Boolean operation.

Algebraic Formulation

To formalize the problem of the patch to patch intersection, one proceeds by writing down all the relevant equations and then algebraically simplify and transform them into a "convenient form".

Equations Consider two prism A-patches (Fig. 9.2) **A** and **B**. Patch **A** (respectively **B**) is built on prisms **D** (**E**) of vertexes $\mathbf{v_i}, \mathbf{v_j}, \mathbf{v_k}$ ($\mathbf{w_i}, \mathbf{w_j}, \mathbf{w_k}$) and normals $\mathbf{n_i}, \mathbf{n_j}, \mathbf{n_k}$ ($\mathbf{o_i}, \mathbf{o_j}, \mathbf{o_k}$). Call the prism's local coordinates, $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \alpha_3)$ ($\boldsymbol{\beta} = (\beta_1, \beta_2, \beta_3)$) for the barycentric part, and, λ (μ) for the height along the interpolated normal $\mathbf{n}(\boldsymbol{\alpha}) = \alpha_1\mathbf{n_i} + \alpha_2\mathbf{n_j} + \alpha_3\mathbf{n_k}$ ($\mathbf{o}(\boldsymbol{\alpha}) = \ldots$). As shown in equation 8.1, the global coordinates generic point in the prism is $\mathbf{v} = \mathbf{D}(\lambda)\boldsymbol{\alpha}$ ($\mathbf{w} = \mathbf{E}(\mu)\boldsymbol{\beta}$). The surface patch is defined as the zero-contour of scalar

field $F(\boldsymbol{\alpha},\lambda)=0$ ($G(\boldsymbol{\beta},\mu)=0$) (see also eq. 8.2) . The scalar field may be written in B.B. form as

$$F(\boldsymbol{\alpha},\lambda)=\sum_{i+j+k=n} a_{ijk}(\boldsymbol{\alpha},\lambda)B^n_{ijk}(\boldsymbol{\alpha}) \quad \left(resp.\ G(\boldsymbol{\beta},\mu)=\sum_{i+j+k=n} b_{ijk}(\boldsymbol{\beta},\mu)B^n_{ijk}(\boldsymbol{\beta})\right) \ .$$

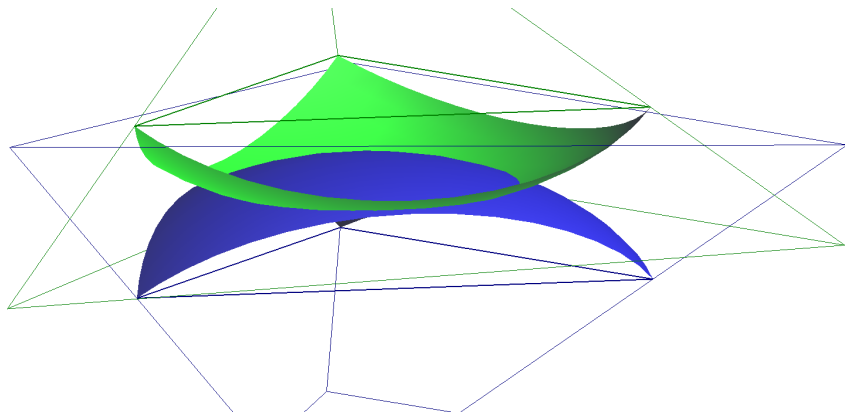


Figure 9.2: Two intersecting patches. Image generated by the author using the prototype software [BPP+08].

Given a certain point $\boldsymbol{\alpha}$ ($\boldsymbol{\beta}$) in the barycentric coordinate, by root finding and selecting the $\lambda \in I_\lambda$, one has a procedural parametric formulation of the surface (see also eq. 8.14)

$$\lambda=\lambda_F(\boldsymbol{\alpha}) \quad (\mu=\mu_G(\boldsymbol{\beta})) \ .$$

Restricting the attention to cubic A-patches, the B.B. coefficients are uniquely determined by the prism data (see chapter.section 8.1). With the sole exception of a_{111} (b_{111}), the coefficients are simple linear expression $a_{ijk}(\lambda)=a^{(1)}_{ijk}\lambda+a^{(0)}_{ijk}$. The boundary curves of a patch have a simple rational parametric form (ie. $\alpha_2=0$ and $\alpha_3=1-\alpha_1$) :

$$\lambda=\lambda_F(\alpha_1)=\frac{\sum_{i+j=n} a^{(0)}_{ijk}B^n_{ij}(1-\alpha_1,\alpha_1)}{\sum_{i+j=n} a^{(1)}_{ijk}B^n_{ij}(1-\alpha_1,\alpha_1)} \quad \left(\mu=\mu_G(\beta_1)=\frac{\sum_{i+j=n} b^{(0)}_{ijk}B^n_{ij}(1-\beta_1,\beta_1)}{\sum_{i+j=n} b^{(1)}_{ijk}B^n_{ij}(1-\beta_1,\beta_1)}\right) \tag{9.1}$$

The intersection between the two surfaces of the patches is implicitly defined by the two surface equations and by equating the coordinate transformation of the two prisms. The coordinate transformation equation is vectorial in a 3D space thus counts as 3 equations. The intersection is a system of 5 non-linear equations in 6 variables thus describing a

curve.

$$\left\{\begin{array}{lcl} \mathbf{D}(\lambda)\begin{pmatrix}1\\ \alpha_1\\ \alpha_2\end{pmatrix} & = & \mathbf{E}(\mu)\begin{pmatrix}1\\ \beta_1\\ \beta_2\end{pmatrix} \\ F(\boldsymbol{\alpha},\lambda) & = & 0 \\ G(\boldsymbol{\beta},\mu) & = & 0 \end{array}\right. \tag{9.2}$$

The objective is to reduce the system 9.2 in a minimal number of equations involving only the barycentric coordinates of one of the two prisms (i.e. $\mathbf{E}$).

Coordinate Transformation Inversion The first focus is to use the first row (the coordinate transformation equation) of the system 9.2 in order to convert coordinates from one prism to the other. Consider the coordinate transformation of prism $\mathbf{D}(\lambda)$ as a λ-family of linear transformations and invert keeping λ symbolic

$$\mathbf{D}(\lambda)^{-1} = \frac{1}{|\mathbf{D}(\lambda)|}\begin{pmatrix} (\mathbf{v_j}(\lambda)-\mathbf{v_i}(\lambda))\times(\mathbf{v_k}(\lambda)-\mathbf{v_i}(\lambda)) \\ (\mathbf{v_k}(\lambda)-\mathbf{v_i}(\lambda))\times\mathbf{v_i}(\lambda) \\ \mathbf{v_i}(\lambda)\times(\mathbf{v_j}(\lambda)-\mathbf{v_i}(\lambda)) \end{pmatrix} .$$

One may then compose with $\mathbf{E}(\mu)$

$$\begin{pmatrix}1\\ \alpha_1\\ \alpha_2\end{pmatrix} = \mathbf{D}(\lambda)^{-1}\mathbf{E}(\mu)\begin{pmatrix}1\\ \beta_1\\ \beta_2\end{pmatrix} . \tag{9.3}$$

By bringing $|\mathbf{D}(\lambda)|$ to the left in the first row one has an equation relating in $\boldsymbol{\beta},\lambda,\mu$. The second and third row define $\boldsymbol{\alpha_1}$ as rationals of $\boldsymbol{\beta},\lambda,\mu$. Denote these as $Q(\boldsymbol{\beta},\lambda,\mu)=0$ and $\boldsymbol{\alpha}(\boldsymbol{\beta},\lambda,\mu)$ respectivelly.
An operational example is, given any point $(\overline{\boldsymbol{\beta}},\overline{\mu})$ in $\mathbf{E}$ coordinates: (1) solve $Q(\overline{\boldsymbol{\beta}},\lambda,\overline{\mu})=0$ for $\overline{\lambda}\in I_\lambda$; (2) compute $\overline{\boldsymbol{\alpha}}=\boldsymbol{\alpha}(\overline{\boldsymbol{\beta}},\overline{\lambda},\overline{\mu})$.

There is a more intuitive, elementary and explicit approach though not mathematically synthetic and neither implementation efficient. Consider the prism $\mathbf{D}$ as a λ-family of planes $L(x,y,z;\lambda)=A(\lambda)x+B(\lambda)y+C(\lambda)z+D=0$ passing through points $\mathbf{v_i}(\lambda)=$

$\mathbf{v_i}+\lambda\mathbf{n_i}$, $\mathbf{v_j}(\lambda)=\mathbf{v_j}+\lambda\mathbf{n_j}$, $\mathbf{v_k}(\lambda)=\mathbf{v_k}+\lambda\mathbf{n_k}$:

$$
\begin{aligned}
A(\lambda) &= \begin{vmatrix} 1 & (\mathbf{v_i}+\lambda\mathbf{n_i})_y & (\mathbf{v_i}+\lambda\mathbf{n_i})_z \\ 1 & (\mathbf{v_j}+\lambda\mathbf{n_j})_y & (\mathbf{v_j}+\lambda\mathbf{n_j})_z \\ 1 & (\mathbf{v_k}+\lambda\mathbf{n_k})_y & (\mathbf{v_k}+\lambda\mathbf{n_k})_z \end{vmatrix} \\
B(\lambda) &= \begin{vmatrix} (\mathbf{v_i}+\lambda\mathbf{n_i})_x & 1 & (\mathbf{v_i}+\lambda\mathbf{n_i})_z \\ (\mathbf{v_j}+\lambda\mathbf{n_j})_x & 1 & (\mathbf{v_j}+\lambda\mathbf{n_j})_z \\ (\mathbf{v_k}+\lambda\mathbf{n_k})_x & 1 & (\mathbf{v_k}+\lambda\mathbf{n_k})_z \end{vmatrix} \\
C(\lambda) &= \begin{vmatrix} (\mathbf{v_i}+\lambda\mathbf{n_i})_x & (\mathbf{v_i}+\lambda\mathbf{n_i})_y & 1 \\ (\mathbf{v_j}+\lambda\mathbf{n_j})_x & (\mathbf{v_j}+\lambda\mathbf{n_j})_y & 1 \\ (\mathbf{v_k}+\lambda\mathbf{n_k})_x & (\mathbf{v_k}+\lambda\mathbf{n_k})_y & 1 \end{vmatrix} \\
D(\lambda) &= \begin{vmatrix} (\mathbf{v_i}+\lambda\mathbf{n_i})_x & (\mathbf{v_i}+\lambda\mathbf{n_i})_y & (\mathbf{v_i}+\lambda\mathbf{n_i})_z \\ (\mathbf{v_j}+\lambda\mathbf{n_j})_x & (\mathbf{v_j}+\lambda\mathbf{n_j})_y & (\mathbf{v_j}+\lambda\mathbf{n_j})_z \\ (\mathbf{v_k}+\lambda\mathbf{n_k})_x & (\mathbf{v_k}+\lambda\mathbf{n_k})_y & (\mathbf{v_k}+\lambda\mathbf{n_k})_z \end{vmatrix}
\end{aligned}
$$

Factor out the λ and define the plane family in λ:

$$
\begin{aligned}
A(\lambda) &= A_0+A_1\lambda+A_2\lambda^2 \\
B(\lambda) &= B_0+B_1\lambda+B_2\lambda^2 \\
C(\lambda) &= C_0+C_1\lambda+C_2\lambda^2 \\
D(\lambda) &= D_0+D_1\lambda+D_2\lambda^2+D_3\lambda^3
\end{aligned}
$$

$$
\begin{array}{cccccccccc}
 & A(\lambda)x & = & A_0x & + & A_1\lambda x & + & A_2\lambda^2 x & & \\
+ & B(\lambda)y & = & B_0y & + & B_1\lambda y & + & B_2\lambda^2 y & & \\
+ & C(\lambda)x & = & C_0z & + & C_1\lambda z & + & A_2\lambda^2 z & & \\
+ & D(\lambda) & = & D_0x & + & D_1\lambda & + & D_2\lambda^2 & + & D_3\lambda^3 \\
\hline
= & L(\lambda) & = & L_0 & + & L_1\lambda & + & L_2\lambda^2 & + & L_3\lambda^3
\end{array}
$$

Given a point $\overline{\mathbf{p}}=\overline{(x,y,z)}$ in global coordinates, finding the roots $\left\{\overline{\lambda}_i\right\}_{i\in\{1,2,3\}}$ of $L(\lambda)$ with $\overline{\lambda}\in I_\lambda$ will give the plane in the family passing through the point $\overline{\mathbf{p}}$ and thus $\overline{\lambda(x,y,z)}$ coordinate of the point in the prism. Once the $\overline{\lambda}$ is determined the $\boldsymbol{\alpha}$ coordinates are easily found inverting the baricentric coordinates in the triangle on points $\mathbf{v_i}(\overline{\lambda})=\mathbf{v_i}+\overline{\lambda}\mathbf{n_i}$, $\mathbf{v_j}(\overline{\lambda})=\mathbf{v_j}+\overline{\lambda}\mathbf{n_j}$, $\mathbf{v_k}(\overline{\lambda})=\mathbf{v_k}+\overline{\lambda}\mathbf{n_k}$ as shown in equation (2.2) (chapter 2).

Both $Q(\beta,\lambda,\mu)=0$ and $L(x,y,z;\lambda)=A(\lambda)x+B(\lambda)y+C(\lambda)z+D=0$ are generally cubic (however there are degenerate cases). To solve symbolically one may using Cardano's method, however it has not been possible to overcome the numerical instabilities of such method. Furthermore, until now the concept of selecting the appropriate root $\overline{\lambda}\in I_\lambda$ is easy if one is guaranteed that the point is on/near the surface and *inside* the

prism. In the boolean algorithm one has to test points from other surfaces and from other prisms. To overcome these difficulties the implementation uses a slightly different approach. Instead of defining the λ family of planes $L(x,y,z;\lambda)$ one defines the λ family of normals to these planes, which is quadratic:

$$\mathbf{n_L}(\lambda) = \big(\mathbf{v_j}(\lambda) - \mathbf{v_i}(\lambda)\big) \times (\mathbf{v_k}(\lambda) - \mathbf{v_i}(\lambda))$$

Given a point $\overline{\mathbf{p}} = \overline{(x,y,z)}$ in global coordinates the equation determining $\overline{\lambda}$ becomes the (generally) cubic:

$$E(\lambda) = \mathbf{n_L}(\lambda) \cdot (\overline{\mathbf{p}} - \mathbf{v_i}(\lambda)) = 0$$

Furthermore the the dot product of $nn(\lambda) = \mathbf{n_L}(\lambda) \cdot \mathbf{n_L}(0)$ is a (generally) quadratic function: The study of this function gives insight on the structure of the prism:

- If $deg(nn(\lambda))$ is zero the three normals are parallel to each other. In this case $E(\lambda)$ is linear.
- If the degree on $deg(nn(\lambda)) = 1$ then two normals are parallel and the plane family $L(\lambda)$ will "flip" once, $nn(\lambda) = 0$ will give the $\tilde{\lambda}$ coordinate of the "flip". In this case $E(\lambda)$ is quadratic. If $\tilde{\lambda} < 0$ the prism is *positive convex* and only $\tilde{\lambda} < \overline{\lambda} < \infty$ are considered. Otherwise it is *negative convex* and $-\infty < \overline{\lambda} < \tilde{\lambda}$.
- If the degree on $deg(nn(\lambda)) = 2$, $E(\lambda)$ is cubic and depending on the $discriminant(nn)$:
  - If $discriminant(nn) = 0$ the three normals will meet at one point and $L(\lambda)$ will "flip" there. $nn(\lambda) = 0$ will give the $\tilde{\lambda}$ coordinate of the "flip". The case is similar to the quadratic except $E(\lambda)$ is cubic. If $\tilde{\lambda} < 0$ the prism is *positive convex* and only $\tilde{\lambda} < \overline{\lambda} < \infty$ considered. Otherwise it is *negative convex* and $-\infty < \overline{\lambda} < \tilde{\lambda}$.
  - If $discriminant(nn) > 0$ the $nn(\lambda) = 0$ will give two $\tilde{\lambda}$ coordinate of the "flips". If both are negative, the prism is *positive convex* and $max(\tilde{\lambda}) < \overline{\lambda} < \infty$. If both are negative, the prism is *negative convex* and $-\infty < \overline{\lambda} < min(\tilde{\lambda})$. Otherwise it is *not convex* and $min(\tilde{\lambda}) < \overline{\lambda} < max(\tilde{\lambda})$.
  - If $discriminant(nn) < 0$ there are no "flips" and $E(\lambda)$ will have one real root and two imaginary roots.

The information is combined by the study of the $E(\lambda)$: slope, $E(\lambda)_3 \lesseqgtr 0$; flex point, $d^2E(\lambda)/d\lambda^2 = 0$; and relative max and min, $dE(\lambda)/d\lambda = 0$ to permit the appropriate selection of $\overline{\lambda}$

Equation Simplification The original formulation of the coordinate conversion (9.3) may be used symbolically, substituting $\boldsymbol{\alpha}(\boldsymbol{\beta},\lambda,\mu)$ in $F(\boldsymbol{\alpha},\lambda)=0$, obtaining

$$\overline{F}(\boldsymbol{\beta},\lambda,\mu)=F(\boldsymbol{\alpha}(\boldsymbol{\beta},\lambda,\mu),\lambda)=0,$$

then use $Q(\boldsymbol{\beta},\lambda,\mu)=0$ to eliminate λ by resultant

$$\overline{\overline{F}}(\boldsymbol{\beta},\mu)=Res_{\lambda,0}(\overline{F}(\boldsymbol{\beta},\lambda,\mu),Q(\boldsymbol{\beta},\lambda,\mu))=0.$$

The intersection equations are now reduced to two equations in the local coordinates of prism **E**

$$\begin{cases} \overline{\overline{F}}(\boldsymbol{\beta},\mu) & = & 0 \\ G(\boldsymbol{\beta},\mu) & = & 0 \end{cases}.$$

Conceptually, one may proceed further, and use the parametric form of the surface $\mu=\mu_G(\boldsymbol{\beta})$ and reduce the intersection to a curve in the barycentric coordinates $\boldsymbol{\beta}$ of prism **E**.

$$F_G(\boldsymbol{\beta})=\overline{\overline{F}}(\boldsymbol{\beta},\mu_G(\boldsymbol{\beta})).$$

Given any line in the barycentric coordinates $\boldsymbol{\beta}$ (eg: a linear constraint) one may by root finding compute the intersection point. However, due to the procedural nature of $\mu_G(\boldsymbol{\beta})$ one may only proceed numerically. The intersection between the surface of patch **A** and a border (ie. $\beta_2=0$) of **B** may be fully developed symbolically by plugging the rational $\mu(\beta_1)$ into $\overline{\overline{F}}(\beta_1,\mu)$ obtaining $F_G(\beta_1)=\overline{\overline{F}}(\beta_1,\mu_G(\beta_1))$.

Notice one may have also proceed by resultants and define:

$$F_G(\boldsymbol{\beta})=Res_{\mu,0}(\overline{\overline{F}}(\boldsymbol{\beta},\mu),G(\boldsymbol{\beta},\mu))=0$$

Even with the aid of computer algebra systems (see appendix A), the symbolic development of such polynomials for a pair of generic patches, has been found too cumbersome. On the other hand the computation of such polynomials for each specific pair of patches is too time consuming at implementation level.

Symmetrically the same intersection curve may be defined in domain $\boldsymbol{\alpha}$ of prism **D**.

Intersection Tracing

The intersection points between the boundary of one patch and the surface of the other may be formulated as the roots of univariate polynomials, albeit of high degree:

$$F_G(\beta_1)=0,\quad F_G(\beta_2)=0,\quad F_G(\beta_3)=0,\quad G_F(\alpha_1)=0,\quad G_F(\alpha_2)=0,\quad G_F(\alpha_3)=0.$$

Methods to approximate *all* the roots of a polynomial exist. This is a powerful starting point for a topologically accurate tracing[BX97] of the intersection curve. Under the

condition $F_G(\boldsymbol{\beta}) = 0$ one could determine its singular points $\nabla F_G(\boldsymbol{\beta}) = \mathbf{0}$, its β_i-extreme points $\partial F_G(\boldsymbol{\beta})/\partial \beta_i = 0$, and its flex points $\mathscr{H}_\beta(F_G(\boldsymbol{\beta})) = \mathbf{0}$. Example of such tracing on simpler polynomials may be found in appendix A. However this symbolic development is premature due to symbolic complexity and implementation cost.

In a prototype implementation a purely numerical approach has been used. Given a certain point $\overline{\boldsymbol{\beta}}$, in **E** barycentric coordinates (Fig. 9.2):

1. compute the height $\overline{\mu}$ along the interpolated normal of the surface G using the procedural form $\overline{\mu} = \mu_G(\overline{\boldsymbol{\beta}})$;
2. solve $Q(\overline{\boldsymbol{\beta}}, \lambda, \overline{\mu}) = 0$ for λ by root finding and selecting $\overline{\lambda} \in I_\lambda$;
3. compute the corresponding barycentric coordinates in prism **D** by $\overline{\boldsymbol{\alpha}} = \boldsymbol{\alpha}(\overline{\boldsymbol{\beta}}, \overline{\lambda}, \overline{\mu})$;
4. Compute in $F(\overline{\boldsymbol{\alpha}}, \overline{\lambda})$.

This procedure effectively computes $F_G(\overline{\boldsymbol{\beta}})$. Furthermore, the membership of the point in the interior of prism **D** is determined. The computation may be used iteratively with a numeric root finding algorithm (such as the false position method) to find an intersection point ($F_G(\boldsymbol{\beta}) = 0$) along any line segment in the $\boldsymbol{\beta}$ barycentric domain.

Operationally, the barycentric domain of *both* prisms is subdivided. The particular subdivision is not important, for simplicity the standard decomposition along iso-coordinates is used (Figure 9.3a).

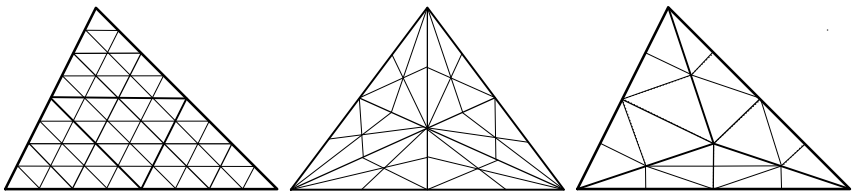


Figure 9.3: Subdivision along: (a) iso-coordinates; (b) barycenter; (c) orthocenter. Figure created by the author for [BPP+08].

Without loss of generality consider the prism **E**, the same operations are symmetrically done in the other prism **D**. For each vertex of the subdivision the corresponding value of the surface scalar function $F_G(\boldsymbol{\beta})$ is computed. Each sub-triangle may either have no sign change (even number of intersections) or a sign change on two sides (odd number of intersections). A simplistic approach is used and only one intersection in case of a sign change is searched. For each intersection point one computes and stores the barycentric coordinates $\boldsymbol{\beta}$, the height μ, and the corresponding coordinates $\boldsymbol{\alpha}$ and λ in the other prism.

All intersection points in the same sub-triangle will be considered in the same connected component of the intersection, thus a tracing of the intersection curve is possible. For the correctness of next steps it is important that: (1) the trace of the intersection is done on both prisms; and; (2) both tracings contain the *same* points. (Figure 9.4, 9.5 and 9.6)

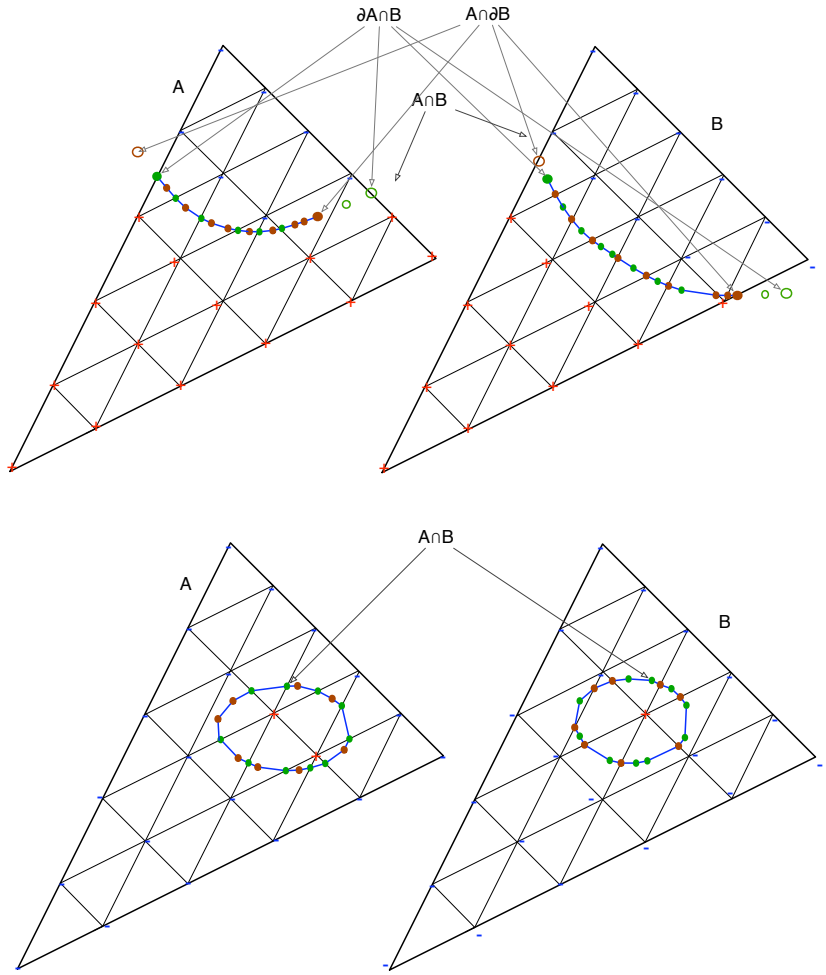


Figure 9.4: Tracing the intersection on both domains. Figure created by the author for [BPP+08].

An advantage of this simplified algorithm is that the topology of the intersection in each sub-triangle is the same as the intersection between two triangles and apt to be processed

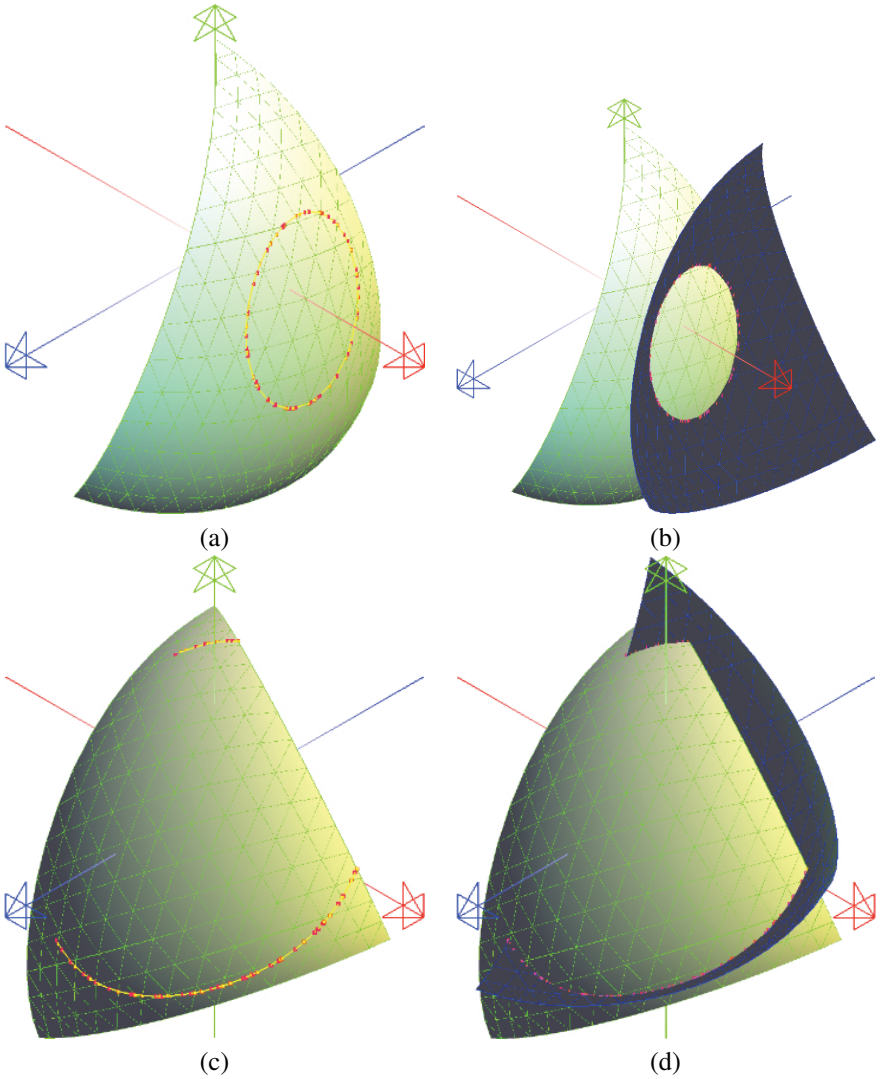
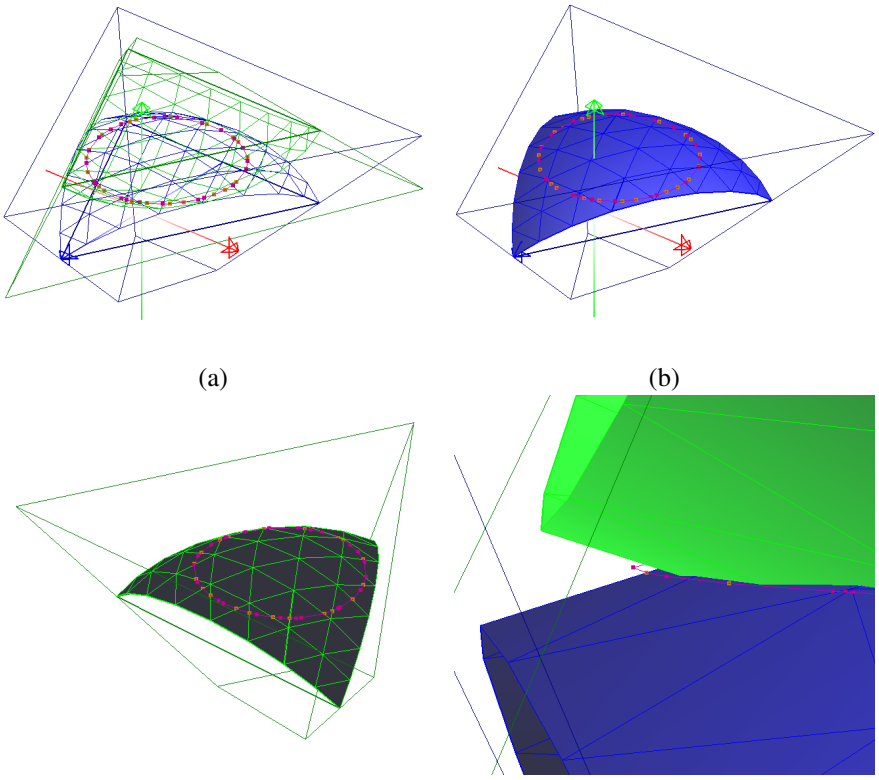


Figure 9.5: Tracing the intersection points on both domains: (a,b) a closed intersection curve between two A-patches; (c,d) two unconnected segments of the intersection curve. Image generated by the author using the prototype software [BPP+08].



(a) (b) (c) (d)

Figure 9.6: Plot and trace of the intersection of two patches: (a) both patches in wireframe, subdivision used in the numeric algorithm is shown; (b) first patch; (c) second patch; (d) detail of intersection, notice the point is more accurately located (by the numeric algorithm) while the surface is a triangular approximation with same scale of the subdivision. Image generated by the author using the prototype software [BPP+08].

by the Boolean algorithm described in section 9.2. Topological errors are possible but their size is limited by subdivision granularity. (Figure 9.7)

Triangulation and New Patch Construction

The final step of our intersection algorithm is to reconstruct triangular (and hence scaffold) support, conforming to the topology of the intersection curve, for all the prism

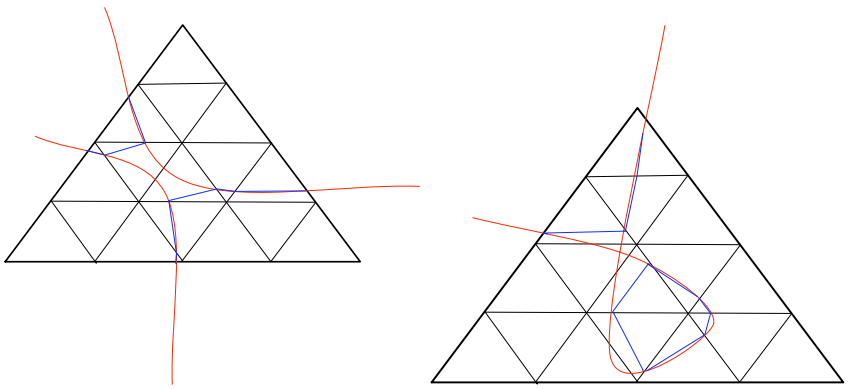


Figure 9.7: Topological approximation. Figure created by the author for [BPP+08].

A-patches, that constitute the result of the Boolean operation. Since the topology of the interesection curve is either an open or a closed curve in both the patch domains, the necessary conforming triangulation can easily be affected in either or both domains.

In both the top and bottom rows of Figure 9.8, the leftmost figures show the intersection topology of the two A-patches (shaded differently). The middle figures (top and bottom) depict the result of the union operation . The rightmost figures, show the patch decomposition into triangles and quads. The quads are further subdivided into two triangles, using any diagonal. The prism scaffolds are easily erected on each triangle based on the normals defined at the vertices. Note, for vertices lying on the intersection curve, there exist two vertex normals (one from each initial A-patch). Hence these vertices are part of two prism scaffold edges.

Note that two adjacent prism scaffolds may intersect each other, since the resulting patches are no longer C^1.

Consider the prism **D** of patch **A** with all traces of the intersection curves given by intersecting patches $\mathbf{B_i}$. The next step is to triangulate the base barycentric coordinates in a way to contain the piecewise linear traces of the intersections. In the prototype implementation, the triangulation is obtained using, in a hierarchical manner, the same subdivision scheme used for the intersection finding . The largest subdivisions without intersections are selected as they are. The sub-triangles of smallest size, containing the intersection trace, are triangulated by linking the intersection points to the vertexes of the sub-triangle (Figure 9.9(a)). Additional operations are needed if a sub-triangle contains more than one trace. Then the original sub-triangles of the subdivision are aggregated to build a new regular triangulation of the result surface of the boolean operation (Figure 9.9(b) and 9.10).

This triangulation scheme is not very sophisticated but it is the most compatible with the

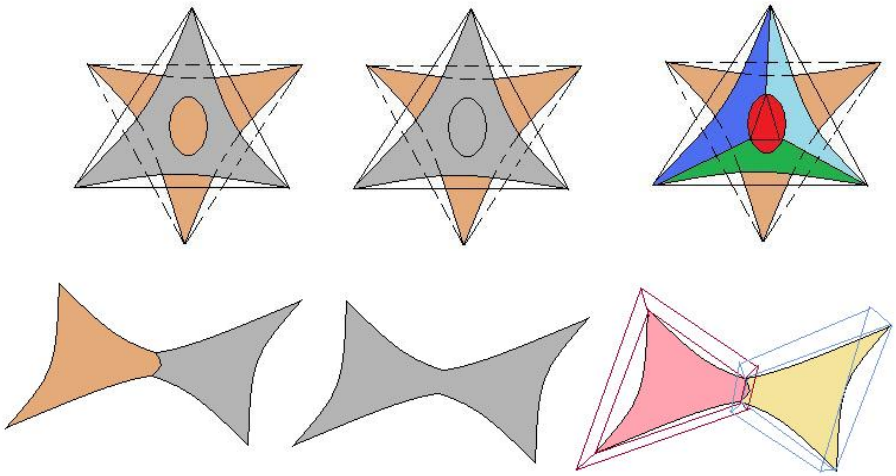
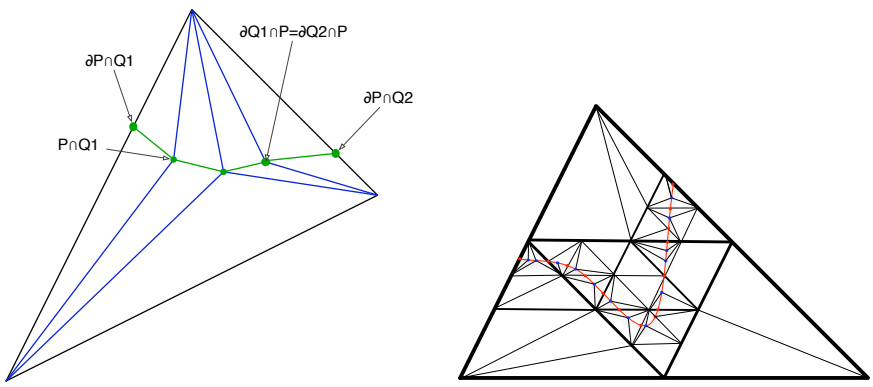


Figure 9.8: Examples of conforming triangular supports of the prism A-patches that constitute the union of original twin A-patches. Shown are parts of the scaffolds of the A-patches for two different topologies of the intersection curve. The triangulation of the quad patches are not shown, just to maintain picture clarity. Figure created by Na Lei for [BPP+08].

current tracing method that already generates the sub-triangles. As mentioned there are more sophisticated tracing methods where the singular, extreme and flex points of the curve are considered. In this case a very sophisticated scheme for triangulation may be adopted. For each curve point not only the coordinates are computed but the local curve tangent. Then the triangulation is operated in BSP (Binary Space Partition) [Nay90] style using the secants between consecutive points and the tangents. At first approximation the intersection curve is piecewise linearly approximated by the secants and the triangles classified as over/under the intersection along this approximation. Notice however that the real curve segment is contained only by the triangle between the secant and the extended tangents(see figure 9.11). One may proceed further and classify this triangle as a possible loci of a refinement. Indeed calculating any other curve point in this triangle one may split locally and create a local refinement and a local classification. This refinement may be applied locally at will producing a converging approximation of the curve. These scheme may be used in a computational framework where the approximation is refined progressively depending on the actual need.

Any triangulation on the barycentric domain, corresponds to a triangulation of the sur-



(a) (b)

Figure 9.9: Triangulation: (a) local triangulation along trace; (b) triangles aggregation. Figure created by the author for [BPP+08].

face. For each point $\boldsymbol{\alpha}_\mathbf{l}$, corresponds a $\lambda_l = \lambda_F(\boldsymbol{\alpha}_\mathbf{l})$ and a $\mathbf{v}_\mathbf{l} = \mathbf{D}(\lambda_l)\boldsymbol{\alpha}_\mathbf{l}$.
The normal for a point not belonging to the intersection is the normal to the surface of the patch $\mathbf{n}_\mathbf{l}^{(\mathbf{F})} = T_D^{-1}\nabla F(\boldsymbol{\alpha}_\mathbf{l}, \lambda_l)$ (see eq. 8.9).
Special care must be given to the normal of an intersection point. This point belongs to both surfaces patches. The intersection curve has both the normal $\mathbf{n}_\mathbf{l}^{(\mathbf{F})}$ to the surface of patch $\mathbf{A}$ and the normal $\mathbf{n}_\mathbf{l}^{(\mathbf{G})} = T_E^{-1}\nabla\overline{\overline{G}}(\boldsymbol{\alpha}_\mathbf{l}, \lambda_l)$ to the surface of patch $\mathbf{B}$. The intersection curve has infinite normals in the span $\mathbf{n}_\mathbf{l} = \alpha\mathbf{n}_\mathbf{l}^{(\mathbf{F})} + \beta\mathbf{n}_\mathbf{l}^{(\mathbf{G})}$. The tangent to the curve is $\mathbf{t}_\mathbf{l} = \mathbf{n}_\mathbf{l}^{(\mathbf{F})} \times \mathbf{n}_\mathbf{l}^{(\mathbf{G})}$. Two options may be considered:

- The point will belong to both the new patch set generated by the split of patch $\mathbf{A}$ and of patch $\mathbf{B}$. The point will have two normals defined by the respective patch surface normals $\mathbf{n}_\mathbf{l}^{(\mathbf{F})}$ and $\mathbf{n}_\mathbf{l}^{(\mathbf{G})}$. However a new kind of prismatic patch must be defined with "sharp" boundary (see figure 9.12(a)(c)) and 9.13(a)(b)(c)).
- The point will be identified on both patches and an unique normal in the span is used (e.g. $\mathbf{n}_\mathbf{l} = 0.5\mathbf{n}_\mathbf{l}^{(\mathbf{F})} + 0.5\mathbf{n}_\mathbf{l}^{(\mathbf{G})}$). The resulting patch will approximate "smoothly" the sharp boundary at the intersection. By construction, its effect is limited to the sub-triangles of highest resolution (see figure 9.12(b)(d) and 9.13(d)(e)(f)).

Given a triangulation $\mathscr{T}$ of the patch A the new patch set may be built simply by re-applying the prismatic A-patch construction algorithm as in Chapter 8 (see figure 9.12 and 9.13).

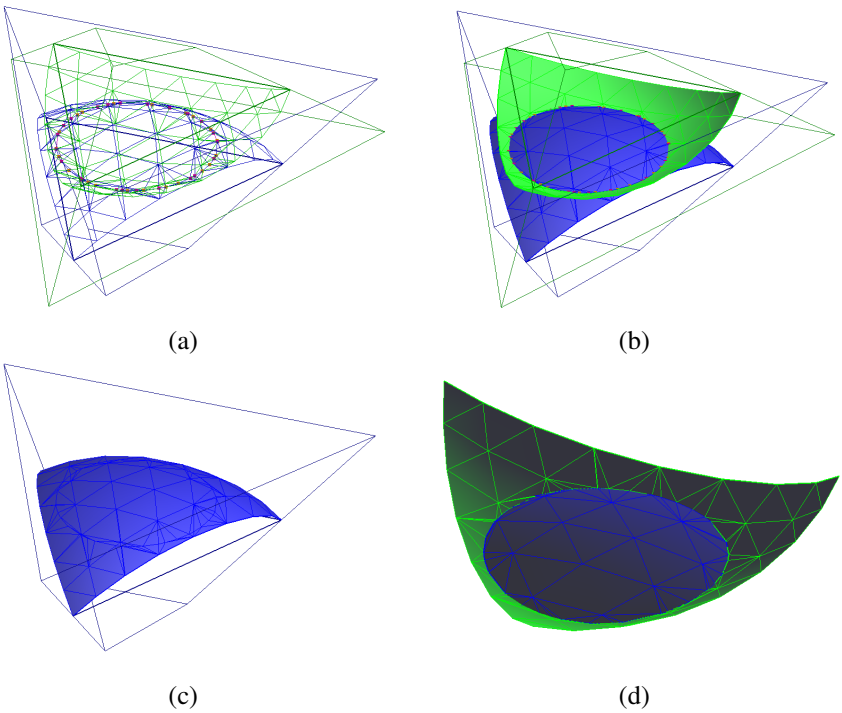


Figure 9.10: Triangulation of two patches: (a) wireframe; (b) solid; (c) second patch; (d) the exterior of first patch and the interior of the first patch, trimmed along intersection.Image generated by the author using the prototype software [BPP+08].

9.4 Examples and Applications

Simple Example In figure (9.13) a sample run with the prototype implementation is shown. Two shells (blue B and green G) are given with a single component intersection curve. Both shells where spheroids of four patches each. The intersection curve is plotted on patch and subdivision boundaries. Then a piecewise linear tracing of the intersection curve interpolates the intersection points. Along the intersection curve segments the sub-triangles are split. Then for each new triangle a new scaffold is built with the adjacent intersection points and the calculated normals therein. In figure (9.13(a)(d)) all the patches exterior to the other shell are selected by the boolean algorithm to produce the shell of the *boolean union* of both shells. In figure (9.13(b)(e)) all the patches interior to the other shell are selected and the normals flipped (*boolean complement*) by

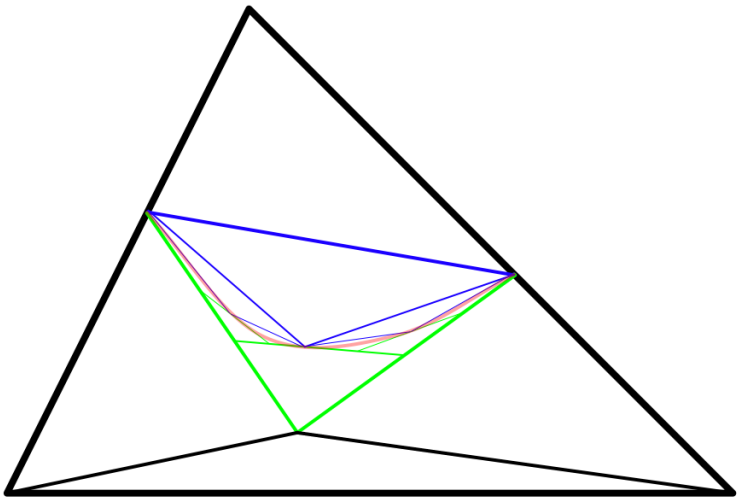


Figure 9.11: Alternative progressive tracing and triangulation. Figure created by the author for [BPP+08].

the boolean algorithm to produce the shell of the *boolean intersection* of both shells. In figure (9.13(c)(f)) all the patches of the green shell interior to the blue shell and all the patches of the blue shell exterior to the green shell are selected and the normals flipped for the patches of the green shell to produce the shell of the *boolean difference* $B-G$. In the first set of examples (9.13(a)(b)(c)) both normals, one for each original shell, are used and the resulting edges are sharp. However the definition of sharp sided prism A-patch is not well defined and at high resolution there is a gap between the patches along the intersection curve outside the intersection points. In the second set of examples (9.13(d)(e)(f)) there is a common normal (e.g. $\mathbf{n_l} = 0.5\mathbf{n_l}^{(\mathbf{F})} + 0.5\mathbf{n_l}^{(\mathbf{G})}$) and there is no gap between on the intersection curve, however the resulting shell is approximated and smoothed on all patches along the intersection.

Stress test In this example we show the Boolean union produced by two very similar objects P_1 and P_2, whose prism scaffolds are displayed in Figures 9.14a and 9.14b, together with the intersection curves drawn on both surfaces. The resulting triangulated subpatches are displayed in Figure 9.14c, whereas the resulting object $P_1 \cup P_2$ is shown in Figure 9.14d.

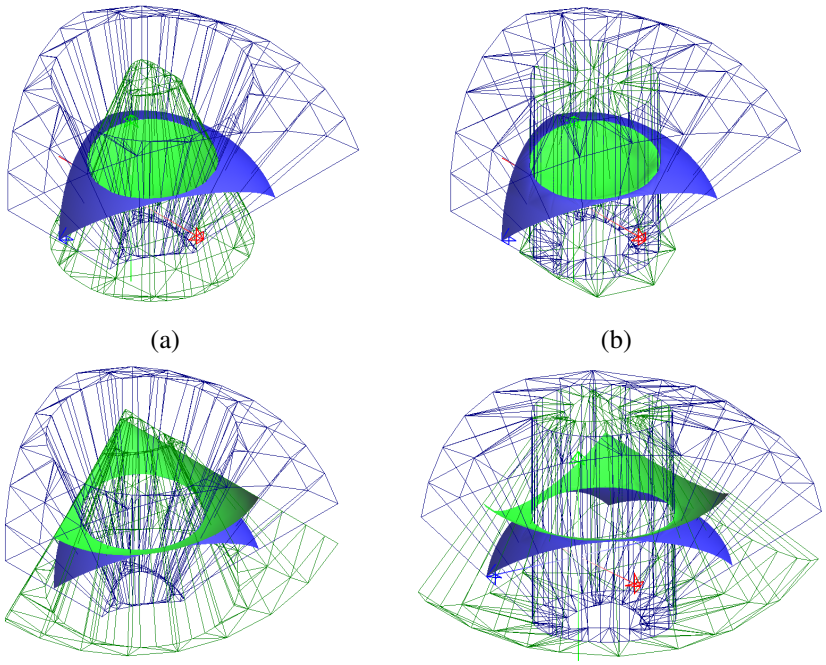


Figure 9.12: New patch construction with new scaffold prisms: top row (a) (b), the exterior of blue patch and the interior of green patch w.r.t. the intersection curve (e.g. boolean difference); bottom row (c) (d), the exterior of both patches (e.g. union); left column (a) (c), two new separate normals for each intersection point and sharp edge; right column (b) (d), single blended for each intersection point normal and smooth edge. Image generated by the author using the prototype software [BPP$^+$08].

Molecular Models The prototype implementation has been used to compute the actual intersection and union in the rest configuration of two ligands proteins and the resulting complex after the docking. Each molecular model includes the solvent excluded surface (SES) [IBR98, STHH90, BC00, BZ06, ZXB06] and is calculated using methods similar to chapter 5 by the TexMol program. Each ligand model has a number of patches in the order of ten thousands. The intersection curve (in purple) of the two ligands has a small geometry with very complex multi component topology. This is a very unfavorable situation for the prototype algorithm. In figures 9.15 and 9.16 the docking of α-Chymotrypsinogen (1CGI-A) and the human pancreatic secretory trypsin

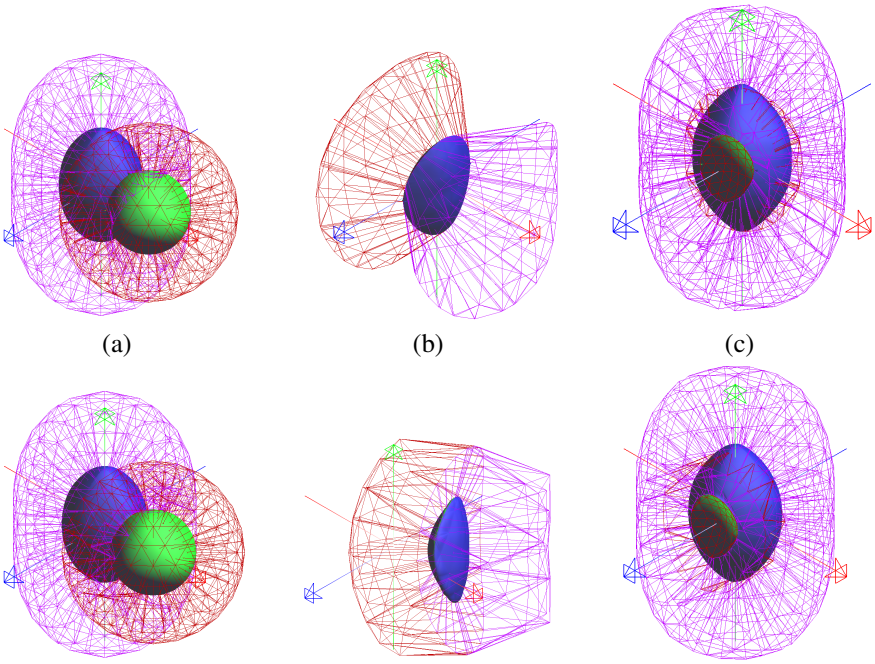


Figure 9.13: Boolean operations on two simple shells: (a)(d) union; (b)(e) intersection; (c)(f) difference; (a)(b)(c) separate normals at intersection; (d)(e)(f) blended normal at intersection. Image generated by the author using the prototype software [BPP+08].

inhibitor (kazal type) variant (1CGI-B) is shown. The models have 64208 and 28852 patches respectively. In figures 9.17 and 9.18 the docking of Trypsin (1PPE-A) and the Trypsin inhibitor CMT-1 (1PPE-B) is shown. The models have 12756 and 5872 patches respectively. In figures 9.19 and 9.20 the docking of Calmodulin (1CKK-A) and RAT CA2+/Calmodulin Dependent protein kinase (1CKK-B) is shown. The models have 13680 and 5252 patches respectively.

Another application is the 3D printing of physical models of molecular models which have seen increasing use in bio-chemistry kits, and more recently in tangible augmented interfaces, for experiential understanding of the complexity of molecular interfaces, especially in docked complexes. Several CAD operations need to be performed on molecular models, including Boolean set operations, thereby allowing, for example, the unioning of bonds, and differences to create holes, before the 3D auto-fabrication

process[GSS+04].

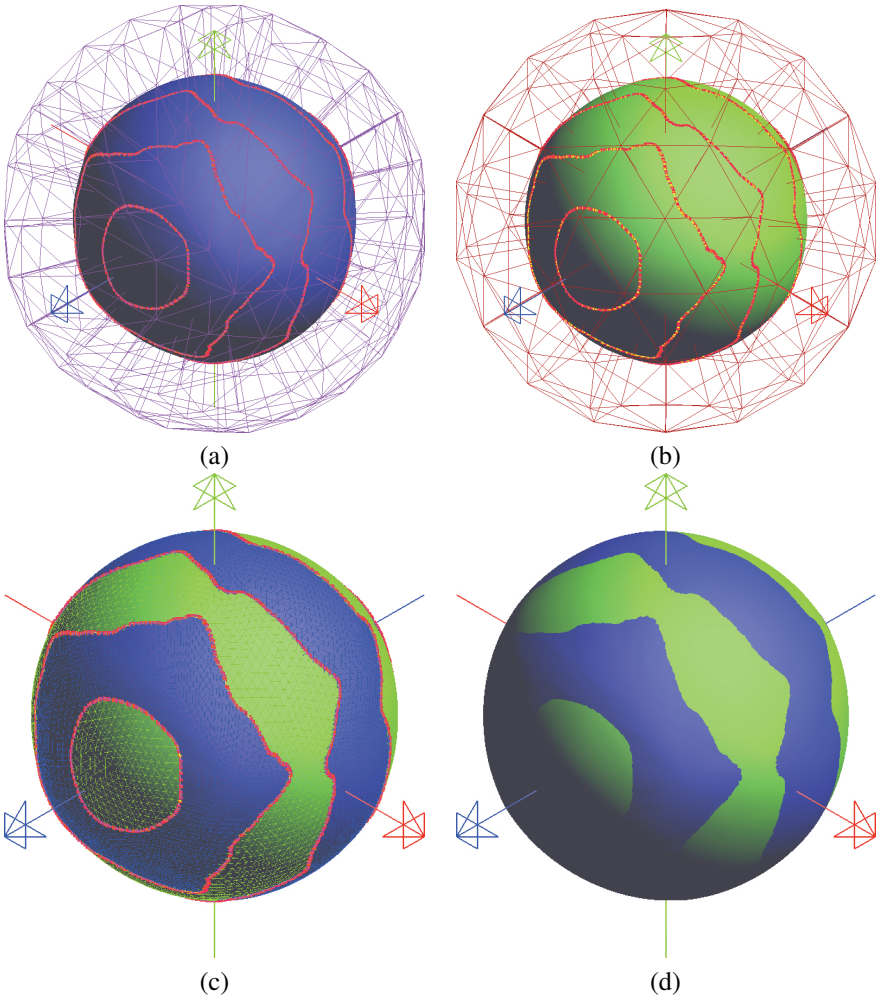


Figure 9.14: (a) The argument object P_1, drawn with the associated prisms. (b) The argument object P_2. (c) The triangulated subpatches between the intersection curves. (d) the object $P_1 \cup P_2$. Image generated by the author using the prototype software [BPP+08].

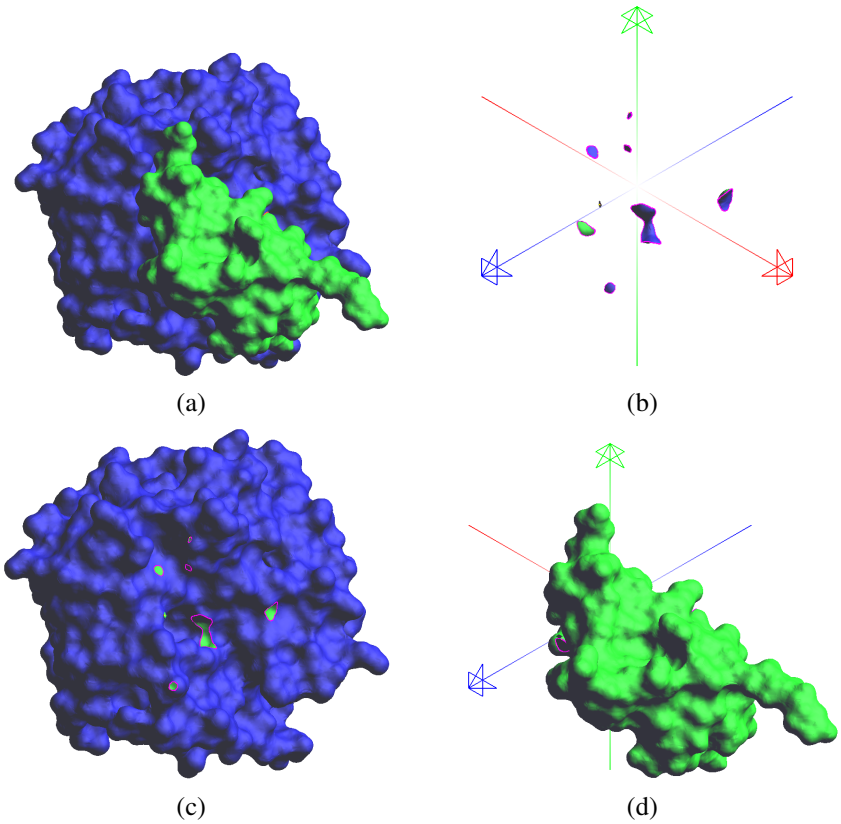
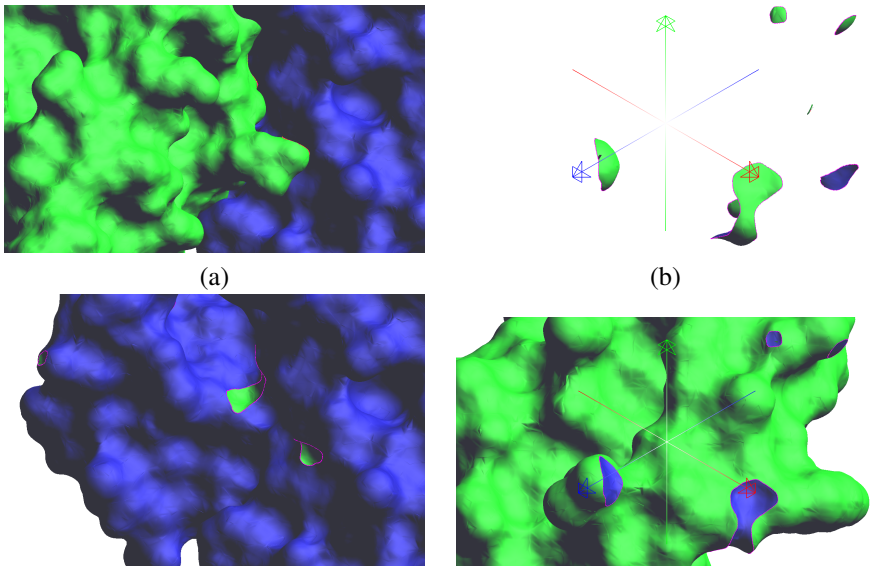


Figure 9.15: Global view of boolean operations between 1CGI-A and 1CGI-B ligands: (a)union; (b) Intersection; (c)(d) difference.Image generated by the author using the prototype software [BPP+08]. Original molecular model courtesy of CVC at ICES.



(a) (b) (c) (d)

Figure 9.16: Detail view of boolean operations between 1CGI-A and 1CGI-B ligands: (a)union; (b) Intersection; (c)(d) difference. Image generated by the author using the prototype software [BPP+08]. Original molecular model courtesy of CVC at ICES.

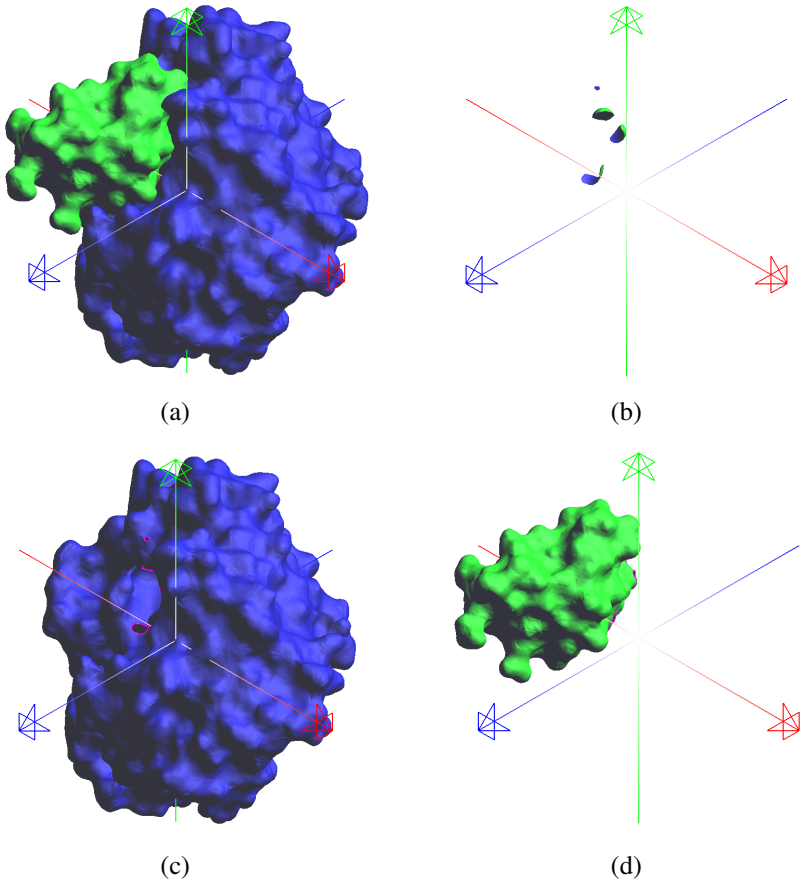


Figure 9.17: Global view of boolean operations between 1PPE-A and 1PPE-B ligands: (a) union; (b) Intersection; (c)(d) difference. Image generated by the author using the prototype software [BPP+08]. Original molecular model courtesy of CVC at ICES.

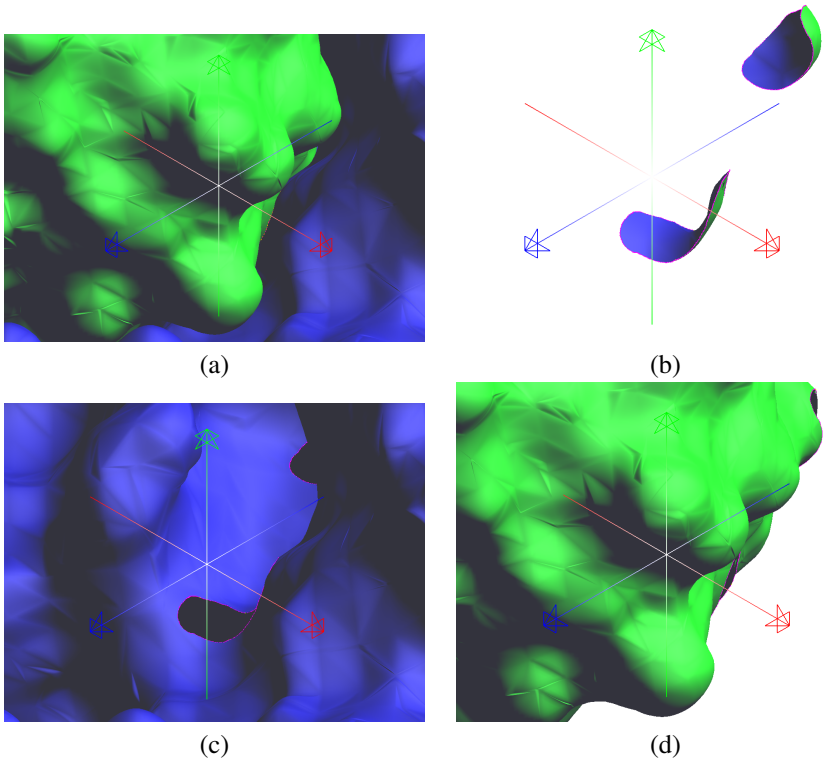


Figure 9.18: Detail view of boolean operations between 1PPE-A and 1PPE-B ligands: (a) union; (b) Intersection; (c)(d) difference.Image generated by the author using the prototype software [BPP+08]. Original molecular model courtesy of CVC at ICES.

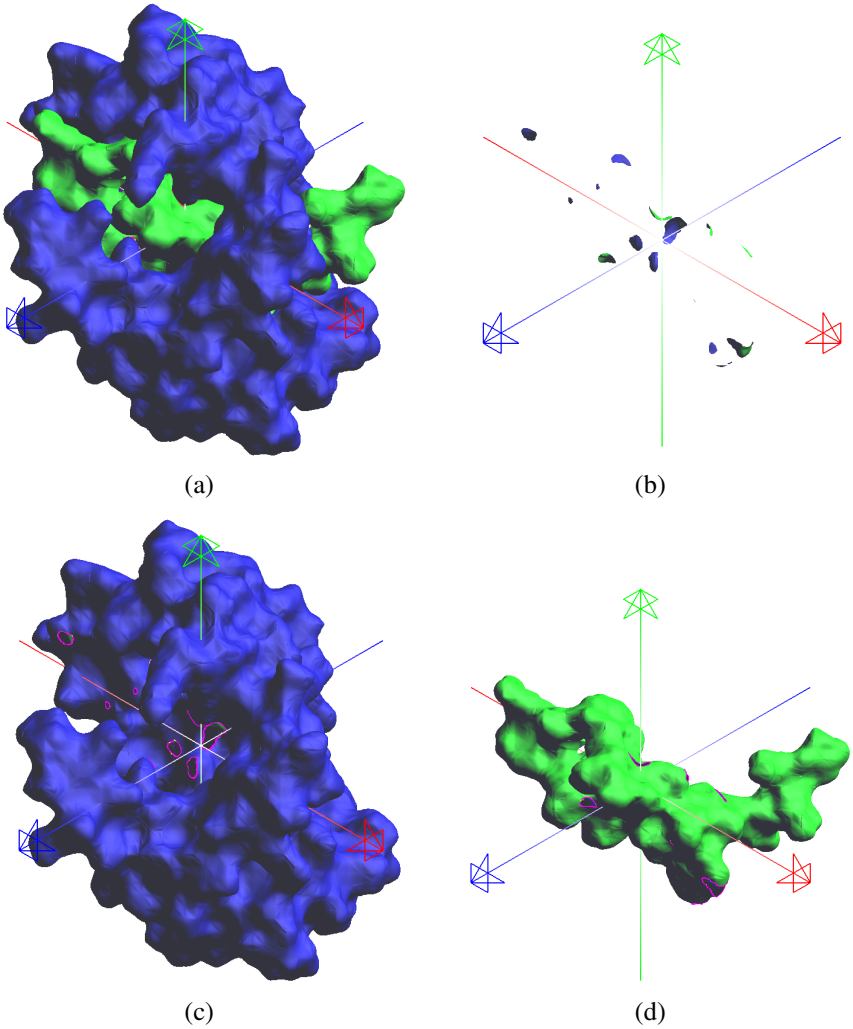
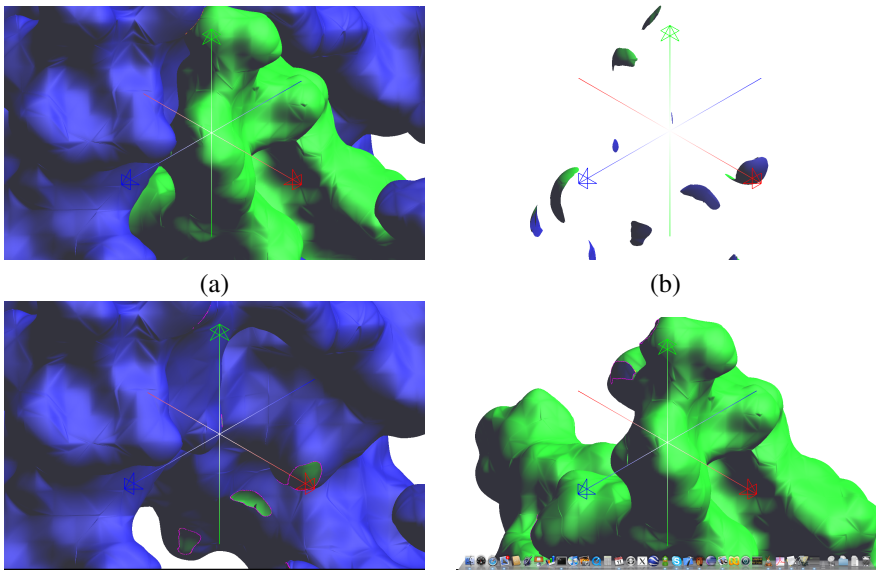


Figure 9.19: Gloabal view of boolean operations between 1CKK-A and 1CKK-B ligands: (a) union; (b) Intersection; (c)(d) difference. Image generated by the author using the prototype software [BPP+08]. Original molecular model courtesy of CVC at ICES.



(a) (b) (c) (d)

Figure 9.20: Detail view of boolean operations between 1CKK-A and 1CKK-B ligands: (a) union; (b) Intersection; (c)(d) difference.Image generated by the author using the prototype software [BPP+08]. Original molecular model courtesy of CVC at ICES.

Chapter 10

Conclusion

Algebraic modeling has never been the main mean for computer based geometric representation. When used in literature, for example in [CS04, LB02, BDST04, BLMP97, BDST04, BCCSX05, ZXB07, ZXB06][ZnBS05, KOB+04, BYA03], the representation scheme was tailored to the actual problem and operations implemented ad-hoc for a particular reconstruction, operation, or computation needed for the problem. There is a wide range of schemes and none of them particularly flexible in terms of degree, scaffold type, dimensionality and interoperability. Notwithstanding the current limitations there is evidence of distinctive advantages. This thesis tries to systemize current representation schemes under an unified point of view. To date the use of algebraic geometry is still in developing phase and far from being a major element in current geometric systems. Much research work and engineering is needed to develop a general algebraic representation and a rich toolset of operations.

10.1 Results

The principal result [BPP+08, BPP07] and practical objective of this thesis are symbolic-numeric methods for Boolean operations on the algebra of curved polyhedra[Req77, Req80, LTH86] whose boundary is triangulated with algebraic patches [Dah89, BCX94, BCX95, BCX95, BX99b, BXHN02, BCX97, Guo92]. The method includes a robust method to trace the intersection curves between two triangular prismatic A-patches, while at the same time locally refining the support triangulation of the result. It is not the "ultimate" solution, boolean operation on curved geometry is yet an open stubborn research problem [KGMM97, Lin00, BKZ01, KCF+02, GMF06]. Though the mathematical solution exists in algebraic geometry [Abh90] few works refer to it explicitly. The development of Boolean set operations is a necessary step for the development of an integrated software package for algebraic finite elements.

The prototype implementation provides for the computation of Boolean operations between two free-form solids, modeled by prism A-patches, however, the general technique is applicable to any A-patch scheme. For example, using tetrahedral A-patches the scaffolding, boolean algorithm, the triangulation and new patch construction are much more complex, but, the polynomials involved are much simpler to manipulate symbolically. From the prototype implementation several important facts have been learned:

- Intersection curve point finding is very accurate with current methods.
- As mentioned, tracing may further enhanced using more sophisticated algebraic methods.
- Current triangulation scheme is poor regarding the number of new triangles (and patches) produced. The quality of the triangles is acceptable but not optimal.
- When input patch sets have very different granularity numerical problems are greatly amplified.
- Performance has not been an objective, however it has been found to be comparable to a standard polyhedral boolean algorithm applied by approximating the input patches with a polyhedral surface of same granularity as the subdivision used in the tracing phase. The resulting object of the polyhedral boolean algorithm is less accurate then the output of the triangulation phase (without new patch construction). This is due to the more accurate point finding (see figure 9.6d).

Hopefully in the future this work may be further developed:

- The implementation code should be further optimized, numerically solidified and organized. It should then be integrated in a geometric application, for example the geometric language PLaSM [Pao03, Pa]. Further improvements in the efficiency involve spatial progressive indices, and multiresolution evaluators, and with local refinement.
- Apply the method to a broader set of real world application and examples. Operating on both acquired models and generated ones. Also, the extension to other A-patch schemes may further enhance the real world use.
- Develop symbolic and topologically exact methods on tracing and triangulation. Under this point of view the ongoing work with the integration of PLaSM [Pao03, Pa] with GANITH [BR90b, Por07b] (see appendix A) enables to interoperate numerical and symbolic methods easily.

Another objective of the thesis has been the rationale to present the algebraic modeling under an unified point of view. This is the result of an ongoing knowledge transfer

between the CVC (Center for Computational Visualization) of the ICES (Institute of Computational Engineering and Sciences) at the University of Texas at Austin and both the PLM group and the SICS at the University Roma Tre. Current A-splines and A-patch representation schemes are too rigid (i.e. fixed support domain, degree, and dimensionality) to constitute a general geometric representation carrier. Hopefully this thesis will help in the further research and engineering towards a general and flexible algebraic modeling.

10.2 Wider Context

At the PLM group at the University "Roma Tre" there is ongoing work to the development of algorithms and software that couple molecular- and cellular-scale simulations of biological systems, in order to understand the impact of chemical and biological agents on synaptic transmission at NMJs (Neuro-Muscular Junctions). Previous computational work in this area has either considered the molecular components of the NMJ in isolation or has used highly simplified models of the NMJ with only gross molecular information. Unfortunately, such methods do not provide a robust framework for understanding the physiological effects of anti-cholinergic agents. For example, modeling of the isolated components ignores the possibility of competition and compensation between the multiple biomolecular targets present in the NMJ. Likewise, models which use only the gross features of the NMJ morphology rely on parameterization against existing experimental data which severely restricts the range of conditions under which the models can work reliably.

The first step in this modelling activity is the collection of structural data for the NMJ and its biomolecular components. All of data from high resolution tomography are assembled into a finite element mesh. The goal of this initial meshing step is to provide a coarse geometric description of the domain to serve as a template for placement of the more accurate biomolecular structural information. AChE and AChR molecules are treated as spherical cavities in the NMJ domain. These spherical cavities must be "filled" with detailed biomolecular finite element meshes based on structural models obtained from existing X-ray data.

One of the objectives of the method presented in this thesis is to contribute to a computational framework [BDP08] where to generate multi-scale detailed meshes, driven by structural data from high-resolution tomography, and to support highly efficient finite element algorithms and software (in this case to simulate the diffusion of ACh in the NMJ). Furthermore, the large size of a bio-simulation mesh demands the ability to coarsen meshes through surface and volumetric simplification methods. The effectiveness of these types of adaptive finite element techniques rests on fast and robust low-level mesh primitives, including Booleans operators, for both refinement and unrefinement.

Another interesting future direction consists in the direct modeling within both space and time dimensions. In fact, the implicit geometric approach used here and the associated linear scaffolding can be extended to work in higher dimensions. The most flexible finite element techniques for evolution equations are based on spacetime discretizations, whereby the space and time domains are subdivided into finite elements. Evolution equations are then discretized in both space and time on each element through low-order polynomial approximation and numerical quadrature. Treating space and time in a unified way in the discretization provides a framework for both mesh refinement and coarsening where needed in both space and time, driven by rigorously-derived *a posteriori* error indicators.

Appendices

Appendix A

Algebraic Geometry in GANITH and PLaSM

PLaSM [Pao03, Pa], (Programming LAnguage for Solid Modeling) is a design language, strongly influenced by FL (programming at Function Level), a novel approach to functional programming developed by the Functional Programming Group leaded by John Backus and John Williams at the IBM Research Division in Almaden in the first nineties [BWW90, JWW90]. PLaSM is a geometric extension of FL, allowing for a powerful algebraic calculus with dimension-independent geometric objects, that include polyhedral complexes and parametric polynomial and rational manifolds (curves, surfaces, curved solids, and higher-dimensional objects). In this environment for geometric computations, a complex shape is generated as an assembly of component shapes, highly dependent from each other, where (a) each part may result from computations with other parts, (b) a generating function is associated to each, (c) geometric expressions may appear as actual parameters of function calls.

The design of PLaSM, the open-source Programming Language for Solid Modeling, started in 1992, long time before the internet-based access to virtual worlds was even imagined. The second optimized and extended version of the language interpreter, written in Common Lisp, and characterized by an unique multidimensional approach to geometric modeling [PPV95], was available in 1994. The version 3, extended with animations, colors and cameras, and exporting to OpenInventor and VRML, came in 1999. The following version 4 was completely rewritten in Scheme and C++, with javascripted animations. The novel version 5, introduced here, is being deployed with a fast geometric engine and with rewriting optimizations in the interpreter, aiming to automatically exploit multi-core processing. The XGP (eXtreme Geometric Processing) engine is using a novel multidimensional decompositive representation, based on progressive BSP trees [SPP07] and on the Hasse graph of the generated $d-$complex; at upper level is us-

ing HPC (Hierarchical Polyhedral Complex) structures. The language allows for powerful operations, including progressive Booleans, the Cartesian product of cell complexes, the extraction of their $k-$skeletons $(k = 1, \ldots, d)$, and the Minkowski sum of a polyhedral complex with parallelotopes, allowing for sweeping and offset. Current extensions aim to encode the object topology as a sparse matrix, using a tensor representation of the chain complex underlying the cell decomposition [ADS07].

The GANITH algebraic geometry toolkit [BR90b, Por07b] manipulates arbitrary degree polynomials and power series. It can be used to solve a system of algebraic equations and visualize its multiple solutions. Example applications of this for geometric modeling and computer graphics are curve and surface display, curve-curve intersections, surface-surface intersections, global and local parameterizations, implicitizations, and inversions. It also incorporates subsystems for techniques for A-splines, multivariate interpolation and least-squares approximation to an arbitrary collection of points and curves.

There are four major components of GANITH: the user interface, controller, numerical and graphics subsystem, and the algebra subsystem. The first three components reside in one process, and the algebra subsystem in another process. The algebra subsystem is written in Common Lisp, and the others in C. All graphics and numerical calculations are performed in C, while symbolic and exact calculations are done in Lisp. The two processes communicate with each other using Unix sockets, and may run on different hosts.

The main functionalities of Ganith are:

- Graphic visualization of points, curves, surfaces in both parametric and implicit form
- Symbolic algebraic manipulation of arbitrary degree polynomials with infinite precision rational numbers.
- Subsystems for ad-hoc functionalities such as:
  - A-splines
  - Interpolation
  - Parametrized Families
  - Scattered Data
  - CT Data Visualization

The author has recently ported the GANITH [BR90b, Por07b] algebra subsystem to PLT [PLT] Scheme [Var98, ASS84] the host language of PLaSM [Pao03, Pa] and integrated in a prototype version of PLaSM. This appendix is actually the first documentation on this project. It is the hope of the author to further develop this work by porting and

integrating in PLaSM both the visualization techniques of Ganith and the A-spline/A-patches techniques described in this thesis including the boolean operators. This would result in a sophisticated environment and programming language to manipulate symbolically and numerically both parametric and algebraic geometry.

A.1 Polynomial Operations

All the operations on the euclidean ring of polynomials are implemented and integrated into the PLaSM language. The algebra subsystem represents polynomials of arbitrary degree and number of variables in *Recursive Canonical Form* (*RCF*). In this form, a polynomial in the variables $x_1, \ldots, x_n$ is represented either as a constant, or as a polynomial in x_n whose coefficients are (recursively) polynomials in the remaining variables $x_1, \ldots, x_{n-1}$. The variable x_n is sometimes referred to as the *main variable*. A strength of this form (for purposes of implementation) is that multivariates "look like" univariates, making it easy to modify algorithms for univariate polynomials to handle multivariates.

In PLaSM, RCF polynomials have been integrated by defining a new data-type. The constructor takes the string representation of the polynomial and converts it into the internal RCF representation. The destructor takes a RCF object and outputs it string representation. Variables may be composed of multiple characters.

```
DEF APOL=mkrcf:'x1∧3−3∗x1∧2∗x2+3∗x1∗x2∧2+1';APOL;
% RCF poly: 3∗x1∗x2∧2−3∗x1∧2∗x2+x1∧3+1 %

UKRCF:APOL;
% '3∗x1∗x2∧2−3∗x1∧2∗x2+x1∧3+1' %

ISRCF:APOL;
% True %
```

Numerical constants are automatically considered polynomials. Predicates are available to recognize them.

```
ISRCF:666;
% True %

UKRCF:42;
% '42' %

RCF_CONSTANT?:149;
% True %

RCF_CONSTANT?:APOL;
% False %

RCF_ZERO?:1;
% False %
```

```
RCF_ZERO?:0;
%True%
```

All the operation in the polynomial ring are available and the operator symbols are overloaded (** is power elevation).

```
3*APOL + MKRCF:'x1+x2' - 1;
% RCF poly: 9*x1*x2^2-9*x1^2*x2+x2+3*x1^3+x1+2 %

APOL * MKRCF:'x3+1';
% RCF poly: 3*x1*x2^2*x3-3*x1^2*x2*x3+x1^3*x3+x3+3*x1*x2^2-3*x1^2*x2+x1^3
    +1 %

(APOL**3);
% RCF poly: 27*x1^3*x2^6-81*x1^4*x2^5+108*x1^5*x2^4+27*x1^2*x2^4-81*x1^6*
    x2^3-54*x1^3*x2^3+36*x1^7*x2^2+45*x1^4*x2^2+9*x1*x2^2-9*x1^8*x2-18*
    x1^5*x2-9*x1^2*x2+x1^9+3*x1^6+3*x1^3+1 %
```

Polynomial is an Euclidean ring, the result of a division $dividend/divisor$ is defined by the pair $(quotient, remainder)$ where $dividend = quotient * divisor + remainder$ and $degree(remainder) < degree(divisor)$. Division is performed w.r.t. the main variable. The results are undefined if the division cannot be performed. Pseudo-division is also supported, pseudo-division of A by B is the same as division of $(b^{m-n+1}) * A$ by B, where $b = ldcf(B)$.

```
APOL RCF_DIVIDE MKRCF:'x2^2+1';
% < RCF poly: 3*x1 , RCF poly: -3*x1^2*x2+x1^3-3*x1+1 > %

APOL**2 RCF_DIVIDE APOL;
% < RCF poly: 3*x1*x2^2-3*x1^2*x2+x1^3+1 , 0 > %

APOL RCF_PDIVIDE MKRCF:'x2^2+1';
% < RCF poly: 3*x1 , RCF poly: -3*x1^2*x2+x1^3-3*x1+1 > %

APOL**2 RCF_PDIVIDE APOL;
% < RCF poly: 81*x1^4*x2^2-81*x1^5*x2+27*x1^6+27*x1^3 , 0 > %
```

It is possible to inspect a polynomial, that is extract variables and their coefficients:

$rcf\_vars(poly)$ list of vars in poly;

$rcf\_coef(poly, var, e)$ the coefficient of var^e in poly, 0 if var is not present in poly, var need not be the main variable of poly;

The last two examples is actually a little PLaSM program: $TEST\_COEFS\_0$ builds the cartesian product of $APOL$ with its variables and with integers $0, 1, 2, 3$. Then the coefficient is applied on all combinations.

```
RCF_COEF:<MKRCF: 'x1+x2+x1*x2-1' , 'x1' ,1>;
% RCF poly: x2+1 %

RCF_VARS:APOL;
% < 'x2' , 'x1' > %

DEF TEST_COEF_0=CART:<<APOL>,RCF_VARS:APOL,0 .. 3>; TEST_COEF_0;
% < < RCF poly: 3*x1*x2^2-3*x1^2*x2+x1^3+1 , 'x2' , 0 > , < RCF poly: 3*
    x1*x2^2-3*x1^2*x2+x1^3+1 , 'x2' , 1 > , < RCF poly: 3*x1*x2^2-3*x1^2
    *x2+x1^3+1 , 'x2' , 2 > , < RCF poly: 3*x1*x2^2-3*x1^2*x2+x1^3+1 , '
    x2' , 3 > , < RCF poly: 3*x1*x2^2-3*x1^2*x2+x1^3+1 , 'x1' , 0 > , <
    RCF poly: 3*x1*x2^2-3*x1^2*x2+x1^3+1 , 'x1' , 1 > , < RCF poly: 3*x1
    *x2^2-3*x1^2*x2+x1^3+1 , 'x1' , 2 > , < RCF poly: 3*x1*x2^2-3*x1^2*
    x2+x1^3+1 , 'x1' , 3 > > %

AA:RCF_COEF:TEST_COEF_0;
% < RCF poly: x1^3+1 , RCF poly: -3*x1^2 , RCF poly: 3*x1 , 0 , 1 , RCF
    poly: 3*x2^2 , RCF poly: -3*x2 , 1 > %
```

Various functions are available to extract useful properties of a polynomial:

rcf_norm(poly) the norm of a polynomial, norm(poly) =
if (poly is constant) then |poly|
else sum of norm(coefficient-i) for i = 0,..,deg(poly);

rcf_max_norm(poly) the max norm of a polynomial max norm(poly) =
max of norm(coefficient-i) for i = 0,..,deg(poly);

rcf_constant_term(poly) the constant term of poly in the main variable;

rcf_terms(poly) all the terms of poly in a list (a list of polynomials);

rcf_total_degree(poly) total degree of poly, i.e. the highest degree of any term (monomial) in the polynomial;

rcf_coeff_denom(poly) a number that when multiplied into poly, will cause poly to have integer coefficients;

rcf_coeff_gcd(poly) the gcd of all the coefficients of poly, assuming they're integers.

```
RCF_NORM:APOL;
% 8 %

RCF_MAX_NORM:APOL;
% 3 %

RCF_CONSTANT_TERM:APOL;
% RCF poly: x1^3+1 %
```

```
RCF_TERMS:APOL;
% < RCF poly: 3*x1*x2^2 , RCF poly: -3*x1^2*x2 , RCF poly: x1^3 , 1 > %

RCF_TOTAL_DEGREE:APOL;
% 3 %

RCF_COEFF_DENOM:(APOL/3);
% 3 %

RCF_COEF_GCD:(APOL*5);
%  5 %
```

A polynomial may be evaluated on any variable. Furthermore any variable may substituted with another polynomial.

rcf_evaluate(*polyvval*) Evaluate poly in variable v at value val. Return a new polynomial that is the evaluated one. The new polynomial will not contain v. The value val must be constant. For non-constant values use *rcf_substitute*.

rcf_substitute(*polys_list*) Apply a variable substitution to all of poly's variables. *s_list* is an associative list of pairs (v_i, p_i), where v_i is a variable and p_i is a polynomial. The substitution $v_i == p_i$ is applied for each i.

```
RCF_EVALUATE:<MKRCF: 'x1^2-x1*x2+1' , 'x1' ,2>;
% RCF poly: -2*x2+5 %

AA:RCF_EVALUATE:TEST_COEF_0;
% < RCF poly: x1^3+1 , RCF poly: x1^3-3*x1^2+3*x1+1 , RCF poly: x1^3-6*x1
    ^2+12*x1+1 , RCF poly: x1^3-9*x1^2+27*x1+1 , 1 , RCF poly: 3*x2^2-3*
    x2+2 , RCF poly: 6*x2^2-12*x2+9 , RCF poly: 9*x2^2-27*x2+28 > %

RCF_SUBSTITUTE:<MKRCF: 'x1^2-x1*x2+1',<<'x2',MKRCF: 'x1+1'>>>;
% RCF poly: -x1+1 %

RCF_SUBSTITUTE:<MKRCF: 'x1^2-x1*x2+1',<<'x2',MKRCF: 'x1'>,<'x1',MKRCF: 'x2'
    >>>;
% RCF poly: x2^2-x1*x2+1 %
```

Moreover a polynomial has also a functional value and may be applied to values (evaluation) or other polynomials (substitution). In this case the lexicographical order of the variables is considered when applying to multiple arguments. This capability opens interesting possibilities when operating with higher-order functions in PLaSM. However this capability is both incompatible with the optimized version of plasm where scheme application is used whenever possible. Furthermore to operate with high order functions it must use the fully lifted combinatorial environment (e.g. the definition of all PLaSM combinators and the lambda substitution must refer PLaSM function application instead of scheme's one). The fully lifted environment is quite a hindrance to performance.

```
IsFun:APOL;
% True %

(mkrcf:'x1+2*x2'):<1,2>;
% 5 %

(mkrcf:'x1+2*x2'):<mkrcf:'x2',mkrcf:'x1'>;
% RCF poly: x2+2*x1 %

APOL:<3,2>;
% 10 %

APOL:<mkrcf:'y^2',mkrcf:'y-1'>;
% RCF poly: y^6-3*y^5+6*y^4-6*y^3+3*y^2+1 %

DEF TEST(x::TT)=K:x + ID ** K:3;
TEST:1:2;
% 9.0 %
TEST:(mkrcf:'x1'):(mkrcf:'x2');
% RCF poly: x2^3+x1 %
TEST:(mkrcf:'x1^5+x2^3'):(mkrcf:'x2^2-x1*x2');
% RCF poly: x2^6-3*x1*x2^5+3*x1^2*x2^4-x1^3*x2^3+x2^3+x1^5 %
(TEST:(mkrcf:'x1^5+x2^3'):(mkrcf:'x2^2-x1*x2')):<mkrcf:'u1u^4-1',mkrcf:'
    v2n'>;
% RCF poly: v2n^6-3*u1u^4*v2n^5+3*v2n^5+3*u1u^8*v2n^4-6*u1u^4*v2n^4+3*v2n
    ^4-u1u^12*v2n^3+3*u1u^8*v2n^3-3*u1u^4*v2n^3+2*v2n^3+u1u^20-5*u1u^16+
    10*u1u^12-10*u1u^8+5*u1u^4-1 %
```

A series of functions useful both for polynomial manipulation and algebraic geometry (in addition to the ones in next section) are available:

$rcf\_homogenize(poly, v)$ Return a homogenous version of *poly* using the new variable v. Useful in algebraic geometry to find loci at infinity.

$rcf\_rroots(poly)$ Returns lists of real roots of univariate *poly*.

$rcf\_pdiff(poly, v)$ The partial derivative of *poly* with respect to variable v.

$rcf\_interpolate(points, vals, var, n)$ Return the polynomial $p(var)$ such that $p(var = points[i]) = vals[i]$, with $i = 0...n-1$

$rcf\_gcd(p_1 p_2)$ Return the greatest common divisor (GCD) of the two polynomials p_1 and p_2. Algorithm is from (Brown, JACM V18 #4 Oct 1971 pp478-504). The presence of such algorithm enables operating on rational functions.

$rcf\_primitive\_part(p)$ Primitive-part of polynomial p. The primitive part of a polynomial is itself divided by its content (see below). The coefficients of the polynomial returned will be relatively prime to each other.

rcf_content(p) The content of polynomial p. The content of a polynomial is the GCD of its coefficients.

```
RCF_HOMOGENIZE:<mkrcf:'x1^2-x1*x2+1','x3'>;
% RCF poly: x3^2-x1*x2+x1^2 %

RCF_PDIFF:<mkrcf:'x1*x2^2+x1*x2-2','x2'>;
% RCF poly: 2*x1*x2+x1 %

RCF_RROOTS:(mkrcf:'x1-3' * mkrcf:'x1+3' * mkrcf:'(x1-1)^3');
% < 3 , -3 , 1.00000351322 , 1.00000351322 , 0.999992973551 > 5

DEF INT_POLY=rcf_interpolate:<<-1,0,1,2>,<1, 0, 1, 2>, 'z'>; INT_POLY;
% RCF poly: -1/3*z^3+z^2+1/3*z %

AA:(rcf_evaluate~[K:INT_POLY,K:'z',ID]):<-1,0,1,2>;
% < 1 , 0 , 1 , 2 > %

RCF_GCD:<mkrcf:'x1^2-1',mkrcf:'x1-1'>;
 % RCF poly: x1-1 %

RCF_CONTENT:(mkrcf:'3*x1*(y2+2)*(3*z3^2+2)');
% RCF poly: 3*x1*y2+6*x1 %

RCF_PRIMITIVE_PART:(mkrcf:'3*x1*(y2+2)*(3*z3^2+2)');
% RCF poly: 3*z3^2+2 %
```

A.2 Algebraic Functions

In addition to the fundamental operations on polynomials the algebra subsystem implements a series of sophisticated algorithms typical of algebraic geometry. These too have been ported to PLaSM, however the graphic visualization is still work in progress. Furthermore the rational parameterization and the conversion between power to Berstein/Bézier is still not effective. For these reason the images have been created with GANITH while the computation output is in PLaSM whenever these function are working properly.

Compactify, Resultant, Realify and Conicoid Parameterization

rcf_compactify(p) Compactify the polynomial p which may be in two or three variables. If it is in two variables x, y, then it is homogenized with the variable z, and the intersection of this surface with $x^2+y^2+z^2-1$ is displayed (see figure A.1.a). Otherwise a homogenizing variable w is used and the resultant of the homogeneous surface and $x^2+y^2+z^2+w^2-1$ with respect to w, is displayed.

rcf_resultant(p_1, p_2, n, v) Compute the resultant $Res_{(v,n)}(p_1, p_2)$, that is, eliminate the variable in $n = 0$ or reduce to a term of degree n. This is the fundamental operator to compute intersections between algebraic varieties.

rcf_realify2d(p) Compute and return the real and imaginary parts of a curve, and their resultant. If p is a polynomial in two variables x and y, the substitution

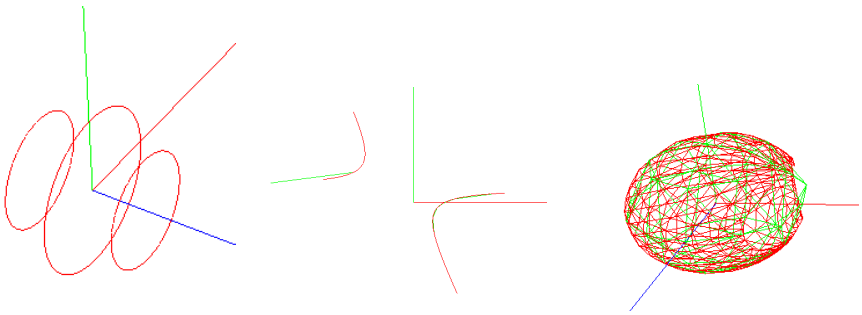
$$\begin{aligned} x &= x + iw \\ y &= y + iz \end{aligned}$$

is made; then the real and imaginary parts are computed, and their resultant w.r.t. w is returned. Their resultant is a polynomial in three variables defining a surface, which is displayed.

rcf_param2dNv$(poly, (v_1, \ldots, v_N), (u_1, \ldots, u_{N-1}), (p_1, \ldots, p_N)$ Create a rational parameterization of the hypersurface of degree 2 and dimension N defined by the equation $poly = 0$. The $(v_1, \ldots, v_N)$ are variables of the polynomial that define the geometric space. The $(u_1, \ldots, u_{N-1})$ are the variables to be used for the parameterization. The $(p_1, \ldots, p_N)$ is an optional point on the surface to use as the base of a parameterization. The result is an ordered list of pairs, with each pair corresponding by position to a variable in v_i. Each pair will be a list of two items, the first being the numerator of the rational function for that variable, and the second being the denominator. The numerator and denominator polynomials will be in the variables of u_i. In figure A.1.b shows both the rendering of the implicit curve and its parameterization.

```
rcf_compactify:(mkrcf:'x^2+y^2-1');
%<<RCF poly: 26244*y^4+52488*x^2*y^2-13284*y^2-27216*x*y+26244*x^4-
    39204*x^2+529 , RCF poly: 972*y+2268*x , RCF poly: -162*y^2-162*x^2+
    23 > , < 7 , 3 > , < RCF poly: -z^2+y^2+x^2 , < RCF poly: z^2+y^2+x^
    2-1 >>> %

rcf_compactify:(mkrcf:'x^2+2*y^2+3*z^2-1');
% RCF poly: 16*z^4+24*y^2*z^2+16*x^2*z^2-8*z^2+9*y^4+12*x^2*y^2-6*y^2+4*x
    ^4-4*x^2+1 %

rcf_resultant:<mkrcf:'x^2+y^2+z^2-1',mkrcf:'y-z',0,'z'>>;
% RCF poly: 2*y^2+x^2-1 %

rcf_resultant:<mkrcf:'x^2+y^2+z^2-1',mkrcf:'y-z',1,'z'>>;
% RCF poly: -z+y %

rcf_resultant:<mkrcf:'x^2+y^2+z^2-1',mkrcf:'x^2+y^2+(z-1)^2-9/16',0,'z'>;
% RCF poly: 1024*y^2+1024*x^2-495 %

rcf_resultant:<mkrcf:'x^2+y^2+z^2-1',mkrcf:'x^2+y^2+(z-1)^2-9/16',1,'z'>;
```

```
% RCF poly: -32*z+23 %

rcf_parameterize:<mkrcf:'2*x*y+y^2+1',<'x' ,'y'>,<'u'>>;
% << RCF poly: -u^2-1 , RCF poly: 2*u > , < RCF poly: u , 1 >> %

rcf_parameterize:<mkrcf:'x^2+2*y^2+z-1',<'x' ,'y', 'z'>,<'u','v'>>;
% << RCF poly: u , 1 > , < RCF poly: v , 1 > , < RCF poly: -2*v^2-u^2+1
    , 1 >> %
```



(A) (B.1) (B.2)

Figure A.1: a: Compactify in two variables: x^2+y^2-1. b: Quadratic parameterization: (1) curve $2xy+y^2+1$; (2) surface $x^2+2y^2+z^2-1$. Image generated with GANITH [BR90a].

Intersection

rcf_intersect2e2d(p_1, p_2) Intersect two plane algebraic curves and display their points of intersection, along with the curves. Given two plane curves $p1(x,y)=0$ and $c2(x,y)=0$ the algorithm returns (pts aux) where:

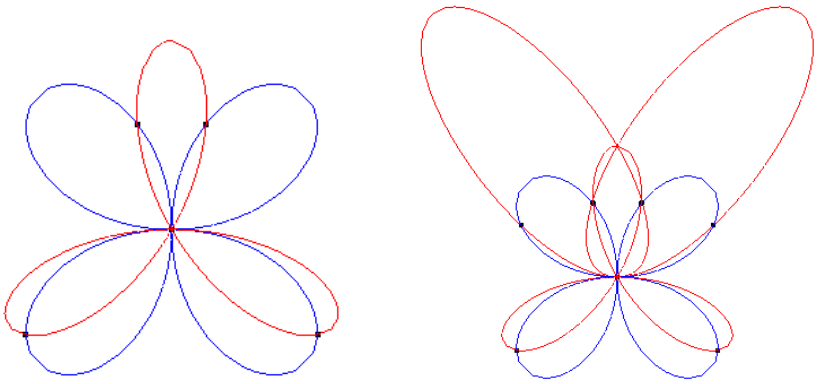
pts = list of intersection points (x y), they may be repeated if multiple;

aux = is the list (res sgcd s11 s10 alpha) where:

- **res** = resultant
- **subres** = sgcd*(s11*y+s10)
- **alpha** = plane transformation parameter

See figure A.2 for GANITH graphic output.

```
rcf_intersect2e2d:<mkrcf:'(x^2+y^2)^2+3*x^2*y-y^3', mkrcf:'(x^2+y^2)^3-4*
    x^2*y^2'>;
% < < < 0 , 0 > , < 0 , 0 > , < 0 , 0 > , < 0 , 0 > , < 0 , 0 > , < 0 , 0
    > , < 0 , 0 > , < 0 , 0 > , < 0 , 0 > , < 0 , 0 > , < 0 , 0 > , < 0
    , 0 > , < 0 , 0 > , < 0 , 0 > , < 0.181635632001 ,
    0.5590169748238133 > , < -0.181635632001 , 0.5590169748238133 > , <
    -0.769420884294 , -0.5590169943759334 > , < 0.769420884294 , -
    0.5590169943759334 > > , RCF poly: 4096*x^18-2560*x^16+80*x^14 , <
    RCF poly: 4*x^6 , RCF poly: -464*x^4+379*x^2-12 , y , RCF poly: -64*
    x^6+124*x^4-4*x^2 , 1 > > %

rcf_intersect2e2d:<mkrcf:'2*x^4-3*x^2*y+y^2-2*y^3+y^4',mkrcf:'(x^2+y^2)^3
    -4*x^2*y^2'>;
% < < < 0 , 0 > , < 0 , 0 > , < 0 , 0 > , < 0 , 0 > , < 0 , 0 > , < 0 , 0
    > , < 0 , 0 > , < 0 , 0 > , < 0 , 0 > , < 0 , 0 > , < 0 , 0 > , < 0
    , 0 > , < 0.191264461949 , 0.5710907930166544 > , < -0.191264461949
    , 0.5710907930166544 > , < -0.734809903348 , 0.3952342840395756 > ,
    < 0.734809903348 , 0.3952342840395756 > > , RCF poly: 729*x^24-2079
    *x^22+10102*x^20-2743*x^18+986*x^16-1263*x^14+45*x^12 , < RCF poly:
    x^4 , RCF poly: 270*x^10-734*x^8+1242*x^6+205*x^4+319*x^2-12 , y ,
    RCF poly: -27*x^12+430*x^10-832*x^8-176*x^6-240*x^4+9*x^2 , 1 > > %
```



(a) (b)

Figure A.2: Curve intersection: (a) $\{(x^2+y^2)^2+3x^2y-y^3=0\} \cap \{(x^2+y^2)^3-4x^2y^2=0\}$; (b) $\{2x^4-3x^2y+y^2-2y^3+y^4=0\} \cap \{(x^2+y^2)^3-4x^2y^2=0\}$. Image generated with GANITH [BR90a].

$rcf\_intersect2e3d(p_1,p_2)$ Intersect the surfaces p_1 and p_2, displaying their curve of

intersection. The intersection curve is projected onto a plane curve first, so this object can be refined by changing the plane curve bounding box. Algorithm returns (crv aux) where

crv = (res s1 s11 -s10)

aux = Transformation parameters $(alpha, beta)$

See figure A.3 for GANITH graphic output.

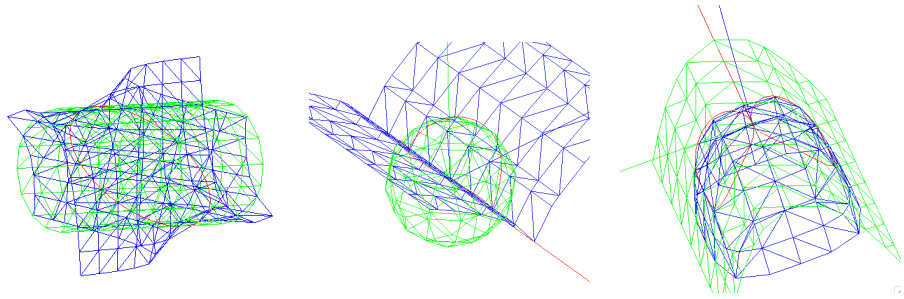
```
rcf_intersect2e3d:<mkrcf:'z^2-y^3',mkrcf:'x^2+y^2+z^2-1'>;
%<<RCF poly: 2197000*y^6-2129400*x*y^5+2737800*y^5+1144260*x^2*y^4-
    442260*x*y^4+455679*y^4-368928*x^3*y^3+341496*x^2*y^3+294840*x*y^3-
    2548260*y^3+79218*x^4*y^2+1479870*x^3*y^2+1713150*x^2*y^2-1479870*x*
    y^2-1792368*y^2-10206*x^5*y-594864*x^4*y+20412*x^3*y+1189728*x^2*y-
    10206*x*y-594864*y+729*x^6+30618*x^5+909792*x^4-61236*x^3-1821771*x^
    2+30618*x+911250 , RCF poly: -148590*y^2+48006*x*y-67554*y-1539*x^2-
    157626*x-3753 , RCF poly: 15730*y^3+378*x*y^2+11259*y^2-3591*x^2*y+
    3591*y+378*x^3+11259*x^2-378*x-11259 > , < RCF poly: 2197000*y^6-
    2129400*x*y^5+2737800*y^5+1144260*x^2*y^4-442260*x*y^4+455679*y^4-
    368928*x^3*y^3+341496*x^2*y^3+294840*x*y^3-2548260*y^3+79218*x^4*y^2
    +1479870*x^3*y^2+1713150*x^2*y^2-1479870*x*y^2-1792368*y^2-10206*x^5
    *y-594864*x^4*y+20412*x^3*y+1189728*x^2*y-10206*x*y-594864*y+729*x^6
    +30618*x^5+909792*x^4-61236*x^3-1821771*x^2+30618*x+911250 , RCF
    poly: -148590*y^2*z+48006*x*y*z-67554*y*z-1539*x^2*z-157626*x*z-3753
    *z-15730*y^3-378*x*y^2-11259*y^2+3591*x^2*y-3591*y-378*x^3-11259*x^2
    +378*x+11259 > , < 7 , 3 >> %

rcf_intersect2e3d:<mkrcf:'z+x^4+y^4',mkrcf:'z+y^2'>;
%<<RCF poly: -y^4+y^2-x^4 , 1 , RCF poly: -y^4-x^4 > , < RCF poly: -y^
    4+y^2-x^4 , RCF poly: z+y^4+x^4 > , < 0 , 0 >> %

rcf_intersect2e3d:<mkrcf:'(x+y)*z-y^2+1',mkrcf:'x^2+y^2-2'>;
%<<RCF poly: y^2+x^2-2 , RCF poly: y+x , RCF poly: y^2-1 > , < RCF
    poly: y^2+x^2-2 , RCF poly: y*z+x*z-y^2+1 > , < 0 , 0 >> %
```

$rcf\_intersect3e3d(p_1, p_2, p_3)$ returns the intersections of the surfaces two by two (just a shortcut for displaying their pairwise intersections (i.e. calling $rcf\_intersect2e3d$ three times).

$rcf\_intersect3e3dtem(p_1, p_2, p_3)$ Computes the intersection points of the the surfaces p_1, p_2, and p_3.

See figure A.4 for GANITH graphic output.

```
rcf_intersect3e3d:<mkrcf:'z^2+y^2-1',mkrcf:'z^2+x^2-1',mkrcf:'z^2-y^3'>;
```



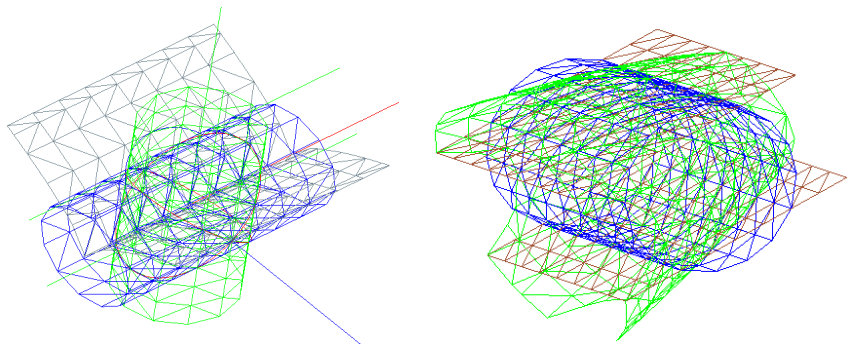
(a) (b) (c)

Figure A.3: Surface intersection: (a) $\{(x+y)*z-y^2+1=0\} \cap \{x^2+y^2-2=0\}$; (b) $\{z+x^4+y^4=0\} \cap \{z+y^2=0\}$; (c) $\{z^2-y^3=0\} \cap \{x^2+y^2+z^2-1=0\}$. Image generated with GANITH [BR90a].

```
% < < < RCF poly: 16900*y^4-10920*x*y^3-1080*x^2*y^2-15080*y^2-7560*x^3*y
    +18480*x*y+8100*x^4-10440*x^2+1600 , RCF poly: -780*y+1260*x , RCF
    poly: 130*y^2-90*x^2-40 > , < 7 , 3 > > , < < RCF poly: 2197000*y^6-
    2129400*x*y^5+1144260*x^2*y^4-456300*y^4-368928*x^3*y^3+673920*x^2*y
    ^3+294840*x*y^3-2737800*y^3+79218*x^4*y^2+1326780*x^3*y^2-110808*x^2
    *y^2-1326780*x*y^2+31590*y^2-10206*x^5*y-568620*x^4*y+20412*x^3*y+
    1137240*x^2*y-10206*x*y-568620*y+729*x^6+30618*x^5+850743*x^4-61236*
    x^3-1703673*x^2+30618*x+852201 , RCF poly: -152100*y^2+49140*x*y-
    1782*x^2-147420*x-3510 , RCF poly: 16900*y^3-3510*x^2*y+3510*y+378*x
    ^3+10530*x^2-378*x-10530 > , < 7 , 3 > > , < < RCF poly: 531441*y^6+
    1062882*y^5+413343*y^4-944784*y^3-1161297*y^2-380538*y+589761 , RCF
    poly: -56862*y^2-43740*y-2430 , RCF poly: 5832*y^3+7290*y^2+2268*y-
    7290 > , < 7 , 3 > > > %

rcf_intersect3e3dtem:<mkrcf:'x^2+y+z^2-3',mkrcf:'x^2+y^2-2',mkrcf:'x^2-1'
    >;
% < <-0.9717549 0.9897984 0.9717549>,
    <-0.9717549 0.9897984 0.9717549>,
    <1.0719498 -0.8264992 -1.5520653>,
    <-0.79833984 1.3150873 1.2784553>,
    <1.0833083 -0.8446808 -1.5780976> >;
```

Weierstrass Preparation, Newton Factorization, Hensel's Lemma

rcf_weierstrass(p,n) The argument p is a polynomial and n is an integer. Factor algebraic polynomial $p(x,y)$ into $p(x,y) = g(x,y)\ h(x,y)$ by Weierstrass Preparation up to degree n. Curves $g(x,y)=0$ and $h(x,y)=0$ are displaced. Returns the pair of polynomials (g,h).



(a) (b)

Figure A.4: Surface triple intersection: (a) $\{z^2+y^2-1\} \cap \{z^2+x^2-1\} \cap \{z^2-y^3\}$; (b) $\{x^2+y+z^2-3\} \cap \{x^2+y^2-2\} \cap \{x^2-1\}$. Image generated with GANITH [BR90a].

$localparam2d(p,s,deg)$ Computes newton powes series aproximation for the *non* singular point in $(0,0)$. Returns parameter expression of the given curve at the origin.

$rcf\_newton(p,n)$ Computes newton powes series aproximation for the singular point in (0,0). Returns a list of pairs, each pair is parameter expression of the given curve at the origin. The parameter is assumed to be positive. That is we got the expression to $p = f(u)$ and $p = f(-u)$. The argument p is a polynomial and n is an integer. Factor algebraic polynomial $p(x,y)$ into

$$p(x,y) = \Pi_{i=1}^{d}(y - n_i(t)), \qquad x = t^{m_i}$$

at origin and display every branch (see fig. A.5(a,c)).

$$\begin{cases} x &= t^{m_i} \\ y &= n_i(t) \end{cases}$$

where $n_i(t)$ is power series up to degree n.

$rcf\_gnewton(p,a,n)$ The argument p is a polynomial, a is a real, and n is an integer. At each of the points, that satisfy

$$\begin{cases} x &= a \\ p(x,y) &= 0 \end{cases}$$

make Newton factorization of degree n and every branch is displayed (see fig. A.5(b,d)).

```
rcf_weierstrass:<mkrcf:'y^4 - 2*y^3 + y^2 - 3*x^2*y + 2*x^4', 6>;
%< RCF poly: y^2+3501972*x^14*y+248484*x^12*y+18441*x^10*y+1462*x^8*y+
    129*x^6*y+14*x^4*y+3*x^2*y-2*y-4844256*x^14-345355*x^12-25806*x^10-
    2068*x^8-186*x^6-21*x^4-6*x^2+1 , RCF poly: y^2-3501972*x^14*y-
    248484*x^12*y-18441*x^10*y-1462*x^8*y-129*x^6*y-14*x^4*y-3*x^2*y+
    224760*x^14+16610*x^12+1308*x^10+114*x^8+12*x^6+2*x^4 > %

rcf_newton:<mkrcf:'x^3 -x^2+ y^2',6>;
%< < RCF poly: -s , RCF poly: 0.02734375*s^6-0.0390625*s^5+0.0625*s^4-
    0.125*s^3+0.5*s^2+1.0*s > , < RCF poly: -s , RCF poly: -0.02734375*s
    ^6+0.0390625*s^5-0.0625*s^4+0.125*s^3-0.5*s^2-1.0*s > , < RCF poly:
    s , RCF poly: -0.02734375*s^6-0.0390625*s^5-0.0625*s^4-0.125*s^3-0.5
    *s^2+1.0*s > , < RCF poly: s , RCF poly: 0.02734375*s^6+0.0390625*s^
    5+0.0625*s^4+0.125*s^3+0.5*s^2-1.0*s > > %

rcf_gnewton:<mkrcf:'x^3 -x^2+ y^2',0,6>;
%< < RCF poly: s , RCF poly: 0.02734375*s^6+0.0390625*s^5+0.0625*s^4+
    0.125*s^3+0.5*s^2-1.0*s > , < RCF poly: s , RCF poly: -0.02734375*s^
    6-0.0390625*s^5-0.0625*s^4-0.125*s^3-0.5*s^2+1.0*s > , < RCF poly: -
    s , RCF poly: -0.02734375*s^6+0.0390625*s^5-0.0625*s^4+0.125*s^3-0.5
    *s^2-1.0*s > , < RCF poly: -s , RCF poly: 0.02734375*s^6-0.0390625*s
    ^5+0.0625*s^4-0.125*s^3+0.5*s^2+1.0*s > > %

rcf_newton:<mkrcf:'y^5 +2*x*y^4 -x*y^2 -2*x^2*y -x^3 +x^4',5>;
%< < RCF poly: -s^3 , RCF poly: -0.3333333333333333*s^5-1.0*s > , < RCF
    poly: s^6 , 0 > , < RCF poly: s^6 , 0 > > %

rcf_gnewton:<mkrcf:'y^5 +2*x*y^4 -x*y^2 -2*x^2*y -x^3 +x^4',1,5>;
%< < RCF poly: 378591227645 75/129961739795077*s^5-95885166 43/94931877133
    *s^4-32632332/69343957*s^3+42909/50653*s^2-18/37*s+1 , RCF poly: s-2
    > , < RCF poly: s+1 , RCF poly: -1308353/4782969*s^5+23974/177147*s
    ^4-1187/6561*s^3-55/243*s^2+2/9*s+1 > , < RCF poly: s+1 , RCF poly:
    -195/256*s^5-181/128*s^4+11/16*s^3+11/8*s^2+1/2*s > > %
```

Local Parameterization

rcf_localpower2d(p,s,n,x_0,y_0) The argument p is a polynomial, s is a variable, n is an integer and x_0,y_0 are reals. At point (x_0,y_0), compute power series expansion of $p(x,y)=0$ in s up to degree n, where (x_0,y_0) can be either simple point or singular point of curve $p(x,y)=0$. If (x_0,y_0) is not on the curve, then the expansion if taken at the nearest point to (x_0,y_0) (see fig. A.6(a,b)).

rcf_localpade2d(p,s,m,n,x_0,y_0) The argument p is a polynomial, s is a variable, n is an integer and x_0,y_0 are reals. At point (x_0,y_0), compute Padè approximation of

degree (m,n) to the power series expansion of $p(x,y)=0$, where s is parameter of Padè approximant, and (x_0,y_0) is the same as localpower2d. (see fig. A.6(c,d)).

```
rcf_localpower2d:<mkrcf:'x^3 -x^2+ y^2','s',6,<0,0>>;
%<<0 , 0> , <<RCF poly: -s , RCF poly: 0.02734375*s^6-0.0390625*s^5
    +0.0625*s^4-0.125*s^3+0.5*s^2+1.0*s > , <RCF poly: -s , RCF poly: -
    0.02734375*s^6+0.0390625*s^5-0.0625*s^4+0.125*s^3-0.5*s^2-1.0*s > ,
    <RCF poly: s , RCF poly: -0.02734375*s^6-0.0390625*s^5-0.0625*s^4-
    0.125*s^3-0.5*s^2+1.0*s > , <RCF poly: s , RCF poly: 0.02734375*s^6
    +0.0390625*s^5+0.0625*s^4+0.125*s^3+0.5*s^2-1.0*s >>> %

rcf_localpower2d:<mkrcf:'x^3 -x^2+ y^2','s',6,<1,0>>;
%<<1 , 0> , <<RCF poly: -7*s^6-2*s^4-s^2+1 , RCF poly: s >>> %

rcf_localpade2d:<mkrcf:'x^3 -x^2+ y^2','s',<3,3>,<0,0>>;
%<<0 , 0> , <<RCF poly: s , 1 , RCF poly: -0.4375*s^3+
    1.4000000000000004*s^2-1.0*s , RCF poly: 0.006250000000000001*s^3+
    0.11250000000000002*s^2-0.8999999999999999*s+1.0 > , <RCF poly: s ,
     1 , RCF poly: 0.4375*s^3-1.4000000000000004*s^2+1.0*s , RCF poly:
    0.006250000000000001*s^3+0.11250000000000002*s^2-0.8999999999999999*
    s+1.0 > , <RCF poly: -s , 1 , RCF poly: -0.4375*s^3-
    1.4000000000000004*s^2-1.0*s , RCF poly: -0.006250000000000001*s^3+
    0.11250000000000002*s^2+0.8999999999999999*s+1.0 > , <RCF poly: -s
    , 1 , RCF poly: 0.4375*s^3+1.4000000000000004*s^2+1.0*s , RCF poly:
    -0.006250000000000001*s^3+0.11250000000000002*s^2+0.8999999999999999
    *s+1.0 >>> %

rcf_localpade2d:<mkrcf:'x^3 -x^2+ y^2','s',<3,3>,<1,0>>;
%<<1 , 0> , <<RCF poly: 3/2*s^4-9/2*s^2+1 , RCF poly: -7/2*s^2+1 ,
    RCF poly: s , 1>>> %
```

Curve Approximation by Piecewise Rational Elements

It is the implementation of the algorithm presented in "Piecewise Rational Approximation of Real Algebraic Curves"[BX97]. Plane curve algorithm has been ported in PLaSM, however output is currently inexact. Both text and graphics output is from GANITH.

piecerational2d$(p,s,m,n,a,b,c,d,\varepsilon,continuity)$ The argument p is a polynomial, s is a variable, *m, n* are integers and *a,b,c,d* and ε are reals; *continuity* is a integer. For the given algebraic curve $p(x,y)=0$, compute piecewise (m,n) rational approximation within the given box

$$\{(x,y): \quad a \le x \le b, \quad c \le y \le d\}$$

and error limit ε, where

$$continuity = \begin{cases} \text{- 1} & c^{-1} & continuity \\ 0 & c^{0} & continuity \\ 1 & c^{1} & continuity \end{cases}$$

piecerationalp2d$(p,s,m,n,a,b,c,d,\varepsilon,continuity,k,polygon)$ Similar to the *piecerational2d* command. The argument k is the modified degree. If *polygon* is 1 the polygon is drawn.

spacecurveiis$(f_1,f_2,a_0,a_1,b_0,b_1,c_0,c_1,m,n,k,\varepsilon,slice)$ For the given space curve defined by the intersection of implicit surfaces $f_1(x,y,z)=0$ and $f_2(x,y,z)=0$, a bounding box defined by $(x,y,z): x\in[a_0,a_1], y\in[b_0,b_1], z\in[c_0,c_1]$, the degree m,n of the rational function to be used, the smoothness index k and error limit ε, compute a parametric piecewise rational approximation.

spacecurvepc$(p_1,q_0,p_1,q_1,p_2,q_2,a_0,a_1,b_0,b_1,c_0,c_1,m,n,k,\varepsilon,slice)$ For the given space curve defined by the parametric form $x(t)=p_0(t)/q_0(t), y(t)=p_1(t)/q_1(t), z(t)=p_2(t)/q_2(t)$, a bounding box defined by $(x,y,z): x\in[a_0,a_1], y\in[b_0,b_1], z\in[c_0,c_1]$, the degree m,n of the rational function to be used, the smoothness index k and error limit ε, compute a parametric piecewise rational approximation with arc-length as parameter.

```
rcf_localpower2d:<mkrcf:'x^3 -x^2+ y^2','s',6,<0,0>>;
%<<0 , 0> , <<RCF poly: -s , RCF poly: 0.02734375*s^6-0.0390625*s^5
    +0.0625*s^4-0.125*s^3+0.5*s^2+1.0*s> , <RCF poly: -s , RCF poly: -
    0.02734375*s^6+0.0390625*s^5-0.0625*s^4+0.125*s^3-0.5*s^2-1.0*s> ,
    <RCF poly: s , RCF poly: -0.02734375*s^6-0.0390625*s^5-0.0625*s^4-
    0.125*s^3-0.5*s^2+1.0*s> , <RCF poly: s , RCF poly: 0.02734375*s^6
    +0.0390625*s^5+0.0625*s^4+0.125*s^3+0.5*s^2-1.0*s>>> %

rcf_localpower2d:<mkrcf:'x^3 -x^2+ y^2','s',6,<1,0>>;
%<<1 , 0> , <<RCF poly: -7*s^6-2*s^4-s^2+1 , RCF poly: s>>> %

rcf_localpade2d:<mkrcf:'x^3 -x^2+ y^2','s',<3,3>,<0,0>>;
%<<0 , 0> , <<RCF poly: s , 1 , RCF poly: -0.4375*s^3+
    1.4000000000000004*s^2-1.0*s , RCF poly: 0.00625000000000001*s^3+
    0.11250000000000002*s^2-0.8999999999999999*s+1.0> , <RCF poly: s ,
     1 , RCF poly: 0.4375*s^3-1.4000000000000004*s^2+1.0*s , RCF poly:
    0.00625000000000001*s^3+0.11250000000000002*s^2-0.8999999999999999*
    s+1.0> , <RCF poly: -s , 1 , RCF poly: -0.4375*s^3-
    1.4000000000000004*s^2-1.0*s , RCF poly: -0.00625000000000001*s^3+
    0.11250000000000002*s^2+0.8999999999999999*s+1.0> , <RCF poly: -s
    , 1 , RCF poly: 0.4375*s^3+1.4000000000000004*s^2+1.0*s , RCF poly:
    -0.00625000000000001*s^3+0.11250000000000002*s^2+0.8999999999999999
    *s+1.0>>> %
```

```
rcf_localpade2d:<mkrcf:'x^3 -x^2+ y^2','s',<3,3>,<1,0>>;
%<<1 , 0> , <<RCF poly: 3/2*s^4-9/2*s^2+1 , RCF poly: -7/2*s^2+1 ,
    RCF poly: s , 1>>> %
```

Power To Bernstein Conversion

bern3d(p) Convert a polynomial $p = f(x,y,z)$ in power basis to a polynomial $F(s,t,u)$ in barycentric coordinates, defined with respect to a default reference tetrahedron. The polynomial $F(s,t,u)$ is displayed in the output window. The default reference tetrahedron has vertices $(n,0,0)$, $(0,n,0)$, $(0,0,n)$, and $(0,0,0)$, where n is the degree of the polynomial. The polynomial may contain free parameters.

berntetra3d$(surf,(x_1,y_1,z_1),(x_2,y_2,z_2),(x_3,y_3,z_3),(x_4,y_4,z_4))$ Similar to the *bern3d* command except the reference tetrahedron has vertices (x_1,y_1,z_1), (x_2,y_2,z_2), (x_3,y_3,z_3), (x_4,y_4,z_4).

Currently unavailable in PLaSM.

```
berntetra3d(a*x^2 + 2*a*x*y + d*x*z - 2*a*x +a*y^2 + e*y*z -2*a*y + c*z^2
    + b*z + a, (2
Tetrahedron set to:
(2.000000 1.000000 1.000000)
(1.000000 2.000000 1.000000)
(1.000000 1.000000 2.000000)
(0.000000 0.000000 0.000000)
Weights for the control points:
w000: 1.000000*A^1
w001: 1.000000*B^1+-1.000000*A^1
w002: 2.000000*E^1+2.000000*B^1+4.000000*C^1+2.000000*D^1+1.000000*A^1
w010: 0.500000*B^1+-2.000000*A^1
w011: 2.500000*E^1+2.000000*C^1+1.500000*B^1+2.000000*A^1+1.500000*D^1
w020: 2.000000*E^1+1.000000*B^1+1.000000*C^1+1.000000*D^1+4.000000*A^1
w100: 0.500000*B^1+-2.000000*A^1
w101: 1.500000*E^1+2.000000*C^1+1.500000*B^1+2.000000*A^1+2.500000*D^1
w110: 1.500000*E^1+1.000000*C^1+1.000000*B^1+4.000000*A^1+1.500000*D^1
w200: 1.000000*E^1+1.000000*B^1+1.000000*C^1+2.000000*D^1+4.000000*A^1

bern3d(a*x^2 + 2*a*x*y + d*x*z - 2*a*x + a*y^2 + e*y*z -2*a*y + c*z^2 + b
    *z + a)
w000: 1.000000*A^1
w001: 1.000000*A^1+1.000000*B^1
w002: 1.000000*A^1+2.000000*B^1+4.000000*C^1
w010: -1.000000*A^1
w011: 1.000000*B^1+-1.000000*A^1+2.000000*E^1
w020: 1.000000*A^1
w100: -1.000000*A^1
w101: 1.000000*B^1+-1.000000*A^1+2.000000*D^1
w110: 1.000000*A^1
```

```
w200: 1.000000*A^1
```

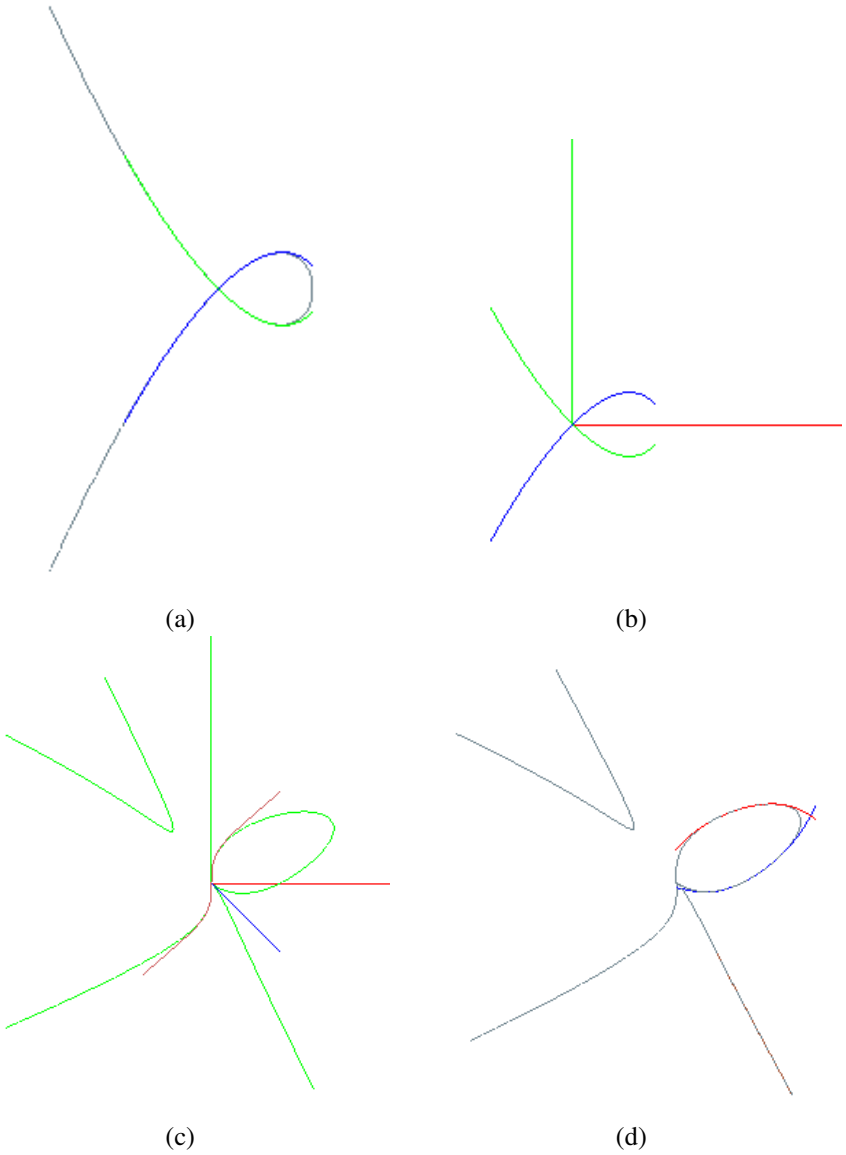


Figure A.5: Newton factorization: (a) rcf_newton(x^3 -x^2+ y^2,6); (b) rcf_gnewton(x^3 -x^2+ y^2,0,6); (c) rcf_newton(y^5 +2*x*y^4 -x*y^2 -2*x^2*y -x^3 +x^4,6); (d) rcf_gnewton(y^5 +2*x*y^4 -x*y^2 -2*x^2*y -x^3 +x^4,1,5). Image generated with GANITH [BR90a].

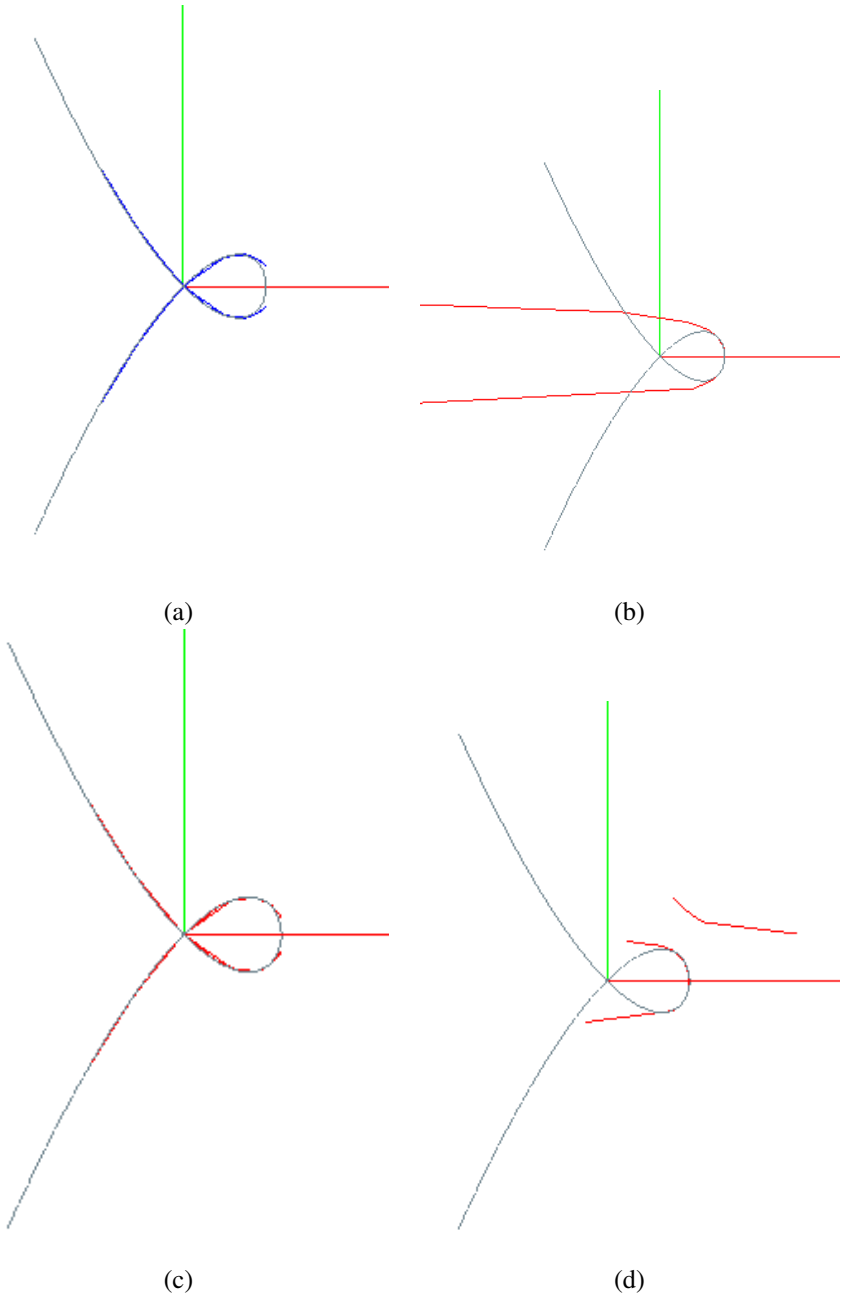


Figure A.6: Local polynomial and rational parameterization: (a) localpower2d (x^3 -x^2+ y^2,s,6,0,0); (b) localpower2d (x^3 -x^2+ y^2,s,6,1,0); (c) localpade2d (x^3-x^2+y^2,s,3,3,0,0); (d) localpade2d (x^3 -x^2+ y^2,s,3,3,1,0). Image generated with GANITH [BR90a].

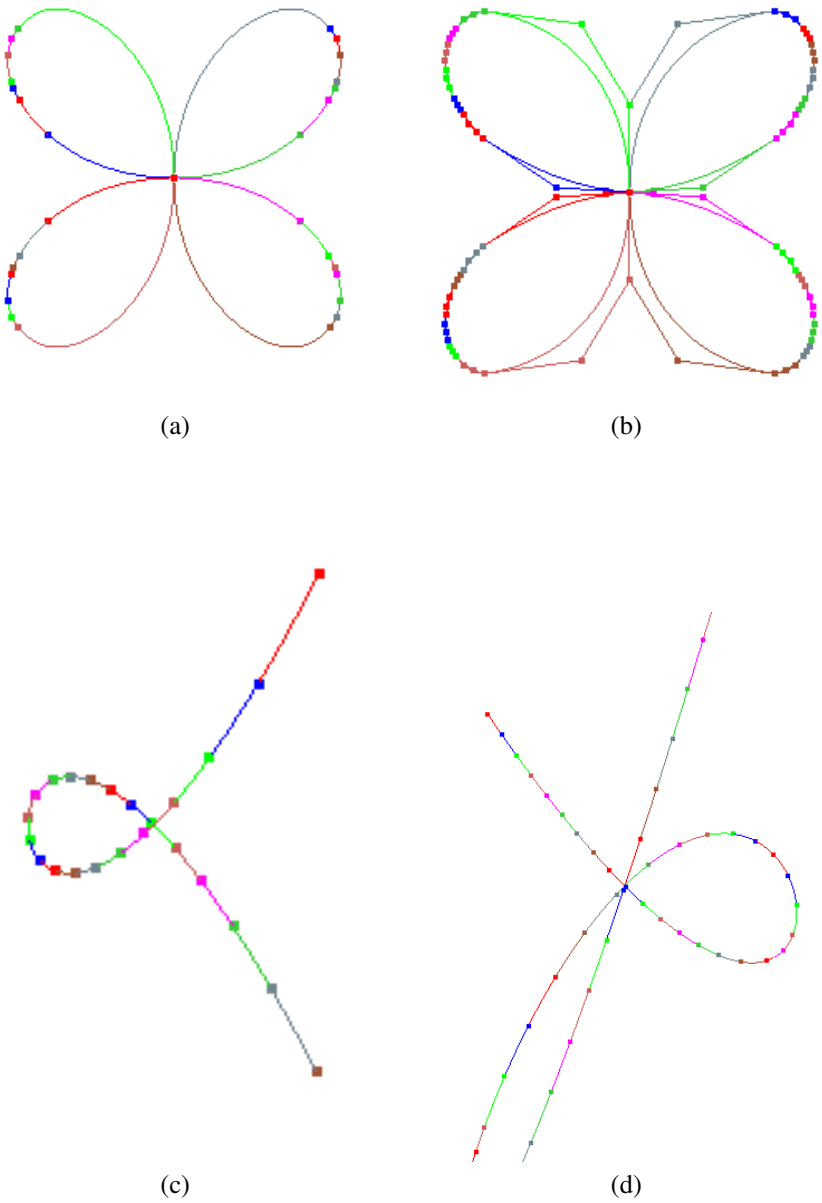


Figure A.7: Piecewise tracing: (a) piecerational2d ((x^2 +y^2)^3 - 4*x^2*y^2,s,3,3 ,-2,2,-2,2,0.05,1); (b) piecerationalp2d ((x^2 +y^2)^3 - 4*x^2*y^2, s,3,1, -2, 2, -2, 2, 0.1,0,2,1); (c) spacecurveiis (y^2-x^2-x^3+0.0*z^1,0.0*x^1+ 0.0*y^1+z^1; (d); spacecurvepc (x*(x-2)*(x+2),1,x*(x-2)*(x+2)*(1+x),1, x*(x-2)*(x+2)*(1-x),1,-8,8,-8,8,-8,8,7,0,3,0.01,3). Image generated with GANITH [BR90a].

Bibliography

[AB88] S. Abhyankar and C. Bajaj. Automatic parameterization of rational curves and surfaces III: Algebraic plane curves. *Computer Aided Geometric Design*, 1988.

[ABG+07] P. Assogna, G. Bertocchi, W. Gehrke, F. Milicchio, A. Paoluzzi, S. Portuesi, G. Scorzelli, M. Vicentino, , and R. Zollo. Vr platform for protection modelling of critical infrastructures. In *Virtual Reality International Conference*, Laval, France, 2007. VRIC 2007.

[Abh90] Shreeram Shankar Abhyankar. *Algebraic Geometry for Scientists and Engineers*. Amer Mathematical Society, 1990.

[ADS07] A. Paoluzzi A. DiCarlo, F. Milicchio and V. Shapiro. Solid and physical modeling with chain complexes. In *ACM Solid and Physical Modeling Symposium*, Beijing, China, 2007. ACM Press.

[ASS84] Hal Abelson, Jerry Sussman, and Julie Sussman. *Structure and Interpretation of Computer Programs*. MIT Press, 1984.

[Baj] Chandrajit Bajaj. Geometric modelling with a-patches : A tutorial.

[Baj90] C. Bajaj. Geometric modeling with algebraic surfaces. In D. Handscomb, editor, *The Mathematics of Surfaces III*, pages 3–48. Conference on the Mathematics of Surfaces, Oxford University Press, 1990.

[Baj92] *Implicit surface patches*. Morgan Kaufmann Publisher Inc., 1992.

[Baj93] C. Bajaj. *The Emergence of Algebraic Curves and Surfaces in Geometric Design*, chapter 1, pages 1–29. Information Geometers Press,, 1993. Directions in Geometric Computing.

[Baj07] Chandrajit Bajaj. Graduate course in geometric modeling and visualization, 2007. http://www.cs.utexas.edu/ bajaj/cs384R07/.

[BC00] D. Bashford and D. A. Case. Generalized born models of macromolecular solvation effects. *Arpc*, 51:129–152, 2000.

[BCCSX05] C. Bajaj, J. Castrillon-Candas, V. Siddavanahalli, and Z. Xu. Compressed representations of macromolecular structures and properties. *Structure*, 13:463–471, 2005.

[BCX94] C. Bajaj, J. Chen, and G. Xu. Free form surface design with A-patches. In *Proceedings of Graphics Interface '94*, pages 174–181, Banff, Alberta, Canada, 1994.

[BCX95] C. Bajaj, J. Chen, and G. Xu. Modeling with cubic a-patches. *ACM Trans. Graph.*, 14(2):103–133, 1995.

[BCX97] C. Bajaj, J. Chen, and G. Xu. Modeling with C^2 quintic a-patches. 1997. Fourth SIAM Conference on Geometric Design, Nashville, TN, (Nov. 1995).

[BDP08] C. Bajaj, A. DiCarlo, and A. Paoluzzi. Protoplasm: A parallel language for scalable modeling of biosystems. *Philosophical Transactions of the Royal Society A*, (366), 2008. To appear.

[BDST04] C. Bajaj, P. Djeu, V. Siddavanahalli, and A. Thane. Texmol: Interactive visual exploration of large flexible multi-component molecular complexes. *Proc. of the Annual IEEE Visualization Conference*, pages 243–250, 2004.

[BI92a] C. Bajaj and I. Ihm. Algebraic surface design with hermite interpolation. *ACM Transactions on Graphics*, 11(1):61–91, 1992.

[BI92b] C. Bajaj and I. Ihm. C^1 smoothing of polyhedra with implicit algebraic splines. *SIGGRAPH92 Computer Graphics*, 26(2):79–88, July 1992.

[BIW93] C. Bajaj, I. Ihm, and J. Warren. Higher order interpolation and least squares approximation using implicit algebraic surfaces. *ACM Transactions on Graphics*, 12(4):327–347, October 1993.

[BK88] C. Bajaj and M-S. Kim. Generation of configuration space obstacles: The case of a moving sphere. *IEEE Journal of Robotics and Automation*, 4(1):94–99, Feb 1988.

[BK90] C. Bajaj and M-S. Kim. Generation of configuration space obstacles: Moving algebraic surfaces. *The International Journal of Robotics Research*, 9(1):92–111, 1990.

[BKZ01] H. Biermann, D. Kristjansson, and D. Zorin. Approximate boolean operations on free-form solids. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 185–194, New York, NY, USA, 2001. Acm Press.

[BLMP97] C. Bajaj, H. Lee, R. Merkert, and V. Pascucci. Nurbs based b-rep models from macromolecules and their properties. *In Proceedings of Fourth Symposium on Solid Modeling and Applications*, pages 217–228, 1997.

[BNK93] V. Bhaskaran, B. Natarajan, and K. Konstandtinides. Optimal piecewise-linear compression of images. In Cohn M. and Storer J., editors, *Data Compression Conference*, pages 168–177. IEEE Computer Society Press, 1993.

[BPP07] C. Bajaj, A. Paoluzzi, and S. Portuesi. Efficient computation of the union of cubic a-patches. San Antonio, TX, USA, November 2007. 10th SIAM Conference in Geometric Design and Computing.

[BPP+08] Chandrajit Bajaj, Alberto Paoluzzi, Simone Portuesi, Na Lei, and Wenqi Zhao. Boolean operations with prism algebraic patches. *Computer-Aided Design and Applications*, 5(5), 2008.

[BR90a] C. Bajaj and A. Royappa. The ganith algebraic geometry toolkit. In *Lecture Notes in Computer Science*, volume 429. Springer-Verlag, 1990.

[BR90b] Chanderjit Bajaj and Andrew V. Royappa. The ganith algebraic geometry toolkit. In *Lecture Notes in Computer Science*, number 429. 1st Annual Conference on the Design and Implementation of Symbolic Computation Systems, Springer-Verlag, 1990.

[Bra78] I. C. Braid. On storing and changing shape information. In *SIGGRAPH '78: Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, pages 252–256, New York, NY, USA, 1978. Acm.

[BWW90] J. Backus, J.H. Williams, and E.L. Wimmers. An introduction to the programming language FL. In D.A. Turner, editor, *Research Topics In Functional Programming*, chapter 9, pages 219–247. Addison-Wesley, Reading, Massachusetts, 1990.

[BX97] C. Bajaj and G. Xu. Piecewise rational approximation of real algebraic curves. *Journal of Computational Mathematics*, 1997.

[BX99a] C. Bajaj and G. Xu. A-splines: local interpolation and approximation using gk-continous piecewise real algebraic curves. *Compuer Aided Geometric Design*, (16):557–578, 1999.

[BX99b] C. Bajaj and G. Xu. Smooth multiresolution reconstruction of free-form fat surfaces. Technical Report TICAM Report 99-08, University of Texas at Austin, 1999.

[BX01] C. Bajaj and G. Xu. Smooth shell construction with mixed prism fat surfaces. In *Geometric Modelling*, pages 19–35, London, UK, 2001. Springer-Verlag.

[BXHN02] C. Bajaj, G. Xu, R. Holt, and A. Netravali. Hierarchical multiresolution reconstruction of shell surfaces. *CAGD*, 19:89–112, 2002.

[BYA03] C. Bajaj, Z. Yu, and M. Auer. Volumetric feature extraction and visualization of tomographic molecular imaging. *Structural Biology*, 144(1-2):132–143, October 2003.

[BZ06] C. Bajaj and W. Zhao. Fast and accurate generalized born solvation energy computations. *Manuscript*, 2006.

[CS04] H. Cheng and X. Shi. Guaranteed quality triangulation of molecular skin surfaces. *IEEE Visualization*, pages 481–488, 2004.

[Dah89] W. Dahmen. Smooth piecewise quadric surfaces. In *Mathematical Methods in Computer Aided Geometric Design*, pages 181–193. Academic Press Professional, Inc., San Diego, CA, USA, 1989.

[DGL87] N. Dyn, J. Gregory, and D. Levin. A 4-point interpolatory subdivision scheme for curve design. *Computer Aided Geometric Design*, (4):257–268, 1987.

[EW79] C. M. Eastman and K. J. Weiler. *Geometric Modeling Using the Euler Operators*. Institute of Physical Planning, Carnegie-Mellon University, 1979.

[Far01] Gerald Farin. *Curves and Surfaces for CAGD: A Practical Guide*. Morgan Kaufmann, 2001.

[GG99] R. Garrett and C. Grisham. *Biochemistry*. Saunders Collge Publishing, New York, 2nd edition, 1999.

[GMF06] A. L. Garcia, J. R. De Miras, and F. R. Feito. Adaptive trimming of cubic triangular Bézier patches. In *SIACG '06: Ibero-American Symposium in Computer Graphics*, Santiago De Compostela, Spain, 2006.

[Gol02] Ron Goldman. *Pyramid Algorithms: A Dynamic Programming Approach to Curves and Surfaces for Geometric Modeling*. Morgan Kaufmann, 2002.

[GSS+04] A. Gillet, M. Sanner, D. Stoffler, D. Goodsell, and A. Olson. Augmented reality with tangible auto-fabricated models for molecular biology applications. *Proc. of IEEE Conference on VIsualization*, (235-242), 2004.

[Guo92] B. Guo. *Modeling Arbitrary Smooth Objects with Algebraic Surfaces*. PhD thesis, Computer Science Department, Cornell University, Ithaca, NY, USA, 1992.

[GW91] T. Garrity and J. Warren. Geometric continuity. *Computer Aided Geometric Design*, 1991.

[HO94] D. Halperin and M. H. Overmars. Spheres, molecules, and hidden surface removal. *ACM SCG '94: Proceedings of the tenth annual symposium on Computational geometry*, pages 113–122, 1994. (New York, NY, USA).

[Hof89] C. M. Hoffmann. *Geometric and Solid Modeling: An Introduction*. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1989.

[IBR98] W. Im, D. Beglov, and B. Roux. Continuum solvation model: computation of electrostatic forces from numerical solutions to the poisson-boltzmann equation. *Cpc*, 111:59–75, 1998.

[JWW90] J. Backus J., H. Williams, and E.L. Wimmers. An introduction of the programming language FL. In *Research Topics in Functional Programming*. Addison Wesley, 1990.

[KCF+02] J. Keyser, T. Culver, M. Foskey, S. Krishnan, and D. Manocha. Esolid—a system for exact boundary evaluation. In *SMA '02: Proceedings of the Seventh ACM Symposium on Solid Modeling and Applications*, pages 23–34, New York, NY, USA, 2002. Acm.

[KGMM97] S. Krishnan, M. Gopi, D. Manocha, and M. R. Mine. Interactive boundary computation of boolean combinations of sculptured solids. *Comput. Graph. Forum*, 16(3):67–78, 1997.

[KOB+04] M. Van Kreveld, R. Van Oostrum, C. Bajaj, V. Pascucci, and D. Schikore. *Contour Trees and Small Seed Sets for Isosurface Generation Topological Data Structures for Surfaces*, chapter 5, pages 71–86. John Wiley and Sons, 2004.

[KWT88] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: active contour models. *Internat. J. Comput. Vision*, pages 321–331, 1988.

[LB02] P. Laug and H. Borouchaki. Molecular surface modeling and meshing. *Engineering With Computers*, 18:199–210, 2002.

[Lin00] L. Linsen. Netbased modelling. In B. Falcidieno, editor, *Proceedings of Spring Conference on Computer Graphics (SCCG)*, pages 259–266, Comenius University, Bratislava, Slowakei, 2000.

[LTH86] D.H. Laidlaw, W. B. Trumbore, and J. F. Hughes. Constructive solid geometry for polyhedral objects. In *SIGGRAPH '86: Proceedings of the 13th International Conference on Computer Graphics and Interactive Techniques*, pages 161–170, New York, NY, USA, 1986. Acm.

[Man88] M. Mantyla. *Introduction to Solid Modeling*. WH Freeman & Co. New York, NY, USA, 1988.

[Mas93] H. Masuda. Topological Operators and Boolean Operations for Complex-based Nonmanifold Geometric Models. *Computer Aided Design*, 25(2):119–29, 1993.

[Nay90] B. F. Naylor. Binary space partitioning trees as an alternative representation of polytopes. *Computer Aided Design*, 22(4):250–252, 1990.

[Nie79] G. Nielson. The side-vertex method for interpolation in triangles. *J. Approx. Theory*, 25:318–336, 1979.

[Pa] Alberto Paoluzzi and al. Plasm web page. www.plasm.net.

[Pao03] A. Paoluzzi. *Geometric Programming for Computer Aided Design*. Wiley, 2003.

[PBCF93] A. Paoluzzi, F. Bernardini, C. Cattani, and V. Ferrucci. Dimension-independent modeling with simplicial complexes. *ACM Transactions on Graphics*, 12(1):56–102, 1993.

[PK89] N.M. Patrikalakis and G. A. Kriezis. Representation of piecewise continuous algebraic surfaces in terms of B-splines. *The Visual Computer*, (5):360–374, 1989.

[PLT] PLT scheme homepage. www.plt-scheme.org.

[Por07a] Simone Portuesi. Doctorate school in complex industrial systems (SICS) student report, second year, March 2007.

[Por07b] Simone Portuesi. Ganith tutorial. Course in Geometric Modeling and Visualization, University of Texas at Austin, Autumn 2007.

[PP92] M. Paluszny and R.R. Patterson. Curvature continuous cubic algebraic splines. *Proc. of SPIE, Curves & Surfaces in Computer Vision and Graphics III*, 1830:48–54, 1992.

[PP93] M. Paluszny and R.R. Patterson. A family of tangent continuous algebraic splines. *Transaction on Graphics*, 12(3):209–232, 1993.

[PPV95] A. Paoluzzi, V. Pascucci, , and M. Vicentino. Geometric programming: A programming approach to geometric design. *ACM Transactions on Graphics*, 14(3):266–306, July 1995.

[PSR89] A. Paoluzzi, A. Santarelli, and M. Ramella. Boolean algebra over linear polyhedra. *Computer-Aided Design*, 21(8):474–484, 1989.

[PT96] Les A. Piegl and Wayne Tiller. *The NURBS Book*. Springer, 1996.

[Req77] A. Requicha. Mathematical models of rigid solid objects. Technical Memo 28, Production Automation Project, University of Rochester, Rochester, NY, November 1977.

[Req80] A. A. G. Requicha. Representations for rigid solids: Theory, methods, and systems. *Computing Surveys*, 12(4):437–464, 1980.

[SAG85] T. Sederberg, D. Anderson, and R. Goldman. Implicitization, inversion, and intersection of planar rational cubic curves. *Computer Vision, Graphics and Image Processing*, (31):89–102, 1985.

[Set96] J. A. Sethian. *Level Set Methods.* Cambridge University Press, 1996.

[SPP07] G. Scorzelli, A. Paoluzzi, and V. Pascucci. Parallel solid modeling using bsp dataow. *Journal of Computational Geometry and Appli- cations*, 2007. To appear.

[SR49] J. Semple and L. Roth. *Introduction to Algebraic Geometry*. Oxford University Press,, 1949.

[STHH90] W. C. Still, A. Tempczyk, R. C. Hawley, and T. Hendrickson. Semianalytical treatment of solvation for molecular mechanics and dynamics. *Jacs*, 112:6127–6129, 1990.

[SZZ88] T. Sederberg, J. Zhao, and A. Zundel. Approximate parameterization of algebraic curve. *Theory and Practice of Geometric Modeling*, pages 33–54, 1988.

[Var98] Various. Revised ^{5} report on the algorithmic language scheme. In J. Rees R. Kelsey, W. Clinger, editor, *Higher-Order and Symbolic Computation*, volume 1. 1998. Available at http://www.schemers.org/Documents/Standards/R5RS/.

[Wal78] R. Walker. *Algebraic Curves.* Springer Verlag New York, 1978.

[War95] J. Warren. Binary subdivision schemes for functions over irregular knot sequences. In Dahlen M., Lyche T., and Schumaker L.L., editors, *Mathematical Methods for Curves and Surfaces*, pages 543–562. Vanderbilt University Press Nashville, 1995.

[XB95] Guoliang Xu and Chandrajit Bajaj. Adaptive model reconstruction by triangular a-patches, 1995.

[XBC00a] Guoliang Xu, Chandrajit L. Bajaj, and Chuan I Chu. Regular algebraic curve segments (I)-Denitions and characteristics. *Computer Aided Geometric Design*, (17):485–501, 2000.

[XBC00b] Guoliang Xu, Chandrajit L. Bajaj, and Chuan I Chu. Regular algebraic curve segments (II)-Interpolation and approximation. *Computer Aided Geometric Design*, (17):503–519, 2000.

[XBC00c] Guoliang Xu, Chandrajit L. Bajaj, and Chuan I Chu. Regular algebraic curve segments (III)-Applications in interactive design and data fitting. *Computer Aided Geometric Design*, (17):503–519, 2000.

[XBE02] Guoliang Xu, Chandrajit L. Bajaj, and Susan Evans. C^1 modeling with hybrid multiple-sided a-patches. *Int. J. Found. Comput. Sci.*, 13(2):261–284, 2002.

[XHB01] G. Xu, H. Huang, and C. Bajaj. C modeling with A-patches from rational trivariate functions. *Comput. Aided Geom. Des.*, 18(3):221–243, 2001.

[ZnBS05] Y. Zhang and C. Bajaj nd BS. Sohn. 3D finite element meshing from imaging data. *Comput. Methods Appl. Mech. Engrg.*, (197), 2005.

[ZXB06] Y. Zhang, G. Xu, and C. Bajaj. Quality meshing of implicit solvation models of biomolecular structures. *Cagd*, 23:510–530, 2006.

[ZXB07] W. Zhao, G. Xu, and C. Bajaj. An algebraic spline model of molecular surfaces modeling. In *ACM Solid and Physical Modeling Symposium*, Beijing, China, 2007. ACM Press. ACM SPM 2007.

Bio-medical Symbolic Modeling with Algebraic Patches

Simone Portuesi

The mathematical description of geometry is of paramount importance in the modeling of systems of all kinds. Standard geometric modeling describes curved objects through parametric functions as the image of compact domains. Alternatively, as in algebraic geometry, one may describe curved geometry as the zero-set of polynomials. Geometric modeling of biological systems highlight certain persistent and open problems more effectively addressed using algebraic geometry.

In this thesis a framework for computer-based geometric representation based on algebraic geometry is introduced. Several algebraic representation schemes known as A-splines and A-patches are detailed as a description of geometry, by using a piecewise continuous gluing of algebraic curves and surfaces. The application of A-splines and A-patches to biological modeling is discussed in the context of protein molecular interface modeling. The rationale is to present an algebraic representation of bio-modeling under an unified point of view. This framework provides a suitable background for the main contribution of this thesis: the formulation and implementation of algorithms for Boolean operations (union, intersection, difference, etc.) on the algebra of curved polyhedra whose boundary is triangulated with A-patches. Boolean operations on curved geometry are yet an open obstinate research problem and its exact solution is only definable within the domain of algebraic geometry. The exact formulation is here used as basis for a geometrically approximate yet topologically accurate solution, closed in the geometric domain of A-patches. The prototype implementation has been applied to pairs of molecular models of ligand proteins in docking configuration. To date, the computational use of algebraic geometry is still experimental and is far from being a major component of current systems. This thesis shows an evidence that representation techniques derived from algebraic geometry have strong potential in bio-medical modeling, still needing much further research and engineering.

ROMA TRE
UNIVERSITÀ DEGLI STUDI

Tesi Dottorato SICS