INTERACTIVE DEFORMABLE MODELING WITH A-PATCHES

A Thesis

Submitted to the Faculty

of

Purdue University

by

Jindong Chen

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 1996

This thesis is dedicated to

my father Sijun, my mother Zhejun, and my wife Ching.

ACKNOWLEDGMENTS

I would like to thank my advisor, Professor Chandrajit Bajaj, for many things culminating in this thesis: for giving me great latitude in this exploratory process; for his constant support, encouragement and guidance; for the many fruitful discussions we have had; and for his invaluable help both within and outside the world of academics.

To my many teachers through the years, especially at The Second Primary School of Haoxian Road, The Primary School of Yuehua Road, The Affiliated Middle School to Southern China Normal University, The University of Science and Technology of China (USTC), The University of Kentucky and Purdue University, I owe a heartfelt debt of gratitude. They led me along the path of knowledge and taught me to learn and grow.

For all that I am and all that I have, I thank my parents. My father, Sijun Chen, and my mother, Zhejun Chen, inculcated in me a thirst for knowledge, stressed the importance of a well rounded education, and encouraged me to strive to excel – to be all I could be. To my wife, my life, Ching – thanks for your spiritual support, thanks for your patience and perseverance and for all your help. Without your love, I would not be able to go through the difficulties in my Ph.D study life as easy, and as enjoyable.

I have learned a tremendous amount of mathematics from Dr. Guoliang Xu, who to me is a teacher rather than a collaborator. Many thanks to all my collaborators and officemates, Dr. Insung Ihm, Dr. Andrew Royappa, Dr. Vinod Anupam, Steven Cutchin, Daniel Schikore, Dr. Xuan Zhang, Susan Evans, Fausto Bernardini, Peinan Zhang, Kwun-nan Lin, Youming Lin, Weiping Zhuang, Guozhong Zhuang, Ping Ning, Hae-Young Lee, Valerio Pascucci, Richard Kennell, and Raimund Merkert. We have had many stimulating discussions on a multitude of topics, often not academic. Many thanks to my classmates, Yue Huang, Aifeng hsu, Dr. Sandeep Kumar, Dr. Tsanchi Li, Dr. Xiangning Liu, Tian Lu, Dr. Lu Xu, Gustavo Rodriguez-Rivera, Dr. Poting Wu, Changxun Zheng, Dr. Aidong Zhang, and Gang Zhu, for the good time we shared in and out of classrooms, exams, and Quofat. Too numerous to thank are the many other friends I made during my stay at Purdue. They made sure there was a life beyond academic pursuits. Many thanks to all my teammates in the Wind, the soccer team in which we have had good time together all these years, on and off the field.

Professor Jörg Peters of my graduate committee offered numerous constructive criticisms during our conversations. Professor John Rice and Professor Benjamin Hillberry, contributed valuable advice.

I gratefully acknowledge the intellectual and material support provided by the fine environs and the great staff of the Computer Sciences Department of Purdue University. Dr. William Gorman and Mary Jo Maslin managed an excellent administration of the graduate program. Richard Bingle, Douglas Crabill, Adam Hammer and Daniel Trinkle were always available to help get troublesome demons out of the system. Georgia Conarroe, Patricia Minniear and Daloris Williamson provided much infra-structural support.

DISCARD THIS PAGE

TABLE OF CONTENTS

Page

Page

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

Appendix
Figure

ABSTRACT

Chen, Jindong. PhD, Purdue University, May 1996. Interactive Deformable Modeling with A-patches. Major Professor: Chandrajit L. Bajaj.

The main impediments to the widespread use of implicit surfaces for geometric modeling are multiple sheets, self-intersections and several other undesirable singularities. Our A-patch technique provides simple ways to guarantee that the constructed implicit surface is single-sheeted and free of undesirable singularities. The technique uses the zero contouring surfaces of trivariate Bernstein-Bézier polynomials to construct a piecewise smooth surface, cubic for C^1 and quintic for C^2. We call such iso-splines *A-patches*, where "A" stands for algebraic. We have designed different algorithms to construct smooth A-patch surfaces that interpolate or approximate scattered 3D point data or simple polyhedra of arbitrary topology.

An A-patch is easy to modify by manipulating the coefficients. The constraints imposed on an A-patch also make it easy to be polygonized quickly. We discusses schemes to edit A-patch models interactively as well as schemes to polygonize an A-patch surface quickly into a triangulated mesh.

The A-patch technique, combined with the physically-based modeling technique, is also suitable for free-form surface design. Most of the existing surface representations fail to support interactive design efficiently and intuitively. Among a few exceptions is the physically-based modeling technique, in which a geometric shape is modeled as an elastic object so that a user can modify it by applying virtual forces. We have developed an elastic model built on top of the A-patch representation. The optimization of the energy of an A-patch is no more complicated than a quadratic

programming problem, which dramatically reduces the computational overhead in general physically based methods.

1. INTRODUCTION

People have believed in piecewise approaches for a long time, probably from the time that they learned how to sew. 集腋成裘 is a Chinese proverb that advises people to take piecewise approaches (literally, a fur coat is made of many pieces of fox fur). Scientists and engineers brought the idea into modern sciences and engineering. In different contexts, we call it "splines" or "finite elements".

Following the steps of the other researchers, this thesis considers the piecewise approach to model free-form surfaces with implicit polynomial surfaces in Bernstein-Bézier form.

The important issues in the piecewise free-form modeling of shapes with arbitrary topology are:

1. the patch representation,
2. the polynomial degree of the patches,
3. the number of patches per face of the input or benchmarking polyhedron,
4. the functional connectivity and nonsingularity of the patches,
5. the conditions for the desired continuity between adjacent patches (how the patches are "stitched" together to form a "smooth" surface),
6. the curvature and higher derivative variation of the patches, especially around the "stitches".

With these issues in mind, we survey piecewise techniques in geometric modeling and visualization, and then outline the rest of the thesis.

1.1 Parametric Surfaces

All polynomial representations can be written implicitly. Parametric representations are just a "popular" subset of implicit polynomial representations.

Different forms of parametric representations have been widely researched and used for a long time. In general, a parametric surface is a vector of (rational) polynomials defined in a parametric domain. Parametric representations have good properties such as ease of evaluation, ease of control, and irreducibility.

B-splines have been widely used as the polynomial basis of choice for parametric polynomial parametric curves and surfaces. However, it is impossible to model a general closed surface or a surface with a handle (higher genus) as a single non-degenerate B-spline. To represent free-form surfaces, a significant amount of recent research have been done in the areas of geometric continuity, non-tensor product patches and in generalizing B-splines. Common schemes include splitting, convex combinations of blending functions, subdivision, and local interpolation by construction.

The rational extension of B-splines, non-uniform rational B-splines (NURBS) [106] has emerged as the *de facto* unified standard in the CAD/CAM industry because it enlarges the variety of shapes to include conics and quadrics, which are commonly used in CAD/CAM systems. However, conics and quadrics are represented by NURBS at the cost of losing some nice properties such as compact storage and easy operation of point location classification. The representation also makes some of the formulations, such as derivatives, complicated, while they are simpler in non-rational representations and implicit representations (conics and quadrics).

1.1.1 Tensor Product Surfaces

Based on the concept of bilinear interpolation, tensor product patches are defined over a rectangular domain. Tensor product Bézier surfaces are defined over a rectangular domain. While linear interpolation fits the "simplest" curve between two points, bilinear interpolation fits the "simplest" surface between four points. The tensor product Bézier surfaces are obtained by repeated applications of bilinear interpolation. Properties of tensor product Bézier patches include affine invariance, convex hull property, and the variation diminishing property. The boundary curves of a patch are Bézier curves whose control polygon is the boundary polygons of the control net of the patch. Hence the patch interpolates the four corners of the control net.

Piecewise bicubic Bézier patches may be used to fit a C^1 surface through a rectangular grid of points. After the rectangular network of curves has been created there are four coefficients left to determine the corner twists (or the mixed partial derivatives $\frac{\partial^2}{\partial u \partial v}$ at the corners) of each patch. These four corner twists can not be specified independently and must satisfy a "compatibility constraint". Common twist estimation methods include zero twists, Adini's twist, Bessel twist, and Brunet's twist. These methods are largely discussed in [28, 29, 30, 42, 55, 60, 74]. To obtain C^1 continuity between two patches, the directions and lengths of the adjacent control edges must be matched across the common control boundary defining the common boundary curve. When the data points deviate from a nice rectangular grid structure, however, there is a problem finding a suitable parameterization and piecewise bicubic interpolation will not work. This problem is also referred to as "vertex inclusion" problem. There are alternatives such as using surfaces of a higher degree, or replacing C^1 requirements with G^1 ones.

Piecewise bicubic Hermite patches are similar to the piecewise bicubic Bézier patches but take points, partials, and mixed partials as input. The mixed partials have effect only on the interior shape of the patch and are also called twist vectors. To obtain C^1 continuity between two patches it is necessary and sufficient to match the points, partials, and mixed partials at the matching corners of the two patches [111].

A B-spline surface is a deformation of a planar domain, tessellated into a regular grid of rectangles. Any B–spline surface can be written in piecewise Bézier form. It is natural for the surface to be treated as a collection of tensor product polynomial patches defined over these rectangles. This leads to notions of parametric continuity (denoted C^k continuity) where smoothness is defined in terms of matching derivatives along patch boundaries.

A bicubic B-spline surface may be used to fit a C^2 surface through a rectangular grid of points. For the bicubic Bézier and Hermite patches an initial global survey of the data is needed to determine appropriate values for the tangent and cross-derivative vectors at the patch corners. The bicubic B–spline surface obtains C^2 continuity by overlapping the control polyhedra of neighboring patches.

Independent of B-spline basis and control points, Coons patches and Gordon surfaces [54] appear more intuitively by generating a surface that interpolates a network of curves, referred to as transfinite interpolation. Coons patches are based on a generalization of ruled, or lofted, surfaces, which fit a surface through two given curves by linear interpolation. Coons patches are generalized to interpolate four boundary curves. They are constructed by composing two lofted surfaces and one bilinear surface, and hence are also called bilinearly blended surfaces. Specifically, a Coons patch can be given in the following general form

$$\boldsymbol{x}(u,v) = \begin{bmatrix} f_1(u) & f_2(v) \end{bmatrix} \begin{bmatrix} \boldsymbol{x}(0,v) \\ \boldsymbol{x}(1,v) \end{bmatrix} \tag{1.1.1}$$

$$+ \begin{bmatrix} \boldsymbol{x}(u,0) & \boldsymbol{x}(u,1) \end{bmatrix} \begin{bmatrix} g_1(v) \\ g_2(v) \end{bmatrix} \tag{1.1.2}$$

$$- \begin{bmatrix} f_1(u) & f_2(u) \end{bmatrix} \begin{bmatrix} \boldsymbol{x}(0,0) & \boldsymbol{x}(0,1) \\ \boldsymbol{x}(1,0) & \boldsymbol{x}(1,1) \end{bmatrix} \begin{bmatrix} g_1(v) \\ g_2(v) \end{bmatrix} \tag{1.1.3}$$

where $x(0,v)$, $x(1,v)$, $x(u,0)$ and $x(u,1)$ are the four boundary curves and f_i, g_i are the blending functions. There are only two restrictions on the blending functions f_i and g_i: each pair must sum to one, and $f_1(0) = g_1(0) = 1$, $f_2(1) = g_2(1) = 0$ in order to interpolate. The first two terms are lofted surfaces while the last term is the tensor product interpolant to the four corners.

A network of curves may be filled in with a C^1 surface using bicubically blended Coons patches. For this the four twists at the data points and the four cross boundary derivatives must be computed. Compatibility problems may arise in computing the twists. If $\mathbf{x}(u, v)$ is twice differentiable, we have $\mathbf{x}_{uv} = \mathbf{x}_{vu}$, but this simplification does not apply here. One approach is to adjust the given data so that the incompatibilities disappear. Or if the data can not be changed, one can use a method known as Gregory's square that replaces the constant twist terms by variable twists that are computed from the cross boundary derivatives. The resulting surface does not have continuous twists at the corners and is rational parametric, which may not be acceptable in certain environments.

The interpolated boundary curves do not have to be vectors of (rational) polynomials. But if they do then Coons patches can be constructed as a tensor B-splines or Bezier patch (or NURBS, in general).

Gordon surfaces are a generalization of Coons patches used to construct a surface $\boldsymbol{g}$ that interpolates a rectangular network of curves, which will be iso-parametric curves

$$\boldsymbol{g}(u_i, v), i = 0, ..., m \tag{1.1.4}$$

and

$$\boldsymbol{g}(u, v_i), i = 0, ..., m \tag{1.1.5}$$

Roughly speaking, a Gordon surface is constructed by taking the sum of the univariate interpolants of the iso-parametric curves, in two directions, and subtracting the tensor product interpolant of the grid points $(g(u_i, v_j))$. Polynomial interpolation or spline interpolation schemes may be used.

Methods for Coons patches and Gordon surfaces can be formulated in terms of boolean sums and projectors, which leads to the triangular generalization of Coons patches.

1.1.2 Triangular Surfaces

Triangular pieces are a fairly standard tool of approximation theory. For complicated surfaces, no rectangular geometry is apparent and the use of triangular patches is advantageous since every surface can be covered with a triangular network. Barnhill [27] was one of the first to apply these triangular methods to vector–valued surface patches. Bézier curves admit a symmetric formulation in terms of affine invariant barycentric coordinates. This concept is generalized to surfaces to yield triangular patches. (See Section 2.1.)

Much of the history of triangular Bernstein-Bézier patches is summarized by [52, 54, 74]. The de Casteljau algorithm for triangular Bézier patches is a generalization of the corresponding algorithm for curves and uses repeated linear interpolation. The control net is a triangular structure, and a degree n patch has a control net of $\frac{1}{2}(n+1)(n-1)$ vertices. These triangular patches are affine invariant under affine parameter transformations, have the convex hull property, and have boundary curves that are Bézier polynomials determined by the boundary control points. Algorithms exists for degree elevation, subdivision, and differentiation.

Farin [52] presents several schemes to interpolate position and derivative information at the vertices and across edges. C^0 nine parameter interpolants give quadratic precision and interpolate position and gradient at each triangle vertex. C^1 interpolants that are at least quintic need input that is up to second derivative data at the vertices and a cross–boundary derivative at each edge midpoint. In general a local piecewise polynomial interpolant that interpolates to all derivatives up to order r of some primitive function f at the vertices of a triangulation and is globally r times differentiable must be of degree $n \geq 4r+1$. For these Hermite interpolants you have to provide derivative data of higher order than the desired order of continuity.

Schmidt [119], and Böhm, Farin, and Hahmann [36] have considered global interpolants to minimize a certain functional (such as strain energy) over the whole triangulated domain. This leads to the solution of a large linear system.

1.1.2.1 Generalized Triangular Surfaces Using Splitting

To prevent continuity constraints from propagating globally, new vertices are introduced to split the original triangles. The new vertices help achieving locality because they are degree-bounded so that the continuity constraints around them can be satisfied by simple averaging.

With splitting schemes every triangle in the triangulation of the data points (also called a "macro–triangle") is split in to several "mini–triangles". Split-triangle interpolants do not require derivative information of higher order than the continuity of the desired interpolant. The simplest of the split-triangle interpolants in the C^1 and C^2 Clough-Tocher interpolant [1, 44]. Each vertex is joined to the centroid, and the macro-triangle is split into three mini-triangles. The first order data that this interpolant requires are position and gradient value at the vertices of the macro–triangle plus some cross–boundary derivative at the midpoint of each edge. There are twelve data per macro-triangle and cubic polynomials are used over each mini–triangle. Alfeld [2] extends the scheme to trivariate case with tetrahedral data. Worsey and Farin [142] generalize to n-dimensional case. The C^1 Powell-Sabin interpolants produce C^1 piecewise quadratic interpolants to C^1 data at the vertices of a triangulated data set. Each macro–triangle is split into several mini triangles. If all angles of the macro–triangle are smaller than 75 degrees, a split into six mini-triangles is performed. Otherwise twelve mini-triangles are generated. Piper [107] and Shirman and Séquin [127] use splitting schemes to locally interpolate mesh vertices with collections of triangular Bézier patches that meet smoothly.

1.1.3 Multi-sided Patches

Multi-sided patch techniques are also developed to compensate the difficulty of fitting tensor-product patches into irregular meshes. Bajaj and Evans [20] give an excellent survey on multi-side patch techniques as well as much of the material covered here.

1.1.4 Spline Surfaces Over Meshes With Arbitrary Topology

The representation of free–form surfaces is one of the major issues in geometric modeling. The problem of constructing a surface from an irregular mesh in space has been considered by many. These surfaces are generally defined in a piecewise manner by smoothing joining several mostly four–sided patches. The patches are given in vector valued parametric form, mapping a rectangular parametric domain into $\mathbb{R}^3$.

Local construction, blending polynomial pieces, and splitting are common approaches to constructing surfaces over irregular meshes. Because of its nonlinear nature and the advantages of a local construction, different approaches have been centered around selecting geometrically meaningful variables that can be fixed (as input or derived from data) so as to arrive at a sufficient and consistent set of linear constraints on the remaining variables. Blending polynomial pieces means constructing k pieces for a k-sided mesh facet such that each piece matches a part of the facet data and a convex combination of the pieces matches the whole. Blending approaches prescribe a mesh of boundary curves and their normal derivatives. However for this approach the existence of a well–defined tangent plane at the data points is not sufficient to guarantee the existence of a C^1 mesh interpolant since the mixed derivatives p_{uv} and p_{vu} are given independently at any point p. Splitting approaches on the other hand expect at least tangent vectors at the data points and sometimes the complete boundary to be given.

Chiyokura and Kimura [43] construct a solid with free–form surfaces. They start with a polyhedral solid which can be edited using local modifications and generate a curve model for the solid consisting of cubic Bézier curves and straight lines. Gregory Patches are used to interpolate the multi–sided faces of the curved model. One step of subdivision is applied to make all faces have four edges. The Gregory Patches are generalizations of bicubic Bézier patches that contain removable singularities at patch corners. The normals along the four boundary curves can be specified with no compatibility constraint, and are computed as a quadratic function. A point on the patch is represented as an addition of sixteen points to which various weights are given.

A class of popular approaches of modeling meshes with arbitrary topology are in conjunction with subdivision methods (Section 1.3). The input mesh is averaged and split before surface patches are fit in. The subdivision steps are sophisticatedly designed so that the surface patches are free of some headaches such as *compatibility* problem and/or "vertex enclosure" problem. The idea is somewhat similar to that of Clough-Tocher and Powell-Sabin interpolants: introducing new vertices that are degree-bounded.

Van Wijk [134] gives a generalization of B–splines that uses subdivision and Coons patches, but this scheme uses bicubic tensor product patches exclusively and imposes relatively strict requirements on the form of the control mesh. The faces of the mesh must have four edges, and at all vertices either three or four edges are allowed to meet or at all vertices an odd number of edges must meet.

Peters [100] describes an algorithm for the interpolation of a mesh of points in space by a piecewise parametric Bézier C^1 surface. At each point the normal can be specified. The surface is constructed locally and consists of cubic and bicubic patches to match the underlying mesh facets. Instead of restricting the boundaries themselves, it constrains the normal direction along the patch boundaries such that the normal along patch boundaries varies linearly. The method encounters a version of the 'compatibility problem' since it uses only one patch per facet, and one additional constraint is added for each point that has an even number of neighbors.

Peters [105] gives an algorithm for the local interpolation of a mesh of cubic curves with 3- and 4-sided facets by a piecewise cubic C^1 surface. Conditions for admissible data for cubic mesh interpolation with splitting are given and a scheme for generating admissible curve meshes from data points and normals is presented. The construction of the surface is local and consists of splitting and averaging. The algorithm guarantees interpolating surfaces without cusps and has a simple extension to n-sided facets with a piecewise quartic surface.

Peters [101] considers the interpolation of a mesh of curves by a smooth regularly parameterized surface with one polynomial piece per facet. Not every mesh with a well–defined tangent plane at the mesh points has such an interpolant. Necessary and sufficient *vertex enclosure* constraints on a mesh of polynomial curves that guarantees the existence of a regular smooth interpolant are given. The curvature of mesh curves emanating from mesh points with an even number of neighbors must satisfy an additional "vertex enclosure constraint". The vertex enclosure constraint is automatically satisfied by the splitting construction and can be satisfied by singularly parameterizing one of the boundary curves. An algorithm for the local interpolation of a cubic curve mesh by a piecewise [bi]quartic C^1 surface is described. The scheme is based on a sufficient constraint that forces the mesh curves to interpolate second–order data at the mesh points. Rational patches, singular parameterizations, and the splitting of patches are interpreted as techniques to enforce the vertex enclosure constraint. Farin, Piper, Shirman, Séquin, and Peters [52, 101, 107, 127] locally interpolate mesh vertices with collections of triangular or rectangular Bézier patches that meet smoothly. In Mann et al. [87] these results are surveyed and it is concluded that local polynomial interpolants generally produce unsatisfactory shapes.

Reif [116] constructs a piecewise G^1 spline surface based on biquadratic rectangular Bézier patches from a set of control points on meshes with arbitrary topology. Geometrical smoothness conditions are used only near the singular vertices of a mesh. He constructs an additional ring of "G–edges" around singular vertices and expresses the smoothness conditions in a system of linear equations.

Peters [102] gives an algorithm for refining an irregular mesh of points into a bivariate C^1 surface. The algorithm generalizes the construction of quadratic splines from a mesh of control points, and an explicit parameterization of the surface with quadratic and cubic pieces is given. When the mesh is regular then a quadratic spline surface is generated. Irregular input meshes with non-quadrilateral mesh cells more or fewer than four cells meeting at a point are allowed and generate spline spaces that generalize the space of quadratic splines. The main idea is to refine the irregular input mesh by the averaging process of Doo–Sabin and generate strips of regular mesh points that isolate regions of irregular points. After the mesh has been refined each mesh point is surrounded by four cells. Control points of symmetric piecewise quadratic box splines over the four-direction mesh are generated and the holes are filled with cubic triangular patches. The cubics are determined by averaging the box spline control points. An optional conversion of the box splines into Bernstein–Bézier form is an evaluation step that creates a large number of patches whose control mesh is closer to the surface than the box spline control mesh. The algorithm can model bivariate open or closed surfaces of general topological structure. However the algorithm generates a large number of patches relative to the number of faces in the control mesh.

Loop [84] constructs a piecewise G^1 spline surface composed of sextic triangular Bézier patches in one-to-one correspondence with the faces of a triangular control mesh. Surfaces of arbitrary topological type are created by approximating any mesh that represents a triangulated 2–manifold. The surface has local support and is affine invariant since the Bézier points of the patches are affine combinations of the vertices of the mesh. A pair of shape parameters is available at each vertex of the mesh for additional local control over the shape of the surface. If some local regularities are present in the structure of the control mesh then the corresponding patches may be parameterized as quintic or even quartic Bézier triangles. In the case of a regular parameterization the surface generated by this method is equivalent to a quartic C^2 triangular B–spline. The question of whether there exists values in the space of the shape parameters so that the affine combinations of mesh vertices are strictly convex has not been resolved.

1.2 Implicit Polynomial Surface

Implicit techniques have been widely used in computer graphics, CAD/CAM and scientific visualization. The quadric surfaces include spheres, cones and cylinders, which are fundamental entities in many solid modeling systems.

A–splines are a suitable polynomial basis for working with piecewise implicit polynomial curves and surfaces. While it is possible to model a general closed surface of arbitrary genus as a single A–spline the geometry of implicit surfaces has proven to be more difficult to specify, interactively control, and polygonize than parametrics. The main shortcoming held against the use of implicit representations is that zeros of polynomials being multivalued may cause the zero contour to have multiple real sheets, self–intersections and several other undesirable singularities. On the positive end, using polynomials of the same degree, implicit polynomial splines have more degrees of freedom compared with parametric and hence potentially are more flexible to approximate a complicated surface with fewer number of pieces and to achieve a higher order of smoothness. The potential of implicits remains largely latent and virtually all commercial and many research modeling systems are based on the parametric representation. An exception is SHASTRA which allows modeling with both implicit and parametric splines [6].

1.2.1 Modeling Irregular Mesh Using Implicit Polynomial Surface

Sederberg [120] showed how various smooth implicit polynomial surfaces in trivariate Bernstein basis can be manipulated as functions in Bézier control tetrahedra with finite weights. He showed that if the coefficients of the Bernstein-Bézier form of the trivariate polynomial on the lines that parallel one edge, say L, of the tetrahedron all increase (or decrease) monotonically in the same direction, then any line parallel to L will intersect the zero contour surface patch at most once. Patrikalakis and Kriezis [98] extended this by considering implicit polynomial surfaces in a tensor product B–spline basis. However the problem of selecting weights or specifying knot sequences for C^1 meshes of implicit polynomial surface patches which fit given spatial data was left open.

The problem of constructing a C^1 mesh of implicit polynomial patches based on an input polyhedron $\mathcal{P}$ has been considered by many. The two most important issues are: (1) continuity between adjacent pieces; (2) single-sheetedness and non-singularity of the surface. They are addressed with different piecewise structures in different schemes. Basically, there are two kind of piecewise structures, simplex-based or non-simplex-based. A simplex-based approach constructs a tetrahedral mesh. The implicit polynomial surface consists of the zero set of the Bernstein-Bézier polynomial defined within each simplex.

A simplex-based approach enforces continuity between adjacent patches by enforcing that the trivariate polynomials are continuous to each other. Similar to the trivariate case, degree-bounded vertices can be introduced to prevent the continuity system from globally propagating. Trivariate Clough-Tocher split [142] and Power-Sabin split are used to introduce degree-bounded vertices. Such a splitting, however, results in a large number of patches. A full trivariate Clough-Tocher split would split a tetrahedron into 12 sub-tetrahedra. However, as only the zero set of the polynomial is of interest, one does not need a complete mesh covering the whole space. A vertex does not need to be fully covered by the trihedral corners of the incident tetrahedra. A "incomplete" vertex would help introducing degree-bounded vertexes. An effective construction of this kind first appears Dahmen [47], and developed by Guo, Lodha, Dahmen and Thamm-Scharr, Bajaj, Chen and Xu [17, 48, 63, 82]. Given a triangulated polyhedron, the construction covers a triangular face by a pair of tetrahedra (called plumbs, or face tetrahedra), one on each side. The surface is set to pass through the three "bottom" vertices of the tetrahedra. Hence a tetrahedron (called a wedge, or an edge tetrahedron) is enough to fill in the gap between tetrahedra built on adjacent faces. The "top" of a face tetrahedron is thus degree-bound to just four simplexes. Face Powell-Sabin split or Clough-Tocher split is enough in different variations. An face split only splits a face of the tetrahedron, analogue to the trivariate case. In a face Clough-Tocher split, the tetrahedron is split into only three sub-tetrahedra. The number of macro patches are greatly reduced. An edge tetrahedron is usually split into two in a C^1 or C^2 scheme. We call this kind of construction the *interpolant (simplicial) hull* approach in the following survey, referring to the fact that surface passes through the vertices of the simplicial mesh. A disadvantage of an interpolant simplicial hull is that, as the surface is pinned to the vertices of the hull, a modification of the surface would involve both the changes of the coefficients and the simplicial hull, whereas if the surface does not have to pass through any vertices, surface modification can be done solely by changing the coefficients of the polynomial, which may lead to simpler formulation in surface optimization.

Dahmen [47] presented an interpolant hull scheme for constructing C^1 continuous piecewise quadric surface patches over a data triangulation in space. In his construction, each triangular face is covered by a tetrahedron. Similar to the Powell–Sabin split [110], the tetrahedron is then face-split and replaced by six subtetrahedra, each of which defines a micro quadric triangular patch. Additional simplexes are then used to fill in the gaps between the simplexes built on adjacent triangular faces. Dahmen's technique however works only if the original triangulation of the data set allows a transversal system of planes, and hence is quite restricted. The most important contribution of this scheme, however, is the construction of the simplicial hull.

Moore and Warren [89] extended the marching cubes scheme to a generic simplex-based scheme, which compute a C^1 piecewise triquadratic B-splines approximation. An effective technique called "signed distance" is used to prevent the unwanted-sheeted problem when employing least square approximation. The unwanted sheeted problem emerges as the input point data, as to the approximating polynomial, is just some zero valued points that are highly unevenly distributed in the domain. Unwanted branches thus could appear in where there is no data point. Hence, auxiliary data points are evenly added in the domain with values being the signed distances to the input data points. The technique, although being a heuristic rather that a guarantee, has been proven to work very well in preventing unwanted branches. However, the auxiliary data points with values of signed distances would enforce the surface patch to be monotonic in one direction, which is more than necessary to prevent unwanted branches, and furthermore restricts the surface from better fitting the original data.

Guo [63] uses cubics in an interpolant hull approach, to create free–form geometric models and enforced monotonicity conditions on a cubic polynomial along the direction from one vertex to a point of the opposite face of the vertex. This condition is difficult to satisfy in general, and even if this condition is satisfied, one still cannot avoid singularities on the zero contour. A face Clough-Tocher split is used to subdivide each tetrahedron of the simplicial hull. Dahmen and Thamm-Scharr [48] independently develop a similar scheme utilizing a single cubic patch per tetrahedron at the cost of less locality and extra perturbation when adjacent faces of $\mathcal{P}$ are coplanar.

Lodha [82] constructed low degree surfaces with both parametric and implicit representations and investigated their properties. A method is described for creating quadratic triangular Bézier surface patches which lie on implicit quadric surfaces. And another method is described for creating biquadratic tensor product Bézier surface patches which lie on implicit cubic surfaces. The resulting patches satisfy all the standard properties of parametric Bézier surfaces, including interpolation of the corners of the control polyhedron and the convex hull property.

Bajaj and Ihm [23] construct low–degree implicit polynomial surfaces in a non-simplex based approach. The resulted surfaces approximate or contain with C^1 continuity any collection of points and algebraic space curves with derivative information. Their Hermite interpolation algorithm solves a homogeneous linear system of equations to compute the coefficients of the polynomial defining the implicit surface. Bajaj, Ihm and Warren [24] extend this idea to C^k (rescaling continuity) interpolate or least squares approximate implicit or parametric curves in space. They show this problem can be formulated as a constrained quadratic minimization problem, where the algebraic distance is minimized instead of the geometric distance.

Bajaj, Ihm and Warren [5, 21, 24] construct implicit surfaces to solve the scattered data fitting problem. Bajaj and Ihm [23] consider an arbitrary spatial triangulation $\mathcal{T}$ consisting of vertices in $I\!R^3$ (or more generally a simplicial polyhedron $\mathcal{P}$ when the triangulation in closed) with possibly normal vectors at the vertex points. Their algorithm constructs a C^1 continuous mesh of real implicit polynomial surface patches over $\mathcal{T}$ or $\mathcal{P}$. 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 C^1 interpolate all the vertices. The curvilinear wireframe is then fleshed to produce a single implicit surface patch of degree at most 7 for each triangular face $\mathcal{T}$ of $\mathcal{P}$. If the triangulation is convex then the degree is at most 5. Similar techniques exist for parametrics [52, 101, 118] however the geometric degree of the solution surfaces tend to be prohibitively high.

In [22, 47, 48, 63, 64, 89], heuristics are provided based on monotonicity and least square approximation to circumvent the multiple sheeted and singularity problems of implicit patches. Bajaj, Chen and Xu [17] construct 3- and 4-sided A–patches that are implicit surfaces in Bernstein–Bézier (BB) form that are smooth and single–sheeted. They give sufficiency conditions for the BB form of a trivariate polynomials within a tetrahedron such that the zero contour of the polynomial is a single sheeted non–singular surface within the tetrahedron and its cubic interpolant hull complex for the polyhedron $\mathcal{P}$ is guaranteed to be is both nonsingular and single sheeted. They distinguish between convex and non-convex facets and edges of the triangulation. For non-convex facets and edges a double-sided tetrahedra is built and for convex facets and edges single-sided tetrahedra are built. A generalization of Sederberg's condition is given for a three–sided patch where any line segment passing through a vertex of the tetrahedron and its opposite face intersects the patch at most once; and for a four–sided patch where any line segment connecting two points at opposite edges intersects the patch at most once. Instead of having coefficients be monotonically increasing or decreasing there is a single sign change condition. There are also free parameters for both local and global shape control. The scheme is extended to a C^2 quintic version [14], in which the one-signe-changed condition is further relaxed. Guo [65] extends the A-patch technique to quadratics using a Powell-Sabin split.

1.2.2 Other Aspects of Implicit Polynomial Surfaces

Sederberg [121, 122] uses cubic surfaces for free-form surface modeling. Surfaces of degree four, such as tori and cyclides, are needed in creating blends in geometric modeling applications. Blinn [33] uses implicit surfaces to modeling electron density maps of molecular structures. Wyvill, McPheeters and Wyvill [143] regard implicit surfaces as *soft objects* and use them to model animate natural phenomena such as smoke, clouds, mountains, coast-lines, living forms, mud, water, and fabrics.

1.2.2.1 Parameterization, Visualization and Interactive Control

As stated above, implicit surfaces are proven to be difficult to control or polygonize. Literature that concerns these issues includes [33, 143, 112, 34, 35, 66, 24, 141] Based on Bernstein-Bézier form, the A-patches technique, also provides a set of methods to parameterize [17], visualize and interactively control implicit surfaces [15, 16].

Witkin and Heckert [141], based on visualizing implicit surfaces as a set of particles, propose promising approaches of parameterization and interactive control of implicits. Pedersen [99] describes subdivision and parameterization schemes that make it easier to decorate implicit surfaces with texture mapping.

1.2.2.2 Implicit Surface Blending

In solid modeling systems and other circumstances, surfaces are often added to provide smooth transition between functional features of the object. Such kind of surfaces are referred as *blending surfaces.* Much work has been done with implicit blending surfaces [68, 69, 70, 78, 79, 80, 81, 88, 96, 137]. Bajaj and Evans [20] give a survey of this subject.

1.2.3 Parametric Surfaces vs Implicit Polynomial Surfaces

A parametric spline is a vector of multivariate (rational) polynomials while an implicit polynomial surface is the zero contour of a multivariate function. The parametric representations possess good properties which include easy to order, easy to generate points on, simpler patches, compact storage, and irreducibility. Implicit representations greatly facilitate the classification problem of whether a given point is on, above, or below a surface. Secondly, the implicit polynomial spline class is closed under basic geometric operations such as offset and intersection. Thirdly, the class of implicit surfaces is larger and includes parametric (rational) polynomial surface.

They contain conics and quadratics. also they are better than NURBS in retaining polynomial form in continuity constraints. they have much more variety in shapes. The last advantage we would like to mention about implicit surfaces is their application in visualization techniques. Implicit surfaces are theoretically the same as iso-surfaces, a class of visualization techniques. Furthermore, implicit surfaces can be used to construct surface on surface easily (see 1.6.1) with lower degree patches.

A compromise is that, parametric and implicit representations co-exist for the same surface model at the cost of storage space. Proper representation is used for proper operations. Implicit representations are used for location classifications while parametric representations are used for quick display. [59, 123, 124, 125] give several implicitization schemes which converts a parametric representation to an implicit one.

1.3 Polyhedron Subdivision

Subdivision techniques can be used to produce generally fair surfaces from arbitrary control meshes. Subdivision consists of splitting and averaging. Each edge or face is split and each new vertex introduced by the splitting is positioned at a fixed affine combination of its neighbor's weights.

The algorithms start with a polyhedral configuration of points, edges, and faces. First, for each face F, a new vertex (C's in Figure 1.3.1) is constructed for each existing vertex (V's) in F. The new vertex is called the *image* of the vertex on the face F. A simple construction is to take C as the midpoint from V to the centroid of the face F. However, any other ratio is fine too [138]. The corner-cutting mesh $\mathcal{T}$ consists of 3 kinds of cells. (1) F-face: constructed by linking all the new vertices C's in the face. An F-face looks like a shrunken version of the original face. (2) E-face: for each edge $E \in \mathcal{P}$, a 4-sided face is constructed by linking the image vertices of its endpoints on the two incident faces of E. For example(See figure 1.3.1), $C_1C_2C_3C_4$ is an E-face corresponding to edge V_1V_2. (3) V-face: for each vertex $V \in \mathcal{P}$, a new cell (Not necessary non-coplanar.) is constructed by linking the image vertices of V. For example (See figure 1.3.1), $C_2C_4C_8$ is a V-face.

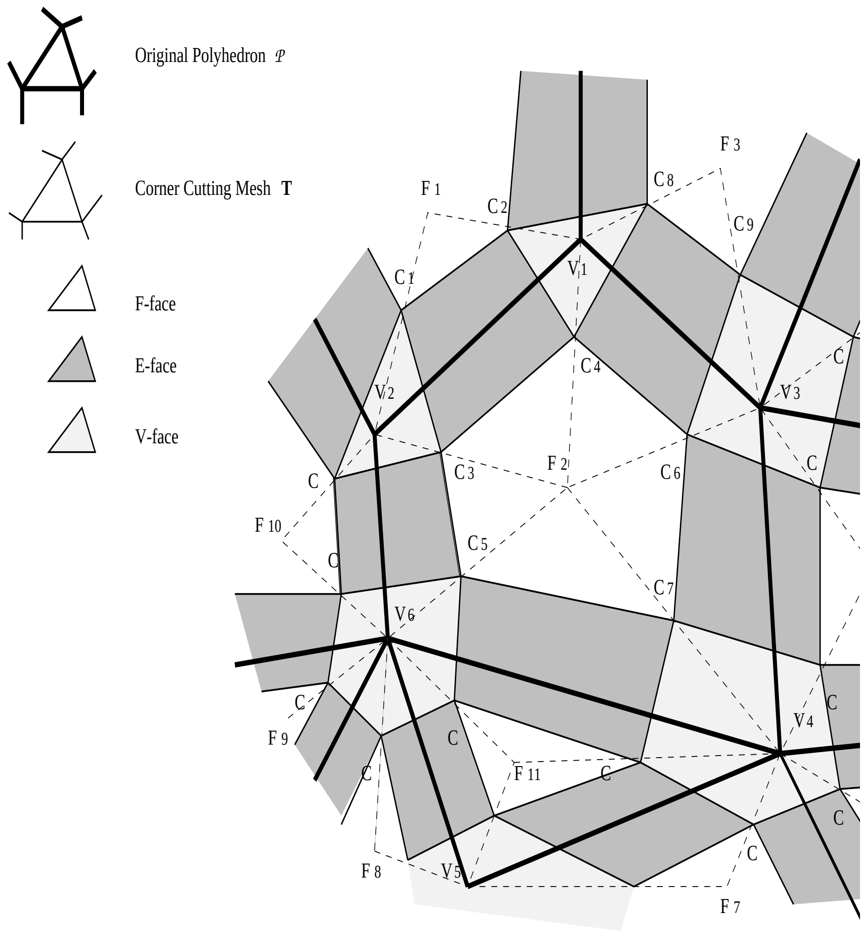


Figure 1.3.1 Traditional corner cutting scheme

The control mesh will in general consist of large regular regions and isolated singular regions. Subdivision enlarges the regular regions of the control net and shrinks the singular regions. Each application of the subdivision algorithm constructs a refined polyhedron, consisting of more points and smaller faces, tending in the limit to a smooth surface. In general the new control points are computed as a linear combination of old control points. The associated matrix is called the *subdivision matrix*.

The earliest of these approaches are the recursive subdivision schemes of Chaikin, Doo, Sabin, Catmull and Clark [38, 41, 49, 50]. These algorithms generate C^1 surfaces that interpolate the centroids of all faces at every step of subdivision. Nasri [93] describes a recursive subdivision surface scheme that is capable of interpolating points on irregular networks as well as normal vectors given at these points. The subdivision scheme developed by Loop [83] splits each triangle of a triangular mesh into four triangles. Each new vertex is positioned using a fixed convex combination of the vertices of the original mesh. The final limit surface has a continuous tangent plane. Hoppe et al. [71] extends Loop's method to incorporate sharp edges into the final limit surface. The vertices of the initial polyhedron are tagged as belonging on a face, edge, or vertex of the final limit surface. Based on this tag different averaging masks are used to produce new polyhedra.

Reif [115] presents a unified approach to subdivision algorithms for meshes with arbitrary topology and gives a sufficient condition for the regularity of the surface. The existence of a smooth regular parameterization for the generated surface near the point is determined from the leading eigenvalues of the subdivision matrix and an associated characteristic map.

As subdivision techniques do not necessarily yield surfaces with analytical representations. However, by careful arrangement, after initial steps of subdivision, the mesh can be used as control nets for piecewise Bézier patches or B-splines [102, 104, 84, 85] (see 1.1.4) or A-patch [18, 19]. The subdivision flavor is thus "taken over" by the subdivision algorithms of the particular parametric representation.

Bajaj, Chen and Xu [18, 19] construct an "inner" simplicial hull after one step of subdivision. Similar to the traditional subdivision schemes, the input polyhedron $\mathcal{P}$ can be used as a control mesh for free-form modeling. $\mathcal{T}$ can be considered as the second level mesh subdivided from mesh $\mathcal{P}$. Also similar to a traditional scheme, simple editing of $\mathcal{P}$ can impose some interesting constraints on the surface, such as interpolating a line or a region. Same as a traditional scheme, the cutting ratios control how deep a corner is smoothed. C^0 features can be mixed into the same model, by allowing zero cutting ratios and patches with singular vertices/edges (weights around the vertices/edges are all zero, compared to coincident control points in the parametric case).

Storry and Ball [128] demonstrate that a B–spline subdivision patch can be fitted into a general $n-$sided area of a bicubic surface with at least tangent plane continuity on the boundary. A Hermite formulation is used for the surface patches, and after the control points have been determined the subdivision algorithm is applied to produce an $n-$sided patch of optimal continuity properties. One degree of freedom is identified and related to shape control. However tangent plane continuity and the existence of a limit for the normal vectors is a rather weak smoothness category since local self–intersections of the surface are still possible.

1.4 Physically-based Deformable Models

The main problem of deformable model techniques are used to solve is the editability of free-form surfaces. A free-form surface often consists of many pieces of small patches. However, an application end user would like to control the shape in a natural and intuitive way rather than playing with tens if not hundreds of control points to get the object that looks right. On the other hand, virtual environment applications require that objects react as if they were "real". The underneath geometric representations (e.g., the structure of the piecewise surface and the control points) should be conceptually hidden from "intuitive" deformations. In order to be "intuitive", physically-based modeling is introduced to simulate soft objects of everyday life. In other words, an object deforms as its reaction to the presence of an external force. Other approaches use methods that are not physics-related, such as hierarchical splines.

Viewed from a more technical angle, the same problem can be stated as the following. We need to deform the object. There are more degrees of freedom than we need. How do we come up with a deformation that looks natural in the context of the application? It looks like an optimization problem, where the objective function is some measurement of "nature effect". The most widely used measurement is the "energy" function, typically the integral of a weighted sum of the second derivatives. Lagrange formula is also used generalize the objects to be dynamic, or elastic. Physically-based modeling is sometime referred to as *variational surfaces*, partially because of the fact that the "energy" function is actually a measurement of the variation of the surface, partially because that the researchers' goals are fair-looking surfaces, intuitive editing methods, rather than a "real" physical model.

1.4.1 Variational Surfaces

Variational constrained optimization plays a central role in the formulation of natural splines. Natural splines minimize the integral of the second derivative squared subject to the interpolation constraints. Surface models based on variational principles have been widely used in computer vision to solve surface reconstruction problems. Similar formulations have been employed in computer graphics for physically based modeling of deformable surfaces. These methods are based on regular finite difference grids of fixed resolution. Constrained optimization based on second–derivative norms has been used in fairing B–spline surfaces. Moreton [91] minimizes variation of curvature to generate surfaces which skin networks of curves while seeking circular or straight line cross–sections. Such schemes can give rise to very fair surfaces but the nonlinearity of their fairness metrics prevents then from being used for interactive surface design.

A typical representation of a deformable surface uses a grid of points where the points are allowed to move in relation to one another. The manner in which the points are allowed to move determines the properties of the deformable surface. One approach is a point mass-spring-hinge model. The points represent the often used abstraction called the point mass. The point mass has no size but does have a finite mass. Other characteristics of points include position and velocity. The spring model defines a distance relationship between two points. This relationship is the force that keeps the model together. The hinge model defines an orientation relationship between four points. Hinges, or torsional springs, maintain angular relationships and thus torques are involved.

Celniker and Gossard [39] proposed a physically based model for interactive free–form surface design in which the surface is modeled using a C^1 mesh of triangular patches, and positions and normals may be controlled along patch boundaries. Interactivity is possible because the surface fairing problem is formulated as a minimization of a quadratic functional subject to linear constraints.

Welch and Witkin [139] use an approach that is closely related to that of Celniker and Gossard, however they consider more general formulations for both surface objective functionals and shape control constraints. Their goal is to present the user with a infinitely malleable piecewise smooth surface with no fixed control or structure of its own. The user interactively manipulates variational curves and surfaces, controlling and combining them through a variety of constraints and objective functions. The user may attach a variety of features to the surface such as points and flexible curves, which then serve as handles for direct interactive manipulation of the surface. Within the constraints imposed by these controls surface behavior is governed by one or more simply expressed criteria such as the surface should be as smooth as possible or conform as closely as possible to a prototype. Formally the surfaces are specified as the solutions to constrained variational optimization problems. They seek shapes that extremize a variety of surface integrals subject to linear geometric constraints. Constraints that are linear in the surface control vector are used since this leads to a constrained optimization problem which can be solved at interactive speeds. The surface approximation formulated as a quadratic objective function with linear constraints can be solved using Lagrange multipliers to enforce a least–squares fit to the constraint matrix or with a penalty–based approach that associates a penalty term with each constraint. Such shapes are not intrinsically linked to any particular surface representation scheme, and the surface representation they use is piecewise polynomial nonuniform B–spline. The nonuniform surface is a simple sum of sparse uniform surface layers which may overlap in arbitrary ways. The resulting surface shape remains a linear function of the control points. The modeler must construct acceptable approximations to such infinite–dimensional variational surfaces using a finite number of control parameters. Automatic subdivision is used to ensure that constraints are met and to enforce error bounds.

Jou and Han [75, 76] propose energy-minimized model with trigonometric functions.

1.4.2 Elastic Dynamic Models

Besides used as editing tools, recently there has been work toward using physically based models as a tool to explore the phenomenon of “natural” motion. Research includes methods for representing deformable surfaces, deformation operations upon surfaces, and how these surfaces interact with an environment acting to deform them. A deformation operation is a geometric operation that takes a set of points and moves them in some manner, such as a warp or bend operation, regardless of any physical relationship.

Terzopoulos, Platt, Barr, Barzel, Fleischer, and Witkin [108, 129, 130, 140] have presented discrete models which are based extensively on the theory of elasticity and plasticity and use energy fields to define and enforce constraints. Haumann [67] used the same approach but used a triangulated model and a simpler physical model based on points, springs, and hinges. Work on deformable models not based on physical modeling has been done by Barr, Cobb, and Sederberg. Barr [32] presented a model for use with solid primitives using a functional map from one space into another to define the deformation. Sederberg and Parry [126] extended this model to use Bernstein polynomials. Cobb [45] defined a set of deformation modeling operations for use with the B–spline surface representation which were designed to modify groups of points in a correct and intuitive way.

Thingvold and Cohen [133] defined a model of elastic and plastic B–spline surfaces which supports both animation and design operations. The basis for the physical model is a generalized point mass-spring-hinge model which has been adapted into simultaneous refinement of the geometric/physical model. Always having a sculptured surface representation as well as the physical hinge/spring/mesh model allows the user to intertwine physical based operations, such as force application, with geometrical modeling. Refinement operations for spring and hinge B–spline models are compatible with the physics and mathematics of B–spline models. The models of elasticity and plasticity are written in terms of springs and hinges, and can be implemented with standard integration techniques to model realistic motions of elastic and plastic surfaces. These motions are controlled by the physical properties assigned and by kinematic constraints on various portions of the surface.

Terzopoulos and Qin [131] develop a dynamic generalization of the nonuniform rational B–spline (NURBS) model. They present a physics–based model that incorporates mass distributions, internal deformation energies, and other physical quantities into the NURBS geometric substrate. A modeler can interactively sculpt curves and surfaces and design complex shapes by adjusting control points and weights or through direct physical manipulation by applying simulated forces and local and global shape constraints. The dynamic behavior results from the numerical integration of a set of nonlinear differential equations that automatically evolve the control points and weights in response to the applied forces and constraints. The equations are derived from Lagrangian mechanics and a finite-element-line discretization. The interactive response of the dynamic NURBS may be modified by varying its mass and damping distributions. Global design requirements may also be achieved by varying physical parameters such as elastic energies. These dynamic NURBS can be used in applications such as rounding of solids, optimal surface fitting to unstructured data, surface design from cross–sections, and free–form deformations.

Qin and Terzopoulos [114] develop a dynamic freeform surface model based on swung NURBS surfaces which is useful for representing objects with symmetries and topological variability. NURBS swung surfaces are formed by swinging one planar NURBS profile curve along a second NURBS trajectory curve. Applications for these dynamic NURBS swung surfaces include interactive sculpting through the imposition of forces, the adjustment of physical parameters such as mass, damping, and elasticity, and surface design with geometric and physical constraints. The equations for motion are derived using the Lagrangian mechanics of an elastic surface and the finite element method. These swung NURBS surfaces are a special case of the dynamic NURBS surfaces in [131].

1.4.3 Other Applications

An application of physically-based techniques is computer vision. Kass, Witkin and Terzopoulos describe an "energy-based"active contour model called *snake*. A snake is an energy-minimizing spline subject to external constraint forces, image forces and internal forces. External constraint forces are the tools for users to guide the snake around features of interest. The image forces pulls the snake toward features such as lines and edges. The internal force restrains the spline from stretching and bending so as to keep the spline in a good shape. Snakes can actively lock onto nearby edges, localizing them accurately.

Terzopoulos, Witkin and Kass [132] describe a framework for 3D object reconstruction based on active symmetry-seeking models. The shapes of the models change in response to externally applied force. The symmetry-seeking property is designed into internal forces that constrain the deformations of the model.

Mumford [92] describe a model called *elastica*, based on trigonometric functions, that minimizes the integral of the square of curvatures. The technique is applied into computer vision for feature recognition.

1.5 Other Deformable Models

1.5.1 Hierarchical Splines

Hierarchical splines are a multi–resolution approach to the representation and manipulation of free–form surfaces. A hierarchal B–spline is constructed from a base surface (level 0) and a series of overlays are derived from the immediate parent in the hierarchy. Forsey [57] presents a refinement scheme that uses a hierarchy of rectangular B–spline overlays to produce C^2 surfaces. Overlays can be added manually to add detail to the surface, and local or global changes to the surface can be made by manipulating control points at different levels.

Forsey and Wang [58] create hierarchical bicubic B–spline approximations to scanned cylindrical data. The resulting hierarchical spline surface is interactively modifiable using editing capabilities of the hierarchical surface representation allowing either local or global changes to surface shape while retaining the details of the scanned data. However oscillations occur when the data has high–amplitude or high–frequency regions.

1.5.2 Free-form Deformations (FFD and EFFD)

A popular method to define spatial deformations is by using trivariate parametric volumes. Sederberg and Parry [126] presents a free-form deformation (FFD) technique in which a user manipulates control points of trivariate Bézier volumes. Coquillart [46] extends FFD to operate on volumes with a topology different from a cube. Joy [77] studies the applicability of trivariate modeling, calling it parametric hyperpatch and providing some higher-level editing operations. An integration of

FFD with superquadrics for solid modeling is described in Gödükbay and Özgüc [61]. Chadwick, Haumann and Parent [40] used FFD for animation.

1.5.3 Simple Constrained Deformations (Scodef)

Borrel and Rappoport [37] describe a model called *Simple constrained Deformations (Scodef)* The user defines a set of constraint points, giving a desired displacement and radius of influence for each. Each constraint point determines a local B-spline basis function centered at the constraint point, falling to zero for points beyond the radius. The deformed image of any point in space is a blend of these basis functions, using a projection matrix computed to satisfy the constraints. The deformation operates on the whole space regardless of the final shape of the deformed objects, and this shape can be fine-tuned by adjusting the radius of influence of each constraint point. The computations required by the technique can be done very efficiently, and real-time interactive deformation editing on current workstations is possible.

1.6 Surface Optimization and Applications

We give a survey about surface optimization techniques. By optimization, we mean the optimization of some function defined on the surface and auxiliary data. Hence we tend to consider data fitting, surface reconstruction from scattered data, physically-based modeling, deformable surfaces, variational surfaces as instances of optimization in different contexts.

1.6.1 Reconstructions and Surface on Surface

The problem here is the reconstruction of surfaces and scalar fields defined over it (surface-on-surface), from scattered trivariate data. The data points are assumed sampled from the surface of a 3D object, and the sampling is assumed to be *dense* for unambiguous reconstruction. Laser range scanners are able to produce a dense sampling, usually organized in a rectangular grid, of an object surface. Some 3D scanners are also able to measure the RGB components of the object color (i.e. three scalar fields) at each sampled point. When the object has a simple shape, this grid of points can be a sufficient representation. However, multiple scans are needed for objects with more complicated geometry, e.g. objects with holds, handles, pockets cannot be scanned in a single pass. Other applications, for example recovering the shape of a bone from contour data extracted from a CT scan, require reconstruction of a surface from data points organized in slices. The approach of considering the input points as unorganized has the advantage of generating cross-derivatives by a uniform treatment of all spatial directions.

Bajaj, Bernardini and Xu [8] reconstruct the sampled surface using A-patches. Their scheme effectively utilizes an incremental Delaunay 3D triangulation for a more adaptive fit; the dual 3D Voronoi diagram for efficient point location in signed distance computations and cubic implicit surface patches. Furthermore, in the same time they also compute a C^1 smooth approximation of the sampled surface-on-surface. Bajaj, Bernardini and Xu [7] have also developed a similar method based on tensor-product Bernstein-Bézier patches.

A different, three-step solution is given by Hoppe et al [71, 72, 73]. In the first phase, a triangular mesh that approximates the data points is created. In a second phase, the mesh is optimized with respect to the number of triangles and the distance from the data points. A third step constructs a smooth surface from the mesh.

The problem of modeling and visualizing just surface-on-surface arises in several physical analysis application areas: characterizing the rain fall on the earth, the pressure on the wing of an airplane and the temperature on the surface of a human body. A number of methods have been developed for dealing with this problem.

Currently known approaches for approximating surface-on-surface data however possess restrictions either on the domain surfaces or the surface-on-surface. The domain surfaces are usually assumed to be spherical, convex or genus zero. The surface-on-surface are not always polynomial [31, 95], or rather higher order polynomial [117], or a large number of pieces [2] compared to the approach of [8]. The method of [2] is a C^1 Clough-Tocher scheme that splits a tetrahedron into 4 subtetrahedra, uses quintic polynomials and requires C^2 data on the vertices of each subtetrahedron. Another Clough-Tocher scheme [142] requires only C^1 data at the vertices, for again constructing a C^1 function which is a cubic polynomial over each subtetrahedron, however splits the original tetrahedron into 12 pieces. A C^1 scheme [117] that does not split each tetrahedron uses degree 9 polynomials and requires C^4 data at the vertices. In extending the method of [117] to a C^2 scheme, requires degree 17 polynomials and C^8 data at the vertices of each tetrahedron. Compared to these approaches, the C^1/C^2 construction of [26] has no splitting and uses much lower degree polynomials (cubic/quintic) requiring only C^1/C^2 data respectively, at the vertices of each tetrahedron.

1.7 Outline of This Thesis

In this section, an outline is given for the rest of the thesis.

1.7.1 Preliminaries

Chapter 2 gives preliminary information that is used throughout the rest of the thesis. It includes brief introductions of the Bernstein-Bézier form, curvatures of implicit curves and surfaces, elastics and dynamics, and finally quadratic programming.

1.7.2 C^1 A-patches

Chapter 3 is based on [17]. We present a sufficient criterion for the Bernstein-Bezier (BB) form of a trivariate polynomial within a tetrahedron, such that the real zero contour of the polynomial defines a smooth, connected and single sheeted implicit polynomial surface patch. We call this an A-patch. We present algorithms to build a mesh of cubic A-patches to interpolate a given set of scattered point data in three dimensions, respecting the topology of any surface triangulation T of the given point set. In these algorithms we first specify "normals" on the data points, then build a simplicial hull consisting of tetrahedra surrounding the surface triangulation T and finally construct cubic A-patches within each tetrahedron. The resulting surface constructed is C^1 (tangent plane) continuous and single sheeted in each of the tetrahedra. We also show how to adjust the free parameters of the A-patches to achieve both local and global shape control.

1.7.3 C^2 A-patches

Chapter 4 is based on [14]. We present an efficient algorithm to construct C^2 smooth meshes of quintic A-patches that interpolate or approximate the vertices of a given triangulated polyhedron $\mathcal{P}$.

Similar to the C^1 cubic approach, a simplicial hull Σ is constructed based on $\mathcal{P}$. A quintic A-patch is then constructed within each tetrahedron of Σ so that the piecewise C^2 implicit polynomial surface interpolates or approximates the vertices of $\mathcal{P}$ with given C^2 data. We guarantee that the implicit polynomial surface is smooth and functional, namely, fully connected and free of singularities and unwanted branches.

1.7.4 Approximating A-patch Splines

Chapter 5 is based on [18, 19]. We present an efficient algorithm to construct an inner simplicial hull Σ based on a given polyhedron $\mathcal{P}$ in three dimensional space. Piecewise smooth C^1 and C^2 A-patches can then be constructed within Σ to approximate polyhedron $\mathcal{P}$. The scheme can be also compared to polyhedron subdivision, in the sense that a polyhedron $\mathcal{P}$ can be used as a "control net", which is "corner-cut" by the free-form A-patches. We also use patches with restricted singularity to model C^0 features.

1.7.5 Interactive Control and Rapid Display of A-patches

Chapter 6 is based on [15, 16]. We present algorithms to utilize the extra degrees of freedom of each A-patch for local shape control. A *ray shooting* scheme is also given to rapidly generate polygonal approximations of A-patches for graphic display.

A distributed implementation of this scheme gives nice "real time" performance on rendering the A-patches to support interactive shape control.

1.7.6 Elastic A-patch

In Chapter 7, we develop some energy-based surface optimization schemes for A-patch surface models. We proposed the hyperenergy model for A-patch surfaces. A hyperenergy of an A-patch is the integral of a weighted sum of the second derivatives of the functional surface. The optimization of such a hyperenergy is a typical quadratic programming problem with linear equality and inequality constraints, which can be solved efficiently.

1.7.7 Summary

Chapter 8 concludes the thesis. A summary of the thesis is given, followed by directions of related future work.

1.7.8 Appendix

A brief introduction to the SplineX modeling system is given in the appendix. The modeling system is used to implement most of the thesis.

2. PRELIMINARIES

In this chapter, we give some preliminaries as well as notations that are used in the following chapters.

2.1 Barycentric Coordinates and Bernstein-Bézier Polynomials

2.1.1 Convex Hull, Affine Hull

Let $\{p_1, \ldots, p_j\} \in I\!R^3$ with $j \leq 4$. Then the *convex hull* of these points is defined by

$$[p_1p_2...p_j] = \{p \in I\!R^3 : p = \sum_{i=1}^{j} \alpha_i p_i, \alpha_i \geq 0, \sum_{i=1}^{j} \alpha_i = 1\}$$

and the *affine hull* is defined by

$$\langle p_1p_2...p_j\rangle = \{p \in I\!R^3 : p = \sum_{i=1}^{j} \alpha_i p_i, \sum_{i=1}^{j} \alpha_i = 1\}.$$

The interior of the convex hull $[p_1p_2...p_j]$ is denoted by

$$(p_1p_2...p_j) = \{p \in I\!R^3 : p = \sum_{i=1}^{j} \alpha_i p_i, \alpha_i > 0, \sum_{i=1}^{j} \alpha_i = 1\}.$$

2.1.2 Barycentric Coordinates

Let $p_1,\ p_2,\ p_3,\ p_4 \in I\!R^3$ be affine independent. Then the tetrahedron with vertices $p_1,\ p_2,\ p_3$, and p_4, is $V = [p_1p_2p_3p_4]$. For any $p = \sum_{i=1}^{4} \alpha_i p_i \in V$, $\alpha = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)^T$ is the barycentric coordinate of p, given as

$$\alpha_1 = \frac{volume([pp_2p_3p_4])}{volume([p_1p_2p_3p_4])}$$

$$\alpha_2 = \frac{volume([p_1pp_3p_4])}{volume([p_1p_2p_3p_4])}$$

$$\alpha_3 = \frac{volume([p_1 p_2 p p_4])}{volume([p_1 p_2 p_3 p_4])}$$

$$\alpha_4 = \frac{volume([p_1 p_2 p_3 p])}{volume([p_1 p_2 p_3 p_4])}$$

Let $p = (x, y, z)^T$, $p_i = (x_i, y_i, z_i)^T$. Then the barycentric coordinates relate to the Cartesian coordinates via the following relation

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} \tag{2.1.1}$$

or

$$\begin{aligned} \begin{bmatrix} x \\ y \\ z \end{bmatrix} &= \begin{bmatrix} x_1 - x_4 & x_2 - x_4 & x_3 - x_4 \\ y_1 - y_4 & y_2 - y_4 & y_2 - y_4 \\ z_1 - z_4 & z_2 - z_4 & z_2 - z_4 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} + \begin{bmatrix} x_4 \\ y_4 \\ z_4 \end{bmatrix} \\ &= \mathbf{J}^{-1} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} + \begin{bmatrix} x_4 \\ y_4 \\ z_4 \end{bmatrix} \end{aligned} \tag{2.1.2}$$

$\mathbf{J}$ is the Jacobian matrix of α in terms of x,y and z.

$$\mathbf{J} = \frac{\partial(\alpha_1, \alpha_2, alpha_3)}{\partial(x, y, z)} = \begin{bmatrix} x_1 - x_4 & x_2 - x_4 & x_3 - x_4 \\ y_1 - y_4 & y_2 - y_4 & y_2 - y_4 \\ z_1 - z_4 & z_2 - z_4 & z_2 - z_4 \end{bmatrix}^{-1} \tag{2.1.3}$$

2.1.3 Bernstein-Bézier (BB) Form

Any polynomial $f(p)$ of degree n can be expressed as Bernstein-Bezier (BB) form over V as

$$f(p) = \sum_{|\lambda|=n} b_\lambda \, B_\lambda^n(\alpha), \;\; \lambda \in \mathcal{Z}_+^4 \tag{2.1.4}$$

where

$$B_\lambda^n(\alpha) = \frac{n!}{\lambda_1!\lambda_2!\lambda_3!\lambda_4!}\ \alpha_1^{\lambda_1}\alpha_2^{\lambda_2}\alpha_3^{\lambda_3}\alpha_4^{\lambda_4} \tag{2.1.5}$$

is Bernstein polynomial, $|\lambda| = \sum_{i=1}^4 \lambda_i$ with $\lambda = (\lambda_1, \lambda_2, \lambda_3, \lambda_4)^T$, $\alpha = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)^T$ $= \sum_{i=1}^4 \alpha_i e_i$ is barycentric coordinate of p, $b_\lambda = b_{\lambda_1\lambda_2\lambda_3\lambda_4}$ (as a subscript, we simply write λ as $\lambda_1\lambda_2\lambda_3\lambda_4$) are called *control points*, and $\mathcal{Z}_+^4$ stands for the set of all four dimensional vectors with nonnegative integer components. The following basic facts about the BB form will be used in this thesis. The first is derived from the directional derivative formulas (see [54]).

Lemma 2.1.1 If $f(p) = \sum_{|\lambda|=n}\ b_\lambda B_\lambda^n(\alpha)$, then

$$b_{(n-1)e_i+e_j} = b_{ne_i} + \frac{1}{n}(p_j - p_i)^T \nabla f(p_i), \quad j = 1, 2, 3, 4; \quad j \neq i \tag{2.1.6}$$

$$\begin{aligned} b_{(n-2)e_i+e_j+e_k} &= -b_{ne_i} + b_{(n-1)e_i+e_j} + b_{(n-1)e_i+e_k} \\ &+ \tfrac{1}{n(n-1)}(p_j - p_i)^T \nabla^2 f(p_i)(p_k - p_i), \quad j \neq i, k \neq i \end{aligned} \tag{2.1.7}$$

where

$$\nabla f(p) = [\frac{\partial f(p)}{\partial x}\ \frac{\partial f(p)}{\partial y}\ \frac{\partial f(p)}{\partial z}]^T. \tag{2.1.8}$$

(2.1.6) can be found in [62](p.23). (2.1.7) is derived from directional derivative formulas (see [53] p.310). They will be used to determine the control points around a vertex from the given C^1 and C^2 data at that vertex.

Lemma 2.1.2 ([54]) Let $f(p) = \sum_{|\lambda|=n}\ a_\lambda B_\lambda^n(\alpha)$ and $g(p) = \sum_{|\lambda|=n}\ b_\lambda B_\lambda^n(\alpha)$ be two polynomials defined on two tetrahedra $[p_1p_2p_3p_4]$ and $[p_1'p_2p_3p_4]$, respectively. Then
(i) f and g are C^0 continuous at the common face $[p_2p_3p_4]$ if and only if

$$a_\lambda = b_\lambda, \quad for \quad any \quad \lambda = 0\lambda_2\lambda_3\lambda_4, \quad |\lambda| = n \tag{2.1.9}$$

(ii) f and g are C^1 continuous at the common face $[p_2p_3p_4]$ if and only if (2.1.9) holds and

$$b_{1\lambda_2\lambda_3\lambda_4} = \beta_1 a_{1\lambda_2\lambda_3\lambda_4} + \beta_2 a_{0\lambda_2\lambda_3\lambda_4+0100} + \beta_3 a_{0\lambda_2\lambda_3\lambda_4+0010} + \beta_4 a_{0\lambda_2\lambda_3\lambda_4+0001} \tag{2.1.10}$$

(iii) f and g are C^2 continuous at the common face $[p_2p_3p_4]$ if and only if (2.1.9)-(2.1.10) holds and

$$\begin{aligned} b_{1\lambda_2\lambda_3\lambda_4} &= \beta_1^2 a_{2\lambda_2\lambda_3\lambda_4} + 2\beta_1\beta_2 a_{0\lambda_2\lambda_3\lambda_4+1100} + 2\beta_1\beta_3 a_{0\lambda_2\lambda_3\lambda_4+1010} + 2\beta_1\beta_4 a_{0\lambda_2\lambda_3\lambda_4+1001} \\ &+ \beta_2^2 a_{0\lambda_2\lambda_3\lambda_4+0200} + 2\beta_2\beta_3 a_{0\lambda_2\lambda_3\lambda_4+0110} + 2\beta_2\beta_4 a_{0\lambda_2\lambda_3\lambda_4+0101} \\ &+ \beta_3^2 a_{0\lambda_2\lambda_3\lambda_4+0020} + 2\beta_3\beta_4 a_{0\lambda_2\lambda_3\lambda_4+0011} + \beta_4^2 a_{0\lambda_2\lambda_3\lambda_4+0002} \end{aligned} \tag{2.1.11}$$

where $\beta = (\beta_1, \beta_2, \beta_3, \beta_4)^T$ are defined by the relation $p_1' = \beta_1 p_1 + \beta_2 p_2 + \beta_3 p_3 + \beta_4 p_4, \quad |\beta| = 1$.

Relation (2.1.10) is called *coplanar condition.*

In Lemma 2.1.2, if $\beta_2 = \beta_3 = 0$, that is p_1', p_4 and p_1 are collinear, then (2.1.10) and (2.1.11) become

$$a_{0\lambda_2\lambda_3\lambda_4+0001} = \mu_1 a_{1\lambda_2\lambda_3\lambda_4} + \mu_2 b_{1\lambda_2\lambda_3\lambda_4} \tag{2.1.12}$$

$$\mu_1^2 a_{2\lambda_2\lambda_3\lambda_4} - \mu_1 a_{0\lambda_2\lambda_3\lambda_4+1001} = \mu_2^2 b_{2\lambda_2\lambda_3\lambda_4} - \mu_2 b_{0\lambda_2\lambda_3\lambda_4+1001} \tag{2.1.13}$$

respectively, where $\mu_1 = -\frac{\beta_1}{\beta_4}, \mu_2 = \frac{1}{\beta_4}$, that is $p_4 = \mu_1 p_1 + \mu_2 p_1'$.

Degree Elevation. The polynomial $f(p) = \sum_{|\lambda|=n} b_\lambda B_\lambda^n(\alpha)$ can be written as one of degree $n+1$ (see e.g. [54]): $f(p) = \sum_{|\lambda|=n+1} (Eb)_\lambda B_\lambda^{n+1}(\alpha), \quad \lambda \in \mathcal{Z}_+^4$, where $(Eb)_\lambda = \frac{1}{n+1}\sum_{i=1}^4 \lambda_i b_{\lambda-e_i}$.

Variation Diminishing Property ([54],p.54). Let $y(t) = \sum_{i=0}^n b_i B_i^n(t)$, then $y(t)$ has no more intersections (counting the multiplicities) with any line than does the polygon $\{i/n, b_i\}_{i=0}^n$ in $[0,1]$.

Collecting the base functions B_λ^m's into a vector $\boldsymbol{B}^m$ and the coefficients b_λ's into a vector $\boldsymbol{b}^m$, equation 2.1.4 is rewritten as

$$F(\alpha) = \mathbf{b}^{m\mathsf{T}}\mathbf{B}^m \tag{2.1.14}$$

The coefficient b_λ's are listed in $\boldsymbol{b}$ in the increasing-backwards-lexicographic order. That is, in variate case as an example, coefficient of $w^i x^j y^k z^l$, b_{ijkl}, precedes b_{pqrs} if and only if $l < s$ or $l = s$ and b_{ijk} precedes b_{pqr}. If the variables are taken to be, e. g., $wxyz$, then in a second degree polynomial the coefficient order is

$$\begin{array}{lll} ww & wx & xx \\ wy & xy & \\ yy & & \\ zw & zx & \\ zy & & \\ zz & & \end{array}$$

Let $\Lambda(\lambda)$ be the one to one function that maps a λ to a index of vector $\boldsymbol{B}$ or $\boldsymbol{b}$. Namely,

$$\mathbf{b}[\Lambda(\lambda)] = \boldsymbol{b}_\lambda \tag{2.1.15}$$

$$\mathbf{B}[\Lambda(\lambda)] = \boldsymbol{B}_\lambda \tag{2.1.16}$$

Such a vector form is used in later chapters. In an implementation, these two mappings can coded into look-up tables for the lower dimensions and lower degree that are frequently used.

2.1.4 Transformation Between Bases

$$g(\alpha_1, \alpha_2, \alpha_3) = f(x(\alpha), y(\alpha), z(\alpha)) \tag{2.1.17}$$

The gradient.

$$\nabla f(x, y, z) = \mathbf{J}^T \nabla g(\alpha). \tag{2.1.18}$$

The Hessian

$$\nabla^2 f(x, y, z) = \boldsymbol{J}^{\mathsf{T}} \nabla^2 g(\alpha) \boldsymbol{J} \tag{2.1.19}$$

$$= \begin{pmatrix} \boldsymbol{J}_1{}^{\mathsf{T}} \nabla^2 g \boldsymbol{J}_1 & \boldsymbol{J}_1{}^{\mathsf{T}} \nabla^2 g \boldsymbol{J}_2 & \boldsymbol{J}_1{}^{\mathsf{T}} \nabla^2 g \boldsymbol{J}_3 \\ \boldsymbol{J}_2{}^{\mathsf{T}} \nabla^2 g \boldsymbol{J}_1 & \boldsymbol{J}_2{}^{\mathsf{T}} \nabla^2 g \boldsymbol{J}_2 & \boldsymbol{J}_1{}^{\mathsf{T}} \nabla^2 g \boldsymbol{J}_3 \\ \boldsymbol{J}_3{}^{\mathsf{T}} \nabla^2 g \boldsymbol{J}_1 & \boldsymbol{J}_3{}^{\mathsf{T}} \nabla^2 g \boldsymbol{J}_2 & \boldsymbol{J}_3{}^{\mathsf{T}} \nabla^2 g \boldsymbol{J}_3 \end{pmatrix} \tag{2.1.20}$$

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

2.1.4.1 Partial Derivatives

The partial derivatives of a BB polynomial $F^m(\alpha) = \boldsymbol{b}^{m\mathsf{T}}\boldsymbol{B}^m$ of degree m are given by

$$\begin{aligned}
F^m_{\alpha_j} &= \mathbf{b}^{m\mathsf{T}}\boldsymbol{B}^m_{\alpha_j}(\alpha) = \boldsymbol{b}^{m\mathsf{T}}\mathcal{D}^m_{\alpha_j}\boldsymbol{B}^{m-1}(\alpha) = \boldsymbol{b}^{m-1\,\mathsf{T}}_{\alpha_j}\boldsymbol{B}^{m-1}(\alpha)\\
F^m_{\alpha_j\alpha_k} &= \boldsymbol{b}^{m\mathsf{T}}\mathbf{B}_{\alpha_j\alpha_k}(\alpha) = \boldsymbol{b}^{m\mathsf{T}}\mathcal{D}^m_{\alpha_k}\mathcal{D}^m_{\alpha_j}\boldsymbol{B}^{m-1}(\alpha) = \boldsymbol{b}^{m-2\,\mathsf{T}}_{\alpha_j\alpha_k}\boldsymbol{B}^{m-1}(\alpha)
\end{aligned}$$

where

$$\begin{aligned}
\boldsymbol{b}^{m-1\,\mathsf{T}}_{\alpha_j} &= \boldsymbol{b}^{m\mathsf{T}}\mathcal{D}^m_{\alpha_j}\\
\boldsymbol{b}^{m-2\,\mathsf{T}}_{\alpha_j\alpha_k} &= \boldsymbol{b}^{m\mathsf{T}}\mathcal{D}^m_{\alpha_j}\mathcal{D}^{m-1}_{\alpha_k}
\end{aligned}$$

and $\mathcal{D}_{\alpha_j}$ $\mathcal{D}_{\alpha_k}$ are some matrices that are constant to $\mathbf{b}$. In particular, $\mathcal{D}_{\alpha_j}$ is a $\binom{m+d}{d}\times\binom{(m-1)+d}{d}$ matrix, each of whose elements

$$\mathcal{D}[s,t] = \begin{cases} 1, & \Lambda^{-1}(s)_j - 1 = \Lambda^{-1}(t)_j \\ -1, & \Lambda^{-1}(s)_{d+1} + 1 = \Lambda^{-1}(t)_{d+1} \\ 0, & otherwise \end{cases}$$

In the bivariate quadratic case,

$$\mathcal{D}_{\alpha_1} = 2\begin{bmatrix} 1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 0\\ -1 & 0 & 1\\ 0 & -1 & 0\\ 0 & 0 & -1 \end{bmatrix}, \qquad \mathcal{D}_{\alpha_2} = 2\begin{bmatrix} 0 & 0 & 0\\ 1 & 0 & 0\\ 0 & 1 & 0\\ -1 & 0 & 0\\ 0 & -1 & 1\\ 0 & 0 & -1 \end{bmatrix}.$$

We further denote

$$\mathcal{D}^m_{\alpha_j\alpha_k} = \mathcal{D}^m_{\alpha_j}\mathcal{D}^{m-1}_{\alpha_k}$$

2.2 Curvatures of Implicit Polynomial Curves and Surfaces

Given an implicit polynomial function $F(x,y)=0$, without loss of generality, we assume that in a neighborhood of the point (x,y), $y=y(x)$, namely, z is a function of x and y. Then we have

$$y_x = \frac{F_x}{F_y} \tag{2.2.21}$$

The curvature of an A-spline curve

$$\begin{aligned}\kappa &= \frac{(-F_x,F_y)\nabla^2(F)(-F_x,F_y)^T}{(F_x^2+F_y^2)^{3/2}} \\ &= \frac{\nabla F^{\mathsf{T}}\mathbf{P}\nabla^2(F)\mathbf{P}^T\nabla F}{} \\ (\nabla F^{\mathsf{T}}\nabla F)^{3/2} &= \frac{\nabla S^{\mathsf{T}}\mathbf{JPJ}^T\nabla^2(S)\mathbf{JP}^T\mathbf{J}^T\nabla S}{(\nabla S^{\mathsf{T}}\mathbf{JJ}^T\nabla S)^{3/2}}\end{aligned} \tag{2.2.22}$$

where the Hessian matrix

$$\nabla^2(F) = \begin{pmatrix} F_{xx} & F_{xy} \\ F_{xy} & F_{yy}\end{pmatrix} = \mathbf{J}^T\nabla^2(S)\mathbf{J}, \quad \nabla^2(S) = \begin{pmatrix} S_{uu} & S_{uv} \\ S_{uv} & S_{vv}\end{pmatrix} \tag{2.2.23}$$

the Jacobian matrix

$$\mathbf{J}(\mathbf{p}) = \begin{pmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y}\end{pmatrix} \tag{2.2.24}$$

and the permutation matrix

$$\boldsymbol{P} = \begin{pmatrix} 0 & 1 \\ -1 & 0\end{pmatrix}. \tag{2.2.25}$$

Given an implicit function $F(x,y,z)=0$, without loss of generality, we assume that in a neighborhood of the point (x,y,z), $z=z(x,y)$, namely, z is a function of x and y. Then we have

$$z_x = \frac{F_x}{F_z}, \qquad z_y = \frac{F_y}{F_z} \tag{2.2.26}$$

$$z_{xx} = -\left(\frac{F_x}{F_z}\right)_x \tag{2.2.27}$$

$$= -\frac{(F_{xx}-F_{xz}z_x)F_z-(F_{zx}-F_{zz}z_x)F_x}{F_z^3} \tag{2.2.28}$$

$$= -\frac{F_{zz}F_x^2 + F_{xx}F_z^2 - 2F_{xz}F_xF_z}{F_z^3} \tag{2.2.29}$$

$$= -\frac{(F_x\frac{\partial}{\partial z} - F_z\frac{\partial}{\partial z})^2 F}{F_z^3} \tag{2.2.30}$$

Similarly,

$$z_{yy} = -\frac{F_{zz}F_y^2 + F_{yy}F_z^2 - 2F_{yz}F_yF_z}{F_z^3} \tag{2.2.31}$$

$$= -\frac{(F_y\frac{\partial}{\partial z} - F_z\frac{\partial}{\partial z})^2 F}{F_z^3} \tag{2.2.32}$$

$$z_{xy} = -\frac{(F_{zz}F_xF_y + F_{xy}F_z^2 - F_{yz}F_xF_z - F_{xz}F_yF_z)}{F_z^3} \tag{2.2.33}$$

$$= \frac{(F_z\frac{\partial}{\partial x} - F_x\frac{\partial}{\partial z})(F_z\frac{\partial}{\partial y} - F_y\frac{\partial}{\partial z})F}{F_z^3} \tag{2.2.34}$$

The Gaussian curvature

$$K = \frac{z_{xx}z_{yy} - z_{xy}^2}{(1 + z_x^2 + z_y^2)^2} \tag{2.2.35}$$

$$= \frac{\nabla^{\mathsf{T}} F \begin{pmatrix} F_{yy}F_{zz} - F_{yz}^2 & F_{yz}F_{zx} - F_{zz}F_{xy} & F_{xy}F_{yz} - F_{yy}F_{zx} \\ F_{yz}F_{zx} - F_{zz}F_{xy} & F_{zz}F_{xx} - F_{zx}^2 & F_{xy}F_{zx} - F_{xx}F_{yz} \\ F_{xy}F_{yz} - F_{yy}F_{zx} & F_{xy}F_{zx} - F_{xx}F_{yz} & F_{xx}F_{yy} - F_{xy}^2 \end{pmatrix} \nabla F}{(F_x^2 + F_y^2 + F_z^2)^2} \tag{2.2.36}$$

The mean curvature

$$H = \frac{1}{2}\frac{(1 + z_y^2)z_{xx} + (1 + z_x^2)z_{yy} - 2z_xz_yz_{xy}}{(1 + z_x^2 + z_y^2)^{\frac{3}{2}}} \tag{2.2.37}$$

$$= \frac{1}{4}\frac{\nabla^{\mathsf{T}} F \begin{pmatrix} -F_{zz} - F_{yy} & F_{xy} & F_{xz} \\ F_{xy} & -F_{xx} - F_{zz} & F_{yz} \\ F_{xz} & F_{yz} & -F_{xx} - F_{yy} \end{pmatrix} \nabla F}{(F_x^2 + F_y^2 + F_z^2)^{\frac{3}{2}}} \frac{F_z}{|F_z|} \tag{2.2.38}$$

The the two principal curvatures κ_1 and κ_2 are the roots of the following quadratic equation.

$$\kappa^2 - 2H\kappa + K = 0 \tag{2.2.39}$$

2.3 Elastics and Dynamics

In this section, we adopt some paragraphs from [130] to give a brief review of physics preliminaries which are used in the physically-based models described later. Let $\boldsymbol{a}$ be the intrinsic or material coordinates of a point in a body Ω. For a solid body, $\boldsymbol{a} = [a_1, a_2, a_3]$; for a surface, $\boldsymbol{a} = [a_1, a_2]$; for a curve, $\boldsymbol{a} = [a_1]$. The Euclidean three-dimensional positions of points in the body are given by a time-varying vector-valued function of the material coordinates $\boldsymbol{r}(\boldsymbol{a}, t)$. The natural rest state is specified by $\boldsymbol{r}^{\,0}(\boldsymbol{a})$.

The distance between two points $\boldsymbol{a}$ and $\boldsymbol{a} + d\boldsymbol{a}$ is

$$dl = \sum_{i,j} G_{ij} da_i da_j, \tag{2.3.40}$$

where the symmetric matrix

$$G_{ij}(\boldsymbol{r}(\boldsymbol{a})) = \frac{\partial \boldsymbol{r}}{\partial a_i} \cdot \frac{\partial \boldsymbol{r}}{\partial a_j} \tag{2.3.41}$$

is known as the metric tensor of first fundamental form. The shape of a body is determined by the Euclidean distances between nearby points. As the body deforms, these distances change. G_{ij} describes how much the body is stretching.

The metric tensor determines the shape of a 3D solid. However, the metric tensor is not enough to determine the shape of a surface or a space curve. From the fundamental theorem of surfaces [56] one also needs the curvature tensor B, to determine the shape of a surface.

$$B_{ij}(\boldsymbol{r}(\boldsymbol{a})) = \boldsymbol{n} \cdot \frac{\partial^2 \boldsymbol{r}}{\partial a_i \partial a_j} \tag{2.3.42}$$

B_{ij} describes how much the body is bending.

For a space curve, the metric tensor and curvature tensor are scalars called the arc length $s(\boldsymbol{r}(a))$ and the curvature $\kappa(\boldsymbol{r}(a))$. Arc length and curvature, along with the torsion $\tau(\boldsymbol{r}(a))$, determine the shape of a space curve.

A reasonable strain energy for elastic bodies is a norm of the difference between the fundamental forms of the deformed body and the fundamental forms of the natural, undeformed body. This norm measures the amount of deformation away from the natural state. Typically, we use the strain energy

$$E_{curve}(\boldsymbol{r}) = \int_{\Omega} \left[\alpha(s - s^0)^2 + \beta(\kappa - \kappa^0)^2 + \gamma(\tau - \tau^0)^2\right] da \tag{2.3.43}$$

$$E_{surface}(\boldsymbol{r}) = \int_{\Omega} \left(\|G - G^0\|_{\alpha}^2 + \|B - B^0\|_{\beta}^2\right) da_1 da_2 \tag{2.3.44}$$

$$E_{solid}(\boldsymbol{r}) = \int_{\Omega} \|G - G^0\|_{\alpha}^2 da_1 da_2 da_3 \tag{2.3.45}$$

where $\|\cdot\|_{\alpha}$ and $\|\cdot\|_{\beta}$ are the weighted matrix norms.

The dynamic equation, or Lagrangian formula, is given as

$$\frac{\partial}{\partial t}(\rho \frac{\partial \boldsymbol{r}}{\partial t}) + \gamma \frac{\partial \boldsymbol{r}}{\partial t} + \frac{\delta E(\boldsymbol{r})}{\delta \boldsymbol{r}} = \boldsymbol{f}(\boldsymbol{r}, t), \tag{2.3.46}$$

where ρ is a mass density function, γ is a damping density function, E is the strain energy, and $\boldsymbol{f}(\boldsymbol{r}, t)$ some forcing function.

To simulate the dynamics of an elastic model, the system of ordinary differential equations is integrated through time. These equations are integrated using a numerical step-by-step procedure, which converts the system of nonlinear ordinary differential equations into a sequence of linear systems, the intermediate solutions of which can also can be utilized as an animation of the surface moving towards the optimum.

Most of the elastic models that are built on parameterizations other than the intrinsic, are referred to as "thin-plate-under-tension" models.

2.4 Quadratic Programming

The material of this section is adapted from [86]. The general quadratic program can be expressed as

$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{2}\boldsymbol{x}^\top \mathbf{Q}\boldsymbol{x} + \boldsymbol{x}^\top \boldsymbol{c} & & \\ \text{subject to} \quad & \boldsymbol{a}_i^\top \boldsymbol{x} = b_i, & & i \in E \\ & \boldsymbol{a}_i^\top \boldsymbol{x} \le b_i, & & i \in I. \end{aligned} \tag{2.4.1}$$

E and I are index sets for equality and inequality constraints. The matrix $\mathbf{Q}$ is symmetric and positive semidefinite (if not actually positive definite).

2.4.1 Equality Constraints

A quadratic program is greatly simplified, and can be solved in closed form, if it contains equality constraints only. Thus we consider in some detail the quadratic program

$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{2}\boldsymbol{x}^\top \mathbf{Q}\boldsymbol{x} + \boldsymbol{x}^\top \boldsymbol{c} \\ \text{subject to} \quad & \mathbf{A}\boldsymbol{x} = \boldsymbol{b} \end{aligned} \tag{2.4.2}$$

In this case it is clear that a unique solution exists if the matrix $\mathbf{A}$ is of full rank and the matrix $\mathbf{Q}$ is positive definite on the subspace $M = \{\boldsymbol{x} : \mathbf{A}\boldsymbol{x} = \mathbf{0}\}$. This follows from the general necessary conditions. (Note that if $\mathbf{Q}$ were only positive semidefinite on M, the solution would not be unique.)

The Lagrange necessary conditions for this problem are

$$\begin{aligned} \mathbf{Q}\boldsymbol{x} + \mathbf{A}^\top \boldsymbol{\lambda} \quad & +\boldsymbol{c} = \mathbf{o} \\ \mathbf{A}\boldsymbol{x} \quad & -\boldsymbol{b} = \mathbf{o} \end{aligned} \tag{2.4.3}$$

One can show nonsingular under the conditions stated above. Hence the system is solvable with unique solutions.

2.4.2 Active Set Method

The general quadratic program with inequality constraints is almost always solved by an active set method (see Section 11.3 in [86]). There is an especially simple version for the case where Q is positive definite. In this version, at iteration k a point

$\boldsymbol{x}_k$ is given that is feasible for all constraints and satisfies all the equality constraints of the current working set W_k. The working set always includes the equality constraints E and possibly some of the inequality constraints I. The quadratic program corresponding to the working set is the defined. by translating to the point $\boldsymbol{x}_k$, int the form

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\boldsymbol{d}_k^\top \boldsymbol{d}_k + \boldsymbol{g}_k^\top \boldsymbol{d}_k \\ \text{subject to} & \boldsymbol{a}_i^\top \boldsymbol{d}_k = 0, \qquad i \in W_k, \end{array} \tag{2.4.4}$$

where $\boldsymbol{g}_k = \boldsymbol{c} + \boldsymbol{Q}\boldsymbol{x}_k$. This program has only equality constraints and can be solved by solving system 2.4.3. If $\boldsymbol{d}_k = \mathbf{o}$, the current point $\boldsymbol{x}_k$ is optimal with respect to the working set. If $\boldsymbol{d}_k \neq \mathbf{o}$ and $\boldsymbol{x}_k + \boldsymbol{d}_k$ is feasible for all constraints, then $\boldsymbol{x}_k + \boldsymbol{d}_k$ becomes the new $\boldsymbol{x}_{k+1}$. If $\boldsymbol{x}_k + \boldsymbol{d}_k$ is not feasible, a search of the form $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha \boldsymbol{d}_k$ is made, and α_k is selected as large as possible to maintain feasibility. At this point a new inequality constraint is satisfied by equality, and this constraint is adjoined to the working set W_{k+1}. The general move is therefore $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$, where

$$\alpha_k = \text{minimum}\left\{1, \frac{b_i - \boldsymbol{a}_i^\top \boldsymbol{x}_k}{\boldsymbol{a}_i^\top \boldsymbol{d}_k}\right\}. \tag{2.4.5}$$

The process may proceed in this fashion, repeatedly adjoining constraints to the working set, but eventually, since there are only a finite number of constraints, a point is obtained that is a minimum over the current working set. At this point the corresponding components of $\boldsymbol{\lambda}$ for constraints in the working set, as determined by the solution of 2.4.3, are examined. If they are nonnegative for all members of I in the working set, the current point is optimal. If, on the other hand, at least one λ_i is negative for $i \in I$, then one such i (usually the most negative) is dropped from the working set and the process is continued. Termination occurs in a finite number of steps, since there is only a finite number of working sets.

In practice an important feature of the active set method for quadratic programming is that the required matrix inverse does not need to be computed from scratch at each step, but can ebe updated efficiently as a constraint is either added or dropped from the working set. This can be accomplished by use of tableau format. Recent methods are based on updating matrix factorizations.

3. C^1 A-PATCHES

3.1 Introduction

As surveyed in Section 1.2 of Chapter 1, implicit polynomial surfaces have been widely used. This chapter[1], uses the A-patch technique to model arbitrary surface with C^1 cubic patches.

We consider the computed "normals" at the given data points, and distinguish between "convex" and "non-convex" faces and edges of the triangulation. These concepts are formally defined in section 3.4. We use a single cubic A-patch per face of T except for the following two special cases. For a non-convex face, if additionally the three inner products of the face normal and its three adjacent face normals have different signs, then in this case one needs to subdivide the face using a single face Clough-Tocher split, yielding C^1 continuity with the help of three cubic A-patches for that face. Furthermore for coplanar adjacent faces of T, we show that the C^1 conditions cannot be met using a single cubic A-patch for each face. Hence for this case we again use face Clough-Tocher splits for the pair of coplanar faces yielding C^1 continuity with the help of three cubic A-patches per face. See also the examples and figures in section 7 where the savings in patches becomes evident.

In section 3.2, we present a sufficient criterion for the Bernstein-Bezier (BB) form of a trivariate polynomial within a tetrahedron such that the real zero-contour of the polynomial is smooth (non-singular) and a single sheeted algebraic surface. We call this an A-patch. In section 4, we describe how to build a interpolant simplicial hull consisting of tetrahedra surrounding a surface triangulation T of the set of scattered data points in 3D. We then show in section 5 how a mesh of cubic A-patches can be used to construct a C^1 interpolatory surface, respecting the topology of the surface

[1]The material covered in this chapter is mostly in common with [17].

triangulation T. In section 6, we show how to adjust the free parameters of the A-patches to achieve both local and global shape control. This C^1 cubic A-patch fitting algorithm is quite appropriate for free form design. In analogy to the final smoothing of an artist's rough sketches, complicated smooth models can be directly formed by first creating a rough polyhedral model of the desired object and then using the fitting algorithms to produce a C^1 smooth solid with extra local and global parameters for fine shape control.

3.2 Sufficient Conditions of an A-Patch

Problem. Given a list of data points $P = \{p_1, \dots p_k\} \in I\!R^3$ and a surface triangulation T of these points, construct a mesh of low degree algebraic surfaces such that the composite surface is single sheeted C^1 continuous and has the same topology as T.

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

Definition 3.2.1 *Smooth vertex condition.* For each $i (1 \leq i \leq 4)$, there is at least one non-zero $b_{\lambda_1\lambda_2\lambda_3\lambda_4}$ for $\lambda_i \geq n-1$.

Definition 3.2.2 *Smooth edge condition.* For each pair $(i,j)(1 \leq i,j \leq 4, i \neq j)$, there is either at least one non-zero $b_{me_i+(n-m)e_j}$ for $m = 0, 1, \cdots, n$, or the polynomials

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

and

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

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 [53], p. 312), it is easy to show that the surface is smooth there if the smooth vertex/edge conditions above are satisfied.

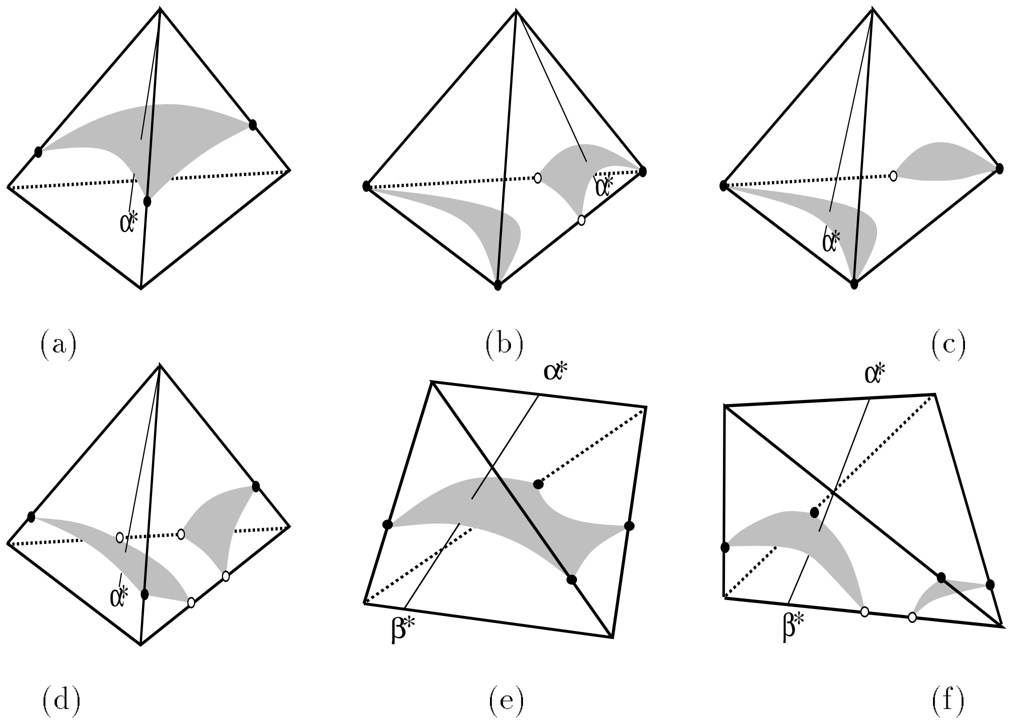


Figure 3.2.1 Three-sided (a, b, c and d) and four-sided patches (e and f). Some of them are disconnected. The filled vertices mark the boundaries of the patches

Definition 3.2.3 *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, α^*) with $\alpha^* \in S_j = \{(\alpha_1, \alpha_2, \alpha_3, \alpha_4)^T : \ \alpha_j = 0, \alpha_i > 0, \ \sum_{i \neq j} \alpha_i = 1\}$ intersects S_F at most once (counting multiplicities), then we call S_F a *three-sided* j*-patch* (see Figure 3.2.1).

Definition 3.2.4 *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 (α^*, β^*) with $\alpha^* \in (e_i e_j)$ and $\beta^* \in (e_k e_\ell)$ intersects S_F at most once (counting multiplicities), then we call S_F a *four-sided* ij*-*$k\ell$*-patch* (see Figure 3.2.1).

It is easy to see that if S_F is a four-sided ij-$k\ell$-patch, it is then also a ji-ℓk-patch, a ℓk-ji-patch, and so on.

Lemma 3.2.1 The three-sided j-patch and the four-sided ij-$k\ell$-patch are smooth (non-singular).

Proof: Let $g(\alpha_1, \alpha_2, \alpha_3) = F(\alpha_1, \alpha_2, \alpha_3, 1 - \alpha_1 - \alpha_2 - \alpha_3)$. The smoothness of the surface patch S_F requires that $\nabla g(\alpha_1, \alpha_2, \alpha_3) \neq 0$ for every $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)^T$ on S_F. We prove only the smoothness of the three-sided j-patch. The proof of smoothness of the four-sided patch is similar.

Suppose the three-sided j-patch is not smooth. There will then be a point $\alpha^* = (\alpha_1^*, \alpha_2^*, \alpha_3^*, \alpha_4^*)^T \in S_F$ in the interior of S such that $\nabla g = 0$. Since

$$\frac{\partial g}{\partial \alpha_i} = \frac{\partial F}{\partial \alpha_i} - \frac{\partial F}{\partial \alpha_4}, \qquad i = 1, 2, 3,$$

we have

$$\frac{\partial F}{\partial \alpha_1} = \frac{\partial F}{\partial \alpha_2} = \ldots = \frac{\partial F}{\partial \alpha_4}.$$

Using Euler's formula[136] for homogeneous polynomials

$$\sum_{i=1}^{4} \alpha_i \frac{\partial F}{\partial \alpha_i} = 4F$$

and $\sum_{i=1}^{4} \alpha_i = 1$, we have

$$\frac{\partial F}{\partial \alpha_i} = 0, \quad i = 1, \ldots, 4.$$

Let $p_1 \in S_j$ and $t = t^* \in (0, 1)$ such that

$$\alpha^* = t^* e_j + (1 - t^*) p_1 = \alpha(t^*).$$

That is $F(\alpha(t^*)) = 0$. And furthermore

$$\left. \frac{\partial F(\alpha(t))}{\partial t} \right|_{t=t^*} = \sum_{i=1}^{4} \frac{\partial F}{\partial \alpha_i} \frac{\partial \alpha_i}{\partial t} = 0.$$

This implies that t^* is a double zero of $F(\alpha(t))$, a contradiction to the definition of the three-sided patch. $\square$

Theorem 3.2.1 Let $F(\alpha) = \sum_{|\lambda|=n} b_\lambda B^n_\lambda(\alpha)$ satisfy the smooth vertex and smooth edge conditions and j $(1 \leq j \leq 4)$ be a given integer. If there exists an integer $k(0 \leq k < n)$ such that

$$b_{\lambda_1\lambda_2\lambda_3\lambda_4} \geq 0, \qquad \lambda_j = 0, 1, \ldots, k-1, \tag{3.2.1}$$

$$b_{\lambda_1\lambda_2\lambda_3\lambda_4} \leq 0, \qquad \lambda_j = k+1, \ldots, n \tag{3.2.2}$$

and $\sum_{\substack{|\lambda|=n \\ \lambda_j=0}} b_\lambda > 0$ if $k > 0$, $\sum_{\substack{|\lambda|=n \\ \lambda_j=m}} b_\lambda < 0$ for at least one $m(k < m \leq n)$, then S_F is a three-sided j-patch.

Proof: For the sake of simplicity, we assume $j = 4$. Let $p = (y_1, y_2, y_3, 0)^T \in S_4$ (i.e., $y_i > 0, \sum_{i=1}^3 y_i = 1$),

$$\alpha(t) \;=\; te_4 + (1-t)p = ((1-t)y_1, (1-t)y_2, (1-t)y_3, t)^T \tag{3.2.3}$$

for $t \in (0, 1)$. Then

$$\begin{aligned} F(\alpha(t)) &= \sum_{|\lambda|=n} \tfrac{b_\lambda n!}{\lambda!} y_1^{\lambda_1} y_2^{\lambda_2} y_3^{\lambda_3} (1-t)^{\lambda_1+\lambda_2+\lambda_3} t^{\lambda_4} \\ &= \sum_{|\lambda|=n} \tfrac{b_\lambda(\lambda_1+\lambda_2+\lambda_3)!}{\lambda_1!\lambda_2!\lambda_3!} y_1^{\lambda_1} y_2^{\lambda_2} y_3^{\lambda_3} \tfrac{n!}{(\lambda_1+\lambda_2+\lambda_3)!\lambda_4!} (1-t)^{\lambda_1+\lambda_2+\lambda_3} t^{\lambda_4} \\ &= \sum_{\ell=0}^n \left(\sum_{\substack{|\lambda|=n \\ \lambda_4=\ell}} b_\lambda B^{n-\ell}_{\lambda_1\lambda_2\lambda_3}(y_1, y_2, y_3) \right) B^n_\ell(t) \\ &= \sum_{\ell=0}^n B_\ell(y_1, y_2, y_3) B^n_\ell(t). \end{aligned} \tag{3.2.4}$$

By (3.2.1) and (3.2.2), $B_0 > 0$ if $k > 0$, $B_\ell \geq 0$, for $\ell = 1, \ldots, k-1$, $B_\ell \leq 0$, for $\ell = k+1, \ldots, n$. If $B_n = \ldots = B_{n-m+1} = 0$; $B_{n-m} < 0$ for some m with $0 \leq m \leq n-k-1$, then $F(\alpha(t))$ can be written as

$$F(\alpha(t)) = (1-t)^m \sum_{\ell=0}^{n-m} C_\ell(y_1, y_2, y_3) B^{n-m}_\ell(t) \tag{3.2.5}$$

where $C_0 > 0$ if $k > 0$, $C_{n-m} < 0$, and the sequence $C_0, C_1 \ldots C_{n-m}$ has at most one sign change. By the variation diminishing property of the functional BB form, the equation $F(\alpha(t))$ has at most one root in (0,1). Finally, we need to show the surface at the boundary of the tetrahedron is smooth. In the proof above, if we allow the intersection to occur at the boundary, then there may be an intersection of higher multiplicity at $t = 0$ or $t = 1$.That is, the surface contains vertices or edges of the tetrahedron. Here the smooth vertex and smooth edge conditions in the theorem guarantee that the surface is also smooth on the boundary of S. □

Theorem 3.2.2 Let $F(\alpha) = \sum_{|\lambda|=n} b_\lambda B^n_\lambda(\alpha)$ satisfy the smooth vertex and smooth edge conditions and (i, j, k, ℓ) be a permutation of (1, 2, 3, 4). If there exists an integer $k(0 \leq k < n)$ such that

$$b_{\lambda_1\lambda_2\lambda_3\lambda_4} \geq 0; \qquad \lambda_i + \lambda_j = 0, 1, \ldots, k-1, \tag{3.2.6}$$

$$b_{\lambda_1\lambda_2\lambda_3\lambda_4} \leq 0; \qquad \lambda_i + \lambda_j = k+1, \ldots, n \tag{3.2.7}$$

and $\sum_{\substack{|\lambda|=n \\ \lambda_i+\lambda_j=0}} b_\lambda > 0$ if $k > 0$, $\sum_{\substack{|\lambda|=n \\ \lambda_i+\lambda_j=m}} b_\lambda < 0$ for at least one $m(k < m \leq n)$, then S_F is four-sided ij-$k\ell$-patch.

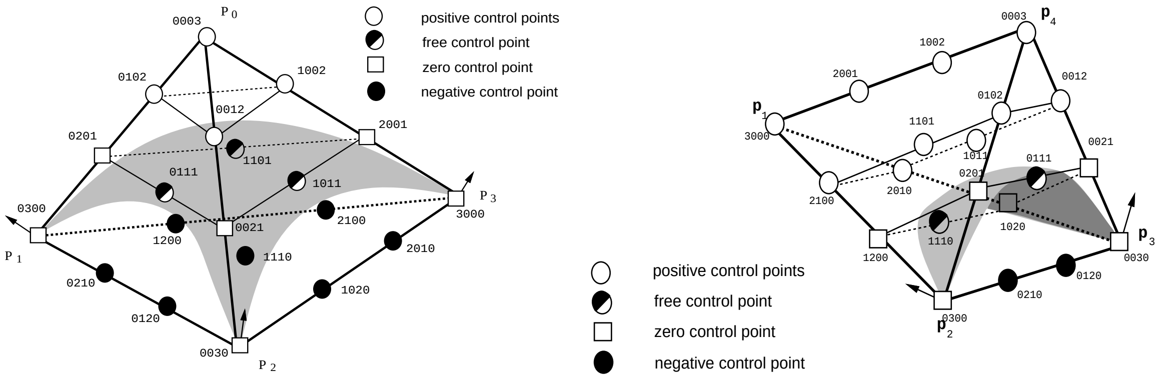
Proof: Without loss of generality, we assume $(i, j, k, \ell) = (1, 2, 3, 4)$. Then the edge $[e_1e_2]$ and $[e_3e_4]$ can be expressed as

$$[e_1e_2] = \{p : p = ue_1 + (1-u)e_2, \ u \in [0,1]\} \tag{3.2.8}$$

$$[e_3e_4] = \{p : p = ve_3 + (1-v)e_4, \ v \in [0,1]\} \tag{3.2.9}$$

and the line segment passing through the two edges is

$$\alpha(t) = t[e_1e_2] + (1-t)[e_3e_4] = (ut, (1-u)t, v(1-t), (1-v)(1-t))^T \tag{3.2.10}$$



(a) (b)

Figure 3.2.2 (a) A three sided patch tangent at p_1, p_2, p_3. (b) A degenerate four sided patch tangent to face $[p_1p_2p_4]$ at p_2 and $[p_1p_3p_4]$ at p_3.

for $t \in (0,1)$. Hence

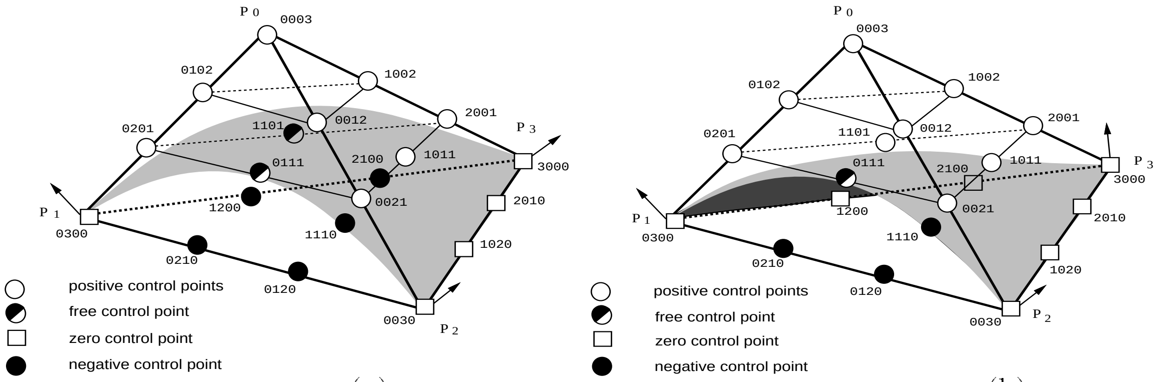
$$\begin{aligned} F(\alpha(t)) &= \textstyle\sum_{|\lambda|=n} \frac{b_\lambda n!}{\lambda!} u^{\lambda_1}(1-u)^{\lambda_2} v^{\lambda_3}(1-v)^{\lambda_4} t^{\lambda_1+\lambda_2}(1-t)^{\lambda_3+\lambda_4} \\ &= \textstyle\sum_{\ell=0}^{n} \left(\sum_{\substack{|\lambda|=n \\ \lambda_1+\lambda_2=\ell}} \frac{b_\lambda(\lambda_1+\lambda_2)!(\lambda_3+\lambda_4)!}{\lambda_1!\lambda_2!\lambda_3!\lambda_4!} u^{\lambda_1}(1-u)^{\lambda_2} v^{\lambda_3}(1-v)^{\lambda_4} \right) \cdot \\ &\quad \frac{n!}{\ell!(n-\ell)!} t^\ell (1-t)^{n-\ell} \\ &= \textstyle\sum_{\ell=0}^{n} B_\ell(u,v) B_\ell^n(t). \end{aligned} \tag{3.2.11}$$

It follows from (3.2.6) and (3.2.7) that $F(\alpha(t))$ has at most one zero in (0,1). Again, the smooth vertex and smooth edge conditions in the theorem guarantee that the surface is smooth on the boundary of S. $\square$

Note. The conditions on the coefficients b_λ in Theorems 3.2.1 and 3.2.2 are sufficient but not necessary. For example if we want some $B_l < 0$, it is not necessary to let every $b_\lambda < 0$, for $|\lambda| = n, \lambda_4 = \ell$.

Some properties of A-patches.

Property a. For a three-sided j-patch, if $b_\lambda = 0$ for $\lambda = (n-\ell)e_m + \ell e_j$, $\ell = 0, 1, \ldots, k (m \neq j, k < n)$, and $b_\lambda \neq 0$ for $\lambda = (n-1)e_m + e_s$, $s \neq j, m$, then the edge $[e_j e_m]$ is tangent with S_F at e_m with multiplicities k. See also Figure 3.2.2(a).



(a) (b)

Figure 3.2.3 (a) A three sided patch interpolating edge $[p_2p_3]$. (b) A three sided patch interpolating edges $[p_2p_3]$ and $[p_1p_3]$.

Proof: Property (a) can be verified by re-considering the proof of Theorem 3.2.1. For example, if $m = 1$, $j = 4$, $(y_1, y_2, y_3) = (1, 0, 0)$ at e_m. Hence

$$B_\ell(y_1, y_2, y_3) = b_{(n-\ell)e_1+\ell e_4} = 0, \qquad \ell = 0, 1, \ldots, k. \tag{3.2.12}$$

Therefore $t = 0$ is the root of $F(\alpha(t))$ with multiplicity $k+1$. On the other hand, e_m is not a singular point of S_F, since $b_\lambda \neq 0$ for $\lambda = (n-1)e_m + e_s$. □

Property b. For a four-sided ij-$k\ell$-patch, if $b_\lambda = 0$ for $\lambda = (n - q_1 - q_2)e_k + q_1e_i + q_2e_j$, $q_1 + q_2 = 0, 1, \ldots, s$; and $b_\lambda \neq 0$ for $\lambda = (n-1)e_k + e_\ell$, then S_F is tangent s times with face $[e_ie_je_k]$ at e_k.

Proof: We illustrate Property (b) by showing that any line passing through edge $[e_ie_j]$ and vertex e_k is tangent to S_F with multiplicity s. In fact, if we take $v = 1$ in the proof of Theorem 3.2.2, we have

$$B_\ell(u, v) = B_\ell(u, 1) = b_{\lambda_ie_i+\lambda_je_j+(n-\ell)e_k} = 0. \tag{3.2.13}$$

Hence $t = 0$ is a root of $F(\alpha(t))$ with multiplicity $s+1$. Again, e_k is not a singular point. □

Note that a four sided patch may degenerate into a two sided patch. See Figure 3.2.2 (b). However, we do not need to treat the degenerate patches any different, but consider it to be a special four sided patch.

Property c. For a three-sided j-patch, if $b_\lambda = 0$ for $\lambda = (n-m)e_i + me_k$, $m = 0, 1, \ldots, n$, then S_F contains the edge $[e_i, e_k]$. If further, $b_\lambda = 0$, for $\lambda = (n-m-1)e_i + me_k + e_j$, $m = 0, 1, \ldots, n-1$, then the S_F is tangent with the face $[e_i e_j e_k]$. See also Figure 3.2.3 (a), (b).

Proof: Similar to the proof of (a). $\square$

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

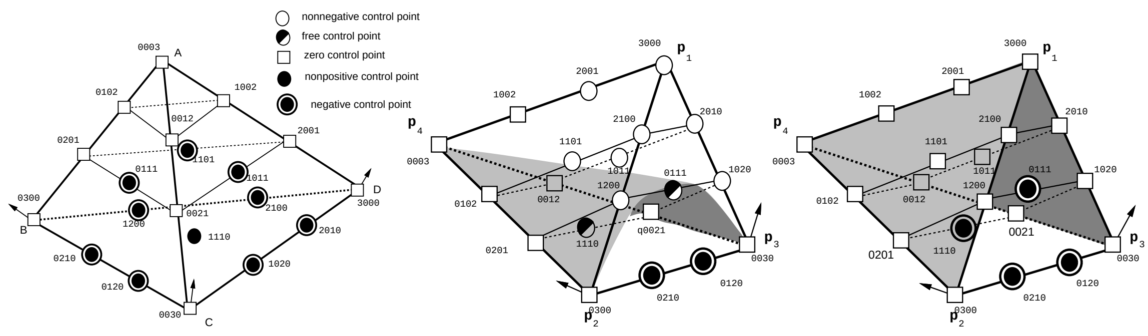
Proof: For Property (d), the mappings are given by the definition. A point $P \in S_F$ maps to $\alpha^* \in S_j$ if and only if line segment (e_j, α^*) intersects S_F at P. And the mapping is one to one for $\alpha^* \in S'_j = \{\alpha \in S_j, F(e_j) \cdot F(\alpha^*) \geq 0\}$, as (e_j, α^*) intersects S_F an odd number times iff $F(e_j) \cdot F(\alpha^*) \geq 0$. $\square$

Property e. For a four-sided ij-$k\ell$-patch, any point $P \in S_F$ 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 $\alpha^* \in (e_i e_j) = \{(\alpha_1, \alpha_2, \alpha_3, \alpha_4)^T : \alpha_k = \alpha_\ell = 0\}$ and $\beta^* \in (e_k e_\ell) = \{(\alpha_1, \alpha_2, \alpha_3, \alpha_4)^T : \alpha_i = \alpha_j = 0\}$. Furthermore, there exists a one to one mapping between S_F and $\{(\alpha_i, \beta_k)^T : F(\alpha^*) \cdot F(\beta^*) \leq 0\}$. If $F(\alpha^*) = 0$, S_F is degenerated and all the points with the same α_k collapse into one point.

Proof: Similar to the proof of *(d)*. $\square$

Hence a three-sided patch can be mapped into a triangular domain while a four-sided patch can be mapped into a quadrilateral domain. In other words, The last two properties actually identify a parameterization of each of three sided and four sided patches. 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 3.2.1 shows some examples. Subsequent



(a) (b) (c)

Figure 3.3.4 SA-patches: relaxed A-patch with singularities. (a) Three sided patch. Vertex A is non-smooth. With the current weight setting, the patch within the simplex is degenerated into three edges $[AB]$, $[AC]$ and $[AD]$. (b)Four sided patch. Vertex P_4 is non-smooth. The surface is singular at that point. (c)Four sided patch. Edge P_1P_4 is non-smooth. With the current weight setting, the patch is degenerated into two linear pieces, $[P_1P_2P_4]$ and $[P_1P_3P_4]$, who intersect with each other at $[P_1P_4]$.

sections detail how a combination of smooth A-patches are pieced together to form a C^1 smooth global surface.

3.3 SA-patches: Relaxed A-patches with Singular Vertices or Edges

One of the major issues over algebraic surface research is how to control singularities. However, controlling does not necessary mean eliminating. One can use patches with singularities to model C^0 features in a model.

Figure 3.3.4 shows the patches with non-smooth vertex and edge (Definitions 3.2.1 and 3.2.2). Please note that although with singularities, the weights of the patches still meet the "one sign change" condition, hence they are still functional.

At a singular point P_0, the gradient (F_x, F_y, F_z) is a zero vector. The normal at this point is not unique. Let P_1 be any point on the surface and within a small neighborhood of P, any smooth curve $[P_1P_0]$ on the surface defines a normal by taking a limit on the normal vector along the curve.

$$\boldsymbol{n}_{[P_1P_0]} = \lim_{P=(x,y,z)\in[P_1P_0], P->P_0} \left(\frac{F_x}{r}, \frac{F_y}{r}, \frac{F_z}{r}\right) \tag{3.3.14}$$

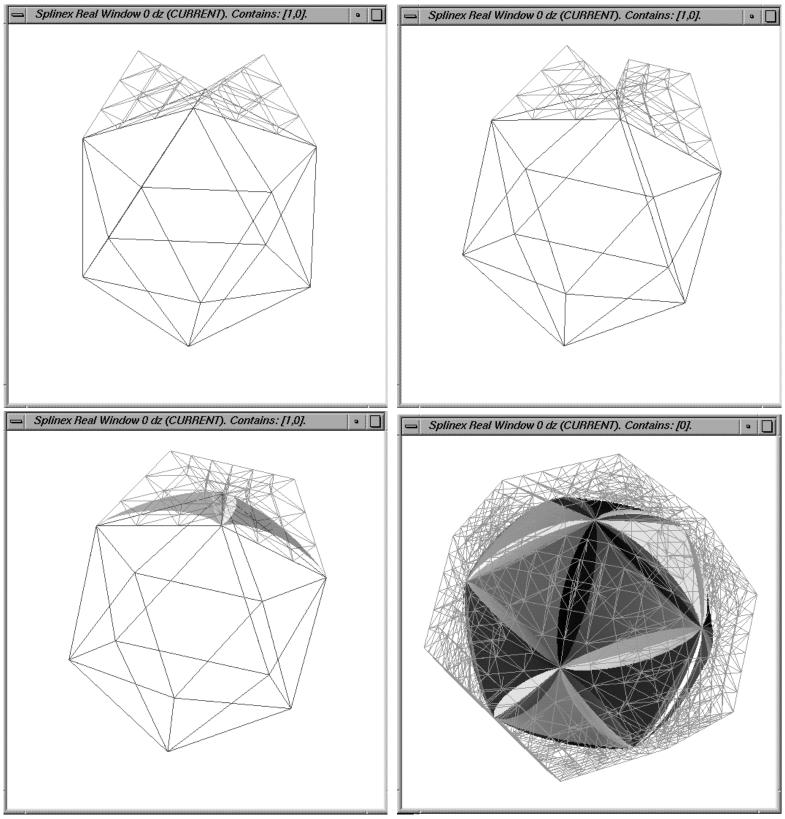


Figure 3.3.5 The construction of a simplicial hull and C^1 cubic A-patches interpolating the vertices of the input icosahedron.

where $r = \sqrt{F_x^2 + F_y^2 + F_z^2}$. In Figure 3.3.4(b), vertex P_4 has infinite number of distinct normals. In Figure 3.3.4(c), however, there are just two different normals for a point at the singular edge, the normals of the neighboring faces.

In Section 5.4, we discuss the possibility of using them to model, with a unified representation, solids that have both C^0 and C^1 features.

3.4 Normals and the Simplicial Hull

For the given point set $P = \{p_1, \ldots, p_k\} \in I\!R^3$ and their surface triangulation T, we first construct a normal set $N = \{n_1, \ldots, n_k\} \in I\!R^3$ for P. That is, for each point p_i, we associate a normal n_i. We will force the constructed surface to interpolate points p_i and at each point have a normal n_i for $i = 1, \cdots, k$. These normals therefore also provide a mechanism to control the shape of the C^1 interpolating surface.

Common approaches to construct these normals at a point p_i include (a) an average of the face normals of the incident faces (b) the gradient of a local spherical fit to the surface triangulation at each vertex. Computing an optimal normal assignment is yet an unsolved problem and we are experimenting with different local and global normal selections schemes [5, 90, 109]. Of course at times the data set can have pre-specified normals and this too can be the input of the C^1 fitting algorithm.

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 the normals all point to the exterior.

Definition 3.4.1 *Convex edge, non-convex edge.*

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

Definition 3.4.2 *Convex face, non-convex face.*

Let $[p_i p_j p_k]$ 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 $[p_i p_j p_k]$ is *positive convex*. If all the three edges are zero convex then we label the face as *zero convex*. A *negative convex* face is similarly defined. All the other cases $[p_i p_j p_k]$ are labeled as *non-convex*.

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 tetrahedra for non-convex faces.

Definition 3.4.3 *Interpolant Simplicial hull.*

An interpolant simplicial hull of T, denoted by $\sum$, is a collection of non-degenerate tetrahedra which satisfies:

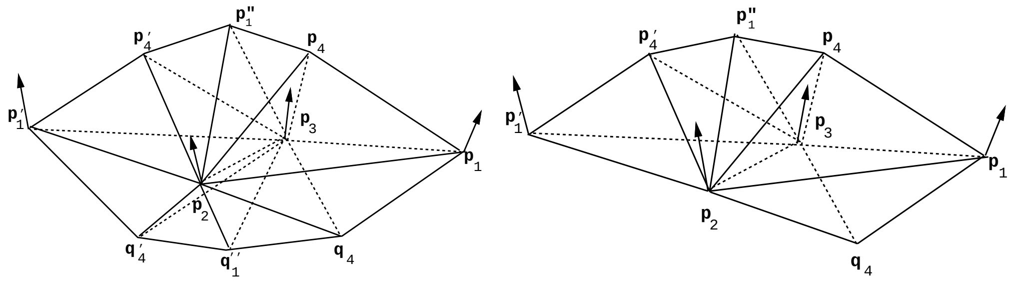


Figure 3.4.1 The construction of double tetrahedra for adjacent non-convex/non-convex faces and convex/non-convex faces

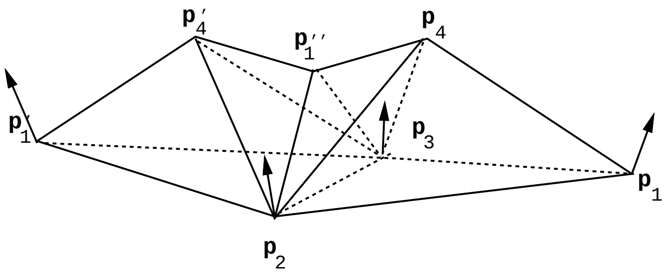
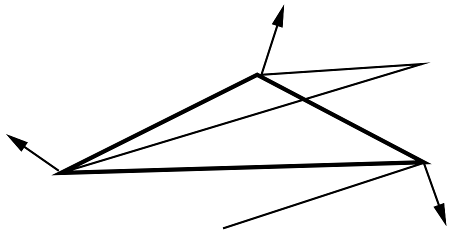


Figure 3.4.2 The construction of single tetrahedra for adjacent convex/convex faces

(1) Each tetrahedron in $\sum$ has either a single edge of T(then it will be called an *edge tetrahedron*) or a single face of T(then it will be called a *face tetrahedron*).
(2) For each face of T there is/are only one/two face tetrahedron/tetrahedra in $\sum$ if the face is convex/non-convex.
(3) Two face tetrahedra that share a common edge do not intersect anywhere else. This condition is referred to as *nonintersection.*
(4) For each edge there is only one pair of common face sharing edge tetrahedra in $\sum$ if the edge is convex, such that the pair fills the region between the two adjacent face tetrahedra in the same side of T.

For each edge there are only two pairs of common face sharing edge tetrahedra in $\sum$ if the edge is non-convex, such that the pairs fills the region between the two adjacent face tetrahedra in the same side of T.
(5) For each vertex, the tangent plane defined by the vertex normal is contained in all the tetrahedra containing the vertex. This condition is called *tangent plane containment.*

It should be noted that, for a given surface triangulation T with normal assignment, there may exist infinitely many interpolant simplicial hulls or no interpolant simplicial hull may exist. We now describe a scheme for constructing an interpolant simplicial hull for the surface triangulation T and prescribed vertex normal assignment. We also enumerate the exceptional configurations where an interpolant simplicial hull of T is difficult and then provide a solution for constructing the interpolant simplicial hull for a locally modified T.
1. Building Face Tetrahedra. For each face $F = [p_1p_2p_3]$ of T, let L be a straight line that is perpendicular to the face F and passes through the center of the inscribed circle of F. Then choose points p_4 and/or 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 tetrahedra $[p_1p_2p_3p_4]$ and $[p_1p_2p_3q_4]$ are formed (*double tetrahedra*). If F is positive convex, then p_4 is chosen on the same side as the direction of the normals, and a single face tetrahedron $[p_1p_2p_3p_4]$ is formed. If F is negative



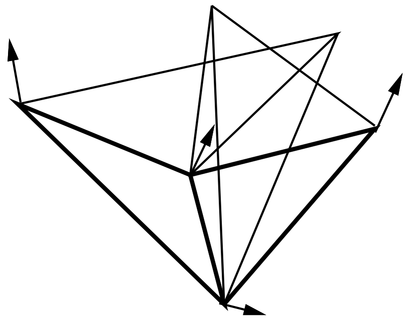


Figure 3.4.3 (a) No tangent plane containment (b) Self-intersecting tetrahedra

convex, then q_4 is chosen on the opposite side as the normals and again a single face tetrahedron $[p_1p_2p_3q_4]$ is formed. Figure 3.4.2 shows the case where both faces are convex and Figure 3.4.1 shows the cases where at least one of the two adjacent faces is non-convex.

A sufficient condition for constructing face tetrahedra with tangent plane containment is that the angle of the assigned normal n_i at each vertex p_i with each of the surrounding face's normals is less than $\pi/2$. If this condition is not met then an exception occurs and we term the vertex as *sharp*. See Figure 3.4.3 (a).

A sufficient condition for adjacent face tetrahedra to be non-intersecting is as follows. For two adjacent faces $F = [p_1p_2p_3]$ and $F' = [p'_1p_2p_3]$, 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 $[p_2p_3]$ the common edge between F and F', let $[p_1p_2p_3p_4]$ and $[p'_1p_2p_3p'_4]$ be the face tetrahedra respectively. Then the two tetrahedra are non-intersecting if the angles $\angle[p_4p_2p_3][p_1p_2p_3] < \frac{1}{2}\angle FF'$ and $\angle[p'_4p_2p_3][p'_1p_2p_3] < \frac{1}{2}\angle FF'$. If this condition is not met then an exception may occur and we term the common edge $[p_2p_3]$ as *sharp*. See Figure 3.4.3 (b).

A heuristic strategy rectifies the *sharp* edge and *sharp* vertex configurations is a local retriangulation of the original surface triangulation T. This strategy has worked well in several of the smoothing examples we have performed.

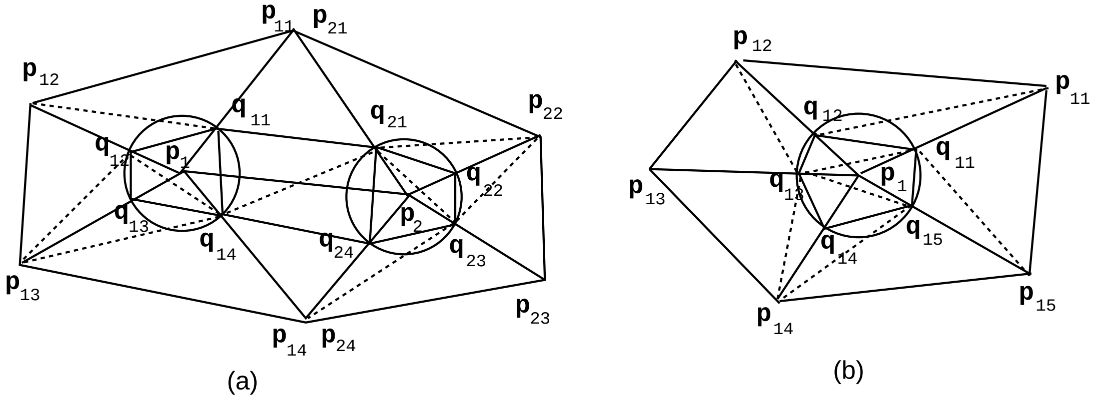


Figure 3.4.4 Retriangulation of (a) sharp edge and (b) sharp vertex

(i) Sharp edge problem. Let $[p_1 p_2]$ be a sharp edge (see Figure 3.4.4(a)), and let $[p_i p_{ij}]$ $(i = 1, 2;\ j = 1, 2, \cdots k_i)$ be the remaining surrounding edges of p_i in adjacency order. Take two spheres $S(p_i, r_i)$ with centers p_i and radius r_i, where r_i are positive numbers that are less than the half of the surrounding edge's lengths $\|p_i - p_{ij}\|$. The smaller r_i is, the sharper the constructed surfaces around edge $[p_1 p_2]$ are Let q_{ij} be the intersection points of $S(p_i, r_i)$ and $[p_i p_{ij}]$. Then $q_{i1}, q_{i2}, ..., q_{ik_i}$ form two closed polygons, and $p_{ij}, p_{ij+1}, q_{ij+1}, q_{ij}$ forms a four sided closed polygons and finally, $q_{11}, q_{21}, q_{2k_2}, q_{1k_1}$ forms another four sided closed polygon. Triangulate these polygons (the dotted line in Figure 3.4.4(a)) by connecting adjacent edges of the polygons in the least inner angle order.

(ii) Sharp vertex problem. Let p_1 be a sharp vertex (see Figure 3.4.4(b)), and let $[p_1 p_{1j}]$ $(j = 1, 2, \cdots k)$ be the surrounding edges of p_1 in adjacency order. Take a sphere $S(p_1, r)$ with center p_1 and radius r, where r is positive number that is less than the half of the surrounding edge's lengths $\|p_1 - p_{1j}\|$. The smaller r is, the sharper the constructed surfaces around vertex p_1 are. Let q_{1j} be the intersection points of $S(p_1, r)$ and $[p_1 p_{1j}]$. Then $q_{11}, q_{12}, ..., q_{1k}$ form a closed polygon, and $p_{1j}, p_{1j+1}, q_{1j+1}, q_{1j}$ forms a four sided closed polygon. Triangulate these polygons (the dotted line in Figure 3.4.4(b)) by connecting the adjacent edges of the polygon in the least inner angle.

2. Building Edge Tetrahedra. Let $[p_2p_3]$ be an edge of T and $[p_1p_2p_3]$ and $[p'_1p_2p_3]$ be the two adjacent faces. Let $[p_1p_2p_3p_4]$ and/or $[p_1p_2p_3q_4]$, and $[p'_1p_2p_3p'_4]$ and/or $[p'_1p_2p_3q'_4]$ be the face tetrahedra built for the faces $[p_1p_2p_3]$ and $[p'_1p_2p_3]$, respectively. Then if the edge $[p_2p_3]$ is non-convex, two pairs of tetrahedra need to be constructed. The first pair $[p''_1p_2p_3p_4]$ and $[p''_1p_2p_3p'_4]$ are between $[p'_1p_2p_3p'_4]$ and $[p_1p_2p_3p_4]$. The second pair $[q''_1p_2p_3q_4]$ and $[q''_1p_2p_3q'_4]$ are between $[p'_1p_2p_3q'_4]$ and $[p_1p_2p_3q_4]$. Here $p''_1 \in (p_4p'_4)$ or is above (p_4, p'_4), say

$$p''_1 = \frac{(1-t)}{2}(p_2 + p_3) + \frac{t}{2}(p'_4 + p_4), \quad t \geq 1 \tag{3.4.1}$$

so that p''_1 is above plane $[p_1p_2p_3]$ and plane $[p'_1p_2p_3]$. Similarly, $q''_1 \in (q_4q'_4)$ or is below (q_4, q'_4), say

$$q''_1 = \frac{(1-t)}{2}(p_2 + p_3) + \frac{t}{2}(q'_4 + q_4), \quad t \geq 1 \tag{3.4.2}$$

so that q''_1 is below plane $[p_1p_2p_3]$ and plane $[p'_1p_2p_3]$. If the edge $[p_2p_3]$ is positive/negative convex, only the first/second pair above are needed. If the edge $[p_2p_3]$ is zero convex, no tetrahedron is needed here. It should be noted that p_4 and $p'_4(q_4$ and $q'_4)$ are always visible.

3.5 Construction of a C^1 Interpolatory Surface Using Cubic A-Patches

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

$$f(p_i) = 0, \quad \nabla f(p_i) = n_i, \quad i = 1, 2, \ldots, k \tag{3.5.1}$$

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

3.5.1 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:

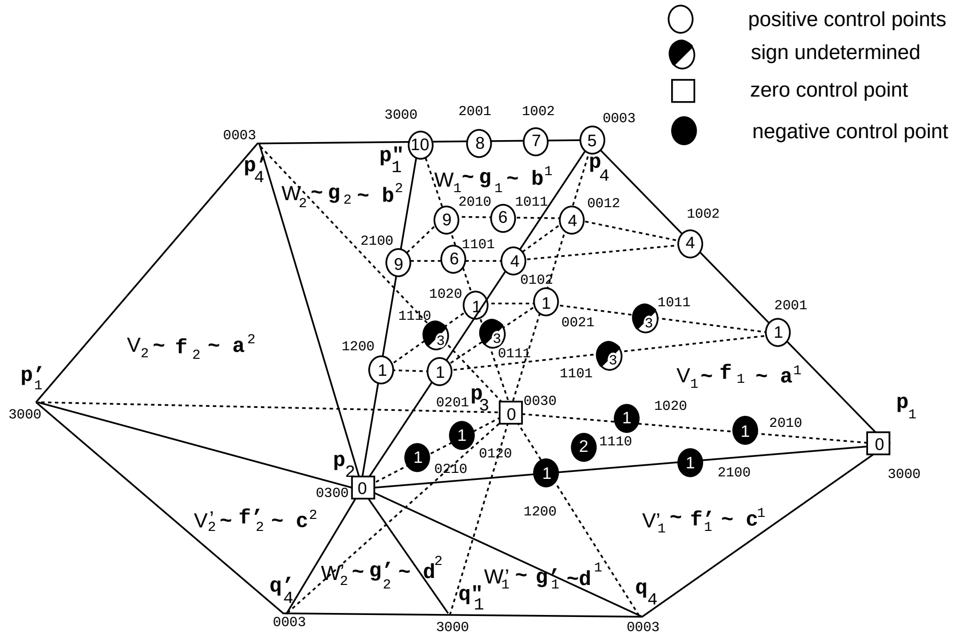


Figure 3.5.1 Adjacent double tetrahedra, functions and control points for two non-convex adjacent faces

(a) Both the faces are non-convex;

(b) Both the faces are convex;

(c) One of them is convex and the other is non-convex.

(a) Both the faces are non-convex

Let $F = [p_1 p_2 p_3]$ and $F' = [p'_1 p_2 p_3]$ be two adjacent non-convex faces. Then we have double tetrahedra $[p_1 p_2 p_3 p_4]$ and $[p_1 p_2 p_3 q_4]$ for F and double tetrahedra $[p'_1 p_2 p_3 p'_4]$ and $[p'_1 p_2 p_3 q'_4]$ for F' (see Figure 3.5.1). Let

$$V_1 = [p_1 p_2 p_3 p_4], \quad V_2 = [p'_1 p_2 p_3 p'_4], \quad W_1 = [p''_1 p_2 p_3 p_4], \quad W_2 = [p''_1 p_2 p_3 p'_4] \tag{3.5.2}$$

$$V'_1 = [p_1 p_2 p_3 q_4], \quad V'_2 = [p'_1 p_2 p_3 q'_4], \quad W'_1 = [q''_1 p_2 p_3 q_4], \quad W'_2 = [q''_1 p_2 p_3 q'_4] \tag{3.5.3}$$

and the cubic polynomials f_i over V_i, g_i over W_i, f'_i over V'_i and g'_i over W'_i be expressed in Bernstein-Bezier forms with coefficients a^i_λ, b^i_λ, c^i_λ, and d^i_λ, $i = 1, 2$, respectively. Now we shall determine these coefficients.

C^0 *Continuity:* If two tetrahedra share a common face, we equate the control points of the associated cubic polynomials on the common face (see Lemma 2.1.2):

$$a^i_{\lambda_1\lambda_2\lambda_3 0} = c^i_{\lambda_1\lambda_2\lambda_3 0}, \quad a^i_{0\lambda_2\lambda_3\lambda_4} = b^i_{0\lambda_2\lambda_3\lambda_4}, \quad b^1_{\lambda_1\lambda_2\lambda_3 0} = b^2_{\lambda_1\lambda_2\lambda_3 0} \tag{3.5.4}$$

$$c^i_{0\lambda_2\lambda_3\lambda_4} = d^i_{0\lambda_2\lambda_3\lambda_4}, \quad d^1_{\lambda_1\lambda_2\lambda_3 0} = d^2_{\lambda_1\lambda_2\lambda_3 0} \tag{3.5.5}$$

Interpolation: Since zero contours of f_i f'_i and g_i and g'_i pass through p_2 and p_3, $a^i_\lambda = b^i_\lambda = c^i_\lambda = d^i_\lambda = 0$ for $i = 1, 2$ and $\lambda = 0300, 0030$.

Normal Condition: From (3.5.1) and (2.1.6) we have, for $j = 2, 3$

$$\begin{aligned}
a^1_{2e_j+e_1} &= \tfrac{1}{3}(p_1 - p_j)^T n_j, & a^2_{2e_j+e_1} &= \tfrac{1}{3}(p'_1 - p_j)^T n_j \\
a^1_{2e_j+e_4} &= \tfrac{1}{3}(p_4 - p_j)^T n_j, & a^2_{2e_j+e_4} &= \tfrac{1}{3}(p'_4 - p_j)^T n_j, \\
b^1_{2e_j+e_1} &= \tfrac{1}{3}(p''_1 - p_j)^T n_j, & d^1_{2e_j+e_1} &= \tfrac{1}{3}(q''_1 - p_j)^T n_j, \\
c^1_{2e_j+e_4} &= \tfrac{1}{3}(q_4 - p_j)^T n_j, & c^2_{2e_j+e_4} &= \tfrac{1}{3}(q'_4 - p_j)^T n_j
\end{aligned} \tag{3.5.6}$$

C^1 *Conditions:* At present, set $a^i_{2e_4+e_j}$, $c^i_{2e_4+e_j}$, $j = 1, 2, 3, 4$, b^i_{2001}, and d^i_{2001} to any value (free parameters) and determine the other control points

1. *Interface of* $[p_2 p_3 p_4]$ *and* $[p_2 p_3 p'_4]$. Suppose

$$\begin{aligned}
p''_1 &= \beta^1_1 p_1 + \beta^1_2 p_2 + \beta^1_3 p_3 + \beta^1_4 p_4, & \beta^1_1 + \beta^1_2 + \beta^1_3 + \beta^1_4 = 1 \\
p''_1 &= \beta^2_1 p'_1 + \beta^2_2 p_2 + \beta^2_3 p_3 + \beta^2_4 p'_4, & \beta^2_1 + \beta^2_2 + \beta^2_3 + \beta^2_4 = 1
\end{aligned} \tag{3.5.7}$$

   Then, the C^1 conditions require (see Lemma 2.1.2)

$$b^i_{1\lambda_2\lambda_3\lambda_4} = \beta^i_1 a^i_{1\lambda_2\lambda_3\lambda_4} + \beta^i_2 a^i_{0\lambda_2\lambda_3\lambda_4+0100} + \beta^i_3 a^i_{0\lambda_2\lambda_3\lambda_4+0010} + \beta^i_4 a^i_{0\lambda_2\lambda_3\lambda_4+0001} \tag{3.5.8}$$

   for $\lambda_2\lambda_3\lambda_4 = 002, 101, 011, 110$. Hence b^i_{1002}, b^i_{1101}, and b^i_{1011} are defined, leaving a^i_{1011} and a^i_{1101} to be determined. Equation (3.5.8) for $\lambda_2\lambda_3\lambda_4 = 110$ will be treated later.

2. *Interface at* $[p_2 p_3 p''_1]$. Let

$$p''_1 = \mu_1 p_4 + \mu_2 p'_4 + \mu_3 p_2 + \mu_4 p_3, \quad \mu_1 + \mu_2 + \mu_3 + \mu_4 = 1 \tag{3.5.9}$$

then C^1 conditions require

$$b^i_{\lambda_1\lambda_2\lambda_3 0+1000} = \mu_1 b^1_{\lambda_1\lambda_2\lambda_3 1} + \mu_2 b^2_{\lambda_1\lambda_2\lambda_3 1} + \mu_3 b^i_{\lambda_1\lambda_2\lambda_3 0+0100} + \mu_4 b^i_{\lambda_1\lambda_2\lambda_3 0+0010} \quad (3.5.10)$$

for $\lambda_1\lambda_2\lambda_3 = 200, 110, 101, 011$. Hence b^i_{3000}, b^i_{2100}, and b^i_{2010}. are defined. The equation for $\lambda_1\lambda_2\lambda_3 = 011$ will be treated later together with (3.5.8).

3. *Interface between* $[p_2p_3q_4]$, $[p_2p_3q_1'']$ *and* $[p_2p_3q_4']$. 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 (3.5.8)–(3.5.10) hold when the quantities $a's$, $b's$, $\beta's, \mu's$ are substituted by $c's$, $d's$, $\gamma's, \eta's$ respectively. The two untreated equations left are

$$d^i_{1110} = \gamma^i_1 a^i_{1110} + \gamma^i_2 a^i_{0210} + \gamma^i_3 a^i_{0120} + \gamma^i_4 c^i_{0111} \quad (3.5.11)$$

$$d^i_{1110} = \eta_1 c^1_{0111} + \eta_2 c^2_{0111} + \eta_3 a^i_{0210} + \eta_4 a^i_{0120} \quad (3.5.12)$$

where the coefficients γ_i and η_i are defined by

$$\begin{aligned} q_1'' &= \gamma^1_1 p_1 + \gamma^1_2 p_2 + \gamma^1_3 p_3 + \gamma^1_4 q_4, \quad && \gamma^1_1 + \gamma^1_2 + \gamma^1_3 + \gamma^1_4 = 1 \\ q_1'' &= \gamma^2_1 p_1' + \gamma^2_2 p_2 + \gamma^2_3 p_3 + \gamma^2_4 q_4', \quad && \gamma^2_1 + \gamma^2_2 + \gamma^2_3 + \gamma^2_4 = 1 \\ q_1'' &= \eta_1 q_4 + \eta_2 q_4' + \eta_3 p_2 + \eta_4 p_3, \quad && \eta_1 + \eta_2 + \eta_3 + \eta_4 = 1 \end{aligned} \quad (3.5.13)$$

4. *Interface between* $[p_1p_2p_3]$ *and* $[p_1'p_2p_3]$. Let

$$\begin{aligned} q_4 &= \alpha^1_1 p_1 + \alpha^1_2 p_2 + \alpha^1_3 p_3 + \alpha^1_4 p_4, \quad && \alpha^1_1 + \alpha^1_2 + \alpha^1_3 + \alpha^1_4 = 1 \\ q_4' &= \alpha^2_1 p_1' + \alpha^2_2 p_2 + \alpha^2_3 p_3 + \alpha^2_4 p_4', \quad && \alpha^2_1 + \alpha^2_2 + \alpha^2_3 + \alpha^2_4 = 1 \end{aligned} \quad (3.5.14)$$

Then we have

$$c^i_{0111} = \alpha^i_1 a^i_{1110} + \alpha^i_2 a^i_{0210} + \alpha^i_3 a^i_{0120} + \alpha^i_4 a^i_{0111} \quad (3.5.15)$$

Now we treat the equations (3.5.8), (3.5.10), (3.5.11), (3.5.12) and (3.5.15). It follows from (3.5.8), (3.5.10), (3.5.11) and (3.5.12) that

$$\mu_1 a^1_{0111} + \mu_2 a^2_{0111} + \mu_3 a^i_{0210} + \mu_4 a^i_{0120} = \beta^i_1 a^i_{1110} + \beta^i_2 a^i_{0210} + \beta^i_3 a^i_{0120} + \beta^i_4 a^i_{0111} \quad (3.5.16)$$

$$\eta_1 c^1_{0111} + \eta_2 c^2_{0111} + \eta_3 a^i_{0210} + \eta_4 a^i_{0120} = \gamma^i_1 a^i_{1110} + \gamma^i_2 a^i_{0210} + \gamma^i_3 a^i_{0120} + \gamma^i_4 c^i_{0111} \quad (3.5.17)$$

Therefore, (3.5.15)–(3.5.17) form a linear system with six equations and six unknowns a^i_{0111}, a^i_{1110}, c^i_{0111} for $i = 1, 2$. It is important to point out that this is not an independent system (see Theorem 3.5.1 for the solvability of the system). It has 4 independent equations and has infinitely many solutions. In fact, if we assume p_1, p_2, p_3, p'_1 are not coplanar and then denote

$$\begin{array}{rclr} p_4 & = & \theta^1_1 p_1 + \theta^1_2 p_2 + \theta^1_3 p_3 + \theta^1_4 p'_1, & \theta^1_1 + \theta^1_2 + \theta^1_3 + \theta^1_4 = 1 \\ p'_4 & = & \theta^2_1 p_1 + \theta^2_2 p_2 + \theta^2_3 p_3 + \theta^2_4 p'_1, & \theta^2_1 + \theta^2_2 + \theta^2_3 + \theta^2_4 = 1 \\ q_4 & = & \vartheta^1_1 p_1 + \vartheta^1_2 p_2 + \vartheta^1_3 p_3 + \vartheta^1_4 p'_1, & \vartheta^1_1 + \vartheta^1_2 + \vartheta^1_3 + \vartheta^1_4 = 1 \\ q'_4 & = & \vartheta^2_1 p_1 + \vartheta^2_2 p_2 + \vartheta^2_3 p_3 + \vartheta^2_4 p'_1, & \vartheta^2_1 + \vartheta^2_2 + \vartheta^2_3 + \vartheta^2_4 = 1 \end{array} \tag{3.5.18}$$

then we can derive from (3.5.16) and (3.5.17) that

$$a^i_{0111} = \theta^i_1 a^1_{1110} + \theta^i_2 a^i_{0210} + \theta^i_3 a^i_{0120} + \theta^i_4 a^2_{1110} \tag{3.5.19}$$

$$c^i_{0111} = \vartheta^i_1 a^1_{1110} + \vartheta^i_2 a^i_{0210} + \vartheta^i_3 a^i_{0120} + \vartheta^i_4 a^2_{1110} \tag{3.5.20}$$

Actually, this means any group of four weights (e.g. a^1_{1110}, a^i_{0210}, a^i_{0120} and a^2_{1110}) defines the same 4-D hyper plane in its own barycentric coordinates (e.g. $[p_1 p_2 p_3 p'_1]$). Therefore, besides $a_0 210^1$ and $a_0 120^1$ (or $c_0 210^1$ and $c_0 120^1$, there are only 2 degree of freedom left. We choose a^i_{1110} (or c^i_{1110}) to be the free parameters. They may be determined by approximating a quadratic (see Section 3.6 or [48]).

(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 the control points are needed. That is, we need a^i_λ, b^i_λ for functions f_i and g_i if F and F' are nonnegative convex, or c^i_λ, d^i_λ 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. One way to choose a^i_{1110} (or c^i_{1110}) is to make the cubic approximate a quadratic (see Section 3.6). In particular, $a^i_{1110} = 0$ (or $c^i_{1110} = 0$) if the face is zero convex.

3. We now need only (3.5.19) for unknowns a^1_{0111} and a^2_{0111} if the edge $[p_2p_3]$ is nonnegative convex, or (3.5.20) for unknowns c^1_{0111} and c^2_{0111} if the edge $[p_2p_3]$ 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 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_λ, d^i_λ 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 (see §6).

3. we need only have (3.5.15) for $i = 2$ and (3.5.19) for unknowns a^1_{0111}, a^2_{0111} and c^2_{0111}.

(b3). Both faces are zero convex.

This case in fact is included in case (b1). 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 non-convex face.

Suppose $[p_1p_2p_3]$ is convex, $[p'_1p_2p_3]$ is non-convex. The following are the exceptions:

1. The function f'_1 and g'_i and their control points c^1_λ, d^i_λ are not needed if F is nonnegative convex. The function f_1 and g_i and their control points a^1_λ, b^i_λ are not needed if F is nonpositive convex.

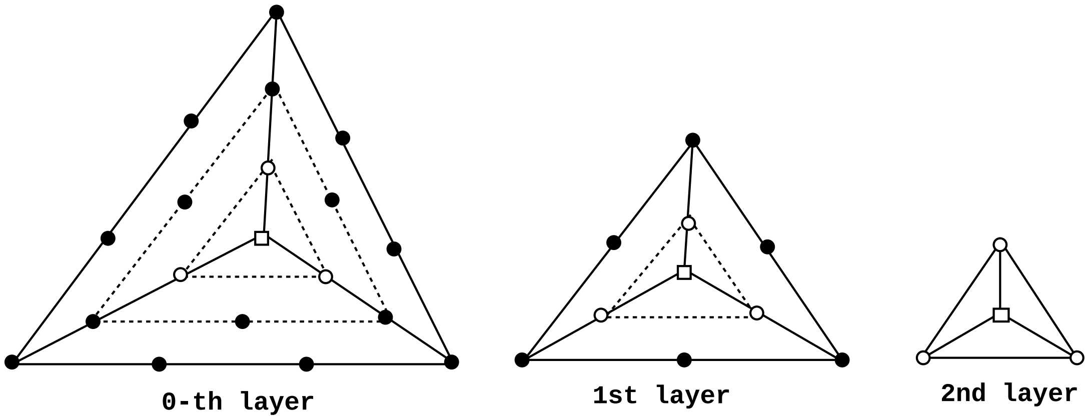


Figure 3.5.2 The control points of 0-th,1st and 2nd layers

2. $a^1_{1110} \leq 0$ and a^2_{1110} can be determined freely as in case (b). In particular, $a^1_{1110} = 0$ if $[p_1p_2p_3]$ is zero convex.

3. For the treatment of equations (3.5.15)–(3.5.17), we need only have (3.5.15) for $i = 2$ and (3.5.19) for unknowns a^1_{0111}, a^2_{0111} and c^2_{0111} if the edge $[p_2p_3]$ is nonnegative convex, or solve (3.5.15) for $i = 2$ and (3.5.17) for unknowns c^1_{0111}, c^2_{0111} and a^2_{0111} if the edge $[p_2p_3]$ is nonpositive convex (see Theorem 3.5.1 (ii) for the solvability of the system).

d. Coplanarity of adjacent faces

In the discussions above, we have assumed that p_1, p_1', p_2, p_3 are affine independent. If p_1, p_1', p_2, p_3 are coplanar, then the coefficient matrices of the linear systems (3.5.16) and (3.5.17) are singular. However, the system (3.5.15)–(3.5.17) are still solvable (see Theorem 3.5.1) taking a^i_{0111} or c^i_{0111} as free parameters. The other unknowns are given directly by these equations. Since the parameters $a^i_{1110}, i = 1, 2$ become now dependent, they are overly determined by the systems (3.5.15)–(3.5.17) around the 3 edges (e.g. edge $[p_1, p_2]$, $[p_2, p_3$ and $[p_3, p_1]$ for a_1110^1), and a solution may be not possible. In this case we split the involved tetrahedron into sub-tetrahedra by subdividing the triangles $[p_1p_2p_3]$ and $[p_1'p_2p_3]$ into three subtriangles at their center points w and w' (a *face Clough-Tocher split*). A solution is now possible where the coefficients are specified as before by regarding w as p_1 and w' as p_1'.

We then need to determine the remaining coefficients over the sub-tetrahedra $U_1 = [p_2p_3p_4w]$, $U_2 = [p_1p_3p_4w]$, and $U_3 = [p_1p_2p_4w]$ such that the C^1 condition is satisfied. In fact, since $w \in [p_1p_2p_3]$, the coefficients on the same layer are C^1 related. For the 0-th layer (see Figure 3.5.2), the control points labeled $\bullet$ are thus already determined. The control points $\circ$ are determined by a coplanar condition with surrounding $\bullet$. Finally, the point $\square$ is determined from the surrounding three points $\circ$ by the coplanar condition.

For the 1st layer (see Figure 3.5.2), the control points labeled $\circ$ and $\square$ are similarly determined as the 0-th layer. For the 2nd layer (see Figure 3.5.2), the control points $\circ$ are arbitrarily chosen and $\square$ is determined by the coplanar condition. Finally, the 3rd layer coefficient is free.

3.5.2 The Solvability of the Related System

Concerning the solvability of the system (3.5.15)–(3.5.17) and its sub-system, we have the following result.

Theorem 3.5.1 Given two affine independent point sets (p_2, p_3, p_4', p_4) and (p_2, p_3, q_4', q_4) as in Figure 3.5.1. (i) The system (3.5.15)–(3.5.17) has four independent equations. If (p_1, p_1', p_2, p_3) is affine independent, then (3.5.16) and (3.5.17) are four independent equations for the unknowns a^i_{0111} and c^i_{0111} for $i = 1, 2$.

(ii) Let

$$\{r_1, \cdots, r_6\} = \{p_1, p_1', p_4, p_4', q_4', q_4\},$$

$$\{x_1, \cdots, x_6\} = \{a^1_{1110}, a^2_{1110}, a^1_{0111}a^2_{0111}, c^1_{0111}, c^2_{0111}\}.$$

For any $1 \le i < j \le 6$, if r_i, r_j, p_2, p_3 are affine independent, then

$$x_k = \phi^k_1 x_i + \phi^k_2 x_j + \phi^k_3 a^1_{0210} + \phi^k_4 a^1_{0120}, \quad k \neq i, j \tag{3.5.21}$$

where ϕ^k_l are defined by $r_k = \phi^k_1 r_i + \phi^k_2 r_j + \phi^k_3 p_2 + \phi^k_4 p_3, \quad \phi^k_1 + \phi^k_2 + \phi^k_3 + \phi^k_4 = 1.$

Proof: (i). The system (3.5.15)–(3.5.17) can be written as $XA = -[a^i_{0210}\ a^i_{0120}]\ B$, where

$$A = \begin{bmatrix} \alpha_1^1 & 0 & \beta_1^1 & 0 & \gamma_1^1 & 0 \\ 0 & \alpha_1^2 & 0 & \beta_1^2 & 0 & \gamma_1^2 \\ \alpha_4^1 & 0 & \beta_4^1 - \mu_1 & -\mu_1 & 0 & 0 \\ 0 & \alpha_4^2 & -\mu_2 & \beta_4^2 - \mu_2 & 0 & 0 \\ -1 & 0 & 0 & 0 & \gamma_4^1 - \eta_1 & -\eta_1 \\ 0 & -1 & 0 & 0 & -\eta_2 & \gamma_4^2 - \eta_2 \end{bmatrix} \tag{3.5.22}$$

$$B = \begin{bmatrix} \alpha_2^1 & \alpha_2^2 & \beta_2^1 - \mu_3 & \beta_2^2 - \mu_3 & \gamma_2^1 - \eta_3 & \gamma_2^2 - \eta_3 \\ \alpha_3^1 & \alpha_3^2 & \beta_3^1 - \mu_4 & \beta_3^2 - \mu_4 & \gamma_3^1 - \eta_4 & \gamma_3^2 - \eta_4 \end{bmatrix} \tag{3.5.23}$$

It follows from (3.5.7),(3.5.9), (3.5.13) and (3.5.14) that

$$\begin{bmatrix} p_1 & p_1' & p_4 & p_4' & q_4 & q_4' & p_2 & p_3 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} = 0 \tag{3.5.24}$$

Hence the rank of the matrix $\begin{bmatrix} A \\ B \end{bmatrix}$ is at most four, that is, the matrix A is singular. Since $\beta_1^1 \neq 0, \beta_1^2 \neq 0$, the first two rows and the last two rows of A are independent. That is, matrix A has rank four. Hence the system (3.5.15)–(3.5.17) has four independent equations. Now we show that if (p_1, p_1', p_2, p_3) is affine independent, then the sub-matrices A_1 and A_2 are nonsingular, where

$$A_1 = \begin{bmatrix} \beta_4^1 - \mu_1 & -\mu_1 \\ -\mu_2 & \beta_4^2 - \mu_2 \end{bmatrix}, \quad A_2 = \begin{bmatrix} \gamma_4^1 - \eta_1 & -\eta_1 \\ -\eta_2 & \gamma_4^2 - \eta_2 \end{bmatrix} \tag{3.5.25}$$

are the sub-matrices of A and they are the coefficient matrices of the equations (3.5.16) and (3.5.17) respectively. This implies that (3.5.16) and (3.5.17) are four independent equations for unknowns a^i_{0111} and $c^i_{0111}, i = 1, 2$. In fact, the affine independency of (p_1, p_1', p_2, p_3) is the necessary and sufficient condition for the nonsingularity of A_1

and A_2. It follows from (3.5.7),(3.5.9) that

$$\begin{bmatrix} p_4 & p_4' \\ 1 & 1 \end{bmatrix} A_1 = - \begin{bmatrix} p_1 & p_1' & p_2 & p_3 \\ 1 & 1 & 1 & 1 \end{bmatrix} B_1, \quad B_1 = \begin{bmatrix} \beta_1^1 & 0 \\ 0 & \beta_1^2 \\ \beta_2^1 - \mu_3 & \beta_2^2 - \mu_3 \\ \beta_3^1 - \mu_4 & \beta_3^2 - \mu_4 \end{bmatrix} \tag{3.5.26}$$

Since $p_4 \neq p_4'$ and $\beta_1^1 \neq 0, \beta_1^2 \neq 0$, matrix A_1 is of full rank if the matrix

$$\begin{bmatrix} p_1 & p_1' & p_2 & p_3 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

is nonsingular. On the other hand, if this matrix is singular, that is p_1, p_1', p_2, p_3 are coplanar, then the matrix A_1 is also singular. Otherwise, p_4, p_4' will lie on the the plane $\langle p_1 p_1' p_2 p_3 \rangle$ by (3.5.26), which yields a contradiction. Similarly, A_2 is nonsingular iff

$$\begin{bmatrix} p_1 & p_1' & p_2 & p_3 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

is nonsingular.

(ii). (a). If p_1, p_1', p_2, p_3 are affine independent, then by (3.5.26) we know that $a_{0111}^i, i = 1,2$ can be expressed as an affine combination of a_{1110}^i and a_{0210}^1, a_{0120}^1. By (3.5.15), $c_{0111}^i, i = 1,2$ can also be expressed as an affine combination of a_{1110}^i and a_{0210}^1, a_{0120}^1.

(b). If we take p_4, p_4', p_2, p_3 or q_4, q_4', p_2, p_3 to be the affine independent set, then the equations (3.5.15)-(3.5.17) are already in the form (3.5.21).

(c). Any other cases can be derived from one of the above cases. □

3.5.3 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 the normals point to the same side of the surface triangulation T. That is the side on which p_4 and p'_4 lie (see Figure 3.5.1). Under this assumption, it follows from Definition 4.1 and equation (3.5.6) that, the control points on the edge, say a^i_{0210}, a^i_{0120} on edge $[p_2p_3]$ (see Figure 3.5.1), are nonpositive if the edge is nonnegative convex, and nonnegative if the edge is nonpositive convex. Now we can divide all the control points into 7 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 0-th layer and on the same side as the normal direction, followed by the 2nd and 3rd layers. Next to the 0-th layer but opposite to the normal, is the -1st layer, then the -2nd and -3rd layers. Now we show that, we can set all the control points on the 2nd and 3rd layer to be positive and the control points on the -2nd and -3rd layers to be 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 (3.5.8) that the 2nd and 3rd layers control points can be positive only if the 2nd layer control points on the neighbor face-tetrahedra are large enough. This is achieved since β^i_4 in (3.5.8) is positive (see the proof of Proposition 3.5.2 for details). Similarly, the control points on the -2nd and -3rd layers can be chosen to be negative. Furthermore, all 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 $[p_2p_3]$ (see Figure 3.5.1), then since a^i_{1110} (or a^i_{0111}) is freely chosen, the smooth edge condition is easily satisfied (see the proof of Proposition 3.5.2). Referring to Figure 5.1, we prove in the following that the patches constructed over V_1 and W_1 are single sheeted. The other patches are similar.

Proposition 3.5.1 If the face $[p_1p_2p_3]$ is nonnegative convex, then the control points can be determined so that the surface over V_1 is a three-sided 4-patch.

Proof: Since the 0-th layer control points are nonpositive, and the second and third layers control points can be set as positive, the defined surface is then a three-sided 4-patch (see Theorem 3.2.1). □

Proposition 3.5.2 If the edge $[p_2p_3]$ is nonnegative convex, then the control points can be determined such that the surface over W_1 is a four-sided 14-23-patch.

Proof: First, the 0-th layer control points are nonpositive, Now we show that the second and third layer control points can be set as positive. Since p'' is above the planes $\langle p_1p_2p_3\rangle$ and $\langle p_1'p_2p_3\rangle$ (i.e., it is at the same side as p_4 of the planes), then $\beta_4^i > 0$. Hence from (5.4) and (5.5), b_{1101} and b_{1011} can be set as positive if a_{0102} and a_{0012} are chosen large enough. Similarly, by (5.3), $b_{1002} > 0$ if a_{0003} is chosen large enough. Also b_{2001} can be set as positive since it is free. Now it follows from (5.8)–(5.10) and $\mu_1 > 0$, $\mu_2 > 0$ that b_{2100}, b_{2010} and b_{3000} can be set as positive. Therefore the surface defined in this way is a four-sided 14-23-patch over W_1 if $[p_2p_3]$ is positive convex. If $[p_2p_3]$ is zero convex, that is $a_{0210}^i = a_{0120}^i = 0$, then by (3.5.16) we can make $a_{0111}^i > 0$ and $b_{1110}^i > 0$ by choosing the free parameter $a_{1110}^i, i = 1, 2$. Hence the 1st layer control points are nonnegative. Hence here the patch over W_1 degenerates to the edge $[p_2p_3]$ and the smooth edges condition is satisfied. However, if the parameter a_{1110}^i are over determined, then a subdivision as in the coplanar case is needed. □

Subdivision. For any face of $T = [p_1, p_2, p_3]$, if it is non-convex and if the three inner products of the face normal and its three adjacent face normals have different signs, then subdivide the double face tetrahedra into 6 subtetrahedra by adding a vertex at the center w of the face (a face Clough-Tocher split). The coefficients are specified as before by regarding w as p_1 (see Figure 3.5.1).

Proposition 3.5.3 If the above subdivision procedure is performed, then the control points can be chosen so that the surface over V_1 is a three-sided 4-patch, and the surface over W_1 is a four-sided 14-23-patch.

Proof: (i). $[p_1' p_2 p_3]$ is non-convex face (see Figure 3.5.1). We show that all the 1st layer's control points over V_i and $W_i, i = 1, 2$ can be set as nonpositive, and the -1st layer's control points over $V_i', i = 1, 2$ and $W_i', i = 1, 2$ can be set as nonpositive. If p_1, p_1', p_2, p_3 are affine independent, then we use the equalities (3.5.19) and (3.5.20). Since both p_4 and p_4' are at the same side of the surface triangulation T, $\theta_1^i \theta_4^i > 0$ for $i = 1, 2$. Assume, without loss of generality, that $\theta_1^1 > 0, \theta_4^1 > 0$ and $\theta_1^2 > 0, \theta_4^2 > 0$ and then $\vartheta_1^1 < 0, \vartheta_4^1 < 0$ and $\vartheta_1^2 < 0, \vartheta_4^2 < 0$. Then by (3.5.19) and (3.5.20), we can take a_{1110}^i large enough such that $a_{0111}^i > 0$ and $c_{0111}^i < 0$, and furthermore, their absolute value can be larger than any specified value. Since the 1st and -1st layer's control points that are determined by the normals are nonnegative and nonpositive, respectively, all the the 1st and -1st layer's control points can be set as nonnegative and nonpositive respectively. Therefore, the surfaces over V_i and V_i' are three-sided 4-patches, and the surfaces over W_i and W_i' are four-sided 14-23-patches (see Theorem 3.2.1 and 3.2.2). If p_1, p_1', p_2, p_3 are coplanar (not affine independent), then by Theorem 3.5.1, all the unknowns can be expressed linearly by $a_{0111}^i, i = 1, 2$ (or $c_{0111}^i, i = 1, 2$). It is easy to see that, we can take $a_{0111}^i > 0$ (or $c_{0111}^i < 0$) large (or small) enough so that $c_{0111}^i < 0$ (or $a_{0111}^i > 0$).

(ii). If $[p_1' p_2 p_3]$ is convex, then the edge $[p_2 p_3]$ is convex also. Then Propositions 3.5.1 and 3.5.2 can be used for this face and edge. As for the face $[p_1 p_2 p_3]$, the discussion above can be used.

Finally, we point out why the splitting is necessary. Consider the face $[p_1 p_2 p_3]$ as an example (see Figure 3.5.1). In order to have $a_{0111}^1, a_{1101}^1, a_{1011}^1$ greater than zero, a_{1110}^1 has to be determined three times by the three C^1 constraints if no splitting is performed. Therefore, in general a solution is impossible without splitting. Also note that, if the three inner products between the face normal and its neighbor's face normals have the same sign (positive or negative), then a_{1110}^1 can be determined so that $a_{0111}^1, a_{1101}^1, a_{1011}^1$ are greater than zero. Hence here we do not need to split the face. $\square$

A three-sided (or four-sided) patch, although by itself may be disconnected in the case of a nonconvex face, it forms a connected piece of surface with the other three-sided (or four-sided) patch of the double tetrahedra.

Theorem 3.5.2 The global piecewise surface constructed is smooth, connected and single-sheeted.

Proof: We consider as the general case double face tetrahedra and double edge tetrahedra.

(Smoothness) The whole surface is smooth as each single piece is smooth.

(Connectedness and Single-sheetedness) Let $S = [p_1p_2p_3p_4]$ and $S' = [p_1p_2p_3q_4]$ be two face tetrahedra sharing face S_4, S_F and $S'_{F'}$ be the two three-sided 4-patches over them. From the single sheeted construction, $F(e_4) < 0$ and $F'(e_4) > 0$ and from the C_0 conditions, for $\alpha^* \in S_4$, $F(\alpha^*) = F'(\alpha^*)$. From the A-patch Property *(d)* in section 3, a polyline $p_4\alpha^*q_4$ intersects the "double" patch $D = S_F \wedge S'_{F'}$ exactly once. Hence D is single-sheeted over the double tetrahedra. In particular, $p_4\alpha^*q_4$ intersects $S'_{F'}$ when $F(\alpha^*) > 0$, S_F when $F(\alpha^*) < 0$, and both S_F and $S'_{F'}$ at α^* when $F(\alpha^*) = 0$ where S_F and $S'_{F'}$ meet. Regard the "double" surface D as a function of α^* denoted as $D(\alpha^*)$. From the fact that the two patches are smooth and at least C^0 to each other,

$$\lim_{\theta^* \to \alpha^*, \theta \in S_4} D(\theta^*) = D(\alpha^*) \tag{3.5.27}$$

Hence D is connected over the double tetrahedra $S \wedge S'$.

Similarly, let $S = [p'_1p_2p_3p_4]$ and $S' = [q'_1p_2p_3q_4]$ be two edge tetrahedra sharing edge $[p_2p_3]$, and S_F, $S'_{F'}$ be the two four sided 14-23-patches over them. From the single sheeted construction, $F(\beta^*) < 0$ for $\beta^* \in [p'_1p_4]$ and $F'(\gamma^*) > 0$ for $\gamma^* \in [q'_1q_4]$. From the C^0 conditions, $F(\alpha^*) = F'(\alpha^*)$, for $\alpha^* \in [p_2p_3]$. Again from Property *(e)*, polyline $\beta^*\alpha^*\gamma^*$ intersects the double patch $D = S_F \wedge S'_{F'}$ exactly once. Hence D is single-sheeted over the double tetrahedra. In particular, $\beta^*\alpha^*\gamma^*$ intersects $S'_{F'}$ when $F(\alpha^*) > 0$, S_F when $F(\alpha^*) < 0$, and both S_F and $S'_{F'}$ at α when $F(\alpha^*) = 0$ where S_F and $S'_{F'}$ meet. Regard D as a function of (α, β), denoted as $D(\alpha, \beta)$, where $(\alpha, 1-\alpha)$

is the barycentric coordinate of α^* and $(\beta, 1-\beta)$ is the barycentric coordinate of β^* or γ^*. From the facts the two patches are smooth and at least C^0 to each other, we have,

$$\lim_{\theta \to \alpha, \phi \to \beta} D(\theta, \phi) = D(\alpha, \beta) \tag{3.5.28}$$

Hence D is connected over the double tetrahedra $S \wedge S'$. □

With theorem 3.5.2, we conclude that the surface is topologically equivalent to the input triangulation.

3.6 Shape Control

From the discussion of §5, there are several parameters that can influence the shape of the constructed C^1 surface. These parameters include (a) the length of the normal if its orientation is fixed, (b) a^i_{1110}, and (c) $a^i_{0102} > 0$, $a^i_{1002} > 0$ $a^i_{0012} > 0$, $a^i_{0003} > 0$ and $b^i_{2001} > 0$ for $i = 1, 2$.

(a). Interactive Shape Control

The influence of the length of a normal at a vertex is as follows: if the normal becomes longer then the surface becomes flatter at this point. Parameter a_{1110} lifts the surface upwards to the top vertex of the tetrahedron, while others push the surface downwards toward the bottom of the tetrahedron. In order to get a desirable surface, one may specify some additional data points in the tetrahedron considered, then approximate these points in the least square sense.

(b). Default Shape Control

Here we only consider the effect of the free parameters, that is, suppose the normals are fixed. The aim of the default choice of these parameters is to avoid producing bumpy surfaces. The commonly used method is to keep the surface patch close to a quadric patch [5, 48].

By least squares approximation of the coefficients of a quadric [48], one can derive that

$$a_{1110} = \frac{1}{4}(a_{1200} + a_{2100} + a_{2010} + a_{1020} + a_{0210} + a_{0120}) \tag{3.6.1}$$

Using the same idea, the other parameters can also be determined. For example, a_λ for $\lambda_4 > 1$ can be determined by the degree elevation formula

$$a_\lambda = \frac{1}{3}\sum_{i=1}^{4} \lambda_i x_{\lambda - e_i}, \quad |\lambda| = 3, \quad \lambda_4 > 1 \tag{3.6.2}$$

where $x_{\lambda - e_i}$ is the solution of the following equations in the least squares sense

$$a_\lambda = \frac{1}{3}\sum_{i=1}^{4} \lambda_i x_{\lambda - e_i}, \quad |\lambda| = 3, \quad \lambda_4 = 0, 1 \tag{3.6.3}$$

In the same way, b_{2001} can be determined. Therefore, under the C^1 conditions, we can define two sets of control points $\{a_\lambda^s\}$ and $\{a_\lambda^q\}$ over V_1, where $\{a_\lambda^s\}$ is yielded from the single sheeted consideration (see Propositions 3.5.1-3.5.3), and $\{a_\lambda^q\}$ comes from approximating a simple (quadratic) surface. Note that the surface defined by $\{a_\lambda^s\}$ above may not be desirable in shape, while the surface defined by $\{a_\lambda^q\}$ above may not be single sheeted. In our implementation we take a finite sequence $0 = t_0 < t_1 < \cdots < t_m = 1$ and consider $\{a_\lambda^{(i)}\} = \{(1 - t_i)a_\lambda^q + t_i a_\lambda^s\}, \quad i = 0, 1, \cdots, m$ selecting the single sheeted surface defined by $\{a_\lambda^{(i)}\}$ for smallest index i. Experiments show that this approach works well and a desirable surface is obtained with $t_i < 0.5$. Examples are shown in Figure 3.7.2.

3.7 Examples

Examples of the interpolant simplicial hull construction and C^1 smoothed triangulations using cubic A-patches are shown in Figures 3.7.1, 3.7.2, 3.7.3, 3.7.4 and 3.7.5. Note in Figure 3.7.4 how the "convex" faces are smoothed by a single cubic A-patch per face, while a face Clough-Tocher splitting occurs for co-planar faces and some "non-convex" faces, as determined by the vertex normals assignment and the adjacent faces.

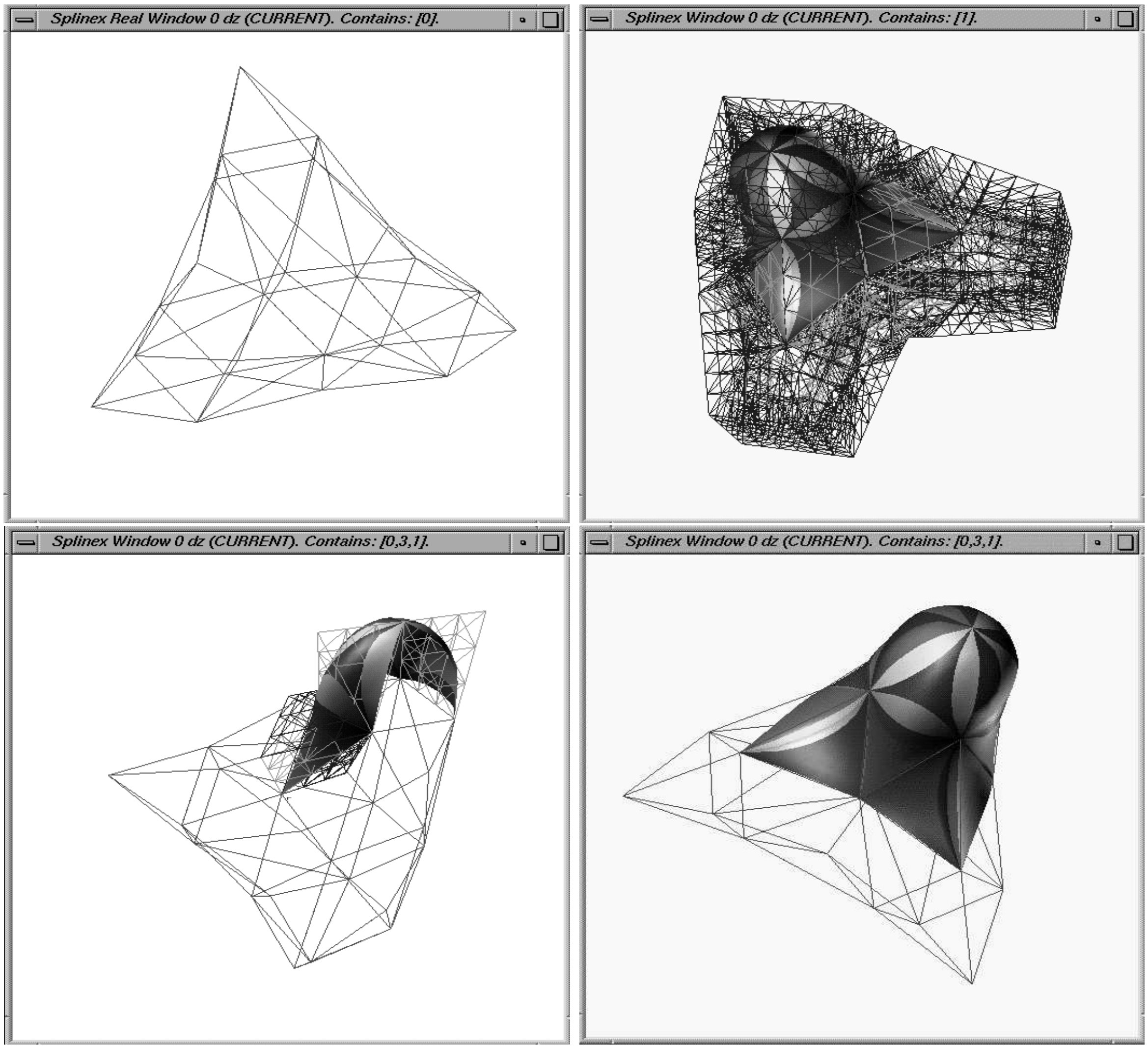


Figure 3.7.1 A surface triangulation, the interpolant simplicial hull and some of the interpolatory C^1 cubic A-patches

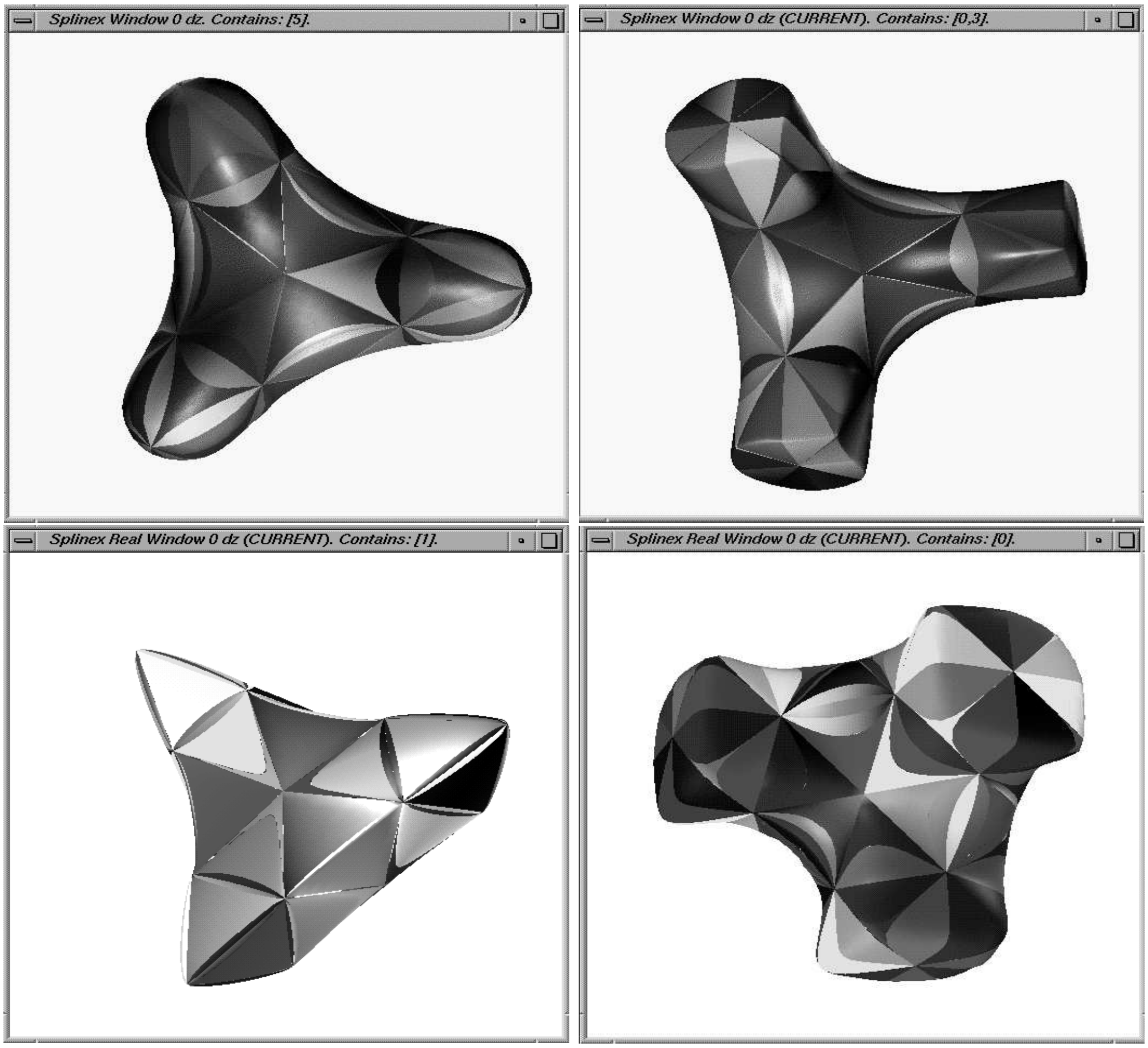


Figure 3.7.2 Different smoothings of the surface triangulation using C^1 cubic A-patches

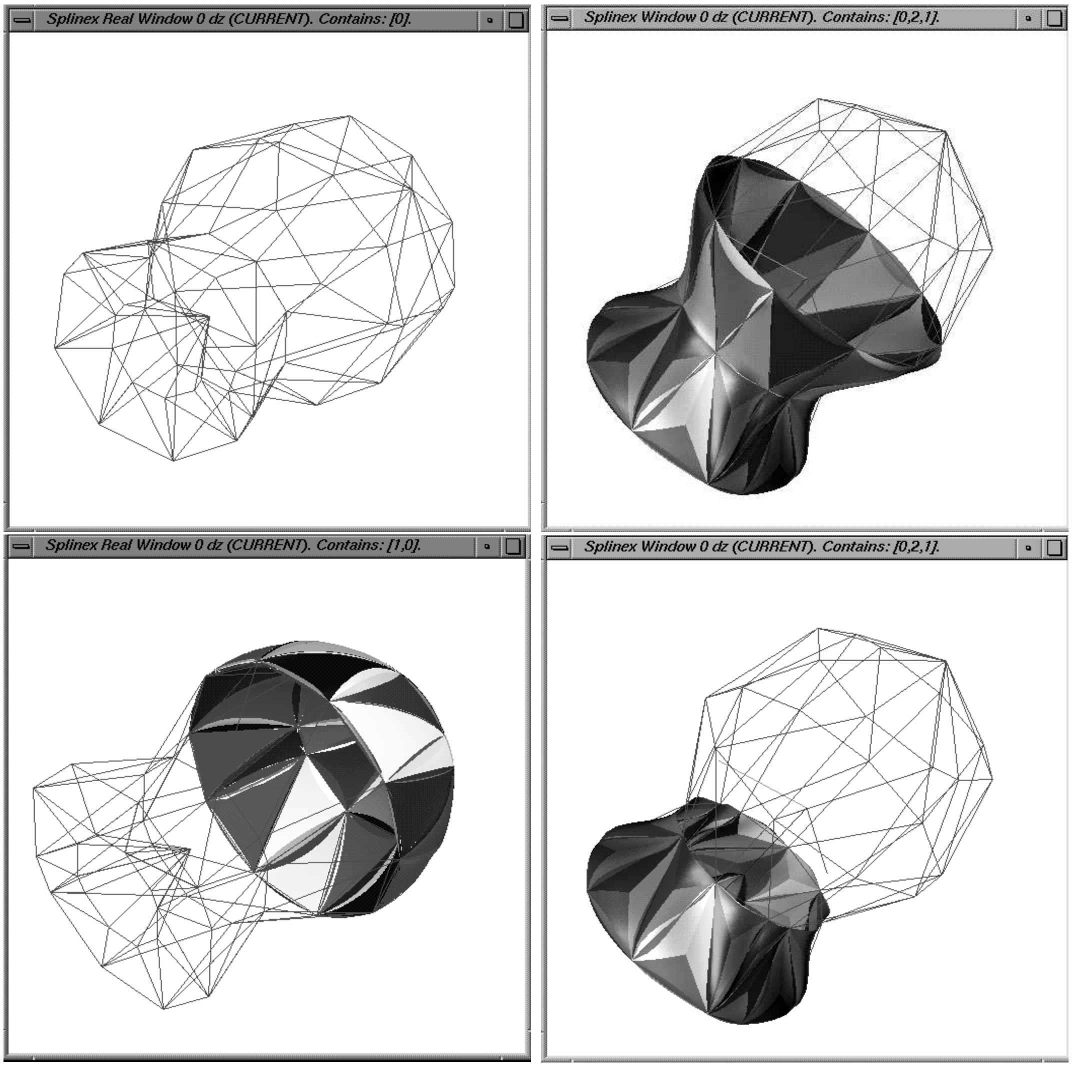


Figure 3.7.3 A surface triangulation and some of the interpolatory C^1 cubic A-patches

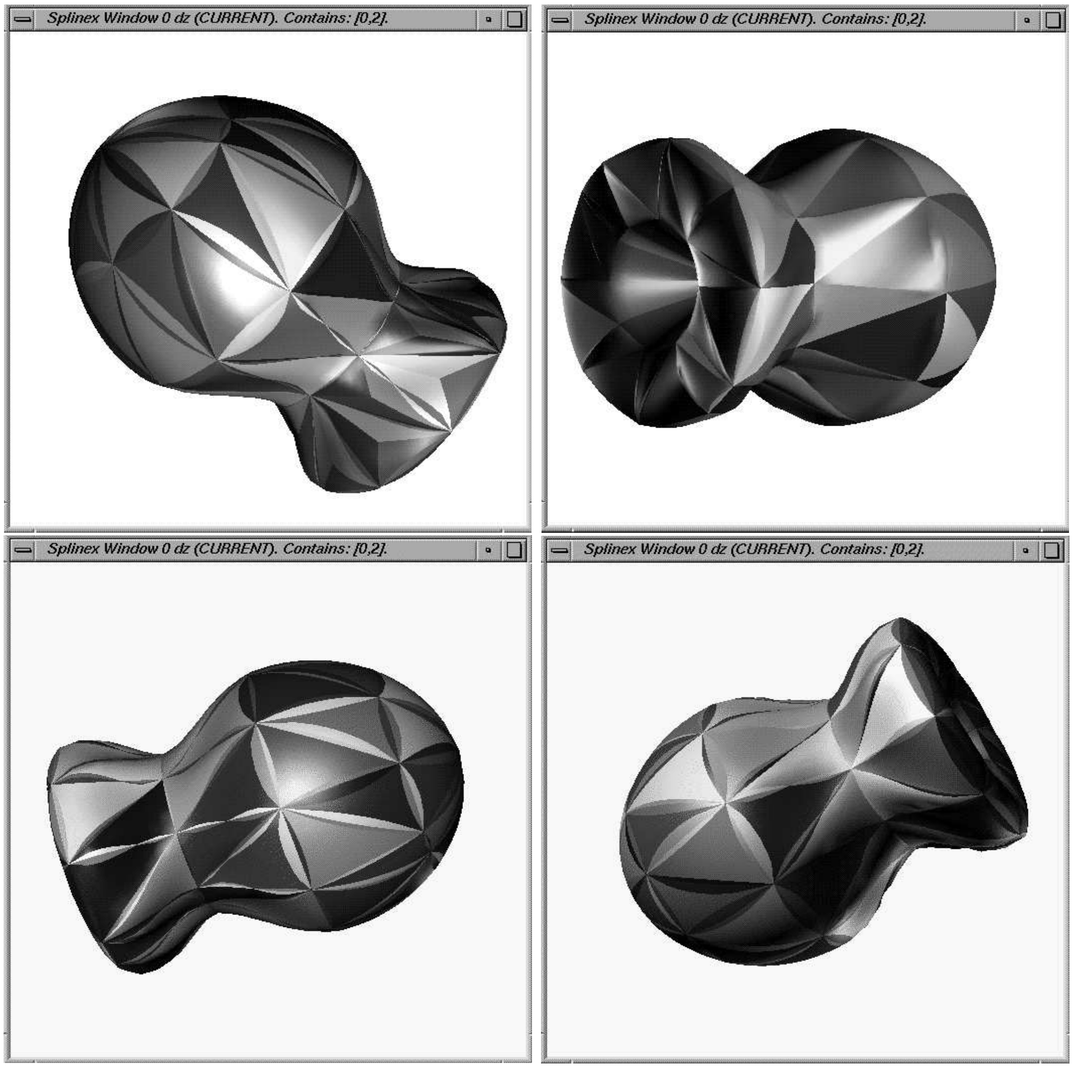


Figure 3.7.4 The complete smoothing of the surface triangulation using C^1 cubic A-patches

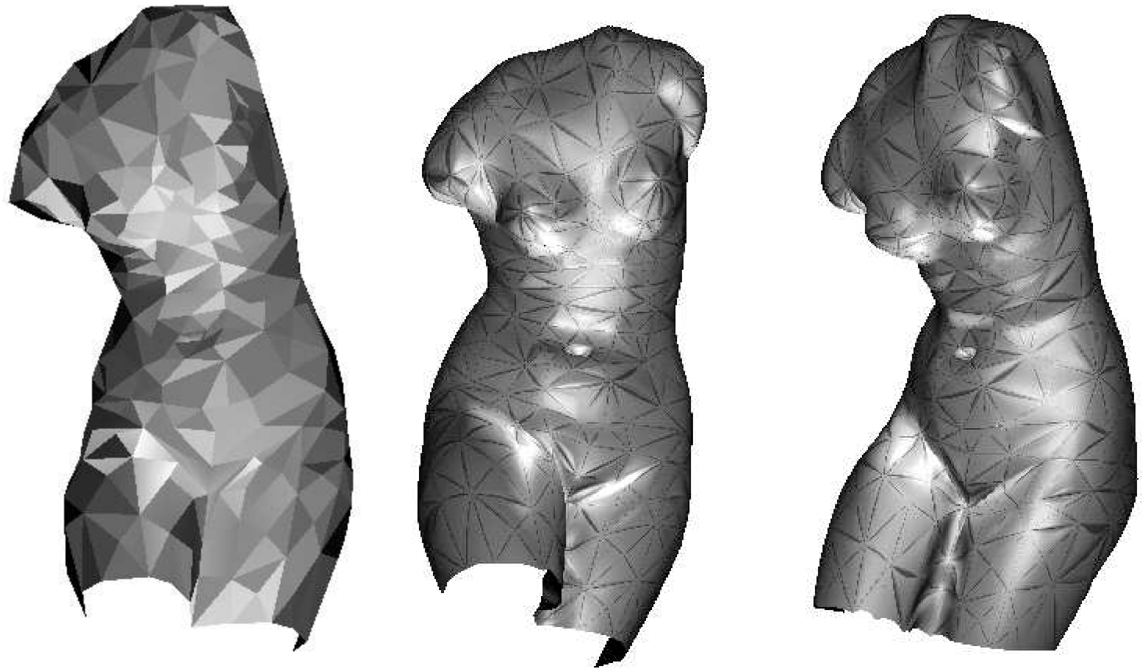


Figure 3.7.5 Venus. (a) Input polyhedral. (b) and (c) smooth shape from different viewpoints.

4. C^2 A-PATCHES

4.1 Introduction

We present a scheme for building a C^2 patch complex with quintic surface patches[1]. Like the C^1 scheme in Chapter 3, a pair of tetrahedra are built on each side of each face of a triangulated polyhedron $\mathcal{P}$. We call them the face tetrahedra. A pair of tetrahedra are built to fill in the gap between each pair of face tetrahedra that share an edge and on the same side of $\mathcal{P}$. A quintic BB polynomial is defined within each tetrahedron so that they are C^2 continuous across their share boundaries. The zero contour of the piecewise C^2 polynomial C^2-interpolates the vertices of $\mathcal{P}$. We guarantee that the zero contour surface is non-singular and topologically equivalent to $\mathcal{P}$. We give sufficient conditions to guarantee that the implicit surface are A-patches. However, the one-signed changed condition, which is used in the C^1 scheme, turns out to be too constraining, and is thus relaxed by subdivisions for the C^2 case.

The rest of this chapter is as follows. Section 4.2 presents details of the C^2 continuity schemes for quintic A-patches. Section 4.3 discuss a set of sufficient conditions which guarantee that the zero contour surface is functional. Section 4.4 provides some implementation details. And finally, Section 4.5 concludes the chapter.

4.2 C^2 Mesh of Quintic Patches

The input of our algorithm is a triangulated polyhedron $\mathcal{P}$, with optional C^2 data at each vertex. C^2 data means that each vertex p is given a gradient vector $\Delta f(p)$ and a Hessian matrix $\Delta^2 f(p)$. The gradient vector and/or Hessian matrix can be specified by a plane or a conic surface $F_p(x, y, z) = 0$ goes through p,

[1]The material covered in this Chapter is mostly in common with [14].

For each vertex that has incomplete C^2 data, a preprocessing step computes reasonable ones by fitting a conic surface to the vertex and its neighboring vertices.

We then construct an interpolant simplicial hull $\sum$ as in [17], except that we impose a stronger tangent containment condition: For a tetrahedral simplex $[p_1p_2p_3p_4]$ built on polyhedral face $[p_1p_2p_3]$,

$$|N_{p_1} \cdot [p_1p_4]| > \zeta, \tag{4.2.1}$$

where N_{p_1} is the normal at p_1 and ζ is a given small positive constant. We will show later that we need this stronger condition to guarantee that piecewise surface is connected within Σ. This stronger condition can usually be enforced by raising p_4.

Once we have established a simplicial hull $\sum$ for the given polyhedron $\mathcal{P}$ and a set of point normals N, we construct a C^2 trivariate piecewise polynomial function F within $\sum$ such that F has the given C^2 data at each vertex. We then further set the free polynomial coefficients so that they are C^2 continuous to each other. We adapted the C^1 and functional cubic scheme from [17] to quintic piecewise polynomial. Please refer to it for a full description of the C^1 scheme.

Figure 4.2.1 illustrates between the weights of neighboring tetrahedral simplex in Σ.

$$\begin{array}{llllllll}
V_1 & = & [p_1p_2p_3p_4], & V_2 & = & [p_1'p_2p_3p_4'], & W_1 & = & [p_1''p_2p_3p_4], & W_2 & = & [p_1''p_2p_3p_4'], \\
V_1' & = & [p_1p_2p_3q_4], & V_2' & = & [p_1'p_2p_3q_4'], & W_1' & = & [q_1''p_2p_3p_4], & W_2' & = & [q_1''p_2p_3p_4'],
\end{array}$$

and the polynomials f_i over V_i, g_i over W_i, f_i' over V_i' and g_i' over W_i' be expressed in Bernstein-Bezier forms with coefficients a_λ^i, b_λ^i and c_λ^i, respectively. Now we shall determine these coefficients step by step. Denote

$$\begin{array}{lllll}
p_1'' & = & \beta_1^1 p_1 + \beta_2^1 p_2 + \beta_3^1 p_3 + \beta_4^1 p_4, & \beta_1^1 + \beta_2^1 + \beta_3^1 + \beta_4^1 = 1 \\
p_1'' & = & \beta_1^2 p_1' + \beta_2^2 p_2 + \beta_3^2 p_3 + \beta_4^2 p_4', & \beta_1^2 + \beta_2^2 + \beta_3^2 + \beta_4^2 = 1 \\
p_1'' & = & \mu_1 p_4 + \mu_2 p_4', & \mu_1 + \mu_2 = 1 \\
q_1'' & = & \gamma_1 q_4 + \gamma_2 q_4', & \gamma_1 + \gamma_2 = 1
\end{array} \tag{4.2.2}$$

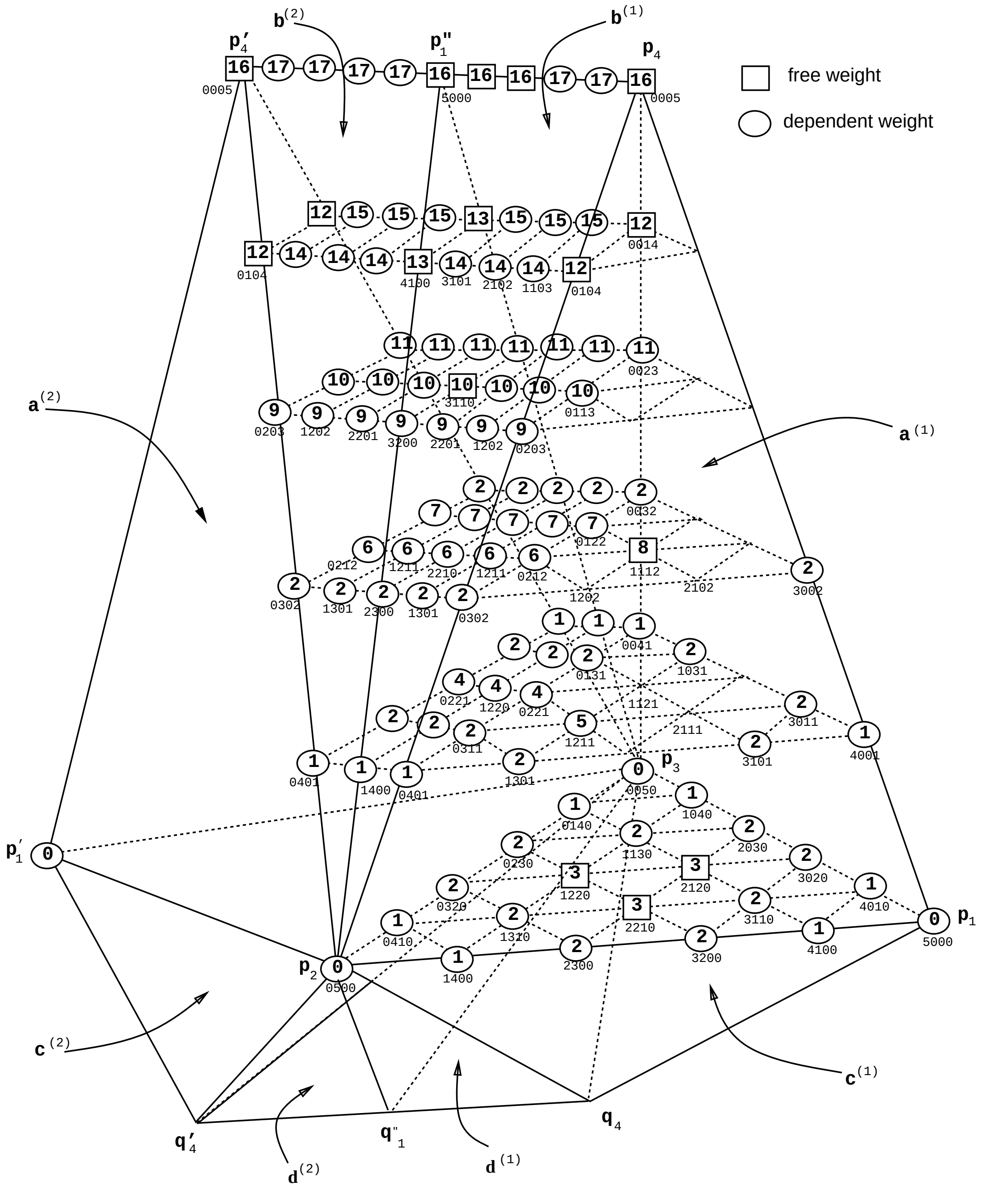


Figure 4.2.1 Adjacent tetrahedra, quintic functions and control points for two non-convex adjacent faces

Algorithm 4.2.1 C^2 quintic scheme (see Figure 4.2.1)

1. The number 0 weights are given by the function values at the vertices. For examples, $a^{(1)}_{5e_i} = f(p_i),\ i = 1, 2, 3$.

2. The number 1 weights are determined by formula (2.1.6).

3. The number 2 weights are determined by formula (2.1.7).

4. The number 3 weights, that is $a^{(i)}_{1220}, a^{(i)}_{2210}$ and $a^{(i)}_{2120}$, are free.

5. The number 4 weights are determined by C^1 condition (2.1.10).

6. The number 5 and 6 weights have to be determined simultaneously. In determining these weights, we need to consider all the C^1 and C^2 conditions related to the tetrahedra surrounding the vertex p_2. Suppose there are k triangles (hence k edges) around p_2.

   Along edge $[p_2p_3]$, there are a "clique" of 8 sets of weights with respect of the 8 tetrahedra around $[p_2p_3]$, $W^{(i)}_{2210}$, $W^{(i)}_{1310}$, $W^{(i)}_{0410}$, $W^{(i)}_{1220}$, $W^{(i)}_{0320}$, $W^{(i)}_{0230}$, $W^{(i)}_{0311}$, $W^{(i)}_{1211}$, $W^{(i)}_{0221}$, $W^{(i)}_{0212}$, $i = 1, 2$, $W = a, b, c, d$. Actually each of the 8 sets is just the same quadratic polynomial in a different BB-form with respect to a different basis. Further studying their relations, we figure that there is a linear relation between $a^{(1)}_{1211}$ and $a^{(2)}_{1211}$, regardless of the number 6 weights or weights $c^{(i)}_{1211}$. However, a number 5 weight, for example a^{1}_{1211}, is shared by two "cliques" around $[p_2p_3]$ and $[p_2p_1]$. Therefore, around vertex p_2, there is a $k \times k$ linear system. One can show that this system is of rank $k-1$ and hence always solvable. $a^{(i)}_{1211}$ are set by solving the system. The other number 5 weights and number 6 weights are set by C^2 condition. Please note that the symmetric linear system with respect to $c^{(i)}_{1211}$ on the other side is trivially satisfied by the solution of $a^{(i)}_{1211}$. See 4.2.1 for further details.

7. The number 7 weights are similarly determined as that of number 6 weights.

8. The number 8 weight $a^{(i)}_{1112}$ are free.

9. The number 9 weights are determined by C^1 and C^2 conditions. Both the number of equations and the number of unknowns are $6k$. See 4.2.2 for details.

10. For the number 10 weights, we have six equations parallel to the equations (4.2.15)–(3.5.16) with all the index changed by the rule

$$\text{the index of the number 10 weight} = \text{the index of the number } 9 - e_2 + e_3 \tag{4.2.3}$$

and seven independent weights. By choosing one of them, say $b^{(i)}_{3110}$, to be a free parameter, the entire system can be solved.

11. The number 11 weights are determined in the same way as the that of number 9 weights.

12. The number 12 and 13 weights are free. The number 14 weights are determined by C^1 and C^2 conditions. That is $b^{(i)}_{1103}$ are defined by (2.1.10), and $b^{(i)}_{2102}$ are defined by (2.1.11). For $b^{(i)}_{3101}$, we have by (2.1.12) and (2.1.13)

$$\mu_1 b^{(1)}_{3101} + \mu_2 b^{(2)}_{3101} = b^{(1)}_{4100}$$

$$-\mu_1 b^{(1)}_{3101} + \mu_2 b^{(2)}_{3101} = \mu_2^2 b^{(2)}_{2102} - \mu_1^2 b^{(1)}_{2102}$$

Hence

$$b^{(1)}_{3101} = \frac{b^{(1)}_{4100} - \mu_2^2 b^{(2)}_{2102} + \mu_1^2 b^{(1)}_{2102}}{2\mu_1}$$

$$b^{(2)}_{3101} = \frac{b^{(1)}_{4100} + \mu_2^2 b^{(2)}_{2102} - \mu_1^2 b^{(1)}_{2102}}{2\mu_2}$$

13. The number 15 weights are similar to that of number 14, the index is changed by the rule (4.2.3).

14. The number 16 weights are free, the number 17's are determined by C^1 and C^2 conditions.

15. The remaining weights with index $\lambda_1\lambda_2\lambda_3\lambda_4$ are determined by C^1 and C^2 conditions (2.1.10) and (2.1.11) for $\lambda_4 \le 2$ and freely chosen for $\lambda_4 > 2$

In summary, the construction steps 1–14 are according to the C^1 and C^2 conditions across the common tetrahedra faces that are over or below the original triangulation. Step 15 is according to the C^1 and C^2 conditions across the common tetrahedra faces that are on the original triangulation. Therefore, the composite polynomial function is global C^2 continuous.

In the algorithm, there are quite some free weights need to be set. whatever values they are assigned to have impacts on the shape of the surface. Instead of an optimization analysis, we proposed two kinds of cheap heuristics. The first method is the following. We first construct a C^1 A-patch surface with cubic patches. After we make sure that the surface looks good, we degree-raised the whole simplicial hull to quintic patches. Now whatever weights a quintic patch has will be used as default values if needed. The second method is to always set the free weight so that the polynomial approximate a polynomial of lower degree. We observe that the first method works better.

4.2.1 Determining Number 5 and 6 Weights of the Quintic Scheme

It follows from (2.1.10) and (2.1.11) that

$$b_{1211}^{(i)} = \beta_1^{(i)} a_{1211}^{(i)} + \beta_2^{(i)} a_{0311}^{(i)} + \beta_3^{(i)} a_{0221}^{(i)} + \beta_4^{(i)} a_{0212}^{(i)} \tag{4.2.4}$$

$$\begin{aligned} b_{2210}^{(i)} &= \beta_1^{(i)}\beta_1^{(i)} a_{2210}^{(i)} + 2\beta_1^{(i)}\beta_2^{(i)} a_{1310}^{(i)} + 2\beta_1^{(i)}\beta_3^{(i)} a_{1220}^{(i)} + 2\beta_1^{(i)}\beta_4^{(i)} a_{1211}^{(i)} \\ &+ \beta_2^{(i)}\beta_2^{(i)} a_{0410}^{(i)} + 2\beta_2^{(i)}\beta_3^{(i)} a_{0320}^{(i)} + 2\beta_2^{(i)}\beta_4^{(i)} a_{0311}^{(i)} \\ &+ \beta_3^{(i)}\beta_3^{(i)} a_{0230}^{(i)} + 2\beta_3^{(i)}\beta_4^{(i)} a_{0221}^{(i)} + \beta_4^{(i)}\beta_4^{(i)} a_{0212}^{(i)} \end{aligned} \tag{4.2.5}$$

for $i = 1, 2$. (4.2.5) can be written briefly as

$$b_{2210}^{(i)} = 2\beta_1^{(i)}\beta_4^{(i)} a_{1211}^{(i)} + \beta_4^{(i)}\beta_4^{(i)} a_{0212}^{(i)} + \gamma \tag{4.2.6}$$

where γ is the known terms in (4.2.5). Since

$$b_{2210}^{(1)} = \mu_1 b_{1211}^{(1)} + \mu_2 b_{1211}^{(2)} \tag{4.2.7}$$

$$\mu_1^2 b_{0212}^{(1)} - \mu_1 b_{1211}^{(1)} = \mu_2^2 b_{0212}^{(2)} - \mu_2 b_{1211}^{(2)} \tag{4.2.8}$$

then by substituting (4.2.4) into (4.2.7) and (4.2.8) and then eliminating $b^{(i)}_{2210}$ from (4.2.6) and (4.2.7) we get three equations related to four unknowns which could be written as:

$$\begin{bmatrix} \beta_4^{(1)} - \mu_1 & -\mu_2 \\ -\mu_1 & \beta_4^{(2)} - \mu_2 \end{bmatrix} \begin{bmatrix} \beta_4^{(1)} & 0 \\ 0 & \beta_4^{(2)} \end{bmatrix} \begin{bmatrix} a^{(1)}_{0212} \\ a^{(2)}_{0212} \end{bmatrix} = \\ - \begin{bmatrix} 2\beta_4^{(1)} - \mu_1 & -\mu_2 \\ -\mu_1 & 2\beta_4^{(2)} - \mu_2 \end{bmatrix} \begin{bmatrix} a^{(1)}_{1211} \\ a^{(2)}_{1211} \end{bmatrix} + \cdots \tag{4.2.9}$$

$$\left[-\mu_1(\beta_4^{(1)} - \mu_1) \quad \mu_2(\beta_4^{(2)} - \mu_2)\right] \begin{bmatrix} a^{(1)}_{0212} \\ a^{(2)}_{0212} \end{bmatrix} - \left[\mu_1\beta_1^{(1)}, -\mu_2\beta_1^{(2)}\right] \begin{bmatrix} a^{(1)}_{1211} \\ a^{(2)}_{1211} \end{bmatrix} = \cdots \tag{4.2.10}$$

where $\cdots$ are known terms. Since the coefficient matrix of (4.2.9) is nonsingular, by solving $[a^{(1)}_{0212}\ a^{(2)}_{0212}]^T$ from (4.2.9) and then substituting it into (4.2.10), we get one equation relating to the unknowns $a^{(1)}_{1211}$, $a^{(2)}_{1211}$. Let the equation be in the form

$$\phi a^{(1)}_{1211} + \psi a^{(2)}_{1211} = \omega \tag{4.2.11}$$

Therefore, these unknowns form a closed chain around the vertex p_2 in one side of the tangent plane at p_2.

Let

$$A = \begin{bmatrix} \phi_1 & \psi_1 & & \\ & \phi_2 & \psi_2 & \\ & & \ddots & \\ \psi_k & & & \phi_k \end{bmatrix} \tag{4.2.12}$$

$$\varpi = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \ddots \\ \omega_k \end{bmatrix} \tag{4.2.13}$$

We observe that system

$$Ax = \varpi \tag{4.2.14}$$

is of rank $k-1$. In other words, it has infinite number of solutions. We have not found a proof of this conjecture yet. We design the algorithm so that, whenever the conjecture is true, we choose the solution that least-squared approximates the default values.

4.2.2 Determining Number 9 Weights of the Quintic Scheme

For $i = 1, 2,$

$$b^{(i)}_{1202} = \beta^{(i)}_1 a^{(i)}_{1202} + \beta^{(i)}_2 a^{(i)}_{0302} + \beta^{(i)}_3 a^{(i)}_{0212} + \beta^{(i)}_4 a^{(i)}_{0203} \tag{4.2.15}$$

$$\begin{aligned} b^{(i)}_{2201} &= \beta^{(i)}_1 \beta^{(i)}_1 a^{(i)}_{2201} + 2\beta^{(i)}_1 \beta^{(i)}_2 a^{(i)}_{1301} + 2\beta^{(i)}_1 \beta^{(i)}_3 a^{(i)}_{1211} + 2\beta^{(i)}_1 \beta^{(i)}_4 a^{(i)}_{1202} \\ &+ \beta^{(i)}_2 \beta^{(i)}_2 a^{(i)}_{0401} + 2\beta^{(i)}_2 \beta^{(i)}_3 a^{(i)}_{0311} + 2\beta^{(i)}_2 \beta^{(i)}_4 a^{(i)}_{0302} \\ &+ \beta^{(i)}_3 \beta^{(i)}_3 a^{(i)}_{0221} + 2\beta^{(i)}_3 \beta^{(i)}_4 a^{(i)}_{0212} + \beta^{(i)}_4 \beta^{(i)}_4 a^{(i)}_{0203} \end{aligned} \tag{4.2.16}$$

and

$$b^{(1)}_{3200} = \mu_1 b^{(1)}_{2201} + \mu_2 b^{(2)}_{2201} \tag{4.2.17}$$

$$\mu_1^2 b^{(1)}_{1202} - \mu_1 b^{(1)}_{2201} = \mu_2^2 b^{(2)}_{1202} - \mu_2 b^{(2)}_{2201} \tag{4.2.18}$$

Substitute (4.2.15) and (4.2.16) into (4.2.18), we have

$$\mu_1 \beta^{(1)}_4 (\mu_1 - \beta^{(1)}_4) b^{(1)}_{0203} - \mu_2 \beta^{(2)}_4 (\mu_2 - \beta^{(2)}_4) b^{(2)}_{0203} = \cdots$$

This is a system of the form

$$A = \begin{bmatrix} \phi_1 & \psi_1 & & \\ & \phi_2 & \psi_2 & \\ & & \ddots & \\ \psi_k & & & \phi_k \end{bmatrix}$$

whose determinant is $\prod_{i=1}^{k+r} \phi_i - (-1)^{k+r} \prod_{i=1}^{k+r} \psi_i$. This matrix is nonsingular in general if the points given are in the general position. Hence the system can be solved.

However, if the surrounding tetrahedra at the same side of p_2 are not closed, the matrix A is in the form of

$$A = \begin{bmatrix} \phi_1 & \psi_1 & & \\ & \ddots & \ddots & \\ & & \phi_k & \psi_k \end{bmatrix}$$

By choosing one unknown, say the l-th to be a free parameter, A can be written as

$$A = \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix}$$

with

$$A_1 = \begin{bmatrix} \phi_1 & \psi_1 & & \\ & \ddots & \ddots & \\ & & \phi_{l-1} & \psi_{l-1} \\ & & & \phi_l \end{bmatrix}, \quad A_2 = \begin{bmatrix} \psi_{l+1} & & & \\ \phi_{l+2} & \psi_{l+2} & & \\ & \ddots & \ddots & \\ & & \phi_k & \psi_k \end{bmatrix}$$

Hence the system of equations breakup into two smaller sub-systems. Each of them can be solved separately.

We identify two kinds of degeneracies as follows.

(1) $\beta_4^{(1)} = \beta_4^{(2)}$, but not equal to zero.

μ_1 and μ_2 need to be set to some values other than 0.5 so that A is not singular.

(2) $\beta_4^{(1)}$ and $\beta_4^{(2)}$ are all zero.

In this case, changing μ_1, μ_2 would not solve the problem. We perturbed the tops of the face simplexes to avoid singularity in A.

4.3 Functional Zero Contouring Quintics

This section gives a set of sufficient conditions to ensure that each quintic is an A-patch. Namely, besides C^2 continuous between adjacent tetrahedra, the zero contouring surface is singly-connected, non-singular and topologically equivalent to $\mathcal{T}$.

In the C^1 cubic scheme, one is able to arrange the weights of the BB polynomial so that there is a layer of weights that separates the nonpositive weights and nonnegative. Such an arrangement turns out to be a simple sufficient condition for an A-patch. However, in the C^2 quintic scheme, after meeting the C^2 conditions, the signs of the weights could alternate through the the layers. In particular, the weights at the 1st, 2nd and 3rd layers can be of any sign. We show in the following text that, one can arrange the 4th, 5th layers to be nonnegative and large enough in absolute value, so that the zero contouring surface is an A-patch.

We first discuss some sufficient conditions to guarantee that a BB polynomial is positive in its domain. Then we discuss some sufficient conditions to make sure a univariate BB polynomial has exactly one root between $(0,1)$. Then we show that the sufficient functional zero contouring conditions for a quintic is an integration of the single-rooted and non-rooted conditions at every rays, for a three-sided patch, shooting from the top vertex p_4 of the tetrahedron to the bottom face, or for a four sided patch, from edge $[p_4, p_1'']$ of the tetrahedron to edge $[p_2, p_3]$. (for a four See Figure 4.2.1.

4.3.1 Enforcing Positivity of a BB Polynomial

4.3.1.1 Univariate

A necessary condition for a univariate polynomial $F(\alpha)$ to be positive in $[0,1]$ is that $F(0) = b_{m0} > 0$ and $F(1) = b_{0m} > 0$. Actually, if we are given $F(0,1) = b_{m0} > \varepsilon$, for some small constant $\varepsilon > 0$, and some other weights excluding $F(1,0) = b_{0m}$, we can always set b_{0m} and other free weights to be large enough so that $F(\alpha) > 0, \alpha \in [0.1]$.

In stead of giving a full proof, we check out several cases that are going to be used in weight setting processes.

For a cubic BB polynomial

$$F3(\alpha) = b_{30}B^3_{30}(\alpha) + b_{21}B^3_{21}(\alpha) + b_{12}B^3_{12}(\alpha) + b_{03}B^3_{03}(\alpha) \qquad (4.3.19)$$

where $b_{30} > 0$, b_{21} are given, set b_{12} and b_{03} so that $F3(\alpha) > 0$ in $[0,1]$.

From the subdivision properties of BB polynomial, if $F3(\alpha) > 0$, then there must exist a subdivision of the BB polynomial where the weights of each piece are all positive. Based on this observation, we subdivide the univariate cubic polynomial $F3(\alpha)$ into two pieces,

$$F3^{(1)} = b^{(1)}_{30}B^3_{30}(\alpha) + b^{(1)}_{21}B^3_{21}(\alpha) + b^{(1)}_{12}B^3_{12}(\alpha) + b^{(1)}_{03}B^3_{03}(\alpha) \qquad (4.3.20)$$

$$F3^{(2)} = b^{(2)}_{30}B^3_{30}(\alpha) + b^{(2)}_{21}B^3_{21}(\alpha) + b^{(2)}_{12}B^3_{12}(\alpha) + b^{(2)}_{03}B^3_{03}(\alpha) \qquad (4.3.21)$$

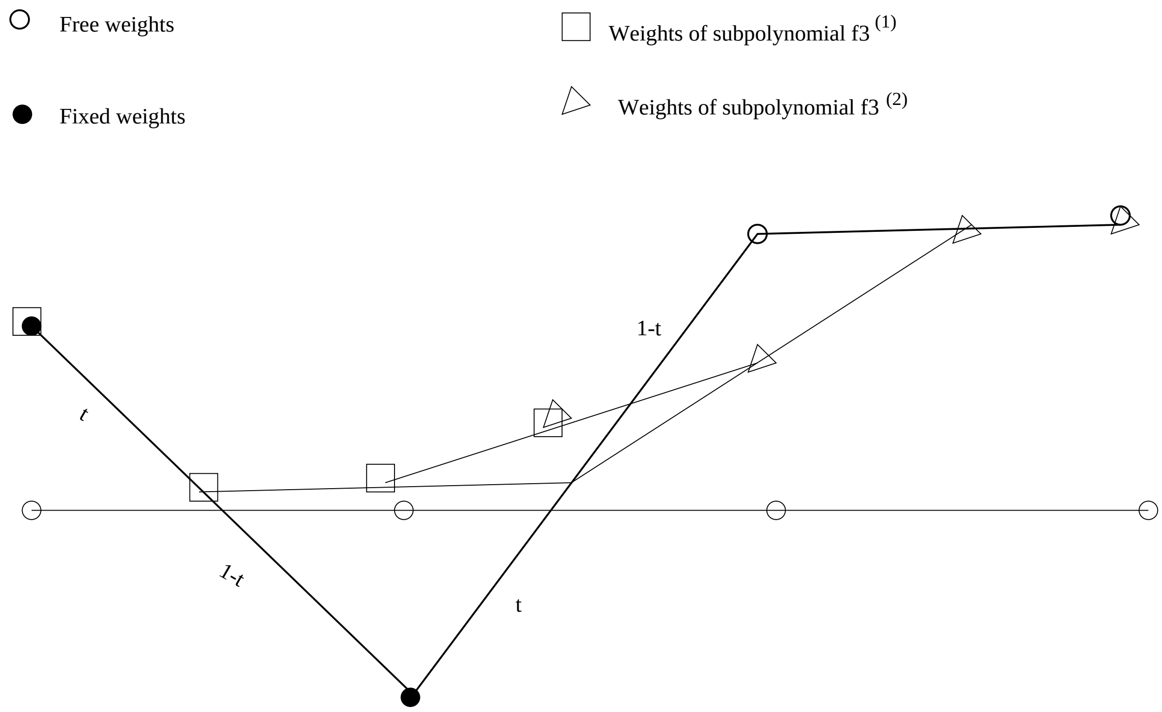


Figure 4.3.2 A cubic polynomial is set positive by subdivision and setting the two subpolynomials positive

at $\alpha = (1-t, t)$ (See figure 4.3.2). We now just need to choose t carefully so that

$$b_{21}^{(1)} = (1-t)b_{30} + tb_{21} > 0 \tag{4.3.22}$$

The other weights are positive functions of b_{12} and b_{03} and hence can be set to be positive by setting b_{12} and b_{03} large enough.

We observe that, if $F3(\alpha) > 0$, then there must exist a subdivision of the BB polynomial where the weights of each piece are all positive. We subdivide the univariate cubic polynomial $F3(\alpha)$ into two pieces,

$$F3^{(1)} = b_{30}^{(1)}B_{30}^3(\alpha) + b_{21}^{(1)}B_{21}^3(\alpha) + b_{12}^{(1)}B_{12}^3(\alpha) + b_{03}^{(1)}B_{03}^3(\alpha) \tag{4.3.23}$$

$$F3^{(2)} = b_{30}^{(2)}B_{30}^3(\alpha) + b_{21}^{(2)}B_{21}^3(\alpha) + b_{12}^{(2)}B_{12}^3(\alpha) + b_{03}^{(2)}B_{03}^3(\alpha) \tag{4.3.24}$$

at $\alpha = t$ (See figure 4.3.2). We now just need to choose

$$t = t_0 < \frac{b_{30}}{b_{30} - b_{21}}$$

so that

$$b_{21}^{(1)} = (1 - t_0)b_{30} + t_0 b_{21} > 0 \tag{4.3.25}$$

The other weights are positive function of b_{12} and b_{03} and hence can be set to be positive. In particular,

$$\begin{aligned} b_{12}^{(1)} &= (1 - t_0)b_{21}^{(1)} + t_0((1 - t_0)b_{21} + tb_{12}) && (4.3.26) \\ &> t_0((1 - t_0)b_{21} + t_0 b_{12}) && (4.3.27) \end{aligned}$$

Hence if

$$(1 - t_0)b_{21} + t_0 b_{12} > 0 \tag{4.3.28}$$

or

$$b_{12} > -\frac{1 - t_0}{t_0} b_{21} \tag{4.3.29}$$

then

$$b_{12}^{(1)} > 0 \quad .$$

It is trivial to verify that if furthermore $b_{03} > 0$, all other control points are positive.

For a quartic BB polynomial,

$$\begin{aligned} F4(\alpha) &= b_{40}B_{40}^4(\alpha) + b_{31}B_{31}^4(\alpha) + b_{22}B_{22}^4(\alpha) + \\ &= b_{13}B_{13}^4(\alpha) + b_{04}B_{04}^4(\alpha) && (4.3.30) \end{aligned}$$

where $b_{40} > \varepsilon$, b_{31} and b_{22} are given, set b_{13} and b_{04} so that $F4(\alpha) > 0$ in $[0, 1]$.

We subdivide the univariate quartic polynomial $F4(\alpha)$ into two pieces,

$$\begin{aligned} F4^{(1)}(\alpha) &= b_{40}^{(1)}B_{40}^4 + b_{31}^{(1)}B_{31}^4 + b_{22}^{(1)}B_{22}^4 + b_{13}^{(1)}B_{13}^4 + b_{04}^{(1)}B_{04}^4 && (4.3.31) \\ F4^{(2)}(\alpha) &= b_{40}^{(2)}B_{40}^4 + b_{31}^{(2)}B_{31}^4 + b_{22}^{(2)}B_{22}^4 + b_{13}^{(2)}B_{13}^4 + b_{04}^{(2)}B_{04}^4 && (4.3.32) \end{aligned}$$

at $\alpha = (1 - t, t)$ (see figure 4.3.3). By choosing t carefully, one makes sure that

$$\begin{aligned} b_{31}^{(1)} &= (1 - t)b_{40} + tb_{31} > 0 && (4.3.33) \\ b_{22}^{(1)} &= (1 - t)^2 b_{40} + 2(1 - t)tb_{31} + t^2 b_{22} > 0. && (4.3.34) \end{aligned}$$

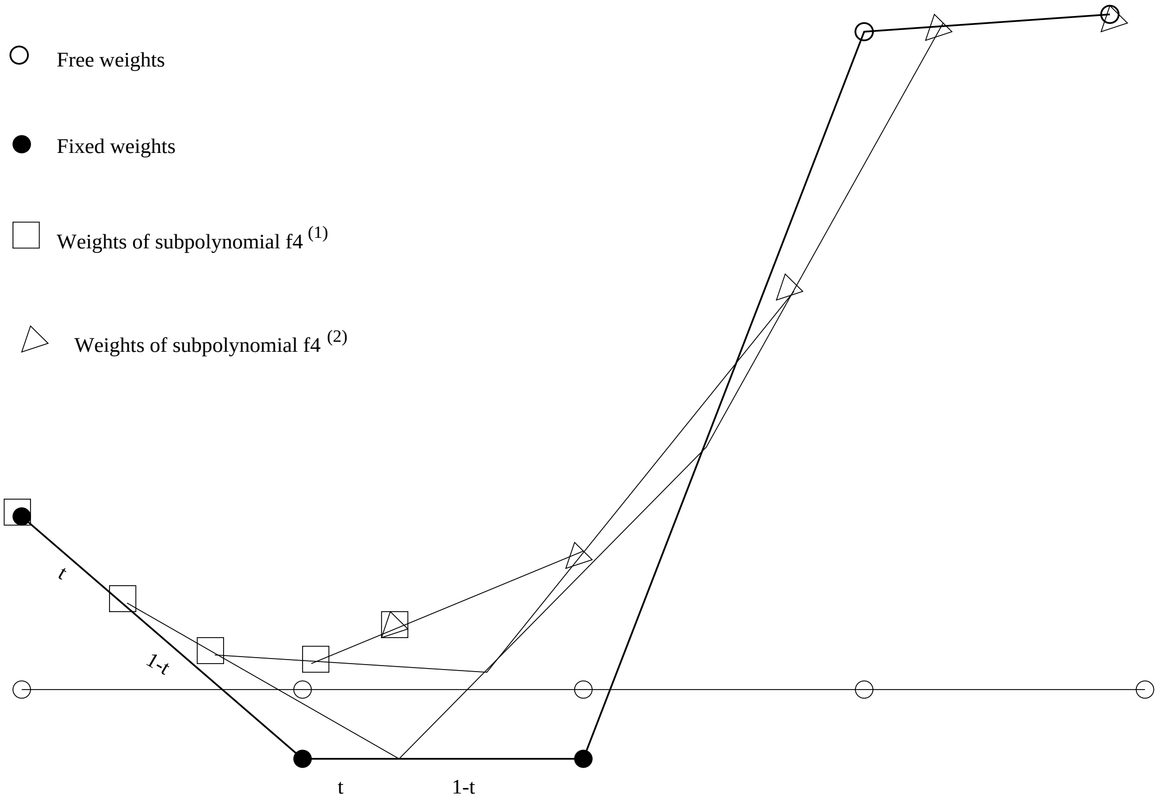


Figure 4.3.3 Making sure a quartic polynomial is positive by looking at its two subpolynomials

The rest of the control points is a positive function of b_{13}, b_{04} and hence can be set to be positive.

Specifically, Let

$$r = \frac{1-t}{t}$$

, inequality (4.3.34) is equivalent to

$$b_{40}r^2 + 2b_{31}r + b_{22} > 0 \tag{4.3.35}$$

Solving this inequality in $r > 0$,

$$r > \frac{-b_{31} + \sqrt{b_{31}^2 - b_{40}b_{22}}}{b_{40}} \tag{4.3.36}$$

Hence by choosing

$$r = r_0 > \frac{-b_{31} + \sqrt{b_{31}^2 - b_{40}b_{22}}}{b_{40}}, \quad b_{22}^{(1)} > 0$$

It is easy to verify that so is $b_{31}^{(1)}$.

Then, in order to make $b_{13}^{(1)} > 0$, as

$$b_{13}^{(1)} = (1-t)b_{22}^{(1)} + t(b_{31}B_{20}^2 + b_{22}B_{11}^2 + b_{13}B_{02}^2) > 0 \tag{4.3.37}$$

or

$$b_{31}B_{20}^2(t_0) + b_{22}B_{11}^2(t_0) + b_{13}B_{02}^2(t_0) > 0 \tag{4.3.38}$$

where $t_0 = \frac{1}{1+r_0}$, or

$$b_{31}r_0^2 + b_{22}2r_0 + b_{13} > 0 \tag{4.3.39}$$

or

$$b_{13} > -\frac{-b_{31} + \sqrt{b_{31}^2 - b_{40}b_{22}}}{b_{40}} \left(b_{31}\frac{-b_{31} + \sqrt{b_{31}^2 - b_{40}b_{22}}}{b_{40}} + 2b_{22} \right) \tag{4.3.40}$$

Furthermore, with $b_{04} > 0$, every other weights are positive.

For a quintic BB polynomial,

$$\begin{aligned} F5(\alpha) &= b_{50}B_{50}^5(\alpha) + b_{41}B_{41}^5(\alpha) + b_{32}B_{32}^5(\alpha) + b_{23}B_{23}^5(\alpha) + \\ & \quad b_{14}B_{14}^5(\alpha) + b_{05}B_{05}^4(\alpha) \end{aligned} \tag{4.3.41}$$

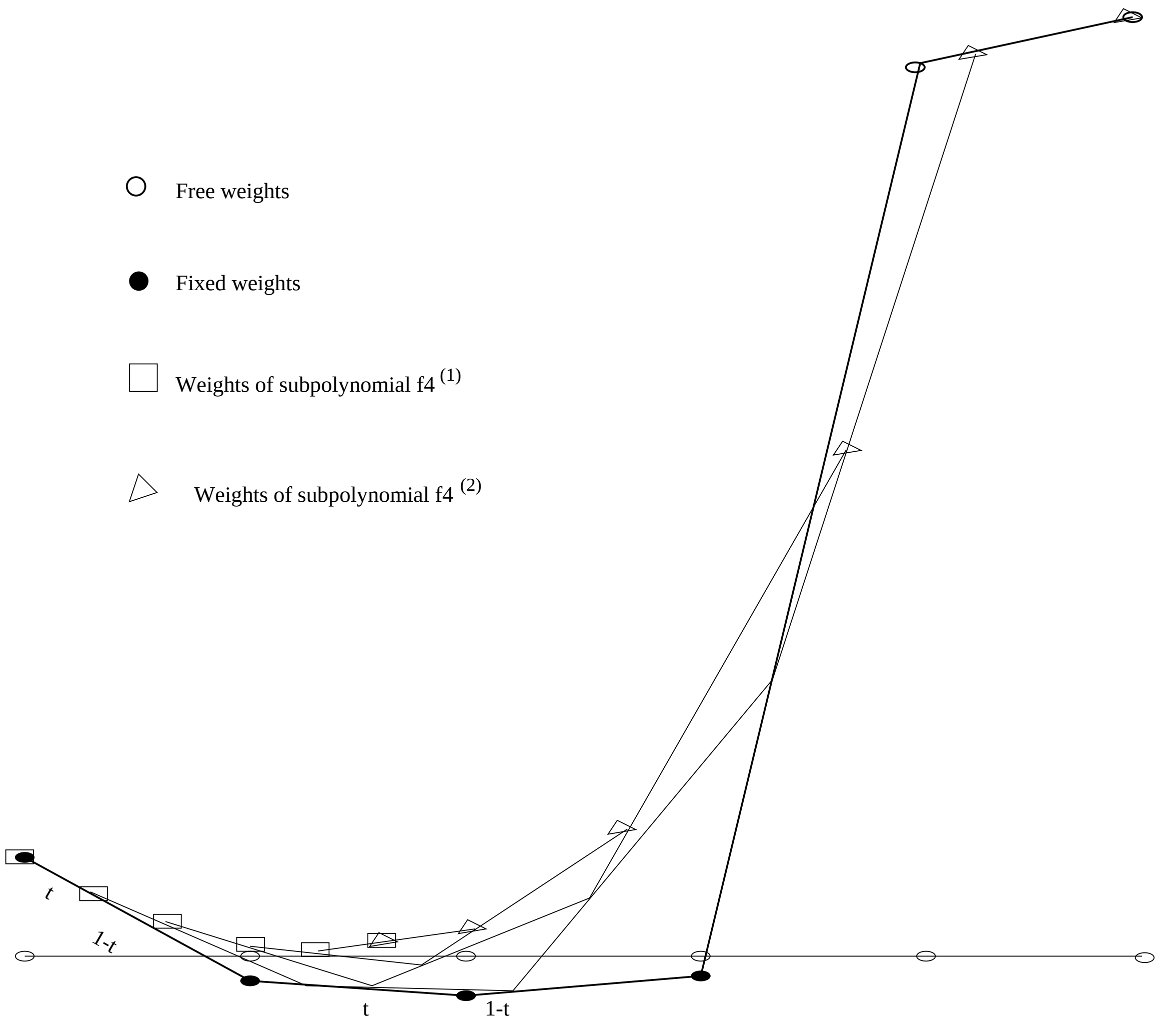


Figure 4.3.4 Making sure a quintic polynomial is positive by looking at its two subpolynomials

where $b_{50} > \varepsilon$, b_{41}, b_{32} and b_{23} are given, set b_{14} and b_{05} so that $F5(\alpha) > 0$ in $[0, 1]$.

We subdivide the univariate quintic polynomial $F5(\alpha)$ into two pieces,

$$F5^{(1)}(\alpha) = b_{50}^{(1)} B_{50}^5 + b_{41}^{(1)} B_{41}^5 + b_{32}^{(1)} B_{32}^5 + b_{23}^{(1)} B_{23}^5 b_{14}^{(1)} B_{14}^5 + b_{05}^{(1)} B_{05}^5 \quad (4.3.42)$$

$$F5^{(2)}(\alpha) = b_{50}^{(2)} B_{50}^5 + b_{41}^{(2)} B_{41}^5 + b_{32}^{(2)} B_{32}^5 + b_{23}^{(2)} B_{23}^5 b_{14}^{(2)} B_{14}^5 + b_{05}^{(2)} B_{05}^5 \quad (4.3.43)$$

at $\alpha = (1-t, t)$ (see figure 4.3.4). By choosing t carefully, one makes sure that

$$b_{41}^{(1)} = (1-t)b_{50} + tb_{41} > 0 \quad (4.3.44)$$

$$b_{32}^{(1)} = (1-t)^2 b_{50} + 2(1-t)tb_{41} + t^2 b_{32} > 0 \quad (4.3.45)$$

$$b_{23}^{(1)} = (1-t)^3 b_{50} + 3(1-t)^2 tb_{41} + 3(1-t)t^2 b_{32} + t^3 b_{23} > 0 \quad (4.3.46)$$

The rest of the control points is a positive function of b_{14}, b_{05} and hence can be set to be positive.

Similar to the cubic and quartic case, we solve

$$(1-t)^3 b_{50} + 3(1-t)^2 tb_{41} + 3(1-t)t^2 b_{32} + t^3 b_{23} > 0 \quad (4.3.47)$$

in $0 < t < 1$, or its equivalence

$$r^3 b_{50} + 3r^2 b_{41} + 3rb_{32} + b_{23} > 0 \quad (4.3.48)$$

with $r = \frac{1-t}{t}$.

Let t_0 be what we choose for t. We next choose b_{14} so that

$$\begin{aligned} b_{14}^{(1)} &= (1-t_0)t_{23}^{(1)} + t_0(b_{41}B_{30}^3(t_0) + b_{32}B_{21}^3(t_0) + \\ &\quad b_{23}B_{12}^3(t_0) + b_{14}B_{03}^3(t_0)) & (4.3.49) \\ &> t_0(b_{41}B_{30}^3(t_0) + b_{32}B_{21}^3(t_0) + b_{23}B_{12}^3(t_0) + b_{14}B_{03}^3(t_0)) > 0 & (4.3.50) \end{aligned}$$

or

$$b_{14} > -(b_{41}r_0^3 + 3b_{32}r_0^2 + 3b_{23}r_0) \quad (4.3.51)$$

Setting $b_{05} > 0$ makes every other weights positive.

4.3.1.2 Bivariate

Given a bivariate degree m BB polynomial $F(\alpha)$ in $[p_1p_2p_3]$, if $F(\alpha) \wedge \alpha_3 = 0$ is fixed and larger than a small constant ε, we can set b_{00m} and any other free weights so that $F(\alpha) > 0$ in the domain.

A sufficient algorithm to achieve this is given as follows. We first construct a "worst case" univariate degree m polynomial

$$Fm^{-}(\alpha) = b_{m0}^{-} B_{m0}^{m}(\alpha) + ... + b_{(m-j)j}^{-} B_{(m-j)j}^{m}(\alpha) + b_{0m}^{-} B_{0m}^{m}(\alpha). \tag{4.3.52}$$

where, for the jth layer $F(\alpha) \wedge \alpha_3 = j$, $0 < j < m$, if a minimum value can be determined from the fixed weights assign it to $b_{(m-j)j}^{-}$. Otherwise mark $b_{(m-j)j}^{-}$ as a free weights and layer j as free layer. We then set the free weights to make sure the univariate polynomial $Fm^{-} > 0$. Then we make sure that the subpolynomial in each free layer is larger than the corresponding free weight in Fm^{-}. By doing that, we make sure that the univariate subpolynomial in any line $[pp_3]$, $p \in [p_1, p_2]$, is positive.

If we have more time or our computer is more powerful, we may obtain more relax result by first subdividing $[p_1p_2]$ into n pieces, $[p_1^{(i)}, p_2^{(i)}]$, $p_1^{(1)} = p_1$, $p_2^{(n)} = p_2$. Then applying the above algorithm to each triangular simplex $[p_1^{(i)}, p_2^{(i)} p_3]$, and then in each layer, setting the free weights so that it is no less than what is required for each subdivided subpolynomial.

Similarly, if b_{00m} is fixed and larger than a small constant ε, we can set the weights of $F(\alpha) \wedge \alpha_3 = 0$ and any other free weights so that $F(\alpha) > 0$ in the domain.

4.3.1.3 Trivariate

There are two kinds of configurations with trivariate BB polynomials. The first case is analog to the bivariate case. For a trivariate degree m BB polynomial $F(\alpha)$ in $[p_1p_2p_3p_4]$, if $F(\alpha) \wedge \alpha_4 = 0$ is fixed and larger than a small constant ε, we can always set b_{000m} and any other free weights so that $F(\alpha) > 0$ for any α over the domain simplex. If $b_{000m} > \varepsilon > 0$ is given, we can always set the weights of subpolynomial in $[p_1p_2p_3]$, namely $F(\alpha) \wedge (\alpha = 0)$ and any other free weights to be large enough so that $F(\alpha) > 0$.

The second case, if $F(\alpha) \wedge \alpha_1 = \alpha_4 = 0$ is fixed and larger than a small constant ε, we can always set the weights of $F(\alpha) \wedge \alpha_2 = \alpha_3 = 0$ and any other free weight to be large enough so that $F(\alpha) > 0$. In particular, the worst case polynomial

$$Fm^-(\alpha) = b^-_{m0}B^m_{m0}(\alpha) + ... + b^-_{(m-j)j}B^m_{(m-j)j}(\alpha) + +...... + b^-_{0m}(\alpha) \tag{4.3.53}$$

where $b^-_{(m-j)j}$ is the minimum of $F(\alpha) \wedge \alpha_1 + \alpha_4 = j$, if the minimum can be determined. Otherwise $b^-_{(m-j)j}$ is marked free.

Summing up, in a simplex tetrahedron, if the subpolynomial in a face is given positive, we can adjust the subpolynomial at its opposite vertex so that the entire polynomial over the simplex is positive and vice versa. If the subpolynomial in an edge is given positive, we can adjust that subpolynomial at its opposite edge so that the entire polynomial over the simplex is positive. The first case is for the three sided patch while the second case is for the four sided case.

Similarly, relax result can be obtained by subdividing $[p_1p_2p_3p_4]$. For the first case (three sided patch), $[p_1p_2p_3p_4]$ is subdivided into $[p_1^{(i)}p_2^{(i)}p_3^{(i)}p_4]$'s, where $[p_1^{(i)}p_2^{(i)}p_3^{(i)}]$ is a sub-triangle of $[p_1p_2p_3]$. For the second case (four sided patch), $[p_1p_2p_3p_4]$ is subdivided into $[p_i^{s1}p_{i+1}^{s1}p_i^{s2}p_{i+1}^{s2}]$'s, where $[p_i^{s1}p_{i+1}^{s1}]$ is a subsegment of $[p_1p_4]$, and $[p_i^{s2}p_{i+1}^{s2}]$ is a subsegment of $[p_2p_3]$.

4.3.2 Sufficient Conditions for Single-rooted Univariate Quintic BB Polynomials

We now tackle the following problem. Give a univariate quintic BB polynomial

$$F5(\alpha) = b_{50}B^5_{50}(\alpha) + b_{41}B^5_{41}(\alpha) + b_{32}B^5_{32}(\alpha) + b_{23}B^5_{23}(\alpha) + b_{14}B^5_{14}(\alpha) + b_{05}B^5_{05}(\alpha) \tag{4.3.54}$$

where b_{50}, b_{41}, b_{32}, and b_{23} are given, set b_{14} and b_{05} so that the polynomial has exactly one root within the interval of $(0, 1)$ if (1) $b_{50} < 0$, or (2) $b_{50} = 0$ and $b_{41} > \eta$, where $\eta > 0$ is a constant.

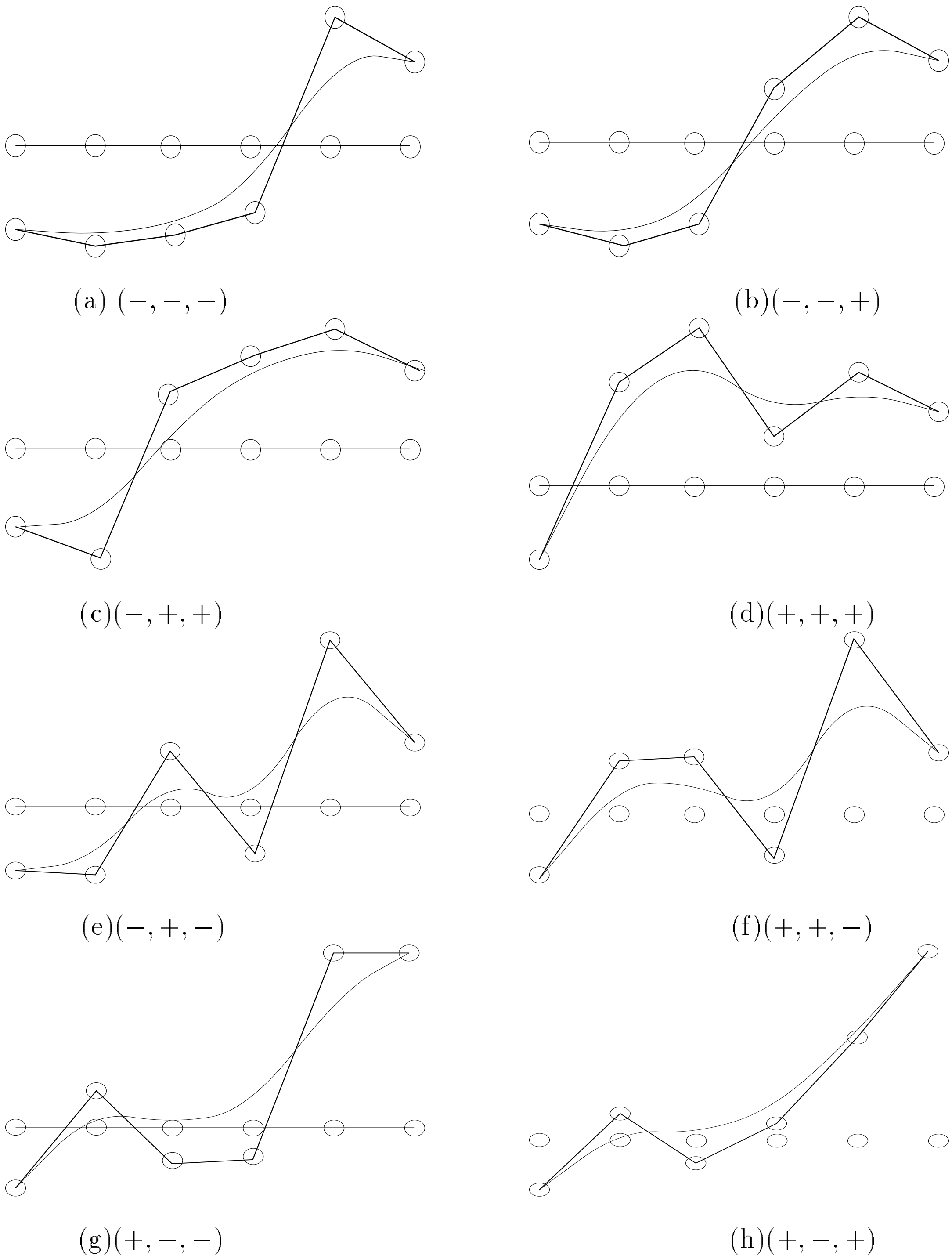
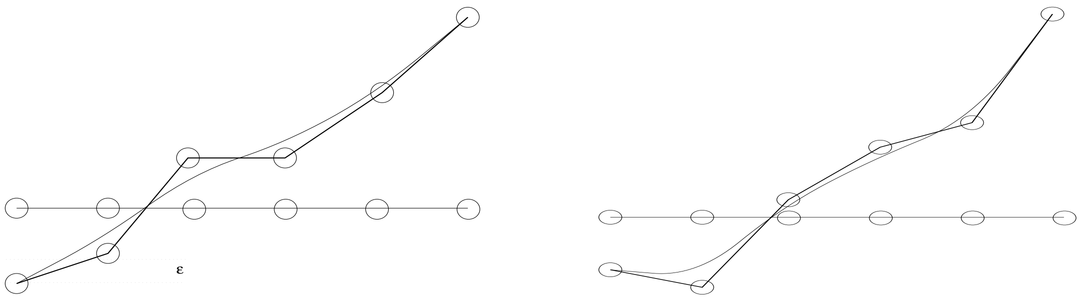


Figure 4.3.5 Quintic BB polynomials are subdivided into 8 cases according to the signs of weights b_{41}, b_{32} abd b_{23}.



(a) (b)

Figure 4.3.6 case $(-,+,-)$ is further subdivided. (a) $b_{50} - b_{41} < -\varepsilon$. (b) Otherwise

We first assume that $b_{50} < 0$. Such a quintic BB polynomial can be classified into the 8 cases in terms of the signs of b_{41}, b_{32} and b_{23}, shown in figure 4.3.5. If we denote each case by a triple consists of the signs of b_{41}, b_{32} and b_{23} respectively, then in cases $(-,-,-)$, $(-,-,+)$, $(-,+,+)$ and $(+,+,+)$, there is only one sign change in the weights, which is a sufficient condition to guarantee that there is exactly one root in $[0,1]$. We call these four cases category (0).

The other four cases are divided into the following two categories in terms of the difference between b_{50} and b_{41}: (1) $b_{50} - b_{41} < -\varepsilon$, where ε is a small positive number, which can be set as $b_{50}/100$; and (2) otherwise. Case $(+,+,-)$, $(+,-,+)$ and $(+,-,-)$ fall into case (1), while case $(-,+,-)$ is further divided into (1) and (2) (See figure 4.3.6).

For a polynomial of category (1), we set b_{14} and b_{05} to make sure that the first derivative of $F5(\alpha)$,

$$\begin{aligned} F5_{\alpha}(\alpha) &= 5((b_{41} - b_{50})B^4_{40}(\alpha) + (b_{32} - b_{41})B^4_{31}(\alpha) + (b_{23} - b_{32})B^4_{22}(\alpha) + \\ &\quad (b_{14} - b_{23})B^4_{13}(\alpha) + (b_{05} - b_{14})B^4_{04}(\alpha)) \end{aligned} \tag{4.3.55}$$

is positive in $[0,1]$. It follow that $F5(\alpha)$ is monotonic and hence single-rooted. The problem is hence reduced to the quartic case we have discussed in previous subsection.

For a polynomial of case (2), we set b_{14} and b_{05} to make sure that the second derivative of $F(\alpha)$,

$$F5_{\alpha\alpha}(\alpha) = 20((b_{32} - 2b_{41} + b_{50})B^3_{30}(\alpha) + (b_{23} - 2b_{32} + b_{41})B^3_{21}(\alpha) +$$

$$(b_{14} - 2b_{23} + b_{32})B^3_{12}(\alpha) + (b_{05} - 2b_{14} + b_{23})B^3_{03}(\alpha)) \qquad (4.3.56)$$

is positive. The positivity of $F_{\alpha\alpha}(\alpha)$ ensure that $F_\alpha(\alpha)$ is monotonic, which ensure that $F(\alpha)$ is single-rooted, providing that $F5(0) = b_{50} < 0$ and $F5(1) = b_{05} > 0$. As $b_{50} < 0$, $b_{41} < 0$, $b_{32} > 0$ and $b_{50} - b_{41} \leq \varepsilon$ we have $b_{32} - 2b_{41} + b_{50} > 0$. Hence the problem is reduced to the cubic case we discuss in last subsection.

In the case that b_{50} and $b_{41} > \eta > 0$, the polynomial has a root at $\alpha = (1, 0)$. As $b_{41} > \eta > 0$, we can set b_{14} and b_{05} large enough so that there is no other roots.

4.3.3 Sufficient Conditions for Quintic A-patches

From definition, a three sided quintic patch is an A-patch if a line segment $[pp_4]$ connecting the top vertex p_4 and any point p on the bottom $[p_1p_2p_3]$ intersects the surface at most once. In other words, the univariate subpolynomial in $[pp_4]$ has at most one root in $[0, 1]$. In the case of a four sided patch, a *ray polynomial* is the univariate subpolynomial in $[p_{14}p_{23}]$, where $p_{14} \in [p_1p_4]$, $p_{23} \in [p_2p_3]$.

We first assume that, at the bottom face $[p_1p_2p_3]$ (where $\alpha_4 = 0$), $F(\alpha) > 0$. Then $[pp_4]$ intersects the surface exactly once. We call such a subpolynomial a *ray polynomial* of the trivariate polynomial in $[p_1p_2p_3p_4]$. Note that how many times $[pp_4]$ intersects with the surface are determined by the ray polynomial in $[pp_4]$. Recalling that, in section 4.2, the weights at the 4th and 5th are free in the C^2 continuity weight setting algorithm, each of such univariate quintic polynomials can be made single-rooted as we discussed above. So if one can check all $[p_4p]$'s for all p in $[p_1p_2p_3]$, to make sure that the ray polynomial has exactly one root in $[0, 1]$, we make sure that the surface is an A-patch. But obviously, it is non-practical to keep track of the infinite number of ray polynomials. Hence, we instead subdivide the bottom face $[p_1p_2p_3]$ into small triangles $[p_1^{(i)}p_2^{(i)}p_3^{(i)}]$, such that the subpolynomial in $[p_1^{(i)}p_2^{(i)}p_3^{(i)}p_4]$ is simpler in the sense that the ray polynomials fall in the same category that we discussed above. We are then able to make sure that all ray polynomials in $[p_4p]$ are single-rooted by making sure that the "worst case" is single-rooted.

Similarly, in the case of a four sided patch $[p_1p_2p_3p_4]$, we subdivide edge $[p_1p_4]$ and edge $[p_2p_3]$ to subdivide the tetrahedron into smaller four sided patches, within each of which we enforce that the surface is an A-patch by treating a "worst" case ray polynomial.

However, it is not practical to subdivide the polynomial into into subpolynomials that exactly fall into individual categories. Instead, we subdivide the polynomial until the individual subpolynomial can be treated the same way. To see that is possible, in last subsection, we classify the univariate quintic subpolynomials into 3 categories. Category (0) can be trivially made single-sheeted by setting b_{14} and b_{05} to be positive. So, if a ray polynomial of a trivariate polynomial is either in category (0) or (1), we can treat them all as if they are all in category (1). Similarly, when a ray polynomial of a trivariate polynomial is either in category (0) or (2). The algorithm that guarantees a category (2) subpolynomial single-rooted rely on the fact that $b_{50}+b_{32}-2b_{41} > 0$, which can apply to some member of categories (0) and (1) as well. Hence if we subdivide the polynomial deep enough, we can get subpolynomial whose ray polynomials are in category (2), or (0), or (1) but with $b_{50}+b_{32}-2b_{41} > 0$.

Based on the above observation, we layout the subdivision scheme in the following way.

Let $F_0(\alpha) = F(\alpha) \wedge (\alpha_4 = 0)$ be the bivariate polynomial in the bottom face, or the 0th layer, $F_1(\alpha) = F(\alpha) \wedge (\alpha_4 = 1)$ be the quartic bivariate polynomial in the 1st layer, $F_2(\alpha) = F(\alpha) \wedge (\alpha_4 = 2)$ be the cubic bivariate polynomial in the 2nd layer. We define the following two polynomials in $[p_1p_2p_3p_4]$

$$F_\Delta(\alpha) = F_0(\alpha) - F_1(\alpha)\alpha \tag{4.3.57}$$

$$F_{\Delta^2}(\alpha) = F_0(\alpha) - 2F_1(\alpha)\alpha + F_2(\alpha)\alpha^2 \tag{4.3.58}$$

We denote the weights of polynomial $F(\alpha)$ as $\boldsymbol{b}(F)$. $\boldsymbol{b}(F) < c$ means the weights of F are all less than c.

Then we keep subdividing $[p_1p_2p_3p_4]$ until for each subpolynomial $[p_1^{(i)}p_2^{(i)}p_3^{(i)}p_4]$ satisfies one of the following cases.

(0) There is only one sign change;

(1) $\boldsymbol{b}(F_\Delta) < -\varepsilon$, which implies that for every univariate subpolynomial in $[p_4 p]$, $b_{50} - b_{41} < -\varepsilon$;

(2) $\boldsymbol{b}(F_1) < 0 \wedge (\boldsymbol{b}(F_{\Delta^2}) \vee \boldsymbol{b}(F_2) < 0 \vee \boldsymbol{b}(F_3) > 0)$ which implies that for every univariate subpolynomial in $[p_4 p]$, $b_{41} < 0$ and $b_{50} - 2b_{41} + b_{32} > 0$, or can be classified into category (0).

In cases (1) and (2), we define the *worst* univariate subpolynomial

$$F_{worst}(x) = b_{50}^{-} B_{50}^5 + b_{41}^{-} B_{41}^5 + b_{32}^{-} B_{32}^5 + +b_{23}^{-} B_{23}^5 + b_{41}^{-} B_{14}^5 + b_{50}^{-} B_{05}^5 \quad (4.3.59)$$

where

$$\begin{aligned} b_{50}^{-} &= Min\{\mathbf{b}(F_0)\} \\ b_{41}^{-} &= Min\{\mathbf{b}(F_1)\} \\ b_{32}^{-} &= Min\{\mathbf{b}(F_2)\} \end{aligned}$$

Enforcing the worst univariate subpolynomial to be single-rooted, we guarantee that all the univariate subpolynomial over $[p_4 p]$ is single-rooted, which implies that the surface is an A-patch.

In the case that $F_0(\alpha)$ is not always positive, we subdivide face $[p_1 p_2 p_3]$ until in each subpolynomial $F_0^{(i)}(\alpha)$ in $[p_1^{(i)} p_2^{(i)} p_3^{(i)}]$ is one of the following

1. Positive. Set the free weights of $F^{(i)}(\alpha)$ so that the trivariate polynomial has no root.

2. Negative. Set the free weights of $F^{(i)}(\alpha)$ so that the trivariate polynomial has exactly one root.

3. The subpolynomial in $[p_1^{(i)} p_2^{(i)} p_3^{(i)}]$ (denoted as $F_4^{(i)}(\alpha)$) is an A-spline [25]. A sufficient condition is that, there exists an integer $j (1 \le j \le 3)$ an integer

$k (0 < k < 5)$ such that

$$b_\lambda \leq 0, \qquad \lambda_j = 0, 1, ..., k-1 \tag{4.3.60}$$

$$b_\lambda \geq 0, \qquad \lambda_j = k+1, ..., 5 \tag{4.3.61}$$

In this case, say $j = 1$, we set the free weights of $F^{(i)}(\alpha)$ so that the bivariate subpolynomial in $[p_2^{(i)} p_3^{(i)} p_4]$ is positive, and furthermore, large enough so that any ray subpolynomial in pp_4, where $p \in [p_2^{(i)} p_3^{(i)} p_4]$, is single-rooted.

At a corner of face $[p_1 p_2 p_3]$, say p_1 or $\alpha = (1, 0, 0, 0)$, the weight $b_{5000} = 0$. From the stronger tangent containment condition 4.2.1 that we have enforced, $b_{4001} > \eta$ for some small constant $\eta > 0$. Thus there exists a small neighborhood $\alpha^*, |\alpha^* - \alpha| < \rho$, where $F_1(\alpha^*) > 0$. Hence it can be treated as an instance of category (1).

In short, the functional zero contour enforcing scheme can be embed into the C^2 scheme as follows. Steps 4 and 8 replace steps 4 and 8 in scheme 4.2.1. Step 11' is inserted between steps 11 and 12.

Algorithm 4.3.2 Functional zero contouring

4. If face $[p_1 p_2 p_3]$ is convex, set $a_{1220}^{(i)}$, $a_{2210}^{(i)}$, $a_{2120}^{(i)}$ to be positive.

8. Set number 8 weight, $a_{1112}^{(i)}$ to be positive.

11$'$ Set number 12 weight to be large enough so that

   (a) we are sure the polynomial in $[p_1 p_2 p_3 p_4]$ is a three sided A-patch, and

   (b) when we compute number 13, 14 and 15 weights later in step 12, they are large enough so that we are sure the polynomial in $[p_1'' p_2 p_3 p_4]$ is a four sided A-patch.

4.3.4 Summary

In summary, we give an outline of the algorithm.

Algorithm 4.3.3 Construction of C^2 A-patch surface

INPUT: a triangulated polyhedron $\mathcal{P}$.

1. Compute the normal at each vertex of $\mathcal{P}$.
2. Construct C^1 cubic interpolant simplicial hull and C^1 A-patches
3. Degree raising the cubic simplicial hull to quintic. Used as default values.
4. Estimate C^2 data (Hessian matrix) at each vertex. interpolate them with the polynomials.
5. Construct piecewise C^2 quintic polynomials.
6. Set functional zero contouring conditions.

OUTPUT: C^2 A-patch surface.

4.4 Implementation

We have presented algorithms for interpolating a three dimensional polyhedron with C^2 quintic A-patches. These algorithms have been implemented in the SplineX toolkit of Shastra [3].

Figure 4.4.7 shows a simple example, quintic A-patches based on 4 triangles sharing a vertex. Figure 4.4.8 shows quintic A-patches built on top of a icosahedron. Note that the sizes of edge patches in pairs are not even, as a conventional interpolant simplicial hull based on an icosahedron, with equal sized edge patch pair, would cause case 1 degeneracy (see Appendix 4.2.2). Figure 4.4.9 shows interactive control of C^2 quintic A-patches. Starting from a sphere, we drag the surface towards the vertices of a cube by changing the weights a_{0113}, a_{1013} and a_{1103} of the quintic face simplexes. Note that the simplicial hull in Figure 4.4.9 suffers case 2 degeneracy (see Appendix 4.2.2). Hence we keep weight a_{1112} as is, but instead change a_{0113}, a_{1013} and a_{1103}.

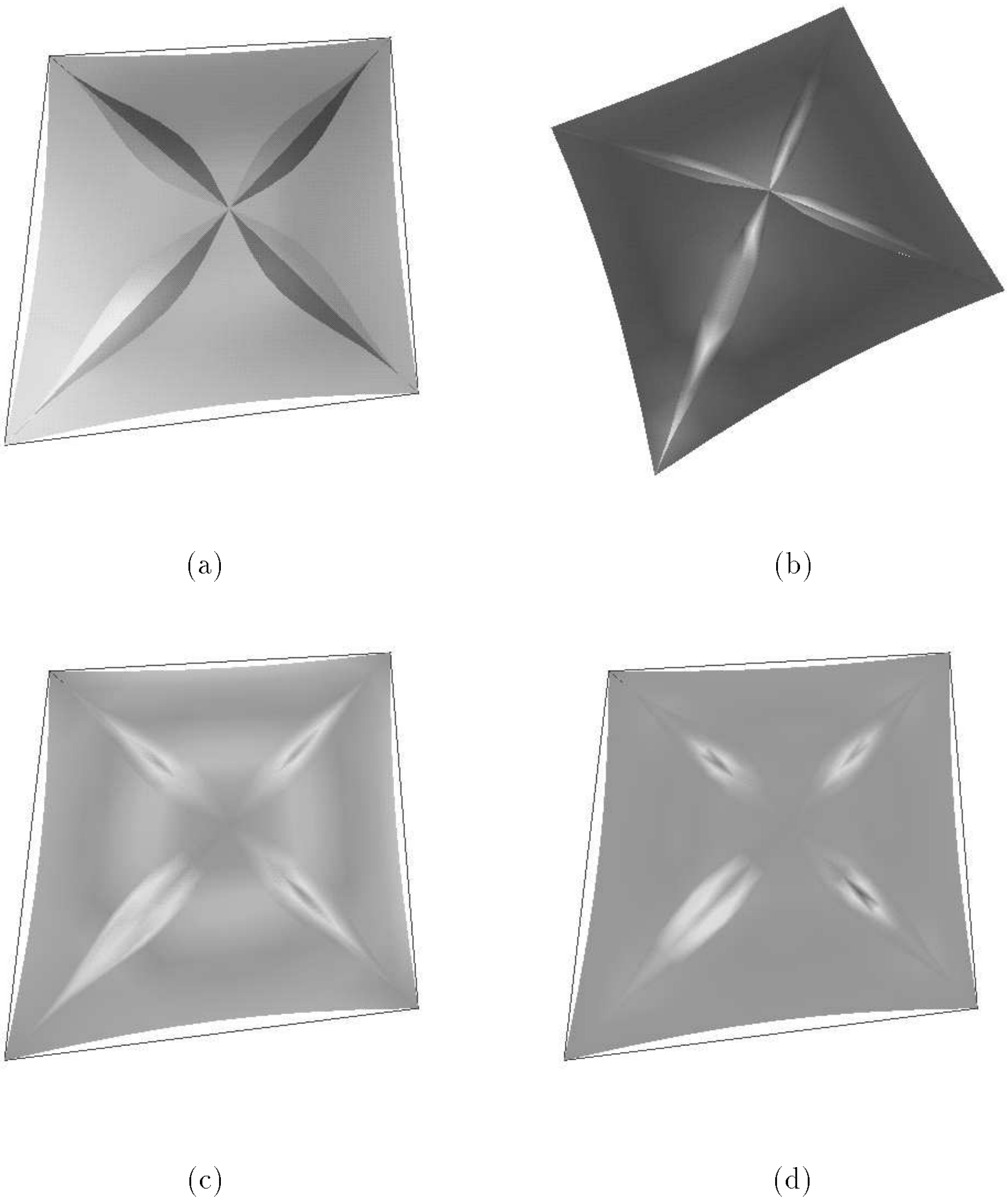
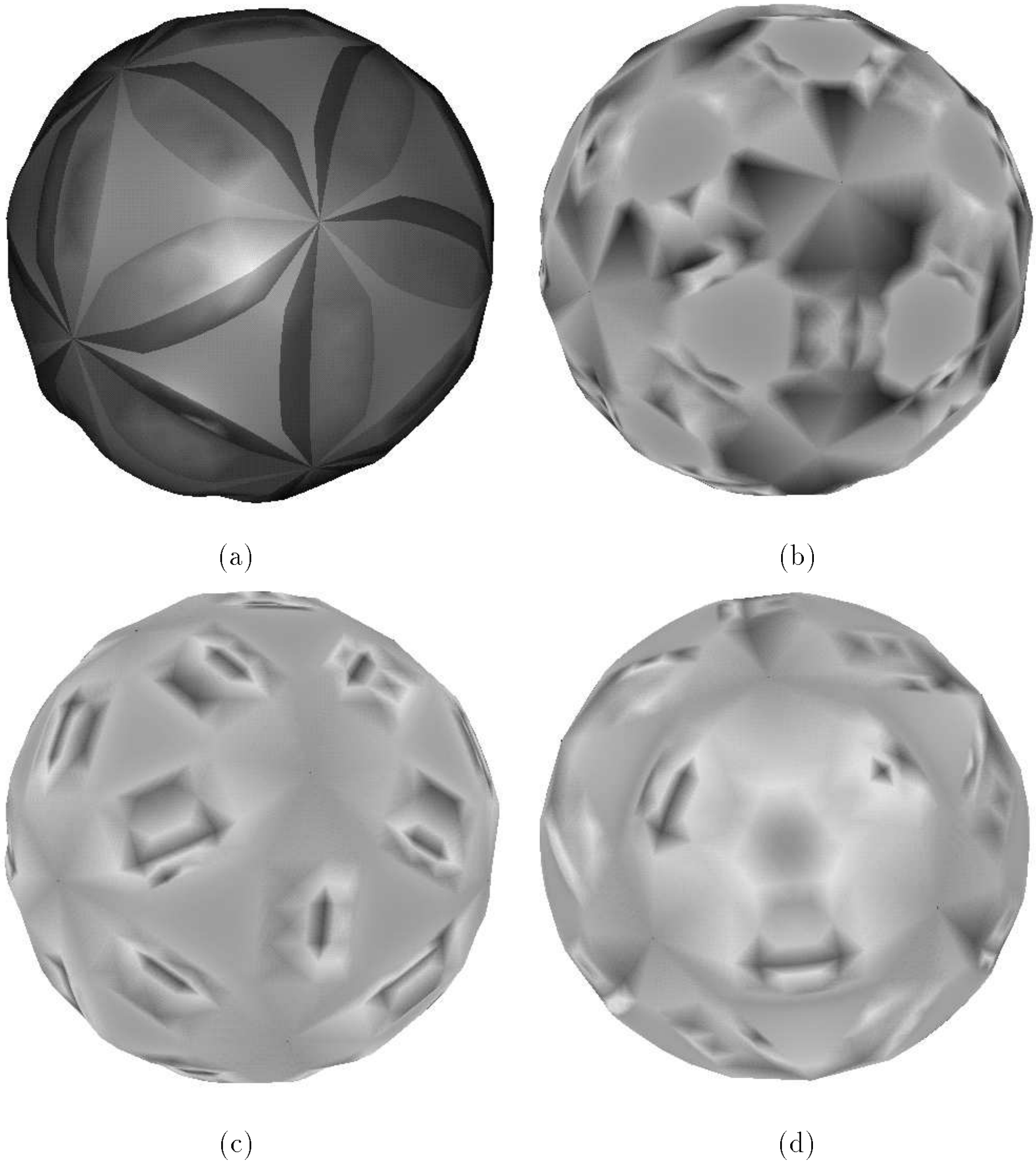


Figure 4.4.7 A simple C^2 example. (a) Drawn in different shades to show the piecewise structure. (b) C^1: showing Gaussian curvature. (c) C^2: showing mean curvature. (d) C^2: showing Gaussian curvature.



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

Figure 4.4.8 A C^2 smooth icosahedron. (a) Drawn in different shades to show the piecewise structure. (b) Gaussian curvature. (c) Mean curvature. (d) Mean curvature. The surface is modified locally around the patch facing up.

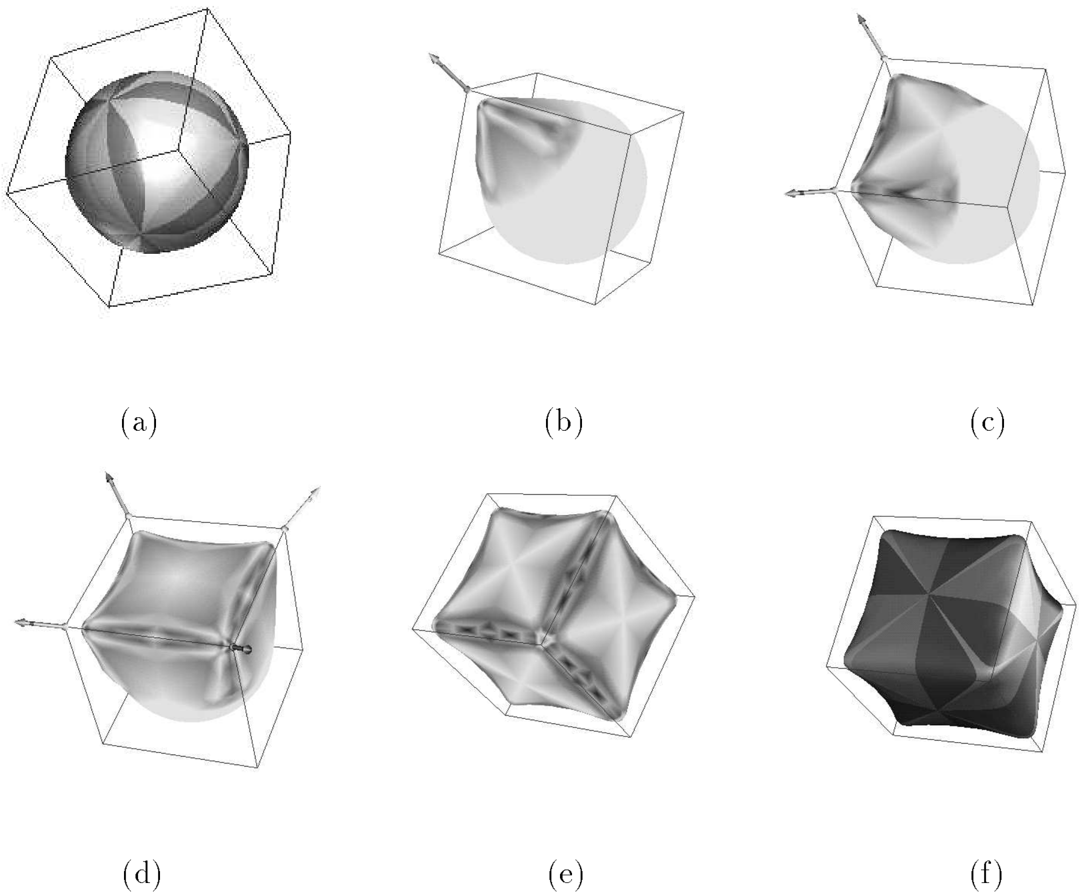


Figure 4.4.9 Interactive control of C^2 A-patches. Starting from a sphere, the quintic piecewise surface is deforming toward a cube. (a) Sphere. (b) The surface is dragged toward a vertex of the cube. Mean curvature map. (c) Toward an edge of the cube. Gaussian curvature. (d) Toward an face of the cube. Mean curvature. (e) Toward the cube. Gaussian curvature. (f) Toward the cube. Shaded in pattern to show the piecewise structure.

4.5 Conclusion

We give piecewise C^2 quintic A-patch scheme to C^2 interpolate or approximate each vertex of a given polyhedron. However, as a first step toward C^2 algebraic splines in BB form, the scheme still has many problems. Compared to the C^1 cubic A-patch scheme [17], this scheme is more complicated in the following aspects. First, we have to solve larger linear system to achieve C^2 continuity. Secondly, it is far more difficult to guarantee that the surface is fully connected, free of singularity and unwanted branches, as in general, the "one sign change" principle is not practical. Thirdly, there are quite many degenerate cases, where the linear systems are singular and we have modified the simplicial hull to remove the degeneracies.

However, it seems to us there is not much room that we can play with quintic surface for C^2 continuity, under the same simplicial hull structure. Further improvements can be made by studying more about the degenerated cases, using higher degree surfaces, or different simplicial hull structures. For example, if we use degree 7 patches, which interpolates some given or pre-calculated C^3 data at each vertex, and C^2 elsewhere, we are able to break up those big systems around the vertices into smaller ones around the edge. Namely, the systems involve no more than 8 simplexes, same level as in the C^1 cubic case.

Besides all these problem, the scheme inherits some common problem of surface fitting, as shown in Figure 4.4.8, although C^2 smooth, the surface looks bumpy. An optimization scheme is hence needed to yield fair surface with respect to this particular problem.

5. APPROXIMATING A-PATCH SPLINES

5.1 Introduction

In this chapter[1], we present an efficient algorithm to construct an "inner" interpolant simplicial hull based on corner-cutting a given polyhedron $\mathcal{P}$ in three dimensions. Both a C^1 smooth mesh with cubic A-patches and a C^2 smooth mesh with quintic A-patches can be constructed to approximate the given polyhedron $\mathcal{P}$.

The interpolatory schemes of [17, 47, 48, 63, 64] all build an "outside" interpolant simplicial hull (consisting of a series of "edge" and "face" tetrahedra) containing the given polyhedron $\mathcal{P}$. Construction of such a simplicial hull is nontrivial for an arbitrary $\mathcal{P}$, even for a convex $\mathcal{P}$ with sharp corners. There are several exceptional situations and degeneracies (co-planarity, hull self-intersection, etc), as well as unwanted undulation in the patches. The new "corner-cutting", "inner" interpolant simplicial hull construction introduced in this chapter can handle an arbitrary simple polyhedron $\mathcal{P}$. This new simplicial hull construction scheme is a three dimensional generalization of the two-dimensional corner-cutting scheme used to construct C^k continuous bivariate A-splines [25].

In this corner cutting scheme, a triangulation $\mathcal{T}$ is constructed based on $\mathcal{P}$, by connecting the adjacent face centroids, and "vertex centroids" to their incident face centroids, where a "vertex centroid" is resulted from a corner cutting process. A "tangent containing" interpolant simplicial hull Σ is constructed based on $\mathcal{T}$. $\mathcal{P}$ serves as an auxiliary "outer" control frame in the simplicial construction to ensure no intersection occurs between neighboring simplexes and furthermore, to provide an appropriate "top" for a tetrahedron built on each triangular face of $\mathcal{T}$.

[1]The material covered in this Chapter is in command with [18, 19].

This corner cutting scheme may also be regarded as a possible A-patch version of the subdivision schemes for smoothing solids. Please see Chapter 1, Section 1.3 for a survey. However, the subdivision scheme presented in this chapter differs from the traditional scheme after a couple of initial steps. Simplicial hull construction and the subsequent weight setting process are free of some common problems in traditional subdivision scheme, such as vertex enclosure problem around an irregular vertex (vertex that has more than 4 incident edges). Compared to parametric ones, the A-patch corner cutting scheme either uses fewer number of patch pieces, or patches of lower degree.

The rest of this chapter is as organized as follows. Section 5.2 presents an interpolant simplicial hull construction scheme for a simple polyhedron $\mathcal{P}$ with convex faces. Section 5.3 gives schemes to to preprocess an input polyhedron into one that meets the restrictions of the algorithm presented in Section 5.2. Section 5.4 discusses some shape control issues, such as editing input polyhedron $\mathcal{P}$, controlling the cutting ratios, and representing C^0 features by using vertex/edge singular A-patches. Finally, Section 5.5 provides some implementation details of our hull construction and polyhedron smoothing scheme.

5.2 Constructing an Interpolant Simplicial Hull for a Simple Polyhedron

In this section, we present an algorithm for constructing an inner interpolant simplicial hull from a simple polyhedron $\mathcal{P}$ with the following restrictions.

1. (Manifold) The polyhedron is manifold (i.e. two faces incident per edge,etc).

2. (Convex face) Every face is a convex polygon.

   A nonconvex polygon in a polyhedra can be easily subdivided into convex regions, even without splitting its neighboring faces. We give the details of this preprocessing step in Section 5.3.

3. (Corner normal) For every vertex, there exist a normal whose inner product with the normal of each incident face is positive.

   This restriction is used as a sufficient condition to guaranteer no intersection between neighboring face tetrahedra. It is actually commonly met by most of the polyhedra we are dealing with. The restriction is not so intuitive. Step 2(i) of the Algorithm 5.2.4 explains how to check if the restriction is satisfied. We give in Section 5.3 an algorithm to preprocess an offending polyhedron.

The above three restrictions are quite reasonable. They are actually commonly met by most of the polyhedra that we are dealing with. Based on such an observation, we put discussion about the preprocessing step in Sectionsec:preprocess.

For the sake of simplicity, we first assume that the input polyhedron has a closed boundary. The open boundary case is discussed in the second half of the section.

5.2.1 A Polyhedron with Closed Boundary

Algorithm 5.2.4 [Polyhedron Subdivision]

INPUT: A simple closed polyhedron $\mathcal{P}$ with convex faces.

1. *Decide the centroid F_i of each face f_i.* We use the term "centroid" in a generalized meaning: F can be any interior point of f_i. It can be part of the input for some special interpolatory effects (see Section 5.4), or by default,

$$F_i = \frac{1}{n}\sum_{j=0}^{n-1} V_j \tag{5.2.1}$$

   where V_j are vertices of f_i and n is the number of the vertices. Assign the face normal to the normal of F_i.

2. *Construction of triangulation $\mathcal{T}$.* (See Figure 5.2.1.)

   (i) *Compute the centroid and normal of each vertex.* For each vertex V_j, compute its centroid O and the normal at the centroid. Let the image of V_j

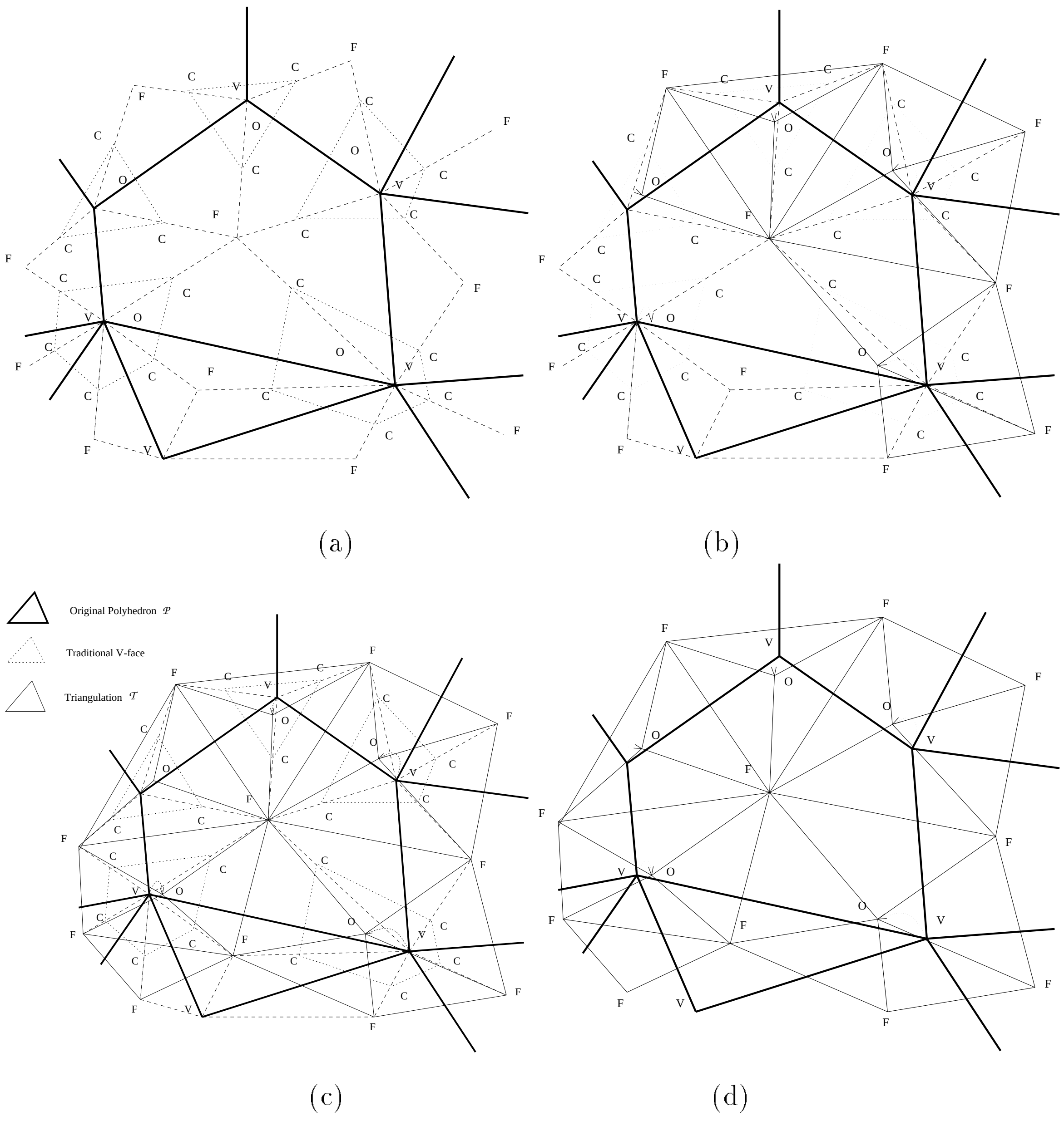


Figure 5.2.1 Construct triangulation $\mathcal{T}$ from polyhedron $\mathcal{P}$. (a) Constructing V-faces by linking the image vertices C's with respect to vertex-incident face pairs. (b) Constructing O's as the centroid of the V-faces. Start construction of triangulation $\mathcal{T}$. (c) A full diagram showing the relationship between $\mathcal{P}$, V-faces and triangulation $\mathcal{T}$. (d) $\mathcal{P}$ and $\mathcal{T}$.

on f_i,

$$C_{ji} = \alpha_{ij} \cdot F_i + (1 - \alpha_{ij}) \cdot V_j, \quad 0 < \alpha_{ij} < 1 \tag{5.2.2}$$

where F_i are the m centroids of the face around V_j and α_i are adjustable parameters. O_j, the centroid of vertex V_j is computed as

$$O_j = \frac{1}{m} \sum_{i=0}^{m-1} C_{ji} \tag{5.2.3}$$

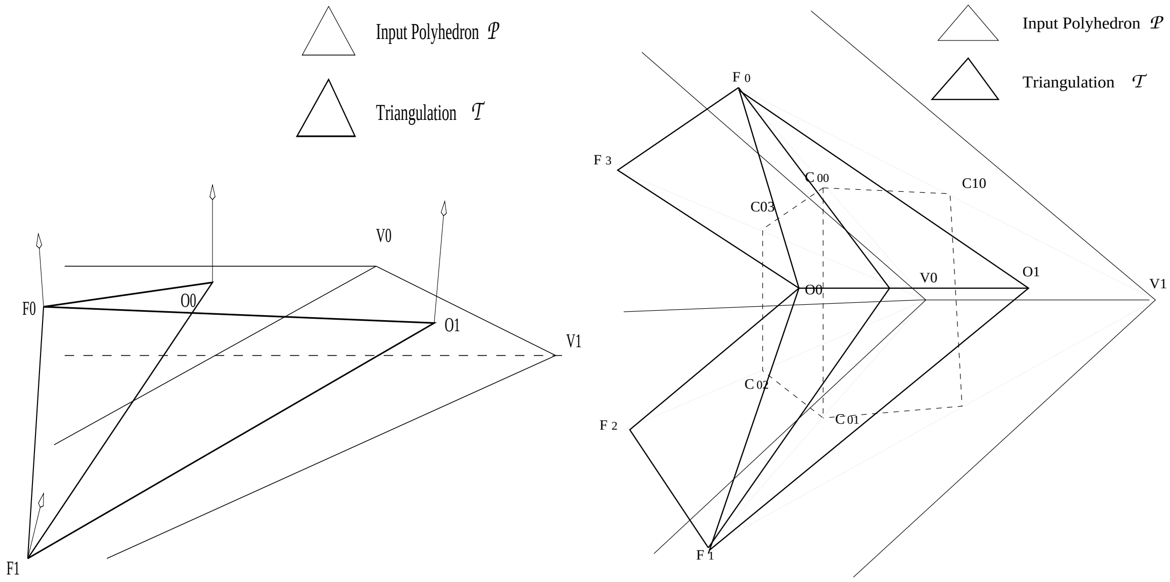
The convex hull of C_{ji}, $i = 0..m-1$, denoted as $\mathcal{C}(V_i)$, is also computed. (Can be done in $\theta(m \log m)$[113, 51].)

Let $H^+_{f_i}$ be the positive half space[2] divided by the plane that contains f_i. In the case that there are more than 2 distinct $H^+_{f_i}$'s, $i = 0, ..., m-1$, compute the intersection of the $H^+_{f_i}$, denoted as $H_j = \bigcap_{i=1}^m H^+_{f_i}$. There are known algorithms to compute the common interior and common exterior of a set of m half spaces in E^3 in time $\theta(m \log m)$ (see [113],section 7.3.2). If H_j is empty, then the input polyhedron offends restriction 3, we go back to the preprocessing stage to have the offending corners "smoothed" (see Section 5.3). If H_j is not empty, then H_j is a m-sided bottomless pyramid whose apex is V_j. Intersect H_j with a sphere centers at V_j and let N be the arithmetic average of all the intersecting points. Set n_{O_j}, the normal of O_j, to be vector $[V_j N]$. As $[V_j N] \subset H_j$, the inner product between n_{O_j} and n_{f_i}, the normal of each face incident to V_j is positive.

If there are at most 2 distinct $H^+_{f_i}$'s, set n_{O_j}, the normal at O_j to be the average of the face normals.

(ii) Triangulation $\mathcal{T}$. Now we construct a triangulation $\mathcal{T}$ by connecting (a) the centroids of adjacent faces, $[F_i F_k]$, and (b) O_j the centroid of a vertex V_j and that of its incident faces, $[O_j F_i]$ and $[O_j F_k]$. Note that each vertex of $\mathcal{T}$ is associated with a normal, pointing to the outside of the solid. Note that a face of $\mathcal{T}$ always consists of a vertex centroid O_j and the two face

[2]Positive half space is the one that contains the normal of the dividing plane



(a) (b)

Figure 5.2.2 Corner V_0 is rather flat. While the normals at F_0, F_1 and O_0 pointing upward, triangle $[F_1F_2O_0]$ are facing downward. Hence the face is normal-skew. (a) A top view and (b) A side view.

centroids F_i and F_k of the two adjacent faces f_i, f_k that are incident to the vertex V_j.

3. *Retriangulate $\mathcal{T}$ to get rid of the faces that are normal-skew.* Triangulation $\mathcal{T}$ is not guaranteed to be free of normal-skew faces. We enumerate the cases where normal-skew faces occur and discuss the methods to avoid them.

   1. See Figure 5.2.2. Corner V_0 is rather flat. While the normals at F_0, F_1 and O_0 pointing upward, triangle $[F_1F_2O_0]$ are facing downward. Hence the face is normal-skew. Figure 5.2.2(a) shows a top view and Figure 5.2.2(b) shows a side view. To eliminate the normal-skew face, we add a new vertex $E_{01} = \frac{1}{4}\sum_{i=0}^{3} C_{0i}$. Replace faces $[F_0F_1O_0]$ and $[F_0F_1O_1]$ by 4 new faces $[F_0E_{01}O_0]$, $[F_0E_{01}O_0]$, $[F_0E_{01}O_1]$ and $[F_0E_{01}O_1]$. Set the normal of E_{01} to be the average of the normals at F_0 and F_1. It can be shown that

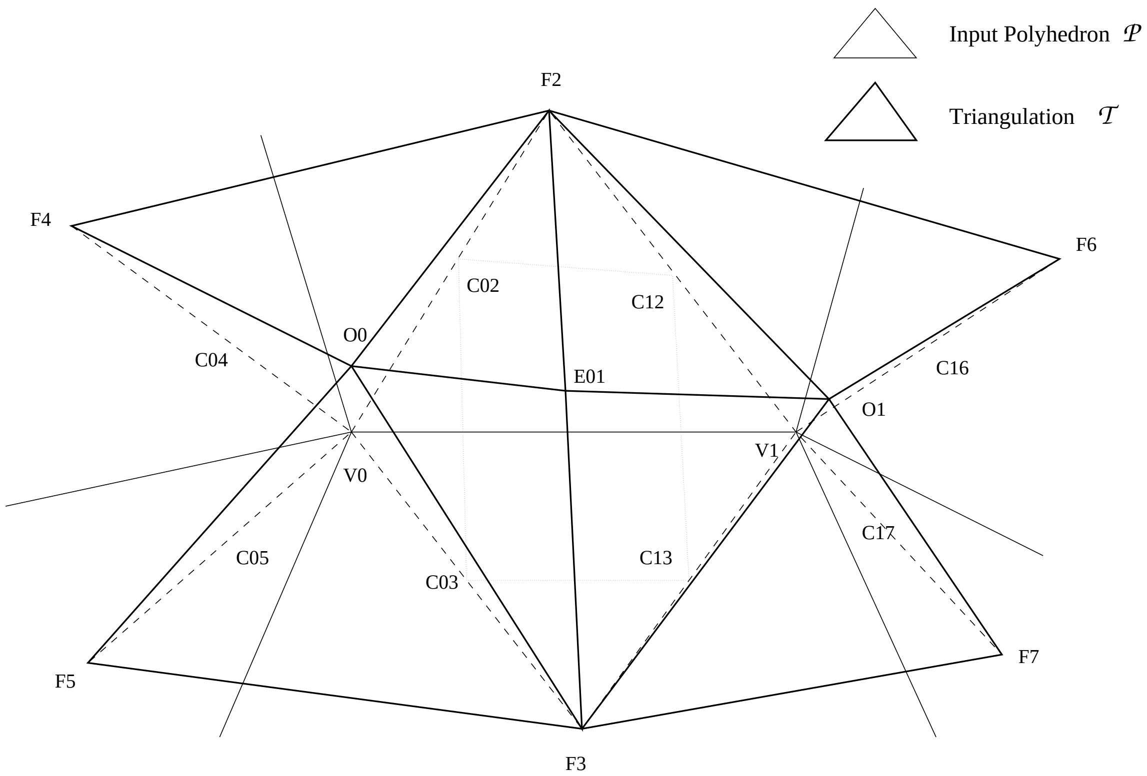
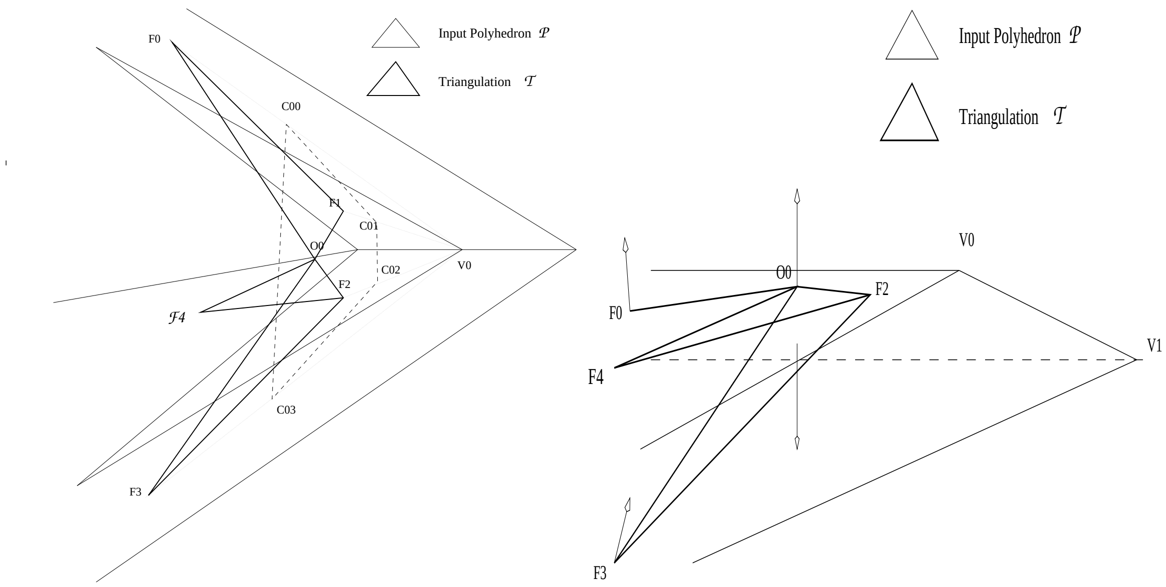


Figure 5.2.3 Subdivide $[F_2F_3O_0]$ and $[F_2F_3O_1]$ by inserting a new vertex E_{01}.



(a) (b)

Figure 5.2.4 Face $[O_0F_2F_3]$ faces downward, as most of the other faces face upward.

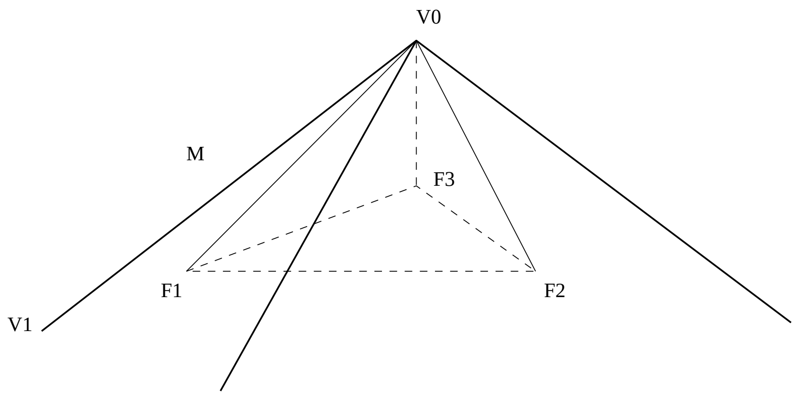


Figure 5.2.5 Construct a face tetrahedron on a trihedral vertex.

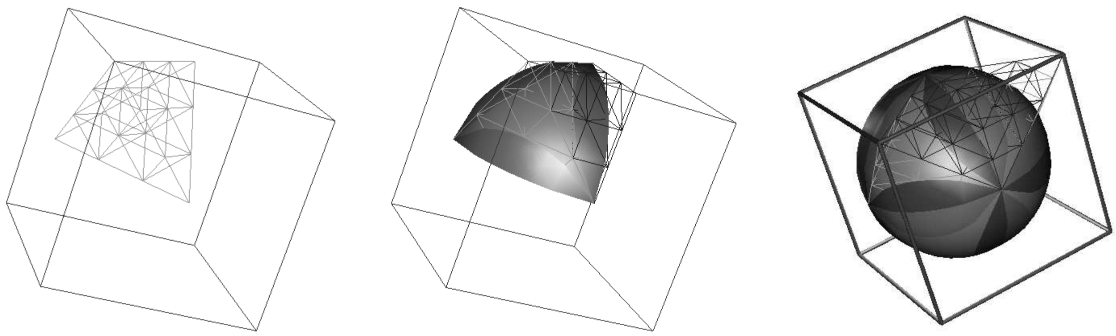


Figure 5.2.6 Corner cuts, inner simplicial hull and C^1 smooth A-patches. Trihedral corners.

the new faces are not normal-skew. Figure 5.2.3 illustrates this further subdivision.

2. See Figure 5.2.4 for a more offending case of normal-skew faces. Again, corner V_0 is rather flat. Even face $[F_0F_1O_0]$ and $[F_2F_3O_0]$ face downward, let alone $[F_0F_3O_0]$. Figure 5.2.4(a) shows a top view an Figure 5.2.4(b) shows a side view. We don't know how to eliminate this offending case by retriangulation. However, if O_0 is close to V_0 enough, face $[F_0F_1O_0]$ and $[F_2F_3O_0]$ face upward as desired. Hence one can always reduce cutting ratios α_i's to get rid of this problem.

4. Face tetrahedra construction based on $\mathcal{T}$ For each vertex V_0, *(i)* if vertex V_0 is a trihedral vertex, and its centroid O_0 is not involved in any retriangulation in step 3, build face tetrahedron $[F_1F_2F_3V_0]$, where F_i is the centroids of a face $f_i \in \mathcal{P}$ around V. (See Figure 5.2.5.)

(ii) if V_0 is non-trihedral, for each incident edge e, let f_2 and f_3 be e's incident faces and V_1 be the other vertex of e. We have the following three cases for a symmetric case where dihedral corner $\angle f_2 - V_1V_2 - f_3$ is less than $\frac{\pi}{2}$.

(a) A face that is not involved with retriangulation in step 3.

Let $[O_0F_2F_3] \in \mathcal{T}$ be a triangular face that is not involved with retriangulation in step 3. We build tetrahedron $[O_0F_2F_3Q]$ (and $[O_0F_2F_3Q']$ if a tetrahedron pair is needed). Let B be a point at edge e so that BF_2F_3 bisects dihedral angle $\angle O_0 - F_2F_3 - O_1$. Let R be the centroid of the intersection between $[O_0F_2F_3]$ and dihedral corner $\angle F_2 - V_0V_1 - F3$. We choose Q(and Q') from the following points.

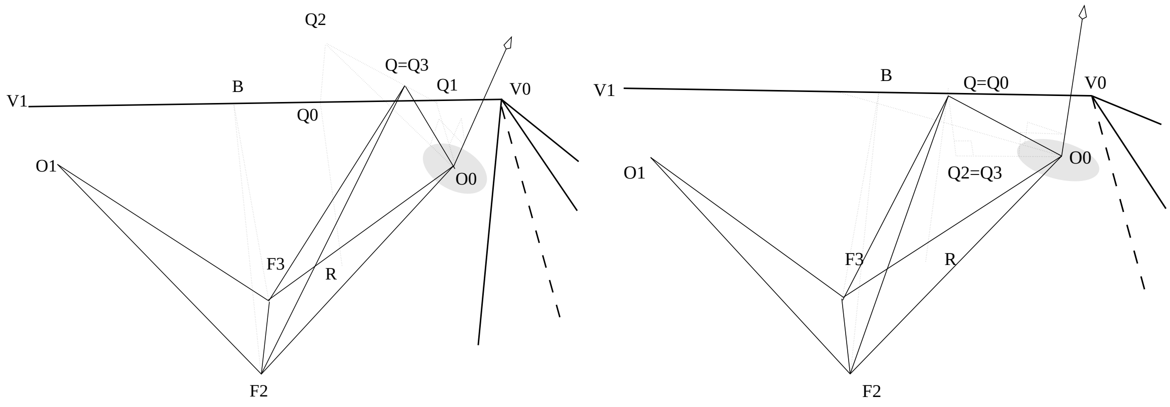
Q_0: Let Q_0 be a point on E so that RQ_0 is parallel to $[BF_2F_3]$.

Q_1: Let Q_1 be the intersection of ray BQ_0 and tangent plane at O_0, if they intersect. Undefined otherwise.

Q_2: Let Q_2 be a point on the tangent plane at O_0 and Q_0Q_2 is (1) coplanar with n_{O_0}; (2) parallel to $[BF_2F_3]$.

Q_3: If Q_1 does not exist, $Q_3 = Q_2$. If $[Q_1A_2]$ is outside the convex hull $\mathcal{C}(V_0)$, $Q_3 = Q_0$. If $[Q_1A_2]$ is inside $\mathcal{C}(V_0)$, $Q_3 = Q_2$. If $[Q_1A_2]$ intersects with $\mathcal{C}(V_0)$, Q_3 is the intersecting point.

Q_4: Let RS be a ray shooting to opposite direction of n_{O_0}. Let Q_4 be the intersection of Q_4 and the tangent plane at O_0, if exists. Undefined otherwise. Normally, Q_4 is in the same side of plane BF_2F_3 as O_0. If that is not the case, let Q_4 be a point on the tangent plane at O_0 and RQ_4 is parallel to BF_2F_3, coplanar with n_{O_0} and $RQ_4 \cdot n_{O_0} < 0$. Please note that as the inner products



(a) (b)

Figure 5.2.7 Face tetrahedron construction at a non-trihedral corner (I): single tetrahedron. (a) $[F_2F_3O_0Q_3]$ is constructed on $[F_2F_3O_0]$. $[F_2F_3O_0Q_0]$ is not tangent containing. (b) $[F_2F_3O_0Q_0]$ is constructed on $[F_2F_3O_0]$. In this example, Q_1 does not exist. Q_0R intersects the tangent plane at O_0. Hence the tetrahedron is tangent containing.

between n_{O_0} and the normal of face f_2 and f_3 are positive, Q_4 is inside the dihedral corner of $\angle f_2 - V_0V_1 - f_3$.

Q_5: Let Q_5 be R.

Check the convexity of $[O_0F_2F_3]$ to decide if a tetrahedron pair is needed. (See Definitions 3.4.2 and 3.4.3(3))

Single sided. If single sided is enough, Q_i,$i = 0, 3$ are the candidates for Q, the top of the tetrahedron. Among tetrahedra $[O_0F_2F_3Q_i]$, $i = 0, 3$, the first one that is tangent containment, in that order, is chosen. Figure 5.2.7 illustrates the two choices of Q.

Justification: $[O_0F_2F_3]$ does not intersect with f_2 and f_3 except at F_2 and F_3. If Q_1 exists, for any point Q_3 on line segment Q_1Q_2, $[O_0F_2F_3Q_3]$ is tangent containing. Otherwise, Q_0 is good enough. We choose them in that order as a heuristic of constructing tetrahedra of better shape.

Double sided. If a tetrahedron pair is needed. Choose Q from the first one among Q_i,$i = 0, 3$, and Q' the first one among Q_j,$j =$

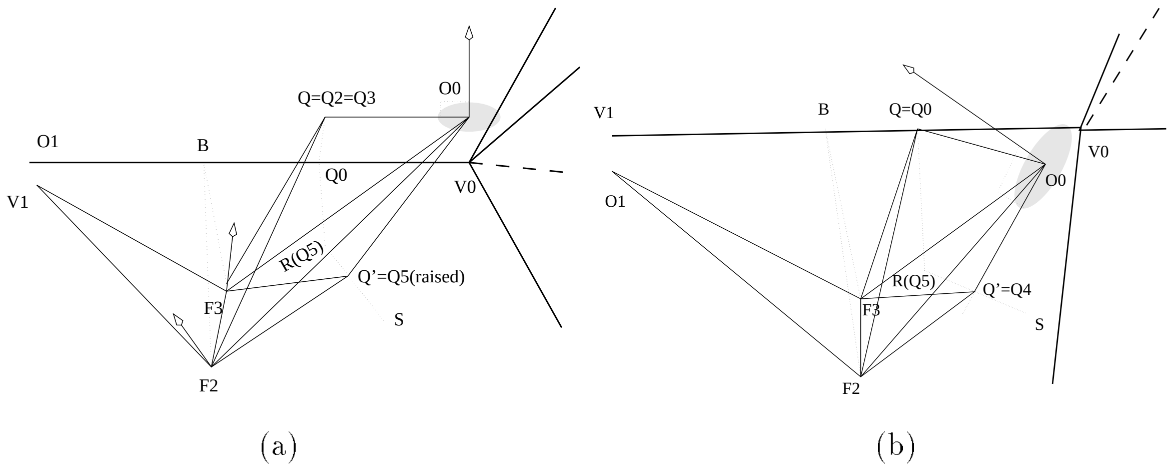


Figure 5.2.8 Face tetrahedron construction at a non-trihedral corner (II): tetrahedron pair. (a) As Q_2Q_5 intersects with all the tangent planes of O_0, F_2 and F_3, Q is chosen to be Q_2 and Q' is chosen to be Q_5. Q_1 and Q_4 do not exist. Q_5 is raised so that the tetrahedron is not flat. (b) $[Q_0Q_4]$ intersects with all the tangent planes, thus chosen as Q and Q'.

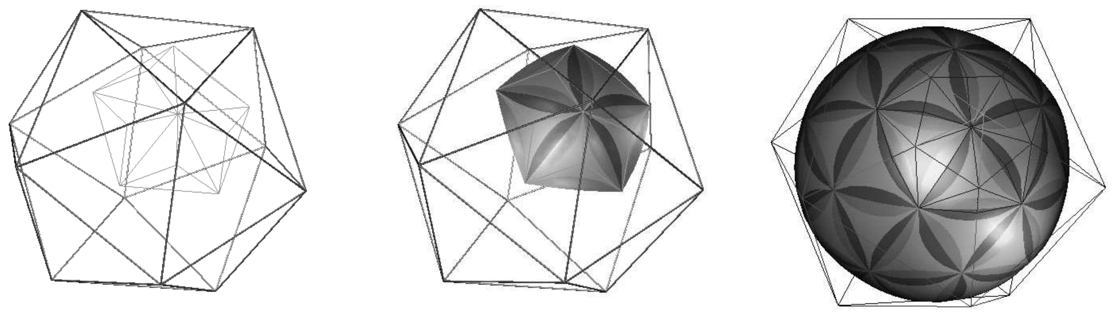


Figure 5.2.9 Corner cuts, inner simplicial hull and C^1 smooth A-patches. Nontrihedral corners.

4, 5 so that $[QQ']$ intersects with the tangent plane at O_0. The tetrahedron pair $[O_0F_2F_3Q]$ and $[O_0F_2F_3Q']$ is tangent containing. Figures 5.2.8 provides some illustrative examples.

Justification: By construction, $[QQ']$ intersects with the tangent plane at O_0. As to the other two tangent planes, all of Q_0,Q_1,Q_2 and Q_3 are outside of the dihedral corner $\angle f_2 - V_0V_1 - f_3$ and both of Q_4,Q_5 are inside the dihedral corner. Hence QQ' intersects with f_2 and f_3, or the tangent planes at F_2 and F_3. It follows that the tetrahedron pair is tangent containing.

(b) A face that is resulted from retriangulations in step(3)(a).

Let $[F_0E_{01}O_0], [F_0E_{01}O_0], [F_0E_{01}O_1], [F_0E_{01}O_1] \in \mathcal{T}$ be faces resulted from retriangulations in step 3(a). Consider a symmetric case $[F_0E_{01}O_0]$. Let Q_0 be where the tangent plane of E_{01} intersects with V_0F_0. Q_1 is a point on the tangent plane of O_0 such that Q_0Q_1 is parallel to normal n_{O_0}. Let Q_2 be the centroid of $[F_0E_{01}O_0]$. Choose Q (and Q' if needed) from $Q_i, i = 0, 1, 2$ so that $[F_0E_{01}O_0Q]$ (or pair $[F_0E_{01}O_0Q]$ and $[F_0E_{01}O_0Q']$) is tangent containing.

5. *Pumping up face tetrahedra.* Without intersecting each other, raise tetrahedra that are too flat. Raise the top of a tetrahedra vertically to the base triangle until it is high enough, for example, 1/8 of the average length of the base triangle. Reduce by half if it cause an intersection with neighboring tetrahedra.

6. *Construction of edge tetrahedra.* Build two edge tetrahedra between each two face tetrahedra that share an edge and whose tops are in the same side of $\mathcal{T}$.

OUTPUT: An interpolant simplicial hull Σ.

From the algorithm we have the following lemma.

Lemma 5.2.1 Every face tetrahedron $[p_ip_jp_kq_l] \in \mathcal{S}_f$ is tangent-containing.

However, as to whether or not neighboring tetrahedra intersect each other, we enumerate the following possible cases. We only enumerate the cases that are not involved with retriangulation in step 3. However, similar arguments are applied to faces with retriangulation.

1. "upper" face tetrahedra that share an edge connecting face centroids, for example $[F_2F_3]$. Impossible since they are separated by $[F_2F_3B]$.

2. "lower" face tetrahedra that share an edge connecting face centroids, for example $[F_2F_3]$. In the construction, RS is vertical to $[RO_0]$, hence the two "lower" face tetrahedra do not intersect if the plane angles $[O_0F_2F_3B]$ and $[O_1F_2F_3B]$ are less than $\frac{\pi}{2}$, which is normally true. Or at least one can choose smaller cutting ration α_i to make sure this.

3. "upper" face tetrahedra that share an edge connecting a face centroid and a vertex centroid, for example $[O_0F_2]$. If Q's are Q_0 or Q_1, they are points on the original input polyhedron, hence the tetrahedra do not intersect each other. If one or both of the Q's are Q_2 or Q_3, as they can be considered as some points vertically raised off the surface of the input polyhedron, along the same normal direction, they are not possible to intersect with each other.

4. "lower" face tetrahedra that share an edge connecting a face centroid and a vertex centroid, for example $[O_0F_2]$. The two Q's are all raised along the same direction. Hence they are not possible to intersect with each other.

So it is followed by the following lemma and theorem.

Lemma 5.2.2 Neighboring face tetrahedra in $\mathcal{S}_f$ do not intersect with each other.

Theorem 5.2.1 Σ is an interpolant simplicial hull.

One important property about the interpolant simplicial hull construction is that, for a convex corner, the simplexes built around the corner are all inside(or all outside) the corner. Hence the constructed surface are all inside (or all outside) the convex corner.

In [12, 17], extra subdivision are needed if the adjacent triangular faces of the triangulation $\mathcal{T}$ are coplanar to each other. Otherwise, all the bottom weights of the coplanar face tetrahedra are related to each other by the continuity constraints so that the locality property of the weight-setting procedure is destroyed and also the single sheeted condition is jeopardized. For the edge convex simplicial hull scheme presented in this section, as the triangulation of the polyhedron is also constructed instead of being given. By selecting "good" centroid combinations, one can construct a hull without any coplanar faces or only with some "trivial" coplanar cases, where coplanar face pair are isolated.

Figures 5.5.21 through 5.5.25 are examples that are smoothed with Algorithm 5.2.4.

5.2.2 Subdividing Polyhedra with Open Boundary

We extend the subdivision scheme to process a polyhedron with open boundary. We consider the shape of an open boundary to be more important than that of its neighboring surface, in the sense that it is more identifiable and easier to manipulate by interactive input from a user. Hence in our subdivision scheme, higher priority is given to the subdivision along a open boundary than to the faces next to it.

We extend the faces that are incident to an open boundary so that the polyhedron can be treated more or less as if it is closed. See Figures 5.2.10, 5.2.11 and 5.2.12. For a vertex V_0 at an open boundary, let V_1 and V_2 be V_0's neighboring vertices along the boundary. f_1 and f_2 are the faces incident to $[V_0V_1]$ and $[V_0V_2]$, respectively. Let F_1, F_2 be a point on $[V_0V_1]$ and $[V_0V_2]$, respectively, be the centroids of faces f_1 and f_2. we discuss the following three cases in detail.

V_0 has two incident faces f_1 and f_2. See Figure 5.2.10. The centroids of faces f_1 and f_2, $F_1 = \frac{V_0+V_1}{2}$, $F_2 = \frac{V_0+V_2}{2}$, as if f_1 and f_2 are extended. $[F_1F_2F_3V_3]$ is constructed to cover the trihedral corner. $[F_1F_2V_3V_0]$ is constructed to be the

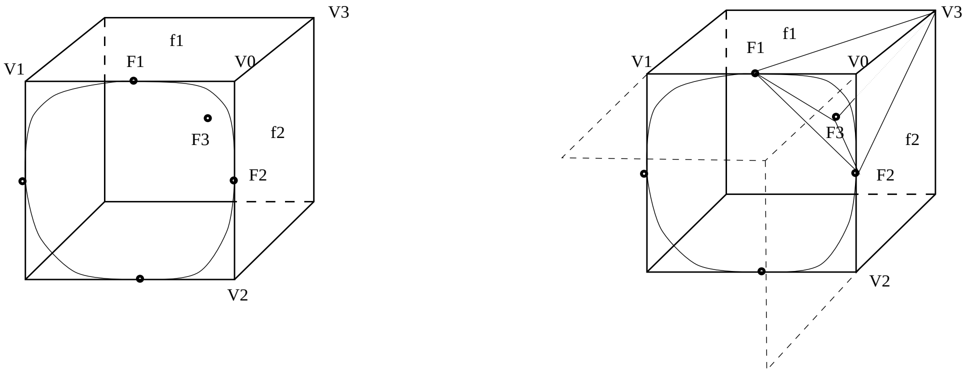


Figure 5.2.10 The boundary vertex V_0 has two incident faces f_1 and f_2. The centroids of faces f_1 and f_2 are set to be $F_1 = \gamma_1 V_0 + (1-\gamma_1)V_1$, $\gamma_1 \in (0,1)$, $F_2 = \gamma_2 V_0 + (1-\gamma_2)V_2$ $\gamma_2 \in (0,1)$, as if f_1 and f_2 are extended.

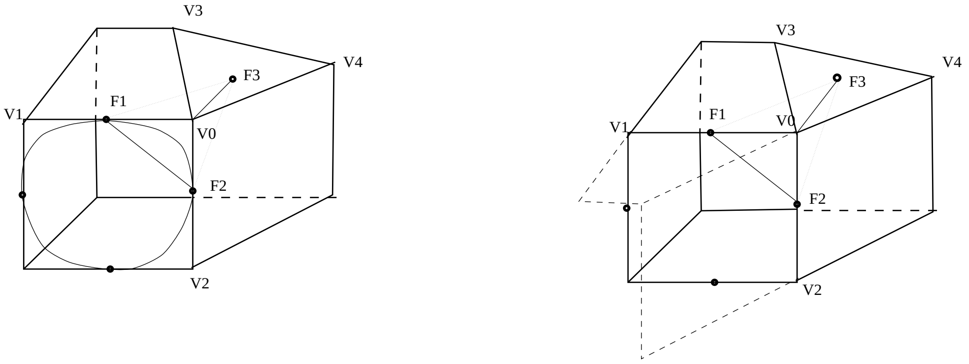


Figure 5.2.11 The boundary vertex V_0 has three incident faces, f_1, f_2 and f_3.

edge tetrahedron. V_0 could be regarded as the midpoint of the edge between the extended faces f_1 and f_2.

V_0 *has three incident faces* f_1*,* f_2 *and* f_3*.* See Figure 5.2.11. Again, faces f_1 and f_2 are extended. V_0 becomes a "non-boundary" trihedral corner. Hence $[F_1F_2F_3V_0]$ is constructed as in the trihedral case of Algorithm 5.2.4.

V_0 has more than three incident faces. See Figure 5.2.12.

$$C_1' = \alpha_1 \cdot F_1 + (1-\alpha_1) \cdot V_0, 0 < \alpha_1 < 1 \tag{5.2.4}$$

$$C_2' = \alpha_2 \cdot F_2 + (1-\alpha_2) \cdot V_0, 0 < \alpha_2 < 1 \tag{5.2.5}$$

$$O_0 = \frac{C_1' + C_2'}{2} \tag{5.2.6}$$

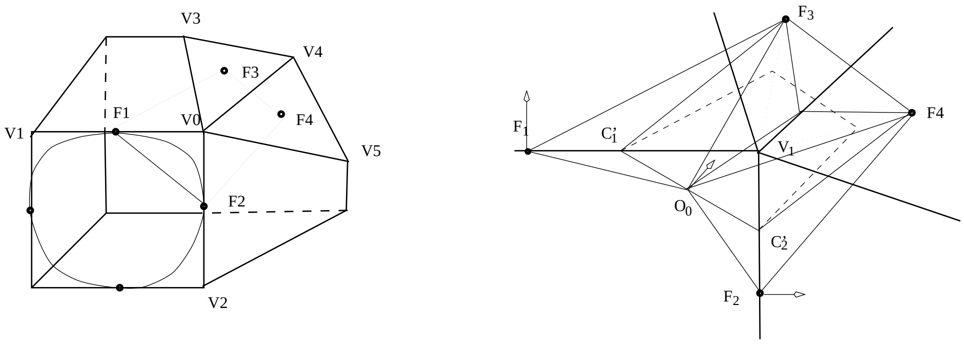


Figure 5.2.12 The boundary vertex V has more than three incident faces, f_1, f_2, f_3, f_4, ...

Compute the normal at O_0 as in Algorithm 5.2.4. Project this normal onto the plane vertical to $C_1'C_2'$ at O_0 and let the projected normal be the adjusted normal at O_0. Note that now $[C_1'C_2']$ lies on the tangent plane at O_0.

The simplicial hull construction around O_0 is just like it is in Algorithm 5.2.4 for triangular faces $[F_1F_2O_0]$, $[F_2F_3O_0]$, etc.. Let $[F_1F_3O_0Q_1]$ and $[F_4F_2O_0Q_2]$ be the face tetrahedra built on top of $[F_1F_3O_0]$ and $[F_4F_2O_0]$. Build edge tetrahedra $[F_1C_1'O_0Q_1]$ and $[F_2C_2'O_0Q_2]$ next to $[F_1F_2O_0Q_1]$ and $[F_1F_2O_0Q_2]$, respectively.

From the above scheme, we have the following theorem about the subdivision.

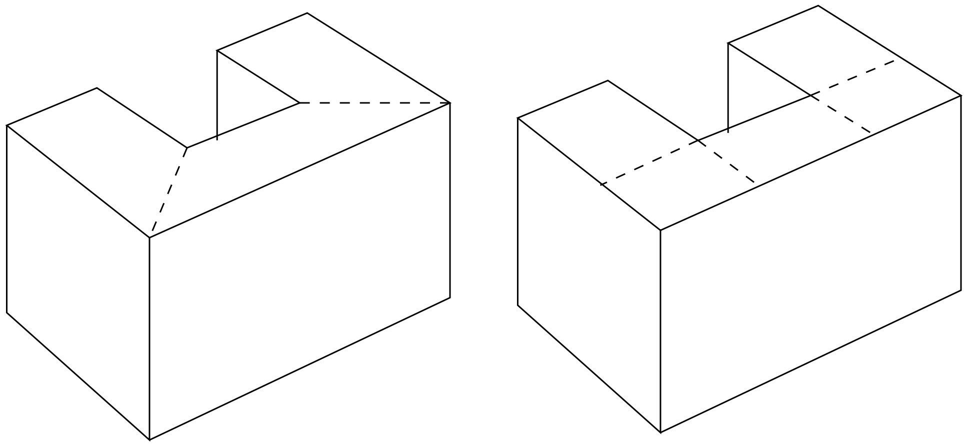
Theorem 5.2.2 If $V_0V_1...V_n$ is the open boundary of a polyhedron, and f_0, f_1, ..., f_{n-1} are the faces incident to edges at the boundary, respectively, then the boundary curve resulted from the above subdivision scheme is

1. C^1 smooth;
2. tangent to face f_i at F_i, where F_i is a point along V_iV_{i+1};
3. the segment between F_iF_{i+1} lies in triangle $[F_iV_iF_{i+1}]$.

The boundary curve is actually an A-spline in 3D[25].

5.3 Preprocessing of an Input Polyhedron

In this section, we discuss the schemes to preprocess an offending input polyhedron into one that meets the restrictions of Algorithm 5.2.4.



(a) (b)

Figure 5.3.13 Subdividing of a nonconvex face into convex faces (I). (a) using existing vertices only. (b) creating new vertices. Note that although the common edges are broken due to the new vertices, the adjacent faces are not subdivided.

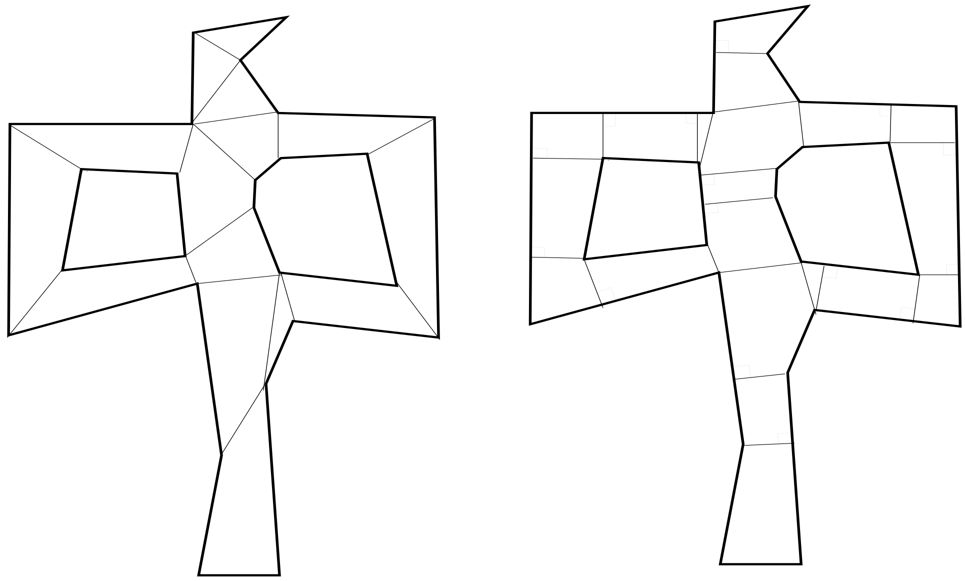
Here we only discuss the preprocessing for the last two restrictions as a non-manifold polyhedron is in general complicated.

5.3.1 Subdivide Nonconvex Polygonal Face on a Polyhedron

As the algorithm 5.2.4 in general does not work with polyhedra that have nonconvex polygonal faces, preprocessing step is needed to subdivide the nonconvex faces into convex ones.

Figure 5.3.13 shows two kinds of subdivisions, using the existing vertices only or adding new vertices. Note that, in the latter method, although new vertices are created, adjacent faces are not subdivided. Instead, only the common edges are broken at the new vertices.

Figure 5.3.14 illustrates two simple algorithms to subdivide a nonconvex polygon into convex regions. In Figure 5.3.14 (a), a nonconvex corners is connected by a new edge to the closest visible existing vertex. In Figure 5.3.14 (b), a nonconvex corners is connected by a new edge to the closest visible existing vertex, or the closet visible point on a non-incident edge, whichever is closer. In both algorithm, the original polygon is subdivided by the new edge. The process is perform recursively



(a) (b)

Figure 5.3.14 Subdividing of a nonconvex face into convex faces (II). (a) Connecting nonconvex corner to the closest visible corner. (b) Connecting nonconvex corner to the closest visible corner or closest visible point on an edge, whichever is closer.

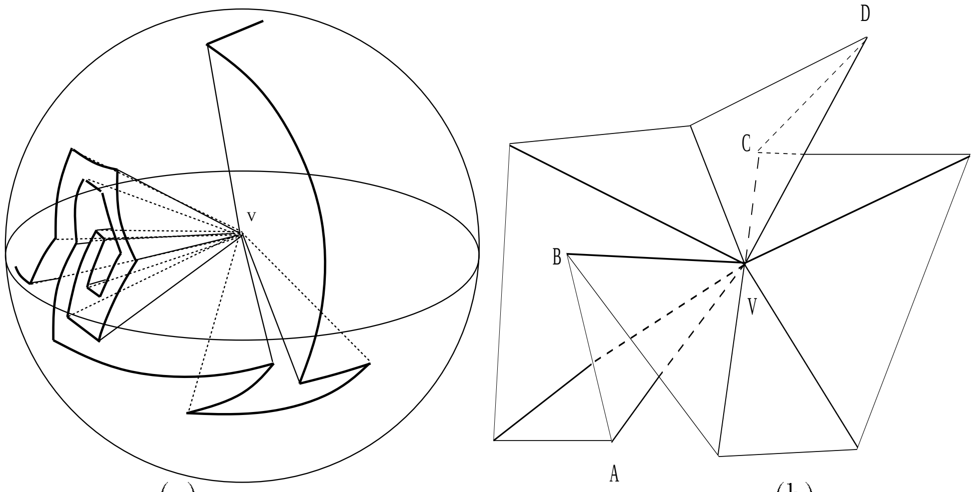
on the new polygons until all the corners are convex. Empirical shows that the second method is better for the next step, polyhedral subdivision. The reason is that the second method creates more “rectangle-like” regions, which leads to more regular tetrahedra when the polyhedron is subdivided.

5.3.2 The “Brick Wall” Subdivision of a Corner Offending Restriction 3

Figure 5.3.15 illustrates two corners that offend restriction 3. We present an algorithm to “smooth out” an offending corner of restriction 3.

The algorithm consists of two passes. The first pass figures out how many subdivision are needed, the second pass computes the final subdivision. Let V be a vertex of m incident faces. Let E_i^0, $i = 0, m-1$ be the midpoints of the m incident edges. The E_i^0's form a space polygon, which is used as the initial configuration of the following iterative process. At the j-th step, if the corner formed by faces

$$[E_i^j E_{(i+1) \bmod m}^j V] \quad i = 0, ..., m-1$$



(a) (b)

Figure 5.3.15 Two offending corners. (a) A "wild" corner intersects with a sphere. Shown in bold is the intersection curve. (b) One of the simplest offending corners. As faces $[VAB]$ and $[VCD]$ are facing downwards from directions with respect to the circular order of V's incident faces, no normal meets restriction 3.

meets restriction 3, stop the iteration and record the current j in k. Otherwise, for $i = 0, m-1$ do

$$E_i^{j+1} = \frac{E_i^j + E_{(i+1)\bmod m}^j}{2}. \tag{5.3.7}$$

It can be shown the limit of this iteration is a disk in space. Hence there exists a k such that the corner formed by $[E_i^k E_{(i+1)\bmod m}^k V]$, $i = 0, ..., m-1$ meets restriction 3. Normally, k is a small number. For the corner shown in Figure 5.3.15(b), $k = 1$ is sufficient.

Note that during the interactive process, the E_i^j's are contracted towards V. But that has nothing to do with our concern, that is, the relations between the normals.

Knowing that k subdivisions are sufficient, we compute the final subdivision in the following way. Initially, let $D_i^0 = E_i^0$, $M_i^0 = \frac{D_i^0 + D_{(i+1)\bmod m}^0}{2}$, $i = 0, ..., m$. For $j = 1, ..., k-1$

$$D_i^j = \frac{1}{k+1-j} M_i^{j-1} + \left(1 - \frac{1}{k+1-j}\right) V \tag{5.3.8}$$

$$M_i^j = \frac{D_i^j + D_{(i+1)\bmod m}^j}{2} \tag{5.3.9}$$

$$(5.3.10)$$

The corner is subdivided into 6 sided cells

$$[M_i^j D_{i+1}^j M_{i+1}^j D_{i+1}^j M_i^j D_i^j]$$

and 4 sided cell

$$[M_i^{k-1} M_{i+1}^{k-1} V].$$

In a 6 sided cell, all the vertices except M_i^j are coplanar; Furthermore, edge

$$[M_i^j D_i^{j+1}]$$

are coplanar with points

$$M_i^{j-1}, M_{i+1}^{j-1}, D_{i+1}^j, M_i^j, D_i^j$$

in

$$[M_i^{j-1} D_{i+1}^{j-1} M_{i+1}^{j-1} D_{i+1}^j M_i^j D_i^j].$$

We call it the "brick wall" corner subdivision as the subdividing cells lined up like a brick wall. Figure 5.3.16 illustrates such a subdivision.

Figure 5.3.17 shows the construction of interpolant simplicial hull on a subdivided corner. Let F_i^{j+1} be the centroid of a 6 sided cell

$$[M_i^j D_{i+1}^j M_{i+1}^j D_{i+1}^j M_i^j D_i^j],$$

F_i^k be that of a 4 sided cell, $[M_i^{k-1} M_{i+1}^{k-1} V]$. In both case, the normal at F_i^{j+1}, denoted as $n_{F_i^{j+1}}$, is computed by connecting F_i^{j+1} to all the vertices in the cell, and then averaging the normals of the triangular faces. For any $i = 0, ..., m-1$ and any $j = 0, ..., k-1$, build tetrahedra $[F_i^j F_{i+1}^j F_i^{j+1} D_{i+1}^j]$ and $[F_i^j F_{i-1}^{j+1} F_i^{j+1} M_i^j]$. It can be shown that $[F_i^j F_{i+1}^j F_i^{j+1} D_{i+1}^j]$ is always tangent containing as face $[F_i^j F_{i+1}^j F_i^{j+1}]$ are always convex. A tetrahedron of the second kind, $[F_i^j F_{i-1}^{j+1} F_i^{j+1} M_i^j]$ may not be tangent containing. If face $[F_i^j F_{i+1}^j F_i^{j+1}]$ is nonconvex, let l be a line that passes M_i^j and intersects the tangent planes of F_i^j, F_{i-1}^{j+1}, F^{j+1} and $[F_i^j F_{i-1}^{j+1} F_i^{j+1}]$ at Q_0, Q_1, Q_2 and Q_4, respectively. Choose the "out-most" pair to be Q and Q' and build tetrahedron pair $[F_i^j F_{i-1}^{j+1} F_i^{j+1} Q]$ and $[F_i^j F_{i-1}^{j+1} F_i^{j+1} Q']$.

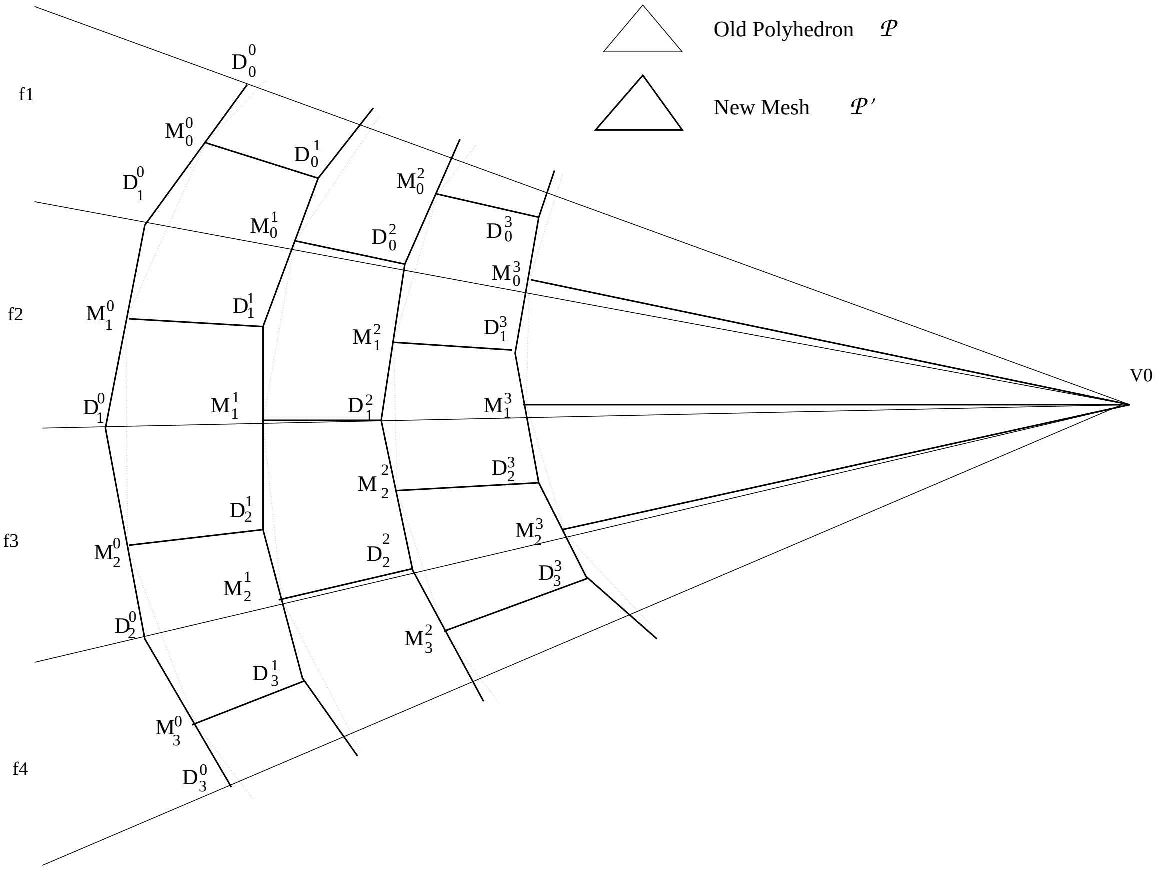


Figure 5.3.16 Subdivision of a corner that offends restriction 3. The corner is divided into 6 sided cells, e.g. $[M_0^1 D_1^1 M_1^1 D_1^2 M_1^2 D_0^2]$, and 4 sided cells, e.g. $[M_1^3 D_2^3 M_2^3 V]$. For a 6 sided cell, e.g. $[M_0^1 D_1^1 M_1^1 D_1^2 M_1^2 D_0^2]$, all vertices but D_1^1 are coplanar, $[M_1^1 D_1^2]$ is coplanar to vertices M_1^0,D_1^1, M_2^0, D_2^1.

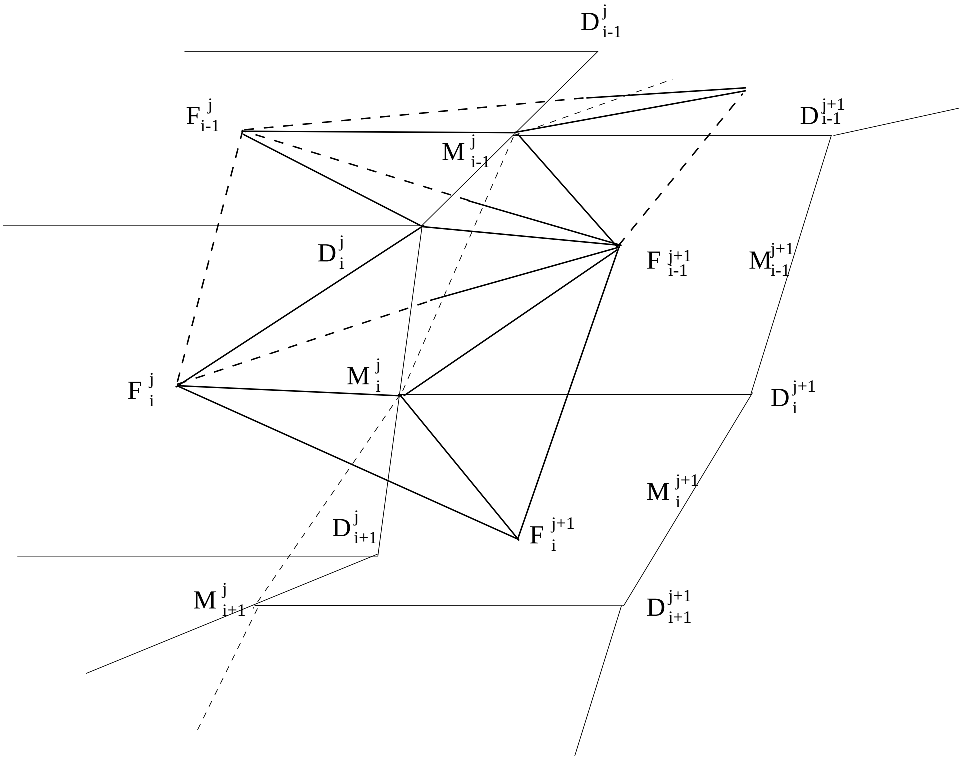


Figure 5.3.17 Construction of interpolant simplicial hull on a subdivided corner.

5.4 Free-form Modeling with A-patches

In this section, we discuss from a higher level about how one can use A-patch for free form modeling.

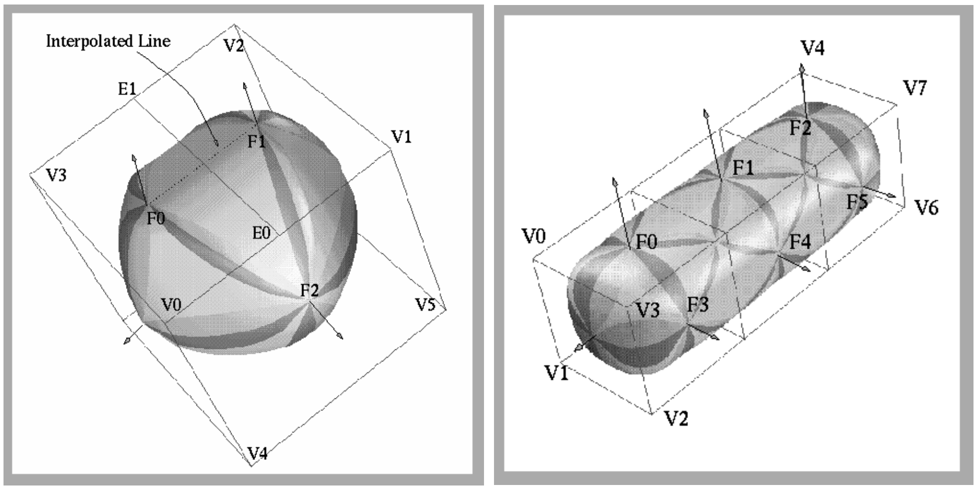
The first level of shape control is the input polyhedron. By making simple modification to the input polyhedron, including changing face centroid positions, one can subsequently change the shape of the resulting smooth surface. The next level of shape control parameters are the cutting ratio in Algorithm 5.2.4 and the free weights within the cubic patches.

In Algorithm 5.2.4, we normally build a single tetrahedron to "cover" a trihedral corner, except when retriangulation of $\mathcal{T}$ is needed near the corner (Step 3 of Algorithm 5.2.4), in which case the corner is treated like a non-trihedral corner. To be simple and intuitive, in the context of this section, we assume that a trihedral corner is always "covered" by a single tetrahedron.

5.4.1 Interpolating Lines and Polygons

One can easily modify the input polyhedron so that the resulting surface smoothly interpolates a straight line segment (see Figure 5.4.18) or a polygonal region (see Figure 5.4.19). We use the special A-patch described in Figure 3.2.3.

In order to interpolating a line segment $[F_0F_1]$ on a face of $\mathcal{P}$, one only needs to subdivide the face into two, with F_0 and F_1, one on each side, and call them the centroids of the corresponding new faces. See Figure 5.4.18(a). In order to interpolate line segment $[F_0F_1]$, $[E_1E_2]$ subdivides face $[V_0V_1V_2V_3]$ into two new faces. Let F_0 and F_1 be the face centroids. An adjacent face, for example, $[V_0V_4V_5V_1]$ is turned into $[V_0V_4V_5V_1E_0]$. Since E_0 is a trihedral corner, tetrahedron $[F_0F_1F_2E_0]$ is built for this corner. By setting the weights on edge $[F_0F_1]$ to be all zero, the three sided patch defined within $[F_0F_1F_2E_0]$ interpolates the edge. Actually, the patch is tangent to face $[V_0V_1V_2V_3]$ throughout $[F_0F_1]$ as it is C^1 continuous with the patch constructed at



(a) (b)

Figure 5.4.18 Interpolating line segments. (a) Interpolating $[F_0F_1]$. (b) Interpolating $[F_0F_1F_2]$ and $[F_3F_4F_5]$, etc.

corner E_1. Figure 5.4.18(b) shows surface that interpolates line segments in adjacent faces.

Interpolation of a polygonal region is a natural extension of the above technique. See Figure 5.4.19(a). In order to interpolates polygon $[F_1F_2F_3F_4]$ on face $[V_0V_1V_2V_3]$, we subdivide the face into $[V_0E_0F_0E_3]$, $[V_1E_1F_0E_0]$, $[V_2E_2F_0E_1]$, $[V_3E_3F_0E_2]$, with F_1, F_2, F_3 and F_4 one on each face. Furthermore, let F_1, F_2, F_3 and F_4 be the centroids of the faces, respectively. Similar to the line-interpolating case, subsequent subdivision constructs surface that C^1 interpolates the polygon. Figure 5.4.19(b) shows a surface that interpolates two squares, one on each of two adjacent faces. Around region $[F_0F_1F_5F_4]$, A fillet blends between the two square. Please note that the interpolated polygons need not to be convex.

5.4.2 Shape Control by Changing Cutting Ratios and Free Weights

The next level of shape control parameters are the cutting ratios (Formula 5.2.2) and the free weights. They control how much the input polyhedron is corner-cut.

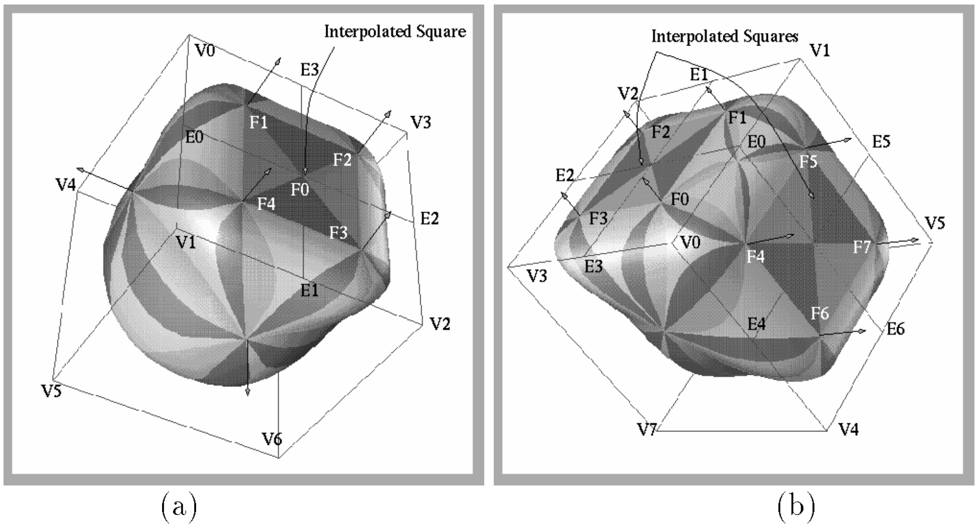


Figure 5.4.19 Interpolating rectangular regions. (a) Interpolating $[F_0F_1F_2F_3]$. (b) Interpolating $[F_0F_1F_2F_3]$ and $[F_4F_5F_6F_7]$.

For a non-trihedral corner, the smaller the ratios α_i's are, the closer the surface is "attracted" to the corner. If α_i are around 0.5, the corner become rounded. See Figure 5.5.24 (a) vs (c), (f) vs (g).

For a trihedral corner, only one tetrahedron built over it, the "cutting ratios" are in the form of free weights a_{0003}, a_{1002}, a_{0102} and a_{0012}(see Figure 3.5.1, the ④ ⑤ weights). The smaller they are in absolute value, the closer the surface is "attracted" to the corner. See Figure 5.5.23(a), (c), (e) and (e), (h), (i) for the effect of changing the free weights a_{0003}, a_{1002}, a_{0102} and a_{0012}. Also see Figure 5.5.25 (a) vs (b), (c) vs (d).

In order to have a simple and unified parameter controlling how much a corner is rounded, for a non-trihedral corner, we impose that all the cutting ratios around a corner V_0 are always set to be the same, α_0^*. For a trihedral corner, we unify all the ④ ⑤ weights to be a^* and let

$$a^* = 2 * \alpha_0^*(3f_3(1/3, 1/3, 1/3) + f_4(1/3, 1/3, 1/3)) \tag{5.4.11}$$

where f_3 is the bivariate BB function defined over the third layer of the simplex (weights with $\lambda_4 = 1$), f_4 is the bivariate BB function defined over the bottom layer of the simplex (weights with $\lambda_4 = 0$). With such a scaling, when $\alpha_0^* = 0.5$, the surface passes through a point O_0 as if the corner were cut like a non-trihedral corner with a cutting ratio of 0.5. Please note that the ④ and ⑤ weights set here are just "suggested" values. The weight setting process described in Chapter 3 may slightly change them to enforce the functional constraints.

The free weights in the edge patch b_{2001} (see Figure 3.5.1) is used to achieve the concave effect shown in Figure 5.5.23 (b), (f) and (g). The larger b_{2001} is in absolute value, the more concave the surface near the midpoint of the edge.

5.4.3 Modeling C^0 Features with Singularity

If we allow the cutting ratios α_i's to be zero, and the vertices and edges of a simplex are not necessary to be smooth(see Definitions 3.2.1, 3.2.2, and Figure 3.3.4

for non-smooth patches), then we would be able to represent C^1 and C^0 surface by a unified model.

If all the cutting zero is zero, we can exactly represent the original input polyhedron with a degenerated configuration. For a non-trihedral corner, the face tetrahedra built around this corner are all degenerated into triangles. See Figure 5.2.7, V_0, O_0 and Q_3 become the same point, according to Algorithm 5.2.4. The surface is degenerated into two edges $[V_0F_2]$ and $[V_0F_3]$, (see also Figure 3.3.4(a)). Imagining that A and B become the same point). For a face tetrahedron built on a trihedral corner, weights a_{0003}, a_{1002}, a_{0102} and a_{0012} (see Figure 3.3.4(a). See also Figure 3.5.1,the ④ and ⑤ weights) are all set to be zero. The surface is degenerated into three edges.

In both cases, the edge patch is $[F_2F_3V_0M]$ (Let M be the midpoint of $[V_0V_1]$). All weights except b_{1110}, b_{0111}, b_{0210} and b_{0120} are set to be zero, hence the surface is degenerated to two branches that coincide with $[F_2V_0M]$ and $[F_3V_0M]$. Figure 3.3.4(c) in Subsectionsec:sing-patches illustrates this kind of singular patch.

In a word, the degenerated edge patches "took over" to represent the polyhedron, while the face patches degenerates into edges. See 5.4.20(f) for a C^0 surface that covers a cube.

By specifying zero cutting ratios in some corners and non-zero ratios in the others, we can achieve surface that mixed C^0 and C^1 features. features (singularities) that coincides with part of part $\mathcal{P}$, and nonsingular surface that approximate the other part of $\mathcal{P}$ smoothly. We enumerate the following three cases.

(1) Touching corner. If the cutting ratios around a corner is zero (the ④ and ⑤ weights are zero in the case of trihedral corner), and the edge patches around this corner are smooth except at P_4 (namely, see Figure 3.5.1, the ④ , ⑤ and ⑦ weights are all zero, also illustrated in Figure 3.3.4(b) of Subsection 3.3(b)), then the surface is touching the corner with a singular point. The surface interpolates the line segments connecting the corner to the face centroids ($[F_2V_0]$ and $[F_3V_0]$in Figure 5.2.7. etc. $[P_2P_4]$ and $[P_3P_4]$ in Figure 3.5.1). See also 5.4.20(a) and (b).

(2) Cutting edge. If the cutting ratios of two neighboring corners are set to be zero, and the two edge patches between them are set to be edge non-smooth (namely, the ④ , ⑤ , ⑦ , ⑧ and ⑨ weights in Figure 3.5.1 are all set to be zero. See also Figure 3.3.4(c)), then the surface is locally degenerated along the edge and coincide with the polyhedron at the part. See Figure 5.4.20(c) and (d).

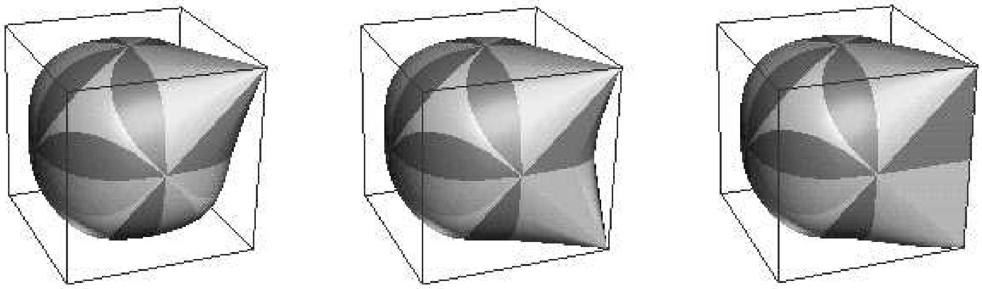
(3) Flat top. If the cutting ratios of the corners of a face are all set to be zero and furthermore all the edge patches between these corners are set to be "cutting edge", then the surface is coincident with this face of $\mathcal{P}$ and C_0 continuous to the rest of the surface. See Figure 5.4.20(f).

Allowing degenerated and singular patch, the configuration space of the surface is extended to a half-closed interval, with C^0 continuous input polyhedron $\mathcal{P}$ at the closed end. Such a unified representation allows C^0 and C^1 continuous surface co-exist in a single model, which will lead to easy creation and manipulation of a surface of this kind.

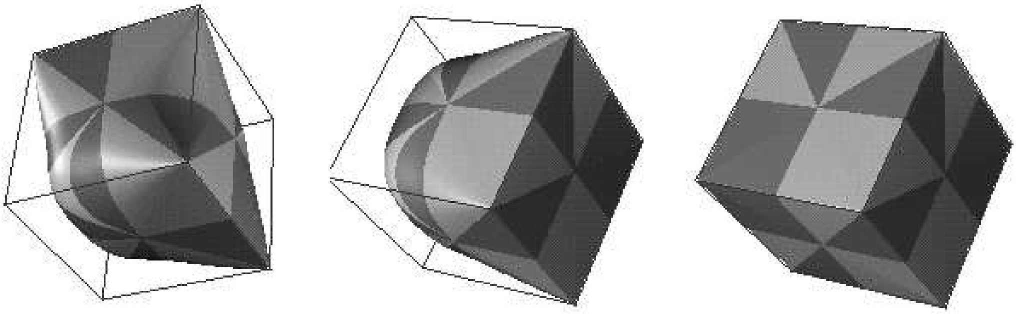
Figure 5.4.20 shows how C^0 and C^1 features can be mixed to approximate a cube in different shapes.

5.5 Implementations

We have presented algorithms for approximating a three dimensional polyhedron with C^1 cubic A-patches. These algorithms have been implemented in the SplineX and SHILP toolkits of the distributed and collaborative geometric design environment SHASTRA [3]. See Figures 5.5.21-5.5.25. We are using them for interactive free-form design. SHILP is an X11 based, interactive solid modeling system, which is used to create a simplicial (face triangulated) polyhedral model of a desired shape. This model could also be the triangulation of an arbitrary surface in three dimensions. This triangulation is C^1 smoothed by a client/server call to a SplineX computation using inter process communication. SplineX is an X11 based, interactive surface modeling toolkit for arbitrary algebraic surfaces (implicit or parametric) in BB form. It allows for the creation of simplex chains (for example, the interpolant simplicial hull



(a) (b) (c)



(d) (e) (f)

Figure 5.4.20 Modeling with singularities. (a) Touching a vertex with a singular point. (b) Touching two vertices with singular points. (c) Cutting an edge with a singular edge on the surface. (d) Cutting two edges. (e) C^0 interpolates a face of a cube. (f) The surface is degenerated into the cube itself. All the edges are singular.

of the triangulation) and the interactive change of control points and weights of the A-patches for shape control. SplineX also has the ability to distribute its rendering tasks (for the display of the individual A-patches) on a network of workstations, to achieve maximal display parallelism.

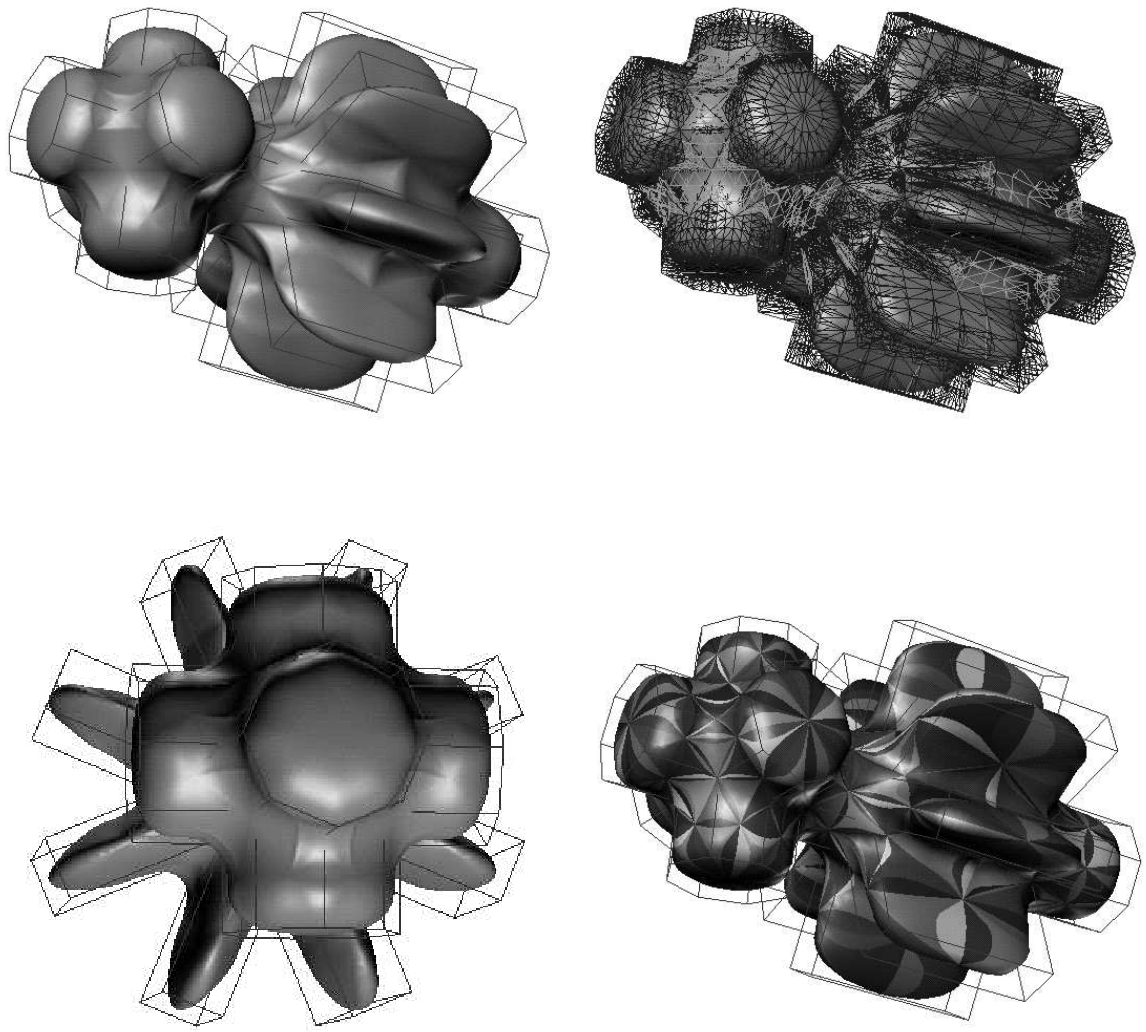


Figure 5.5.21 Smoothing a satellite-like object with cubic A-patches. Upper right also shows the simplicial hull, while bottom right shows the individual cubic a patches.

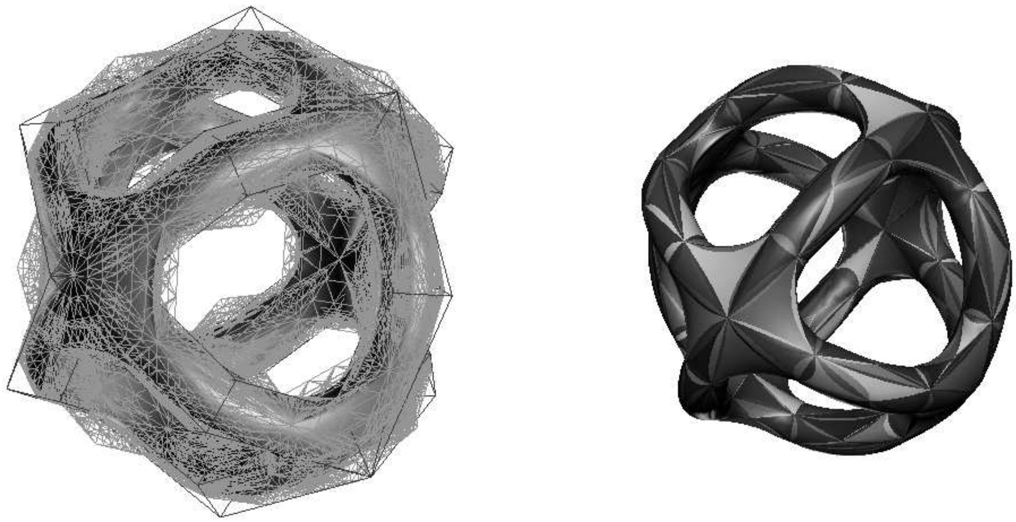
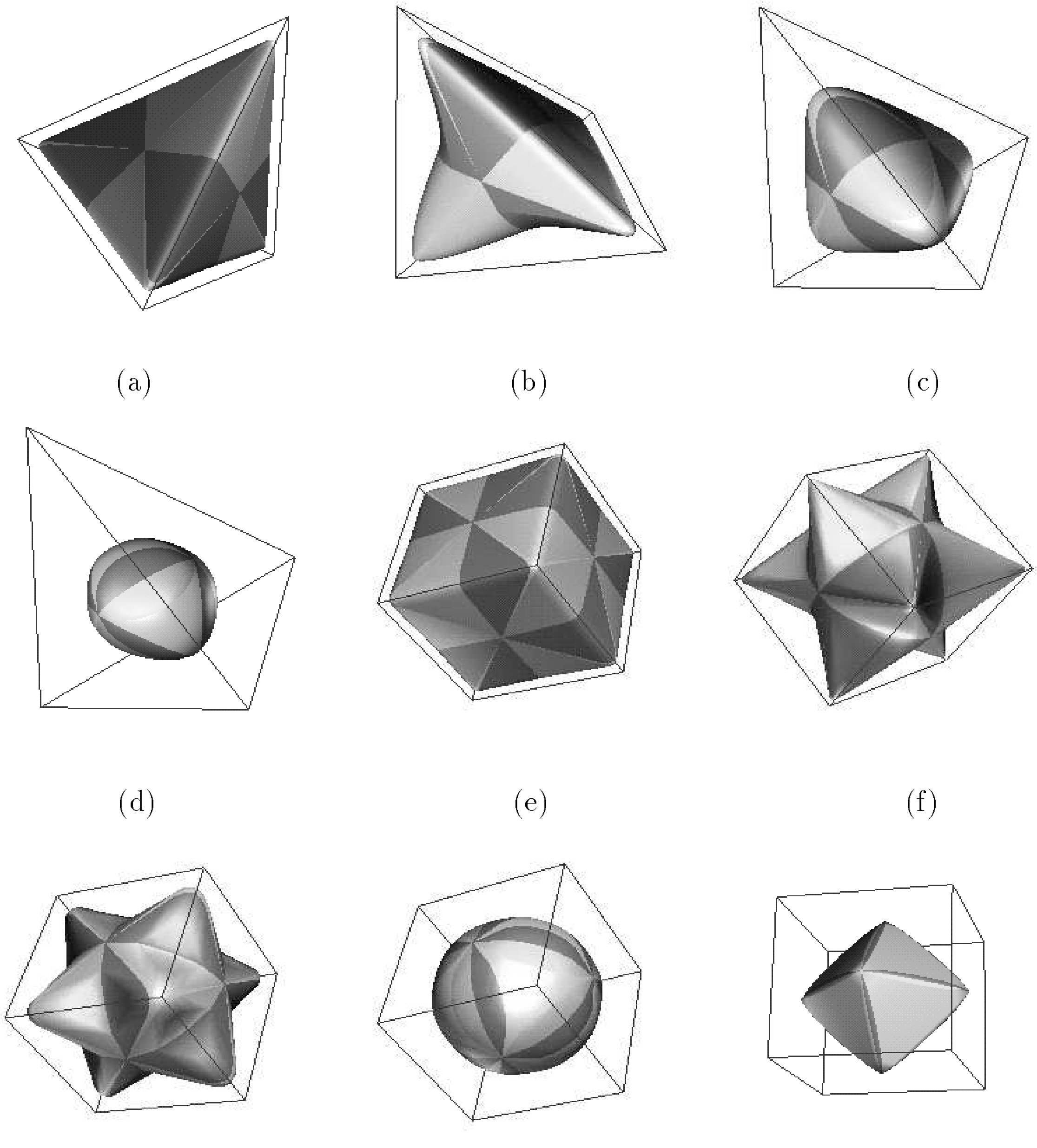


Figure 5.5.22 Smoothing a tri-torus with cubic A-patches

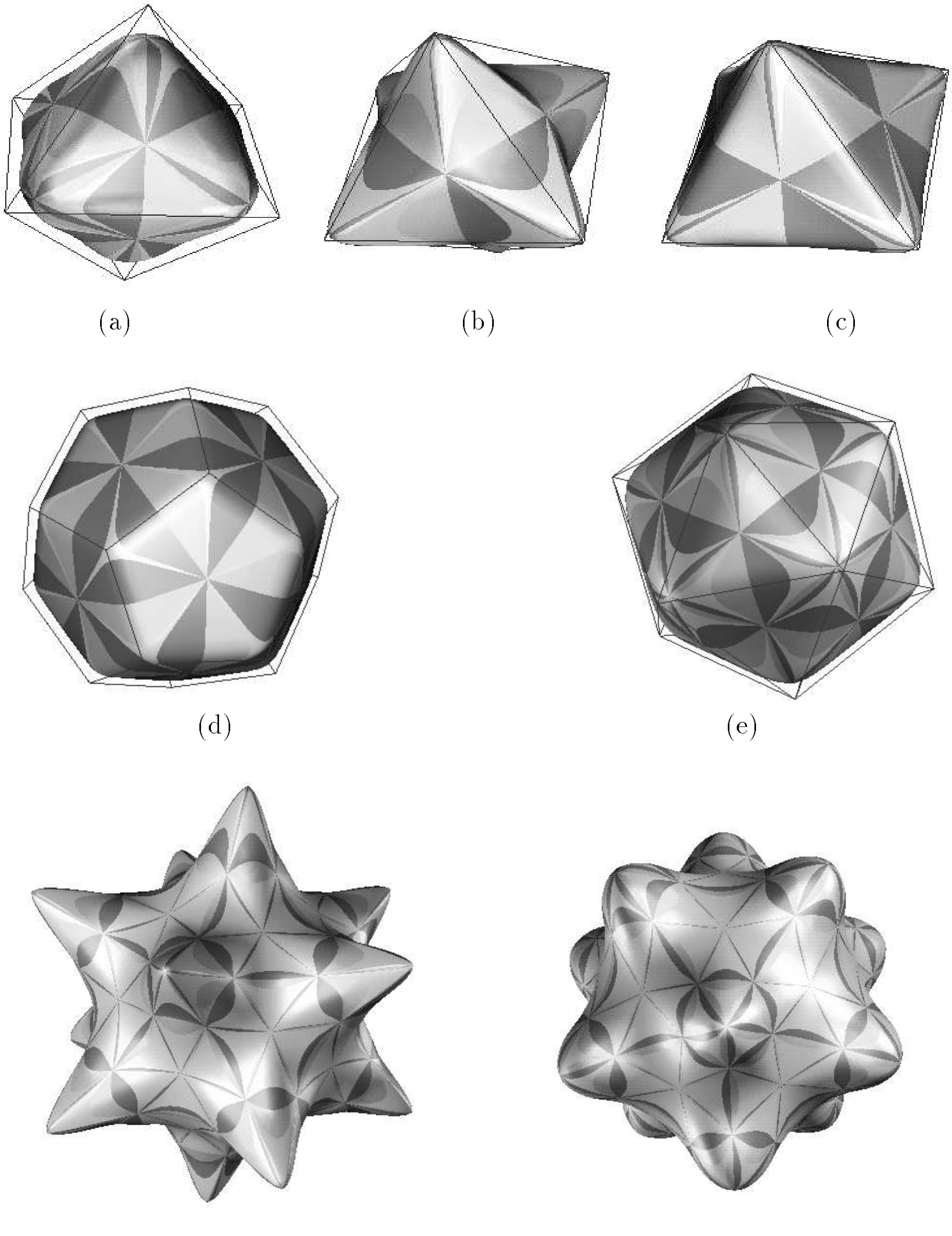


(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 5.5.23 Tetrahedra and cubes. All the corners are trihedral. For a trihedral corner, we control the shape of the neighboring surface by changing only the free weights in the face patch built at the corner and the surrounding edge patches.

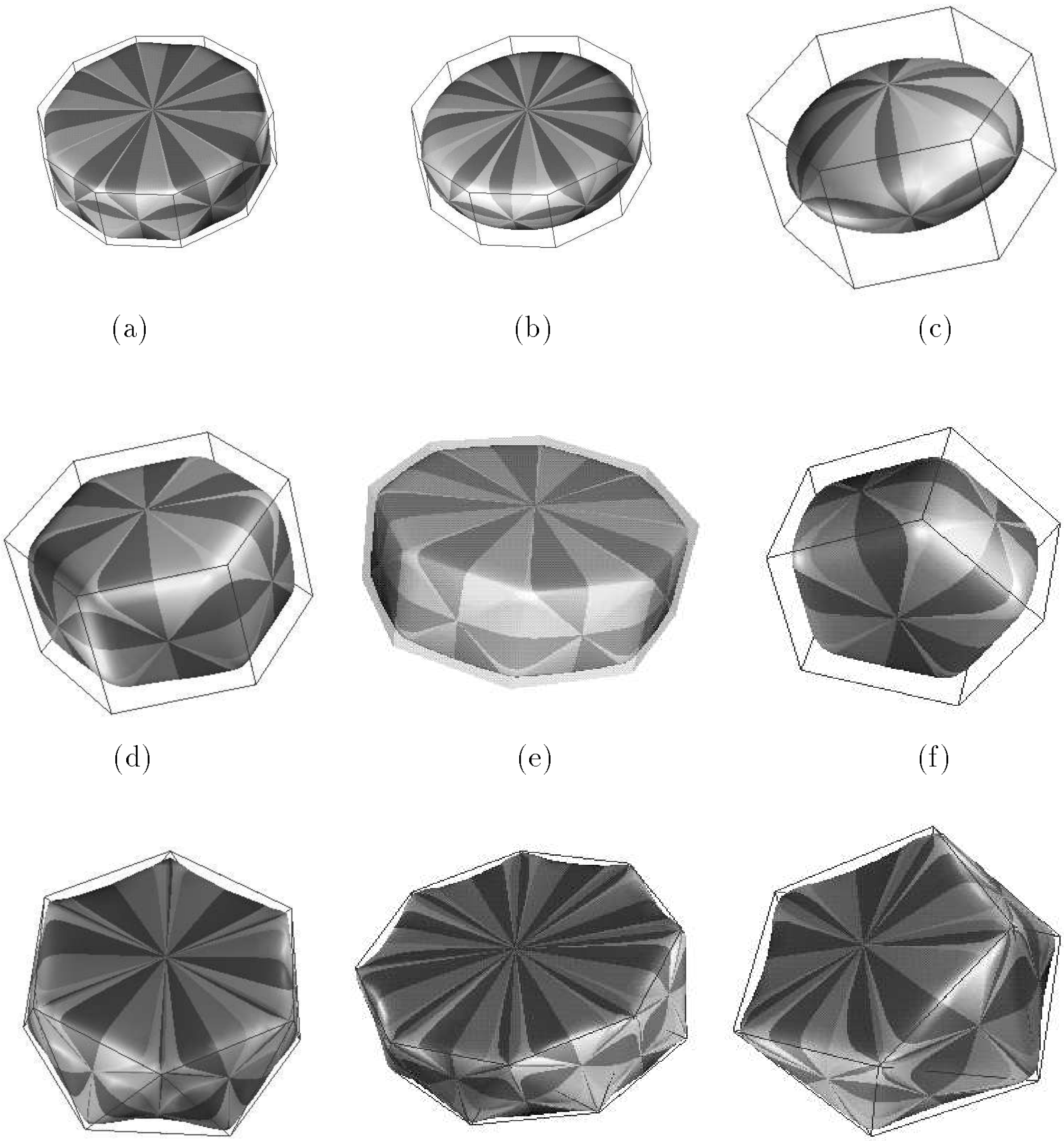


(a) (b) (c)

(d) (e)

(f) (g)

Figure 5.5.24 Octahedron, icosahedron, dodecahedron and stellated dodecahedra. Corners of a dodecahedron are trihedral. Those of the others are non-trihedral. For a non-trihedral corner, we control the shape of the neighboring surface by changing the cutting ratio α_i



(a) (b) (c) (d) (e) (f) (g) (h) (i)

Figure 5.5.25 Different prisms and anti-prisms. All the corners of the prisms are trihedral.

6. INTERACTIVE SHAPE CONTROL AND RAPID DISPLAY

6.1 Introduction

The geometry of implicit surfaces has been proven to be more difficult to control, or polygonize than those of their parametric counterpart (See Chapter 1, Section 1.2.2.1). In the A-patch scheme proposed in Chapter 3, several degrees of freedom remain to modify the resulting C^1 piecewise surface. In this chapter, we utilize these weights for local shape control. We discuss several strategies, their costs and trade-offs.

We also present the *ray shooting* algorithm to rapidly polygonize A-patches. General algebraic surface polygonization schemes, such as oct-tree, polygonize the surface by subdividing a 3D domain. Observing that A-patches can be mapped into 2D domains, we designed the ray shooting algorithm which instead subdivide 2D domains. The algorithm hence dramatically speeds up the polygonization process. In order to display the change of the surface responding to user inputs in real time, an animating ray shooting algorithm is given to quickly modify a polygonization according to small changes of the polynomial coefficients.

The rest of this chapter is organized as follows. Section 6.2 discusses different shape control techniques of A-patches. Section 6.3 proposes the ray shooting algorithms for polygonization and graphics display. Section 6.4 describes a user interface and outlines our implementation, followed by the conclusion of the chapter.

6.2 Shape Controls

6.2.1 The C^1 Smooth and functional Scheme

we reformulate the C^1 cubic A-patch scheme and its representations as suited for the following context.

A piecewise A-patch surface consists of the zero contour of some piecewise smooth BB polynomials defined over a simplicial hull Σ, or a triangulation of a connected region of the space. In particular,

$$F^i(\theta) = \sum_{|\lambda|=m} b^i_\lambda B^m_\lambda(\theta) = \boldsymbol{b}^{i^\top} \mathbf{B}^m = 0 \tag{6.2.1}$$

is the zero contour of a BB polynomial defined with the i-th simplex in Σ. The base functions B^m_λ's are grouped into a vector $\mathbf{B}$ and the coefficients b^i_λ into $\mathbf{b}^i$. Single-sheetedness and non-singularity are enforced by linear inequality constraints

$$\boldsymbol{b}^{i^\top} \mathbf{S} \geq 0 \tag{6.2.2}$$

Continuity of certain degrees and local interpolation of certain degrees are enforced by some linear equality constraints

$$\boldsymbol{b}^\top \mathbf{C}(\mathbf{p}) = 0 \tag{6.2.3}$$

Coefficients of individual simplex $\mathbf{b}^i$ are grouped into a global vector $\mathbf{b}$ and vector $\mathbf{b}$ is a global collection of the coefficient vector $\mathbf{b}^i$ of all simplexes in Σ.

See Figure 3.5.1. Let $V_1 = [p_1p_2p_3p_4]$ and $V_1' = [p_1p_2p_3q_4]$ be a pair of face tetrahedra, $V_2 = [p_1'p_2p_3p_4']$ and $V_2' = [p_1'p_2p_3q_4']$ be the neighboring pair that share edge $[p_2p_3]$, $W_1 = [p_1''p_2p_3p_4]$, $W_2 = [p_1''p_2p_3p_4']$, $W_1' = [q_1''p_2p_3p_4]$, $W_2' = [q_1''p_2p_3p_4']$. be the edge tetrahedra between them, and the polynomials f_i over V_i, g_i over W_i, f_i' over V_i' and g_i' over W_i' be expressed in Bernstein-Bézier form with coefficients a^i_λ, b^i_λ and c^i_λ, respectively.

Denote $\boldsymbol{h}_{XY}$ as the *bary vector* between two tetrahedra that share one common face. For two adjacent tetrahedra, say, W_1 and V_1 in Figure 3.5.1,

$$\boldsymbol{h}_{W_1V_1} = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \beta) \tag{6.2.4}$$

where

$$(\frac{\alpha_1}{\beta}, \frac{\alpha_2}{\beta}, \frac{\alpha_3}{\beta}, \frac{\alpha_4}{\beta})$$

is the barycentric coordinate of vertex p_1'' in tetrahedron V_1. $\boldsymbol{h}_{W_1V_1}$ is actually a homogeneous version of the barycentric coordinates of p_1''. The C^1 condition between them is a set of five equations. For example, between W_1 and V_1, if we put the coefficients of f_i and g_i, a^1_λ and b^1_λ, respectively, into a vector form as follows,

$$\boldsymbol{b}_0 = (a^1_{1200}, a^1_{0300}, a^1_{0210}, a^1_{0201}, b^1_{1200}) \tag{6.2.5}$$

$$\boldsymbol{b}_1 = (a^1_{1110}, a^1_{0210}, a^1_{0120}, a^1_{0111}, b^1_{1110}) \tag{6.2.6}$$

$$\boldsymbol{b}_2 = (a^1_{1020}, a^1_{0120}, a^1_{0030}, a^1_{0021}, b^1_{1020}) \tag{6.2.7}$$

$$\boldsymbol{b}_3 = (a^1_{1101}, a^1_{0201}, a^1_{0111}, a^1_{0102}, b^1_{1101}) \tag{6.2.8}$$

$$\boldsymbol{b}_4 = (a^1_{1011}, a^1_{0111}, a^1_{0021}, a^1_{0012}, b^1_{1011}) \tag{6.2.9}$$

$$\boldsymbol{b}_5 = (a^1_{1002}, a^1_{0102}, a^1_{0012}, a^1_{0003}, b^1_{1002}) \tag{6.2.10}$$

then the sufficient condition we use to ensure C^1 between the neighboring simplexes can be expressed as a set of linear equations as

$$\boldsymbol{h}_{W_1V_1} \cdot \boldsymbol{b}_i = 0, \quad i = 0, ..., 5\ . \tag{6.2.11}$$

In the following weight setting process, $h_{W_1V_1} \cdot b_0 =$ and $h_{W_1V_1} \cdot b_2$ are automatic guaranteed by the fact that the two simplexes shares the same normal and $L_{W_1V_1} \cdot b_1 = 0$ is replaced by a relation directly between two neighboring face tetrahedra,

$$\boldsymbol{h}_{V_1V_2} \cdot (a^1_{1110}, a^1_{0210}, a^1_{0120}, a^1_{0111}, a^2_{1110})^T = 0 \tag{6.2.12}$$

The reason behind this replacement is that, around edge $[p_2p_3]$, the C^1 conditions $h \cdot b_1^T = 0$ between neighboring tetrahedra actually enforce that all the weights are on the same four dimensional hyper plane. Hence setting any four affine independent ones determines the rest. a^1_{1110}, a^1_{0210}, a^1_{0120}, a^2_{1110} are usually set to determine the hyper plane.

In the following algorithms, we mostly describe the setting between V_1 and W_1 and between W_1 and W_2, as the others are trivially symmetric.

Algorithm 6.2.5 C^1 *cubic scheme.*

(1) (At vertices. Interpolating the positions and normals of the vertices. The ⓪ and ① weights)

For face tetrahedron V_1, set $a^1_{3000} = a^1_{0300} = a^1_{0030} = 0.0$, set $(a^1_{2100}, a^1_{2010}, a^1_{2001})$, $(a^1_{1200}, a^1_{0210}, a^1_{0201})$, and $(a^1_{1020}, a^1_{0120}, a^1_{0021})$ according to the normals at the vertices (see Lemma 2.1.1. Let ∇f be the normal at a vertex). For edge tetrahedron W_1, set $b^1_{0300} = b^1_{0030} = 0.0$. Set $(b^1_{1200}, b^1_{0210}, b^1_{0201})$ and $(b^1_{1020}, b^1_{0120}, b^1_{0021})$ according to the normals (Lemma 2.1.1).

(2) (Third level. The ③ weights.)

(i) When $[p_1p_2p_3p'_1]$ is not coplanar, $\alpha_4 \neq 0$, solve 6.2.12 for a^1_{0111}, then solve compute b^1_{0111} accordingly. Similarly, a^1_{1011} and a^1_{1101} are set.

(ii) A degenerated case is when $[p_1p_2p_3p'_1]$ are coplanar, namely, the base faces of two neighboring tetrahedra are coplanar, in which case $\alpha_4 = 0$, a^1_{0111} cannot be expressed in terms of other weights in b_1. In this case, a Clough-Tocher split is needed.

(3) (Second level. Compute the ⑥ weights from ① ③ ④ .)

Setting b_{1101}, b_{1011} of the neighboring tetrahedra by solving C^1 conditions $\boldsymbol{h}_{W_1V_1} \cdot b_3 = 0$ and $\boldsymbol{h}_{W_1V_1} \cdot b_4 = 0$.

(4) (Top level. Compute the ⑦ weights from ④ ⑤)

Setting b_{1002} in the neighboring edge tetrahedra by C^1 condition $\boldsymbol{h}_{W_1V_1} \cdot b_5 = 0$.

(5) (C^1 condition between edge patches. Compute the ⑨ and ⑩ weights from ⑥ and ⑧)

Setting b_{2100}, b_{2010} and b_{3000} according to the C^1 conditions between W_1 and W_2, $\boldsymbol{h}_{V_1V_2} \cdot b_i^{V_1V_2} = 0$.

In Algorithm 6.2.5, a^1_{1110}, a^1_{1002}, a^1_{0102}, a^1_{0012} and a^1_{0003} in f_i and b^1_{2001} in g_i are free. In the following algorithm, Algorithm 6.2.6, we adjust them to enforce single-sheetedness. The description of Algorithm 6.2.6 is actually an add-on of Algorithm 6.2.5. Step n' is to be inserted before step n of Algorithm 6.2.5.

Algorithm 6.2.6 C^1 *Smooth and Single-sheeted*

(1) (Same as in algorithm 6.2.5).

(2′) If $[p_1p_2p_3]$ is not coplanar to the base face of any its neighboring face tetrahedra, a^1_{1110} is left free from the C^1 condition.

(i) If all $a^1_{\lambda_1\lambda_2\lambda_3 0}$ except a^1_{1110} are of the same sign, adjust a^1_{1110} to make sure the BB polynomial defined over $[p_1p_2p_3]$ does not intersect with face $[p_1p_2p_3]$. A simple solution is just setting a^1_{1110} to be of the same sign.

(ii) If one of the three edges $[p_1p_2]$, $[p_2p_3]$, or $[p_3p_2]$ is nonconvex (for example, on $[p_2p_3]$, a^1_{0210} and a^1_{0120} are of different signs), then the weights at the third level, $a^1_{\lambda_1\lambda_2\lambda_3 1}$'s have to be of the same sign as that of the top level weight. If needed, a Clough-Tocher split is performed so that there is enough freedom to ensure this sign restriction. In the case that a Clough-Tocher split is need, namely, V_1 is split into (V_{11}, V_{12}, V_{13}) and V_2 is split into (V_{21}, V_{22}, V_{23}), Adjust a^{11}_{1110}, a^{21}_{1110} to ensure that a^1_{0111}, a^2_{0111} to be positive and c^1_{0111}, c^2_{0111} to be negative.

(2) (Same as in algorithm 6.2.5).

(3′) (functional condition/Second level. ④ $\Leftarrow$ ⑥)

Adjust a_{1002}, a_{0102} and a_{0012} so that they are big enough in absolute value to ensure that b_{1101}, b_{1011} in the neighboring edge tetrahedra are of the same sign as they are related in C^1 conditions $L \cdot b_3 = 0$ and $L \cdot b_4 = 0$.

(3) (Same as in algorithm 6.2.5).

(4′) (functional condition/Top level. ⑤ $\Leftarrow$ ⑥)

Adjust a_{0003} so that it is big enough in absolute value to ensure that b_{1002} in the neighboring edge tetrahedra are of the same sign when setting the C^1 condition between them, namely, $L \cdot b_5 = 0$.

(4) (Same as in algorithm 6.2.5).

(5′) (functional condition/Edge patch. The ⑧ weights) Set edge weight b_{1200} to be of the same sign as b_{0003}.

(5) (Same as in algorithm 6.2.5).

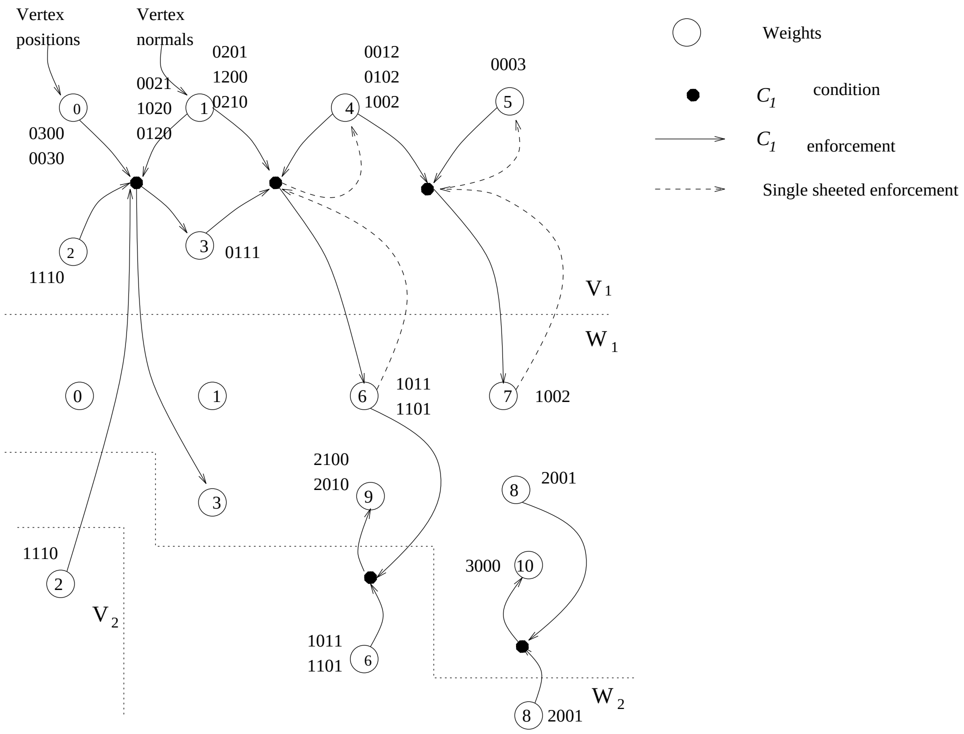


Figure 6.2.1 Control flow of the C^1 smooth and single-sheeted setting.

Figure 6.2.1 extracts the control flow of the algorithm 6.2.6. Based on this control flow diagram, we design several schemes which manipulates the shape of the surface by modification of the weights.

Shape control can be achieved by (1) changing the weights or (2) changing the simplicial hull. The former is affect the surface in a very local manner, and is rather economic, while the latter is suitable for wide range shape editing, but rather expensive. We discuss difference schemes and trade-offs in this section.

6.2.2 Default Weights

In the C^1 smooth and functional scheme, weights a^1_{1110}, a^1_{1002}, a^1_{0102}, a^1_{0012} and a^1_{0003} of V_1 and b^1_{2001} of W_1 are adjustable within some ranges. This freedom, on the one hand, allows us to locally change the shape of the surface, while on the other hand, burdens us with extra work to remove bumpy defects.

6.2.2.1 Approximating Patches of lower degree

One commonly used method is to keep the surface patch close to a lower degree patch [5, 48], which, in our case, is quadric patch. Specifically, we determine a quadric that is least square approximating the known weights of the cubic and then set the unknown weights of the cubic from the quadric by a degree raising formula.

However, the least square optimization is subject to the C^1 and single-sheeted linear constraints. Hence it is actually a quadratic programming problem with inequality constraints.

6.2.2.2 Edge Patch First

The previous method fails to consider the shape of neighboring face patches beyond the fact that they share common normals. Such neglect could lead to unwanted variation in the edge patch between them, especially when the edge patch is thin. An alternative scheme sets the ideal weights of the edge patches first. Then the weights

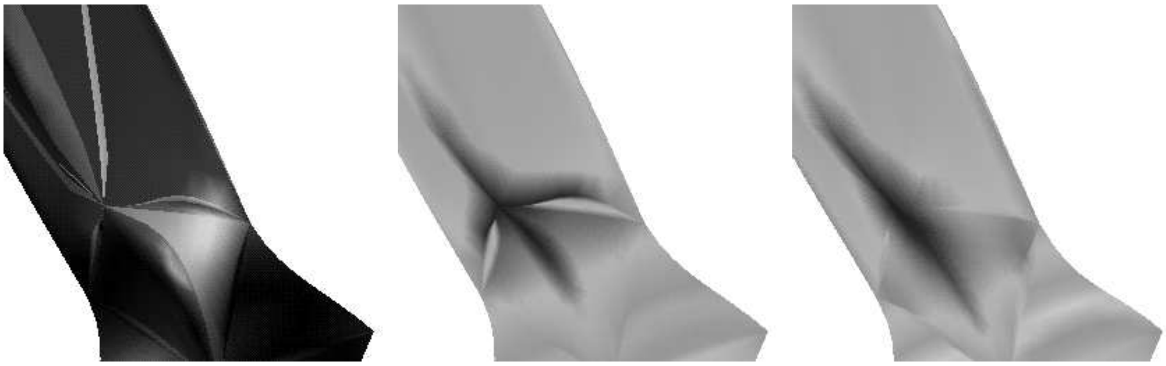


Figure 6.2.2 Setting the edge patches first to reduce unwanted variations. Left: piecewise surface in multi-patch shading mode. Center and Right: shown in an RGB model where the normal vectors become the color RGB vector. Center: face patches are set first. Right: edge patch are set first. One sees that the right picture exhibits a fairer surface

of the face patches "honor" the choice of the edge patches by making sure the ideal edge patch weights are changed the least when the C^1 conditions are set.

Considering edge patch W_1 and W_2. Let ℓ be the intersection of the two tangent plane at p_2 and p_3. The weights around the two vertices are set by the interpolatory and normal conditions. We set the rest of the weights so that $g_1 = 0$ is actually a swept surface parallel to ℓ and the cross section curve approximate a quadric in least square sense. This can be done by a few basis transformations of the polynomials. Set g_2 to be the same as g_1.

By C^1 conditions, the edge patch weights propose values for the neighboring face patches. A face patch, however, take weighted averages of the proposed values from different edge patches around it and set them as default values. In taking the weighted average, smaller edge patch weighs more as for the same BB-represented surfaces, smaller tetrahedra yields larger curvature and larger curvature change for the same amount of change in the weights.

The heuristic methods we discussed above are "cheap" for they do not require any massive computations. "Expensive" global optimization methods can also be used to improve the surface by minimizing the "energy", or some smoothness criteria, of the surface.

6.2.3 Shape Control with Weights

We propose two schemes here which control the shape of a surface by manipulating only the weights. The simplicial hull is fixed.

6.2.3.1 Scheme I: Control with Top Level Weights

In a face tetrahedra V, we can raise or lower weights a_{0003}, a_{1002}, a_{0102} and a_{0012} defining the surface patch without altering its variation (see Figure 6.2.4) ; Weight a_{1110} alters the variation of the surface. Please note that by C^1 conditions, a_{1110} is related to a_{0111}, a_{1011} and a_{1101}, and also there are other weights that could alter the surface geometry (see Figure 6.2.5). Hence the effect of changing a_{1110} in changing variation could be exaggerated or reduced, depending on the geometric relationship between the tetrahedra. Also, as a_{1110} are related to neighboring patches in linear equations, change of this weight propagates further into to neighboring patches, while that of a_{1002}, a_{0102} and a_{0012} do not affect neighboring face patches and that of a_{0003} affects only adjacent edge patches.

In general, a desirable modification involves collaboration of several adjustable weights rather than a single one. Hence an alternative way is to specify some additional data points in the tetrahedra, and then approximate these points in the least square sense.

The following is simple interactive scheme by controlling a_{0003}, a_{1002}, a_{0102} and a_{0012}. This scheme is particular suitable for simplicial hulls constructed by the corner-cutting (approximation) scheme [13].

(1) Vertex control I: Change only a_{0003} in a face patch (3-sided patch). Execute (4)' and (4) of algorithm 6.2.6. This control only affects a_{0003} and b_{1002} in the neighboring 4-sided patches. In short, it affects totally 4 weights in 4 patches. (It is actually 7 weights if we count b_{0003} in an edge patch as a different weight than a_{0003}.)

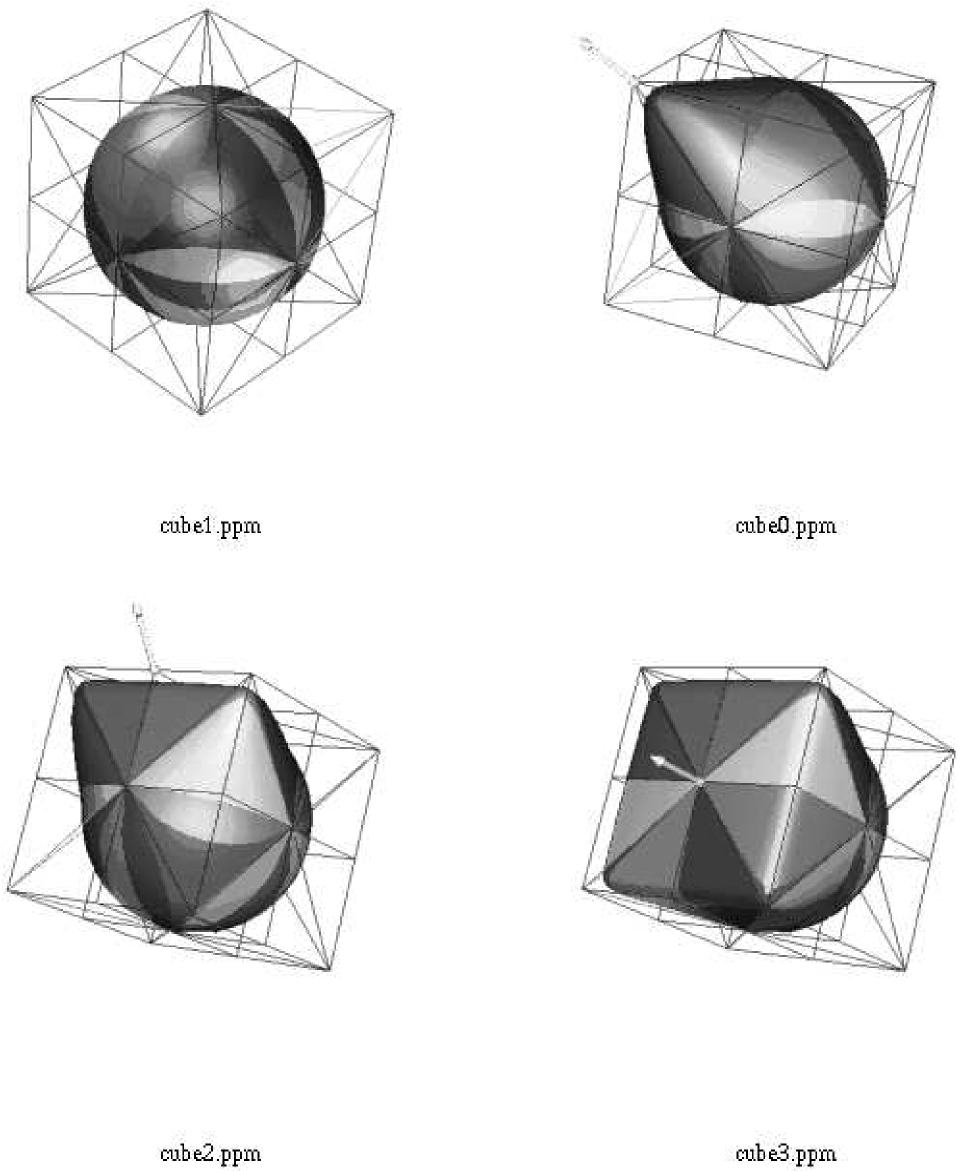


Figure 6.2.3 Interactive control with 3D widgets

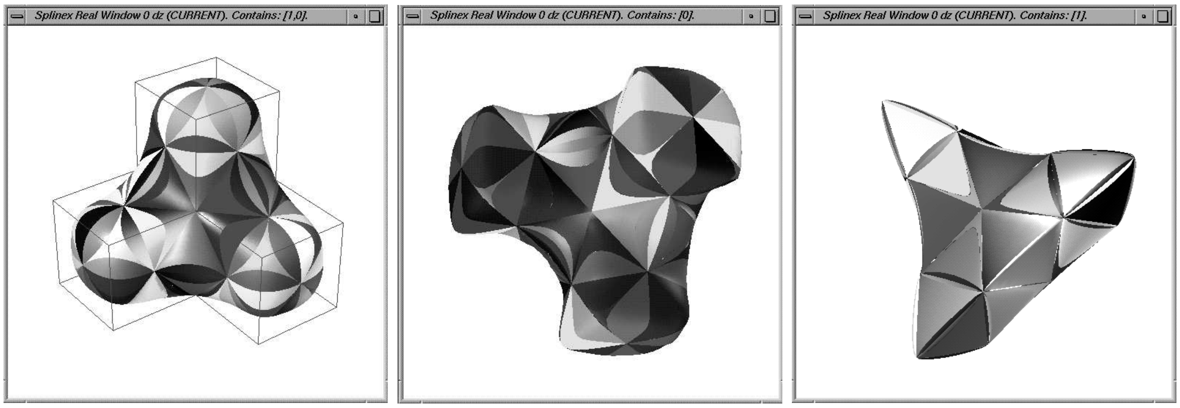


Figure 6.2.4 Corner cutting scheme: smoothing the same tricube with different configurations.

(2) Vertex control II: Let $a_{0003} = a_{1002} = a_{0102} = a_{0012} = t$, for some t. Change t and, do steps (3)', (3), (4)', (4), (5') and (5) of algorithm 6.2.6, in that order. This control affects the top 4 weights in the face patch (3-sided patch), and weights in the 3 edge patches (4-sided patches) face-adjacent to the face patch, and the 3 edge patches face-adjacent to these 3 edge patches. Hence totally 1 face patch and 6 edge patch are affected.

(3) Vertex control III: Change a_{1110} in a face patch(3-sided patch). Starting from $(2)'$, perform algorithms 6.2.6 on this face patch and it neighboring face patches and their neighboring patches, whichever are applicable according to the algorithm. It affects totally 22 to 44 patches. 4 face patch pairs (totally 4 to 8 patches) and 18 to 36 edge patches edge-adjacent to them.

(4) Edge control I: Between to edge patches W_1 and W_2 (see figure 3.5.1), $b^1_{3000}(= b^2_{3000})$, b^1_{2001} and b^2_{2001} are related to each other in a linear equation. Any change done to it will not affects any other weights as they are the last to set in algorithm 6.2.6. There are two degree of freedom left in this linear equation. Change two of them and calculate the third one according to the linear equation. This control only affects two edge patches.

(5) Edge control II: Perform a vertex control to two adjacent face patch.

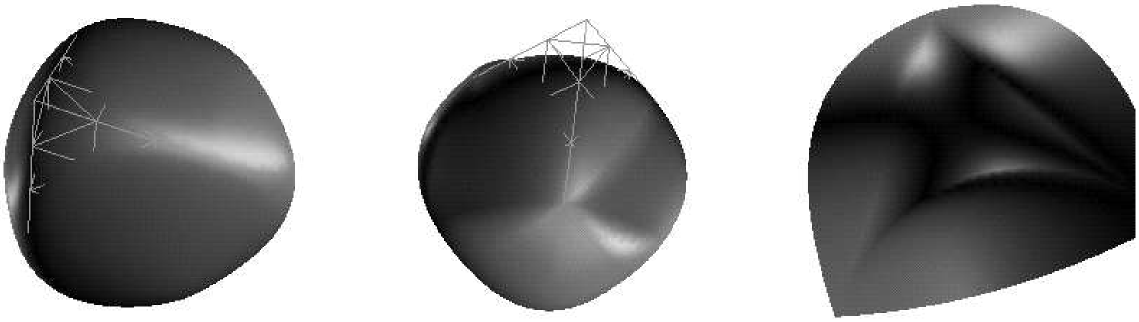


Figure 6.2.5 Changing a_{1110} on a face patch affecting other face patches. From left, $a_{1110} = -0.5, -1.0, 0.5$. The corner is popped up in the center picture while sucked in in the right one

(6) Face control: Perform a vertex control to all face patches share the same vertex at the bottom faces.

6.2.3.2 Scheme II: Control with Normals

At a vertex, if the normal becomes longer then the surface becomes flatter around this vertex. The change of normal length is equivalent to a scaling of all the weights around the vertex by the same ratio. Altering the direction of a normal also changes the shape around the vertex. However, when we desire a direction change we need to check that it does not violate the tangent containment constraints. After a normal is changed, algorithm 6.2.6 need to be performed on every patch incident to the vertex. This control affects every patch incident to the vertex and the edge patches that are edge-adjacent to these patches. It accounts to at least $5n$ and at most $10n$ patches, where n is the number of face patches incident to the vertex.

This scheme is suitable for simplicial hulls constructed by the interpolatory scheme [17].

6.2.4 Shape Control with Simplicial Hull

Only changing the weights may not be enough to edit the surface to a desirable shape. We explore the feasibility of editing the simplicial hull in order of obtain a desirable shape.

6.2.4.1 Interpolation Based Scheme

A straight forward scheme can be derived based on the construction scheme given in Chapter 3. Given a triangulation $\mathcal{T}$ with normal directions at each vertex, the scheme works in two phases. It first builds two tetrahedra (the face tetrahedra) on a triangular face of $\mathcal{T}$, one on each side. The two tetrahedra have to meet the tangent containment constraint and do not intersect with tetrahedra built on adjacent triangular faces. In the second phase, two tetrahedra (edge tetrahedra) are filled in a gap between the face tetrahedra of adjacent triangular faces. Note that the construction scheme highly localized. In the first phase, only the information of the triangular face and its adjacent faces is needed to build the face tetrahedra pair. In the second phase, only the information of two edge-adjacent face tetrahedra is needed to fill in the gap with two edge tetrahedra.

We identify 3 kinds of vertices in a simplicial hull:

- Base vertices. A base Vertex coincides with that of the input triangulation $\mathcal{T}$. A normal is associated with such a vertex.

- Top vertices. A top vertex is the top of the tetrahedron built on a triangular face, or the common top of a group of Clough-Tocher built on top of a triangular face.

- Auxiliary vertices. An auxiliary vertex is either the one shared only by two face-adjacent edge tetrahedra, or the one shared only by three Clough-Tocher sub-tetrahedra.

The position and normal direction of a base vertex can be changed freely. The simplicial hull around the vertex is then adjusted to reassure the tangent-containment constraint. A subdivision of the simplicial hull may be required when it is hard to enforce with the current simplicial hull tangent-containment constraint or the non-self-intersection constraint.

The position of a top vertex can be changed providing that neither the tangent containment constraint nor the non-self-intersection constraint is destroyed.

The auxiliary vertices are not to be changed independently. An auxiliary vertex is enforced to have a fixed relative position with other vertices around it: either the midpoint between to top vertices of two adjacent faces, or the centroid of a triangular face of $\mathcal{T}$.

6.2.4.2 Corner Cutting Based Scheme

Another interactive scheme is based on the approximation scheme presented in Chapter 5.

In Chapter 5, the simplicial construction is highly localized. The position and shape of a simplex in the mesh is determined locally by a vertex of the input polyhedron, the edges and faces incident to it and the cutting ratio. Hence input polyhedron can serve as a control polygon. One can instead modify the input polygon and then the mesh is modified accordingly.

6.3 Rapid Display and Animation Schemes

Interactivity is crucial for interactive control schemes. Hence rapid display and animation schemes are needed to have quick response to any change to the surface. We discuss a rapid "ray shooting" algorithm to polygonize A-patches. We further develop an animation scheme which, derived from the ray shooting algorithm, quickly update the triangulation of the surface to reflect small configuration changes in the surface.

General algebraic surface polygonization schemes, such as oct-tree, polygonize the surface by subdividing a 3D domain. Observing that A-patches can be mapped

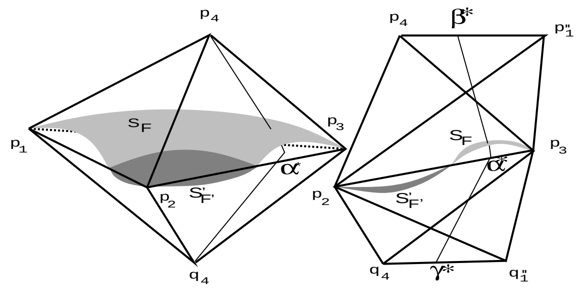


Figure 6.3.6 Ray shooting A-patches

into 2D domains, we design a ray shooting algorithm which instead subdivides 2D domains. The reduction from 3D to 2D dramatically speeds up the polygonization process.

Algorithms to generate polygonal approximations of a three-sided or four-sided patch are suggested by properties (d) and (e) in Section 3.2. Let $F(\alpha) = 0$ and $F'(\alpha) = 0$ be two based sharing 3-sided 4-patch defined over $[p_1p_2p_3p_4]$ and $[p_1p_2p_3q_4]$, respectively (see Figure 3.5.1). If

1. be C^0 continuous to each other, namely, $F(\alpha) = F'(\alpha)$ if $\alpha_4 = 0$.

2. $F(e_4) \cdot F'(e_4) < 0$

then any point v on the surface defined can be one to one mapped to a point α^* on face $[p_1p_2p_3]$. This is because that polyline $p_4\alpha^*q_4$ intersects with the union of two patches exactly once (see Figure 6.3.6). Hence the barycentric coordinate of a point on $[p_1p_2p_3]$ can be used as a parameterization of the union of the two three-side A-patches.

Similarly, for two edge sharing four-sided 14-patches $F(\alpha) = 0$ and $F'(\alpha) = 0$ defined in $[p_1''p_2p_3p_4]$ and $[q_1''p_2p_3q_4]$ (see Figure 3.5.1), if

1. be C^0 continuous to each other, namely, $F(\alpha) = F'(\alpha)$ if $\alpha_2 + alpha_3 = 1$;

2. $F(\alpha) \cdot F'(\alpha) < 0$ if $\alpha_2 + \alpha_3 = 0$.

the union of the two 4-sided A-patches can be parameterized into a quadrilateral domain (s,t), where $(1-s,s)$ is the barycentric coordinate of both a point $\beta^* \in [p_1''p_4]$

and a point $\gamma^* \in [q_1''q_4]$, and $(1-t,t)$ is the coordinate of a point $\alpha^* \in [p_2p_3]$, as polyline $\beta^*\alpha^*\gamma^*$ intersection the union of A-patches once (see Figure 6.3.6). We call such a pair of three-sided or four-sided patches as a *double* A-patch. A pair of three-sided double patches, $[p_1p_2p_3p_4]$ and $[p_1p_2p_3q_4]$ are denoted as $[p_1p_2p_3;p_4q_4]$; A pair of four-sided double patches, $[p_1''p_2p_3p_4]$ and $[q_1''p_2p_3q_4]$ are denoted as $[p_1''q_1'';p_2p_3;p_4q_4]$; We call polyline($p_4\alpha^*q_4$ or $\beta^*\alpha^*\gamma^*$) a *ray* of a double A-patch, and call the parameterizations *ray coordinate*. We denote the *ray coordinate* of a vertex v as $R(v)$.

For a 3-sided 4-patch $[p_1p_2p_3p_4]$, if a ray coordinate is (s,t), then $[p_4,\alpha^*]$, the segment of the ray within the simplex can be expressed as

$$\alpha(u) = ue_4+(1-u)(1-s-t,s,t,0) = ((1-u)(1-s-t),(1-u)s,(1-u)t,u)^T, u \in [0,1] \tag{6.3.13}$$

Substituting 6.3.13 into the barycentric polynomial 2.1.4 defined in $[p_1p_2p_3p_4]$, we get a univariate sub-polynomial defined in $[e_4\alpha^*]$ as follows.

$$F_{3r}(u) = F(\alpha(u)) = \Sigma b_\lambda B_\lambda(\alpha(u)).$$

Let $\alpha_4 = \ell$.

$$\begin{aligned} B_{\lambda_1\lambda_2\lambda_3\ell}(\alpha(u)) &= \frac{n!}{\lambda_1!\lambda_2!\lambda_3!\ell!}\alpha_1^{\lambda_1}\alpha_2^{\lambda_2}\alpha_3^{\lambda_3}\alpha_4^{\ell} \\ &= \frac{n!}{\lambda_1!\lambda_2!\lambda_3!\ell!}(1-s-t)^{\lambda_1}s^{\lambda_2}t^{\lambda_3}(1-u)^{\lambda_1+\lambda_2+\lambda_3}u^{\ell} \\ &= \frac{(n-\ell)!}{\lambda_1!\lambda_2!\lambda_3!}(1-s-t)^{\lambda_1}s^{\lambda_2}t^{\lambda_3}\frac{n!}{(n-\ell)!\ell!}(1-u)^{1-\ell}u^{\ell} \\ &= B^{1-\ell}_{\lambda_1\lambda_2\lambda_3}B^n_{n-\ell,\ell} \end{aligned}$$

Hence

$$F(\alpha(u)) = \Sigma_{\ell=0}^n b_\ell(s,t)B_\ell^n(u)$$

where

$$b_\ell(s,t) = \sum_{|\lambda|=n-\ell} b_\lambda B^{n-\ell}_{\lambda_1\lambda_2\lambda_3}(1-s-t,s,t) \tag{6.3.14}$$

For a 4-sided patch $[p_1p_2p_3]$, if a ray coordinate is (s,t), then $[\beta^*\alpha^*]$, the ray segment within the simplex can be expressed as

$$\alpha(u) = (us, u(1-s), (1-u)t, (1-u)(1-t))^T, u \in [0,1] \tag{6.3.15}$$

Substituting 6.3.15 into the barycentric polynomial 2.1.4 defined in $[p_1p_2p_3p_4]$, we get a univariate sub-polynomial defined in $[\beta^*\alpha^*]$ as follows.

$$F_{4r}(u) = F(\alpha(u)) = \Sigma^n b_\lambda B^n_\lambda(\alpha(u)) \tag{6.3.16}$$

Let $\lambda_1 + lambda_2 = \ell$,

$$\begin{aligned}
B_{\lambda_1\lambda_2\lambda_3\ell}(\alpha(u)) &= \frac{n!}{\lambda_1!\lambda_2!\lambda_3!\lambda_4!}\alpha_1^{\lambda_1}\alpha_2^{\lambda_2}\alpha_3^{\lambda_3}\alpha_4^{\lambda_4} \\
&= \frac{\ell!}{\lambda_1!\lambda_2!}\frac{(n-\ell)!}{\lambda_3!\lambda_4!}\frac{n!}{\ell!(n-\ell)!}(us)^{\lambda_1}(u(1-s))^{\lambda_2}((1-u)t)^{\lambda_3}\cdot \\
&\quad ((1-u)(1-t))^{\lambda_4} \\
&= \frac{\ell!}{\lambda_1!\lambda_2!}s^{\lambda_1}(1-s)^{\lambda_2}\frac{(n-\ell)!}{\lambda_3!\lambda_4!}t^{\lambda_3}(1-t)^{\lambda_4}\frac{n!}{\ell!(n-\ell)!}u^{\lambda_1+\lambda_2}(1-u)^{\lambda_3+\lambda_4} \\
&= B^\ell_{\lambda_1\lambda_2}(s,1-s)B^{n-\ell}_{\lambda_3\lambda_4}(t,1-t)B^n_\ell(u,1-u)
\end{aligned}$$

Hence

$$F_{4r}(u) = F(\alpha(u)) = \Sigma^n_{\ell=0} b_\ell(s,t)B^n_\ell(u) \tag{6.3.17}$$

where

$$b_\ell(s,t) = \sum_{\substack{\lambda_1+\lambda_2=\ell \\ \lambda_3+\lambda_4=n-\ell}} b_\lambda B^\ell_{\lambda_1\lambda_2}(s)B^{n-\ell}_{\lambda_3\lambda_4}(t) \tag{6.3.18}$$

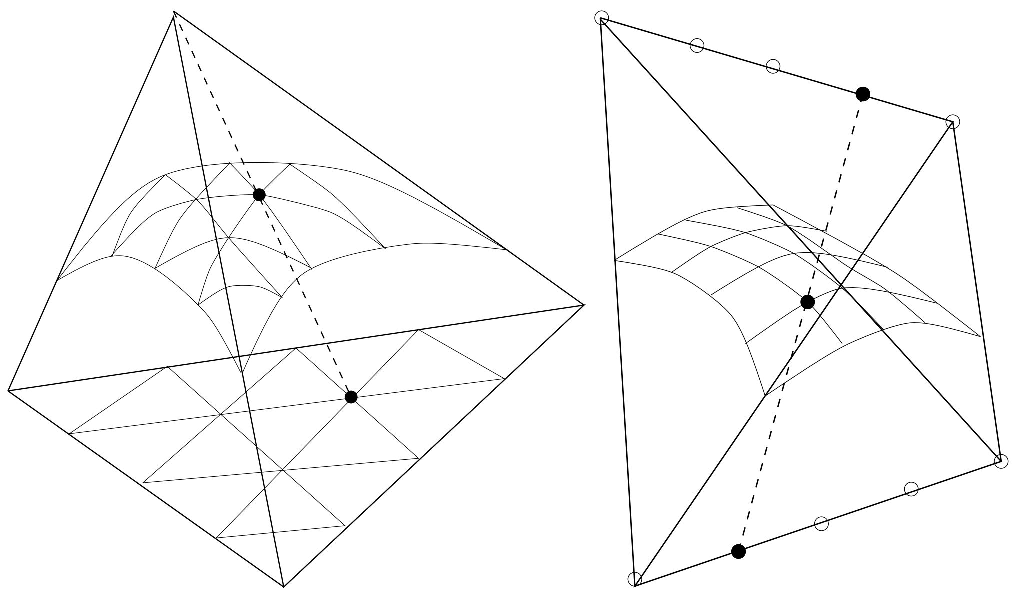
We compute the point on the surface by solving the sub-polynomial $F_r(u) = 0$ defined on the *ray* segment $([p_4, \alpha^*]$ or $[\beta^*\alpha^*]$ with secant method or Newton's method.

6.3.1 An Algorithm for Double A-patches

A simple adaptive polygonization algorithm is then designed for both three-sided and four-sided A-patches. Beginning with some initial triangles, we keep subdividing them until the shape is desirable to some criterion. We list three criteria as follows. Let v_1 and v_2 be two points on an A-patch $F(\alpha) = 0$ defined in simplex $[p_1p_2p_3p_4]$.

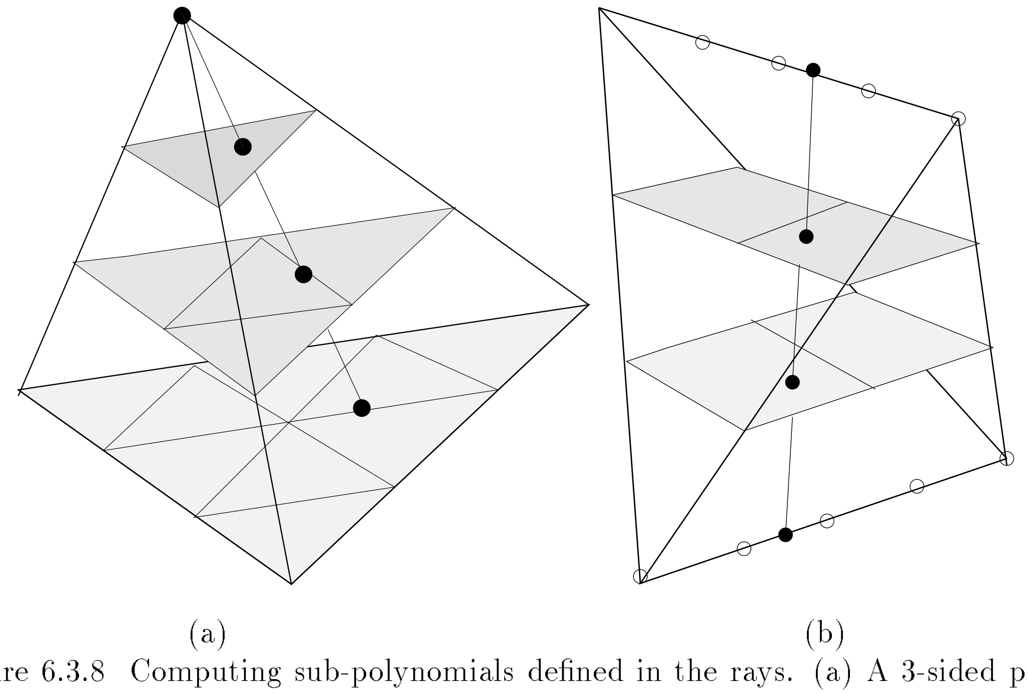
Definition 6.3.1 *Edge length* We say edge $[v_1, v_2]$ is *breakable* if it is longer than a certain threshold value T_l.

Definition 6.3.2 *Normal difference.* We say edge $[v_1v_2]$ is *breakable* if the inner product between the normals (normalized) at the two end points is less than a certain threshold value T_n.



(a) (b)

Figure 6.3.7 Parameterization of A-patches. (a) A 3-sided patch: one to one mapping between a point on the patch and a point at the bottom face (b) A 4-sided patch: one to one matching between a point on the patch and pair of points, one on each edge



(a) (b)

Figure 6.3.8 Computing sub-polynomials defined in the rays. (a) A 3-sided patch. (b)A 4-sided patch

Curvature variation is also a commonly used criterion in adaptive surface polygonization schemes. If both v_1 and v_2 are on the same face of the domain tetrahedron, their curvatures are taken as that of the boundary curve on the face. Otherwise, the curvature of, say v_1, is taken as that of the intersection curve between the surface and the plane spanned by $[v_1 v_2]$ and the surface normal at v_1. See Appendix for formulae of the curvatures for algebraic surface in BB form.

Definition 6.3.3 *Curvature difference.* We say an edge is too *breakable* if the *curvature* difference between two end points are larger than certain threshold value T_c.

For low degree surface, a combination of 6.3.1 and 6.3.2 works good enough and is computationally inexpensive compared to the curvature difference criteria (Definition 6.3.3).

Algorithm 6.3.7 *Ray-shooting*

(1) Initialization. (i) Three-sided double A-patches: Compute vertices p_1, p_2 and p_3 whose ray coordinates are $(0,0)$, $(1,0)$ and $(0,1)$, respectively. Enter $[p_1 p_2 p_3]$ as the first cell in the polygon list.
(ii) Four-sided double A-patch: Compute vertices p_1, p_2, p_3 and p_4 whose ray coordinates are $(0,0)$, $(0,1)$, $(1,1)$ and $(1,0)$, respectively. Enter $[p_1 p_2 p_3]$ and $[p_1 p_3 p_4]$ as the first two cells in the polygon list.

(2) For each edge $[AB]$, where the ray coordinate of the endpoints

$$R(A) = (s_0, t_0), R(B) = (s_1, t_1),$$

if it is *breakable*, compute

$$R(C) = \begin{pmatrix} s \\ t \end{pmatrix} = (1-\zeta) \begin{pmatrix} s_0 \\ t_0 \end{pmatrix} + \zeta \begin{pmatrix} s_1 \\ t_1 \end{pmatrix} \tag{6.3.19}$$

for some $\zeta \in [0,1]$. Compute the position of vertex C. Replace $[AB]$ by $[AC]$ and $[CB]$. Exit if no edge is breakable; go to (3) otherwise.

(3) Triangulate every cell that has broken edges due to step 2. (i) All three edges are broken in step 2: For example, in triangle $[ABC]$, $[AB]$ is broken by D; $[BC]$ by

E; $[CA]$ by F. Replace $[ABC]$ by $[ADF]$, $[BDE]$, $[CFE]$ and $[DEF]$. (ii) Two edges are broken: For example, in $[ABC]$, $[AB]$ is broken by D; $[BC]$ by E. Replace $[ABC]$ by $[ADE]$, $[BED]$ and $[CAE]$. (iii) One edge is broken: For example, in $[ABC]$, $[AB]$ is broken by D. Replace $[ABC]$ by $[ADC]$ and $[BCD]$. Go to (2).

Please note that a four-sided double patch may have some special points where the surface passes through edge p_2p_3. At those points, s, the first component of its ray coordinate, is undefined. For all the points with the same t value coincide at one point.

This ray-shooting algorithm works reasonably fast, it takes only about 30 seconds on an Indigo Extreme to polygonize the satellite object (Figure 5.5.21), which contains over 1392 A-patches. Each A-patch is polygonized into about 40 triangles.

6.3.2 Fast Animation of Changing A-patches

In an interactive control session, a piece of A-patch is changing due to changing of (1) the simplex frame or/and (2) the coefficients. If the changes are sliced into small intervals, of each of which we render a surface, then we have an animation that shows movement of the patch.

We propose a scheme to achieve fast animation of the movement of an A-patch. We store the polygonization in an array of vertices and an array of faces, where the vertices of a face are indexed into the vertex array. Assuming that the change of the surface is relatively small, we only update the vertex array according to the frame or coefficient change. Each vertex in the vertex array still retain the same ray coordinate. For a simplex frame change, we apply the same transformation matrix to each vertex. For a coefficient change, we take the current position of the vertex as the initial value, perform one iteration of Newton's method and call it a good approximation of the new vertex position. Let $F(\boldsymbol{b}, \alpha) = 0$ be the patch defined with a simplex. $\mathbf{b}$ denotes its coefficients. $\boldsymbol{b}_0$ is the old value of the coefficient. α_0 is the old barycentric coordinates of vertex v. $\boldsymbol{R}$ is the ray of vertex v and $(1-u, k)$ is the barycentric coordinates of v in ray $\boldsymbol{R}$. From Taylor's expansion

$$\begin{aligned} 0 &= F(\mathbf{b_0}, \alpha_0) + F_{\mathbf{b}} d\mathbf{b} + F_\alpha d\alpha + \text{higher order terms} && (6.3.20) \\ &\approx F(\mathbf{b}, \alpha_0) + \nabla_\alpha F \boldsymbol{R} dk. && (6.3.21) \end{aligned}$$

For a 3-sided patch $R(v) = (s, t)$. From (6.3.13),

$$d\alpha = -(1 - s - t, s, t, 1)^{\mathsf{T}} du \ .$$

Hence,

$$du = \frac{-F(\boldsymbol{b}, \alpha_0)}{\nabla_\alpha F(1 - s - t, s, t)^{\mathsf{T}}} \qquad (6.3.22)$$

$$u = u_0 + du$$

The barycentric coordinates of v is $((1 - s - t)u, su, tu)$. Hence

$$v = \mathbf{J}^{-1}((1 - s - t)u, su, tu)^T + p_4.$$

Similarly, for a vertex in a 4-sided patch

$$du = \frac{F(\boldsymbol{b}, \alpha_0)}{\nabla_\alpha F(s, 1 - s, -t)^T} \qquad (6.3.23)$$

Hence $v = \mathbf{J}^{-1}(us, (u(1 - s), (1 - u)t)^T + p_4$. At the end of an animation session, we use the complete Newton's method to make sure the final position of the vertex is accurate enough.

6.3.3 An Algorithm for a Group of A-patches

In Chapter 3, a simplicial hull is constructed in the following pattern: a pair of three-sided double 4-patches $[p_1 p_2 p_3; p4q4]$ are connected to another double three-sided 4-patches, $[p_1' p_2 p_3; p_4' q_4']$, by sharing the edge $[p_2 p_3]$ of their base faces; two four-sided double 14-patches $[p_1'' q_1''; p_2 p_3; p_4 q_4]$, $[p_1'' q_1''; p_2 p_3; p_4' q_4']$ are bridging the two adjacent double three-sided A-patches. The three-sided patches are called *face tetrahedra* while the four-sided patches bridging them are called *edge tetrahedra.*

We observe that, in a simplicial hull, a large portion of edge tetrahedra are thin compared to their neighboring face tetrahedra. If we ray-shoot each double patch separately, the polygons on an edge tetrahedron are rather skew and dense while that of its neighboring face tetrahedron are rather balanced and sparse. This is not desirable for display or further processing based on the polygonal representation. The above ray-shooting algorithm is modified to improve the polygonization. Instead of ray-shoot individual double A-patches, we ray-shoot a group of A-patch together, so that edge patches are "melt" into their neighboring face patches.

Specifically, if $[p_1p_2p_3; p_4q_4]$ is double 4-patch and $[p_1''q_1''; p_2p_3; p_4q_4]$, $[p_2''q_2''; p_1p3; p_4q_4]$, and $[p1p2; p_3''q_3''; p_4q_4]$ are the edge tetrahedra around it, then we say that they form an *A-patch group*, denoted as $G = [p_1p_2p_3; p_4q_4; p_1''q_1''; p_2''q_2''; p_3''q_3'']$. An A-patch group has the following properties:

1. A patch in the group is at least C^0 continuous with its neighboring patches;
2. The piecewise surface interpolates vertices p_1, p_2 and p_3;

Triangle face $[p_1p_2p_3]$ is now the *base* face of the group G. Under some cases, the three-sided double patch $[p_1p_2p_3; p_4q_4]$ is split into three three-sided double patch by adding an extra vertex c in the interior of $[p_1p_2p_3]$, which is known as the Clough-Tocher split.

The ray coordinate of a point v in an A-patch group G is with respect to the tetrahedra it is in. Hence we augment the notation of ray coordinate to be $R(v) = (s, t, T)$, where T is the patch that contains v. Points on the boundary faces have different ray coordinates with respect to different tetrahedra. But as the boundary is C^0, the point can be computed in any one of its ray coordinates.

In a patch group, in general the ray coordinate of the midpoint of an edge can no longer be the average of that of the end points as we do in Algorithm 6.3.7, since the end points may not even be in the same double patch. We need to find out in which double patch the midpoint is and figure out its ray coordinate within that double patch. If the mid point is on an edge of the base face, say $[p_2p_3]$, we assume the point is on face $[p_2p_3p_1'']$ or $[p_2p_3q_1'']$ so as to make sure the polygonal mesh spans over the whole patch group.

We give a group ray-shooting algorithm as follows. The breakable criteria (Definitions 6.3.1–6.3.3) are applicable here.

Algorithm 6.3.8 *Group Ray-shooting*

(1) For an A-patch group G, $[p_1p_2p_3, p_4q_4, p_1''q_1'', p_2''q_2'', p_3''q_3'']$, initialize the polygon list with triangle $[p_1p_2p_3]$.

(2) For each edge $[A = (s_0, t_0, T_0), B = (s_1, t_1, T_1)]$ that are *breakable*, Let C be the midpoint of edge $[AB]$, call subroutine *GetRayCoord* (G, C, $RayCoord$) to compute $R(C)$, the ray coordinate of C. Replace edge $[AB]$ by $[AC]$ and $[CB]$. Exit if no edge is broken, go to (3) otherwise.

(3) Retriangulate as in (3) of Algorithm 6.3.7

Subroutine GetRayCoord (G,C,$RayCoord$)

(1) Find the tetrahedron $T \subset G$ such that $C \in T$.

(2) If C is on an edge of the base face $[p_1p_2p_3]$, say, $[p_2p_3]$, Let $RayCoord = (0, s)$ where $(1, 0)$ is the barycentric coordinate of p_1'' (or q_1'') over edge $[p_1''p_4]$ (or $[q_1''q_4]$) and $(1 - s, s)$ is that of C over edge $[p_2, p_3]$

(3) Otherwise, let $RayCoord = R(C)$ with respect to T.

6.3.4 Distributed/Paralleled Rendering

The two algorithms can be speed up by distributing the computation to different machines or parallelizing into different processors. According to the number of servers available, patches are put into approximately the same number of groups and sent over to the servers for polygonization. We have implemented this distributed scheme in SplineX[10]. With servers running on 9 networked Sun Workstation, it takes less than 17 seconds to polygonize the satellite object (Figure 5.5.21) containing over 1392 A-patches. Each A-patch is polygonized into about 40 triangles.

We observes that as Unix processes are heavy-weighted and ray-shooting algorithms are fast, a heavy overhead is paid when performing the distributed computation. Dramatical speed-up can only be achieved when the solid consists a large number of A-patches, namely, there are a large number of patches within each distributed group.

6.4 Conclusion

We discuss issues regarding interactive shape controls and display of A-patches. We provide several simple heuristics to set the free weights to avoid unwanted variation of the surface. We propose simple effective interactive schemes to control the shape of the surface by changing a group of free weights.

We also present the ray shooting algorithms to rapidly polygonize A-patches. An animating ray shooting algorithm is also given to quickly modify a polygonization according to small changes of the coefficients.

We implement the schemes discussed in the SplineX system (see Appendix). In order to achieve better human-computer interactions, we display some feedback functions to help user navigate around the configuration space. User can choose to view, for example, Gaussian curvature or mean curvature by mapping the scalar value to a RGB triple or a gray scale, by showing iso contours on the surface, or by raising the surface along the normal direction to the amount of the curvature. We also develop a set of space widgets, with which user can modifies the solid effectively.

The shape control measurements given in this chapter are all very simple, in an effort to enhance the interactivity to the best. However, while in general effective, they fail to give fair variational surfaces in different cases. More sophisticated methods, such as energy-based optimizations could be used to improve the fairness of the surface. Chapter 7 presents the elastic A-patch model, one of whose purposes is variation surface shape control.

7. ELASTIC A-PATCH

7.1 Introduction

Variational surface modeling and physically-based modeling has received intensive attention recently (Section 1.4, Chapter 1). The motivations for developing an elastic model for the A-patch representation are mainly

- simulation of elastic objects;
- improvement of computer-human interaction in geometric design;
- surface optimization.

We propose two kinds of elastic surface models built on top of the A-patch representation. A curvature based energy model formulates the surface energy as the integral of the sum of the squares of the two principal curvatures, over the surface. Nonlinear optimization techniques are required to minimize this energy. The hyper energy is a simplified energy model we then propose to achieve interactivity. The hyper energy is formulated as the integral of a weighted sum of the squares of the first and second derivatives of the polynomial over the whole domain. Such an energy function is easy to be minimized as it can be formulated as a quadratic programming problem with linear inequality constraints.

The rest of the chapter is organized as follows. Section 7.2 gives a general energy model for A-patches. Section 7.3 proposes a curvature based energy model, and the optimization schemes associated with this model. Section 7.4 describes a simplified energy model. Section 7.5 discusses the optimization algorithms. Section 7.6 extends the optimization algorithms to distributed schemes. Section 7.7 concludes the chapter.

7.2 Energy Models for A-patch

We first describe an abstract energy model for a piecewise A-patch surface. The energy of a model is given as a weighted sum of the *surface energy* E_s, *curve energy* E_c and *external energy* E_e.

$$E = \zeta_s E_s + \zeta_c E_c + \zeta_e E_e$$

The surface energy E_s is the sum of the energy of each surface patch. The curve energy E_c is the sum of the energy of some boundary curves. The external energy E_e is the energy due to external sources.

The optimization of the energy can be described as constrained optimization problem with linear equality constraints (continuity constraints (6.2.3) and inequality constraints (single-sheeted and non-singular constraints (6.2.2).

$$\begin{aligned} \text{Minimize} \quad & E = \zeta_s E_s + \zeta_c E_c + \zeta_e E_e \\ \text{subject to} \quad & \boldsymbol{b}\mathbf{C}(\boldsymbol{p}) = \mathbf{0} \\ & \boldsymbol{b}\mathbf{S} > \mathbf{0} \end{aligned} \tag{7.2.1}$$

Generally speaking, an implicit surface does not have an obvious and simple parameterization as does a parametric surface. Based on this nature of an implicit surface, our first energy model is in terms of curvatures, which is parameterization independent. The model, however, involves high degree rational representations and thus the energy is difficult to minimized quickly. The second energy model approximates with the "energy" of the trivariate function over the tetrahedral mesh.

7.3 A Curvature Based Energy Model

7.3.1 The Model

Here we assume that the natural state is a straight line or a plane, and we only consider the potential energy that are due to bending.

For an A-spline,

$$E = \int_\Omega \kappa^2 d\Omega, \tag{7.3.2}$$

where

$$\kappa = \frac{\nabla S^{\mathsf{T}} \boldsymbol{J}\boldsymbol{P}\boldsymbol{J}^{\mathsf{T}} \nabla^2(S) \boldsymbol{J}\boldsymbol{P}^{\mathsf{T}}\boldsymbol{J}^{\mathsf{T}} \nabla S}{(\nabla S^{\mathsf{T}} \mathbf{J}\mathbf{J}^T \nabla S)^{3/2}}. \tag{7.3.3}$$

Hence, the bending energy

$$E = \int_\Omega \kappa^2 d\Omega$$

See Section 2.2 for the formula of κ.

We define the bending energy of the surface as

$$E = \int_\Omega (\kappa_1^2 + \kappa_2^2) d\Omega = \int_\Omega (H^2 - 2K) d\Omega$$

See Section 2.2 for the

7.3.2 Optimization

In this section, we study an energy-based surface optimization problem. That is, given some freedom in your surface representation, how to minimize some energy function defined over the surface.

An energy-minimized setting is a solution to

$$\nabla_{\boldsymbol{b}} E = \mathbf{o} \tag{7.3.1}$$

or

$$\int_\Omega 2\kappa \nabla_{\boldsymbol{b}} \kappa d\Omega = \mathbf{o} \tag{7.3.2}$$

$$\int_\Omega 2H \nabla_{\boldsymbol{b}} H + 2\nabla_{\boldsymbol{b}} K d\Omega = \mathbf{o} \tag{7.3.3}$$

$$\tag{7.3.4}$$

for an A-spline or an A-patch, respectively.

We already have the curvature formula of an A-patch. However, it is hard to symbolically calculate the energy. The difficulty is due to (1) there is no simple parameterization for an A-patch; (2) the integral function is at least a rational polynomial.

We propose two optimization schemes. The first scheme first restricts the freedom to be one just one variable, or one parameter. Then the energy minimizing equation is reduced to a non-linear univariate equation.

The second one is an iterative optimization scheme. Assume that we have a piecewise linear approximation of the A-patch to start with. Each piece is small enough so that we can assume that the gradient over each piece is pretty much a constant. By this simplification, we are able to reduce system 7.3.1 to a linear system. We repeat the process until the solution converge to a certain degree.

7.3.2.1 Restricted Solutions

System 7.3.1 is in general a nonlinear system of $\boldsymbol{b}$. A nonlinear system is not guaranteed to be solvable. However, by restrict the freedom to be one variable, we reduce to the system to be a univariate non-linear system, which is easy to solve.

Let $\boldsymbol{b}$ be a vector function of some parameter t. For an A-spline curve, system 7.3.1 is reduced to

$$\frac{\partial E(t)}{\partial t} = 0$$

Let

$$f(t) = \frac{dE(t)}{dt} = \int_\Omega (\kappa\kappa_t) d\Omega$$
$$f'(t) = \int_\Omega (\kappa_t^2 + \kappa\kappa_{tt}) d\Omega$$

$f(t)$ and $f'(t)$ are both continues. We can use standard methods, such as Newton's method, to solve for the roots. Please note that the evaluations of $f(t)$ and $f'(t)$ would involve numerical integrations. By choosing the free parameters carefully, we can separate the optimization process independently within each piece.

Bajaj, Chen, Holt and Netravali [11] explore different energy models in the context of A-splines representation. Figure 7.3.1 shows and an optimal quadratic A-spline (in solid) with the curvature-based energy model. The end points are constrained to

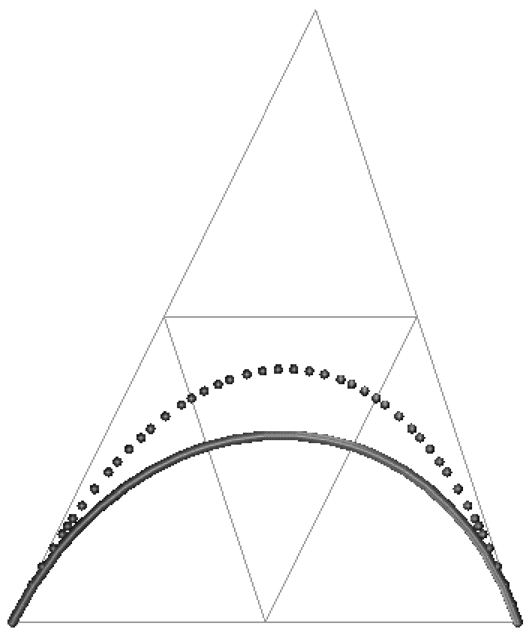


Figure 7.3.1 An optimal quatratic A-spline (in solid) with curvature-based energy model

pass through the vertices of the triangular simplex and furthermore, tangent the two edges. The dotted line shown is the initial configuration.

For a piece of face A-patch, let the number ④ ⑤ weights be functions of some parameter t, or simply set

$$a_{0003} = a_{1002} = a_{0102} = a_{0012} = t^2$$

Then

$$\begin{aligned} f(t) &= \frac{dE(t)}{dt} = \int \frac{\partial}{\partial t}(H^2 - 2K)d\Omega && (7.3.5) \\ f'(t) &= 2\int (\frac{d}{dt}H)^2 + H\frac{d^2}{dt^2}H - \frac{d^2}{dt^2}K)d\Omega && (7.3.6) \end{aligned}$$

$f(t) = 0$ hence can be solved by standard methods, such as Newton's method. However, if we include the edge patches around the face patch, because of the inequality constraints, $f'(t)$ may not be continuous, we use the bisection method or the secant method instead to solve for the root of $f(t) = 0$

7.3.2.2 Linear System Solutions

Another effort we make to overcome the non-linearity of the multivariate system is an iterative scheme, which starts with an initial value $\boldsymbol{b}_0$, repeatedly improves the setting by solving a linear system each each iteration. Instead of integrating along the surface, we take an approximation by integrating along some uniform piecewise linear subdivision of the curve/patch. Methods to generate piecewise linear subdivision of algebraic surface, or specifically A-splines/A-patches can be found in [141, 15].

We first discuss the case of an A-spline. Assume that at iteration i, the weights are given as $\boldsymbol{b}^{(i)}$. We compute a uniform piecewise linear subdivision of the curve $\{(u_j^{(i)}, v_j^{(i)})\}$, $j = 0..n$. Method The subdivision is fine enough so that we can assume that $h(\boldsymbol{b}, u, v) = (\nabla^{\mathsf{T}} S \boldsymbol{J}\boldsymbol{J}^{\mathsf{T}} \nabla S)^{\frac{3}{2}}$ in the denominator of the curvature formula 7.3.3 is a constant.

We substitute $\nabla^2(S(\boldsymbol{b}^i))$ by $\nabla^2(S(\boldsymbol{b}^{i+1}))$. In the k-th piece, the denominator $h(\boldsymbol{b}, u, v)$ is substituted by $\overline{h}_j = \frac{1}{2}(h(\boldsymbol{b}^{(i)}, u_j^{(i)}, v_j^{(i)}) + h(\boldsymbol{b}^{(i)}, u_{j+1}^{(i)}, v_{j+1}^{(i)}))$.

Hence the curvature

$$\kappa_j^{(i)} = \frac{\nabla S(\boldsymbol{b}^{(i)})^{\mathsf{T}} \boldsymbol{J P} J^{\mathsf{T}} \nabla^2 (S(\boldsymbol{b}^{(i+1)})) \boldsymbol{J P}^{\mathsf{T}} J^{\mathsf{T}} \nabla S(\boldsymbol{b}^{(i)})}{\overline{h}_j} \tag{7.3.7}$$

The energy minimizing system

$$\sum_j \nabla_{\boldsymbol{b}^{(i+1)}} E^{(i)} = 0$$

where

$$\begin{aligned} \nabla_{\boldsymbol{b}^{(i+1)}} E^{(i)} &= \int_{\Omega} 2\kappa_j^{(i)} \nabla_{\boldsymbol{b}^{(i+1)}} \kappa_j^{(i)} d\Omega & (7.3.8) \\ &= \frac{1}{\overline{h}_j^2} \int_{\Omega} (2(\boldsymbol{N})^{(i)^{\mathsf{T}}} \nabla (S(\boldsymbol{b}^{(i+1)})) \boldsymbol{N}^{(i)} \boldsymbol{N}^{(i)^{\mathsf{T}}} \\ &\quad \nabla_{\boldsymbol{b}^{(i+1)}} \nabla^2 (S(\boldsymbol{b}^{(i+1)})) \boldsymbol{N}^{(i)}) d\Omega & (7.3.9) \end{aligned}$$

where

$$\boldsymbol{N}^{(i)} = \boldsymbol{J P}^{\mathsf{T}} \boldsymbol{J}^{\mathsf{T}} \nabla S(\boldsymbol{b}^{(i)}.$$

Now the problem is how to symbolically integrate along an implicit curve Ω : $S(u,v) = 0$. We instead integrate along piecewise straight lines $[(u_j^{(i)}, v_j^{(i)}), (u_{j+1}^{(i)}, v_{j+1}^{(i)})]$. Hence

$$\begin{aligned} u^{(i)}(t) &= u_j^{(i)}(1-t) + u_{j+1}^{(i)} t \\ v^{(i)}(t) &= v_j^{(i)}(1-t) + v_{j+1}^{(i)} t \end{aligned}$$

$$d\Omega = \sqrt{(du)^2 + (dv)^2} = \sqrt{(u_{j+1}^{(i)} - u_j^{(i)})^2 + (v_{j+1}^{(i)} - v_j^{(i)})^2} dt = l_j dt$$

The energy minimizing system is approximated as

$$\sum_j \frac{1}{\overline{h}_j^2} \int_0^1 2\boldsymbol{N}^{(i)^{\mathsf{T}}} \nabla^2 (S(\boldsymbol{b}^{(i+1)})) \boldsymbol{N}^{(i)} \boldsymbol{N}^{(i)^{\mathsf{T}}} \nabla_{\boldsymbol{b}^{(i+1)}} \nabla^2 (S(\boldsymbol{b}^{(i+1)})) \boldsymbol{N}^{(i)} l_j dt = \mathbf{o}$$

Note that the each integral function is a polynomial, hence the integration can be symbolically done off-line. Furthermore, the right hand side is linear in terms of $\boldsymbol{b}^{(i+1)}$. Hence $\boldsymbol{b}^{(i+1)}$ can be solved by solving the linear system.

We iterate with this procedure until $||\boldsymbol{b}^{(i+1)} - \boldsymbol{b}^{(i+1)}||$ is less than some ϵ.

In the A-patch case, formulas are similar.

7.4 The Hyperenergy Model

In order to have a "cheap" solution to the A-patch optimization problem, we propose a simplified energy model. Instead of define an energy model over the surface, we define one over the volume of the simplex within which the A-patch is defined. Specifically, for an implicit surface $F(x) = 0$ defined in tetrahedron $[p_0p_1p_2p_3]$, we define an energy model to be the integral of a weighted square sum of the first and second derivatives of the function $F(x)$ over the tetrahedron. We call it the *hyperenergy* of the A-patch. Hyperenergy function is very easy to minimized as it can be expressed as a quadratic polynomial function of the coefficients of $F(x)$. On the other hand, hyperenergy is a reasonable simplification based on the observation that if a surface is fair, its iso-contours are fair also.

7.4.1 Bending Energy

The bending energy is formulated in terms of $\nabla^2 F$, the second derivatives of F. Let matrix

$$\mathbf{H} = \nabla^2 F = \begin{pmatrix} H_{11} & H_{12} & H_{13} \\ H_{12} & H_{22} & H_{23} \\ H_{13} & H_{23} & H_{33} \end{pmatrix} \tag{7.4.10}$$

We propose two kinds of bending hyperenergy.

$$\begin{aligned} \text{Curve: } E_B^P &= \int \beta_{11}H_{11}^2 + \beta_{22}H_{22}^2 + 2\beta_{12}H_{12}^2 d\Omega & (7.4.11) \\ \text{Surface:} E_B^P &= \int \beta_{11}H_{11}^2 + \beta_{22}H_{22}^2 + \beta_{33}H_{33}^2 + 2\beta_{12}H_{12}^2 + \\ & \quad 2\beta_{23}H_{23}^2 + 2\beta_{31}H_{31}^2 d\Omega & (7.4.12) \end{aligned}$$

E_B^P model optimize the surface toward a straight line or a plane.

$$\begin{aligned} E_B^S &= \int \gamma_{11}(H_{11} - H_{22})^2 + 2\gamma_{12}H_{12}^2 d\Omega & (7.4.13) \\ E_B^S &= \int \gamma_{11}(H_{11} - H_{22})^2 + \gamma_{22}(H_{22} - H_{33})^2 + \gamma_{33}(H_{33} - H_{11})^2 + \end{aligned}$$

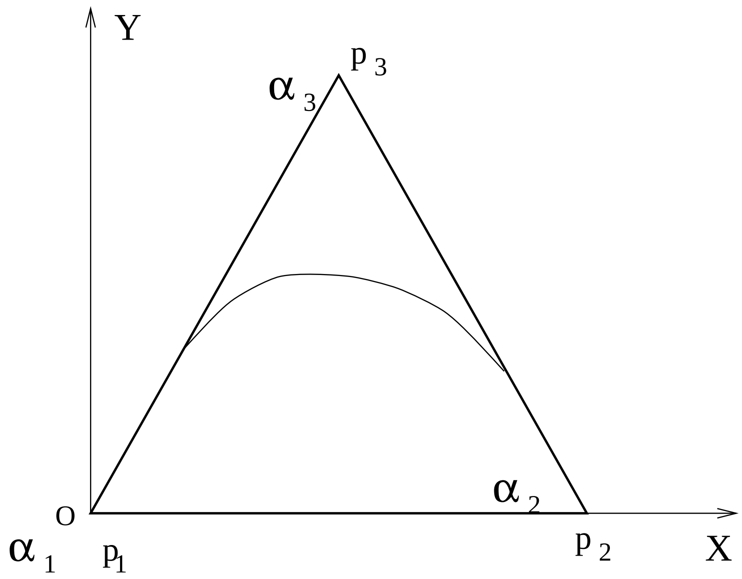


Figure 7.4.2 The hyperenergy model of an A-spline. $[p_1p_2]$ is parallel to X-axis. Only the derivatives in X-axis count.

$$2\gamma_{12}H_{12}^2 + 2\gamma_{23}H_{23}^2 + 2\gamma_{13}H_{31}^2 d\Omega \tag{7.4.14}$$

E_B^S model optimize the surface toward a circle or a sphere.

For an A-spline curve, we are only interested in variation of the curve, which can be approximated by the variation of polynomial along one direction, and is rather independent to the variation in the other direction. Hence, we set the X-axis to be parallel to $[p_1p_2]$, and

$$\beta_{22} = \beta_{12} = 0$$

in Equation (7.4.11).

Similarly, as shown in Figure 7.4.3, we set the XOY plane to be parallel to $[p_1p_2p_3]$ for a 3-sided A-patch, or parallel to $[p_1p_2]$ and $[p_3p_4]$ for a 4-sided A-patch. And we ignore the variations in Z direction by setting

$$\beta_{33} = \beta_{23} = \beta_{13} = 0$$

in Equation (7.4.12).

We may further set

$$\beta_{11} = \beta_{22}$$

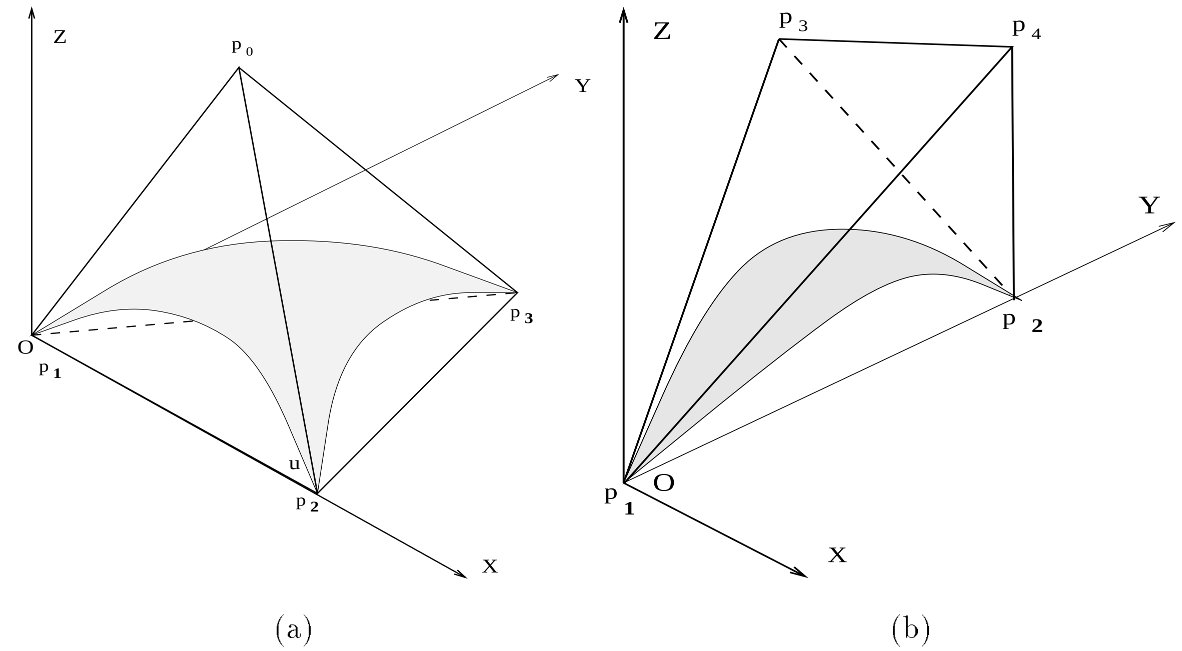


Figure 7.4.3 The hyperenergy model of A-patches. Variation in Z direction is omitted. (a) 3-sided patch: $[p_1p_2p_3]$ is parallel to XOY. (b) 4-sided patch: $[p_1p_2]$ and $[p_3p_4]$ are parallel to XOY.

so that the energy is invariant to the orientation of the XOY frame.

We develop the formulae in the trivariate case. Let the Jacobian matrix

$$\mathbf{J} = (\mathbf{J}_1, \mathbf{J}_2, \mathbf{J}_3) = \begin{pmatrix} J_{11} & J_{12} & J_{13} \\ J_{12} & J_{22} & J_{23} \\ J_{13} & J_{23} & J_{33} \end{pmatrix}$$

Then,

$$H_{12} = \boldsymbol{J}_1{}^{\top} \nabla^2 (S(\boldsymbol{\alpha})) \boldsymbol{J}_2 \tag{7.4.15}$$

$$= \boldsymbol{J}_1{}^{\top} \begin{pmatrix} H_{\alpha_1\alpha_1} & H_{\alpha_1\alpha_2} & H_{\alpha_1\alpha_3} \\ H_{\alpha_1\alpha_2} & H_{\alpha_2\alpha_2} & H_{\alpha_2\alpha_3} \\ H_{\alpha_1\alpha_3} & H_{\alpha_2\alpha_3} & H_{\alpha_3\alpha_3} \end{pmatrix} \boldsymbol{J}_2 \tag{7.4.16}$$

$$= \boldsymbol{J}_1{}^{\top} \begin{pmatrix} \boldsymbol{b}^{\top}\mathcal{D}_{\alpha_1\alpha_1}B & \boldsymbol{b}^{\top}\mathcal{D}_{\alpha_1\alpha_2}B & \boldsymbol{b}^{\top}\mathcal{D}_{\alpha_1\alpha_3}B \\ \boldsymbol{b}^{\top}\mathcal{D}_{\alpha_1\alpha_2}B & \boldsymbol{b}^{\top}\mathcal{D}_{\alpha_2\alpha_2}B & \boldsymbol{b}^{\top}\mathcal{D}_{\alpha_2\alpha_3}B \\ \boldsymbol{b}^{\top}\mathcal{D}_{\alpha_1\alpha_3}B & \boldsymbol{b}^{\top}\mathcal{D}_{\alpha_2\alpha_3}B & \boldsymbol{b}^{\top}\mathcal{D}_{\alpha_3\alpha_3}B \end{pmatrix} \boldsymbol{J}_2 \tag{7.4.17}$$

$$= \sum_{i=1}^{3}\sum_{j=1}^{3} J_{1i}J_{2j}\boldsymbol{b}^{\top}\mathcal{D}_{\alpha_i\alpha_j}B \tag{7.4.18}$$

$$= \boldsymbol{b}^{\top} \left(\sum_{i=1}^{3}\sum_{j=1}^{3} J_{1i}J_{2j}\mathcal{D}_{\alpha_i\alpha_j} \right) B \tag{7.4.19}$$

$$= \boldsymbol{b}^{\top} \left(\boldsymbol{J}_1{}^{\top} \nabla^2 \mathcal{D} \boldsymbol{J}_2 \right) B \tag{7.4.20}$$

where

$$\nabla^2\mathcal{D} = \begin{pmatrix} \mathcal{D}_{\alpha_1\alpha_1} & \mathcal{D}_{\alpha_1\alpha_2} & \mathcal{D}_{\alpha_1\alpha_3} \\ \mathcal{D}_{\alpha_1\alpha_2} & \mathcal{D}_{\alpha_2\alpha_2} & \mathcal{D}_{\alpha_2\alpha_3} \\ \mathcal{D}_{\alpha_1\alpha_3} & \mathcal{D}_{\alpha_2\alpha_3} & \mathcal{D}_{\alpha_3\alpha_3} \end{pmatrix}$$

Note that, in Equation 7.4.15, the second factor

$$\boldsymbol{J}_1{}^{\top} \nabla^2 \mathcal{D} \boldsymbol{J}_2$$

yields a matrix of $m \times m'$, with m being the number of coefficients in degree n polynomial $F(\alpha)$ (2.1.14), and m' being the number of coefficients in the derivatives, $F_{\alpha_i}(\alpha)$, a degree $n-1$ polynomial. Each entry of $\nabla^2\mathcal{D}$ is itself a matrix of $m \times m'$. This augmented convention are used in the following text.

In general,

$$H_{ij} = \boldsymbol{b}^{\mathsf{T}} \left(\boldsymbol{J}_i{}^{\mathsf{T}} \nabla^2 \mathcal{D} \boldsymbol{J}_j \right) B \tag{7.4.21}$$

$$\begin{aligned}
\int H_{ij}^2 d\Omega &= \int \boldsymbol{b}^{\mathsf{T}} \left(\boldsymbol{J}_i{}^{\mathsf{T}} \nabla^2 \mathcal{D} \boldsymbol{J}_j \right) B B^{\mathsf{T}} \left(\boldsymbol{J}_i{}^{\mathsf{T}} \nabla^2 \mathcal{D} \boldsymbol{J}_j \right) \boldsymbol{b} d\Omega \\
&= \boldsymbol{b}^{\mathsf{T}} \boldsymbol{J}_i{}^{\mathsf{T}} \nabla^2 \mathcal{D} \boldsymbol{J}_j \left(\int B B^{\mathsf{T}} d\Omega \right) \boldsymbol{J}_i{}^{\mathsf{T}} \nabla^2 \mathcal{D} \boldsymbol{J}_j \boldsymbol{b} \qquad (7.4.22) \\
&= \boldsymbol{b}^{\mathsf{T}} \mathbf{K}_B^{ij} \boldsymbol{b} \qquad (7.4.23)
\end{aligned}$$

where

$$\mathbf{K}_B^{ij} = \boldsymbol{J}_i{}^{\mathsf{T}} \nabla^2 \mathcal{D} \boldsymbol{J}_j \left(\int B B^{\mathsf{T}} d\Omega \right) \boldsymbol{J}_i{}^{\mathsf{T}} \nabla^2 \mathcal{D} \boldsymbol{J}_j$$

The notation

$$\int \mathbf{M} d\Omega,$$

where $\mathbf{M}$ is a matrix, denotes that the integration is taken place in each of $\mathbf{M}$'s elements. BB^{T} is a matrix of $m' \times m'$. We list some of the "integrated" matrix $\int BB^{\mathsf{T}} d\Omega$ as follows.

Bivariate linear:

$$\int B B^{\mathsf{T}} d\Omega = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$$

Bivariate quadratic:

$$\int B B^{\mathsf{T}} d\Omega = \frac{1}{180} \begin{pmatrix} 6 & 3 & 1 & 3 & 1 & 1 \\ 3 & 4 & 3 & 2 & 2 & 1 \\ 1 & 3 & 6 & 1 & 3 & 1 \\ 3 & 2 & 1 & 4 & 2 & 3 \\ 1 & 2 & 3 & 2 & 4 & 3 \\ 1 & 1 & 1 & 3 & 3 & 6 \end{pmatrix}$$

$$\int BB^{\mathsf{T}} d\Omega = \frac{1}{1120} \begin{pmatrix} 20 & 10 & 4 & 1 & 10 & 4 & 1 & 4 & 1 & 1 \\ 10 & 12 & 9 & 4 & 6 & 6 & 3 & 3 & 2 & 1 \\ 4 & 9 & 12 & 10 & 3 & 6 & 6 & 2 & 3 & 1 \\ 1 & 4 & 10 & 20 & 1 & 4 & 10 & 1 & 4 & 1 \\ 10 & 6 & 3 & 1 & 12 & 6 & 2 & 9 & 3 & 4 \\ 4 & 6 & 6 & 4 & 6 & 8 & 6 & 6 & 6 & 4 \\ 1 & 3 & 6 & 10 & 2 & 6 & 12 & 3 & 9 & 4 \\ 4 & 3 & 2 & 1 & 9 & 6 & 3 & 12 & 6 & 10 \\ 1 & 2 & 3 & 4 & 3 & 6 & 9 & 6 & 12 & 10 \\ 1 & 1 & 1 & 1 & 4 & 4 & 4 & 10 & 10 & 20 \end{pmatrix}$$

Table 7.4.1 Bivariate cubic $\int BB^{\mathsf{T}} d\Omega$

$$\int BB^{\mathsf{T}} d\Omega = \frac{1}{1260} \begin{pmatrix} 6 & 3 & 1 & 3 & 1 & 1 & 3 & 1 & 1 & 1 \\ 3 & 4 & 3 & 2 & 2 & 1 & 2 & 2 & 1 & 1 \\ 1 & 3 & 6 & 1 & 3 & 1 & 1 & 3 & 1 & 1 \\ 3 & 2 & 1 & 4 & 2 & 3 & 2 & 1 & 2 & 1 \\ 1 & 2 & 3 & 2 & 4 & 3 & 1 & 2 & 2 & 1 \\ 1 & 1 & 1 & 3 & 3 & 6 & 1 & 1 & 3 & 1 \\ 3 & 2 & 1 & 2 & 1 & 1 & 4 & 2 & 2 & 3 \\ 1 & 2 & 3 & 1 & 2 & 1 & 2 & 4 & 2 & 3 \\ 1 & 1 & 1 & 2 & 2 & 3 & 2 & 2 & 4 & 3 \\ 1 & 1 & 1 & 1 & 1 & 1 & 3 & 3 & 3 & 6 \end{pmatrix}$$

Table 7.4.2 Trivariate quadratic $\int BB^{\mathsf{T}} d\Omega$

$$\int BB^{\mathsf{T}} d\Omega = \tfrac{1}{1260} \cdot$$

$$\begin{pmatrix}
20 & 10 & 4 & 1 & 10 & 4 & 1 & 4 & 1 & 1 & 10 & 4 & 1 & 4 & 1 & 1 & 4 & 1 & 1 & 1 \\
10 & 12 & 9 & 4 & 6 & 6 & 3 & 3 & 2 & 1 & 6 & 6 & 3 & 3 & 2 & 1 & 3 & 2 & 1 & 1 \\
4 & 9 & 12 & 10 & 3 & 6 & 6 & 2 & 3 & 1 & 3 & 6 & 6 & 2 & 3 & 1 & 2 & 3 & 1 & 1 \\
1 & 4 & 10 & 20 & 1 & 4 & 10 & 1 & 4 & 1 & 1 & 4 & 10 & 1 & 4 & 1 & 1 & 4 & 1 & 1 \\
10 & 6 & 3 & 1 & 12 & 6 & 2 & 9 & 3 & 4 & 6 & 3 & 1 & 6 & 2 & 3 & 3 & 1 & 2 & 1 \\
4 & 6 & 6 & 4 & 6 & 8 & 6 & 6 & 6 & 4 & 3 & 4 & 3 & 4 & 4 & 3 & 2 & 2 & 2 & 1 \\
1 & 3 & 6 & 10 & 2 & 6 & 12 & 3 & 9 & 4 & 1 & 3 & 6 & 2 & 6 & 3 & 1 & 3 & 2 & 1 \\
4 & 3 & 2 & 1 & 9 & 6 & 3 & 12 & 6 & 10 & 3 & 2 & 1 & 6 & 3 & 6 & 2 & 1 & 3 & 1 \\
1 & 2 & 3 & 4 & 3 & 6 & 9 & 6 & 12 & 10 & 1 & 2 & 3 & 3 & 6 & 6 & 1 & 2 & 3 & 1 \\
1 & 1 & 1 & 1 & 4 & 4 & 4 & 10 & 10 & 20 & 1 & 1 & 1 & 4 & 4 & 10 & 1 & 1 & 4 & 1 \\
10 & 6 & 3 & 1 & 6 & 3 & 1 & 3 & 1 & 1 & 12 & 6 & 2 & 6 & 2 & 2 & 9 & 3 & 3 & 4 \\
4 & 6 & 6 & 4 & 3 & 4 & 3 & 2 & 2 & 1 & 6 & 8 & 6 & 4 & 4 & 2 & 6 & 6 & 3 & 4 \\
1 & 3 & 6 & 10 & 1 & 3 & 6 & 1 & 3 & 1 & 2 & 6 & 12 & 2 & 6 & 2 & 3 & 9 & 3 & 4 \\
4 & 3 & 2 & 1 & 6 & 4 & 2 & 6 & 3 & 4 & 6 & 4 & 2 & 8 & 4 & 6 & 6 & 3 & 6 & 4 \\
1 & 2 & 3 & 4 & 2 & 4 & 6 & 3 & 6 & 4 & 2 & 4 & 6 & 4 & 8 & 6 & 3 & 6 & 6 & 4 \\
1 & 1 & 1 & 1 & 3 & 3 & 3 & 6 & 6 & 10 & 2 & 2 & 2 & 6 & 6 & 12 & 3 & 3 & 9 & 4 \\
4 & 3 & 2 & 1 & 3 & 2 & 1 & 2 & 1 & 1 & 9 & 6 & 3 & 6 & 3 & 3 & 12 & 6 & 6 & 10 \\
1 & 2 & 3 & 4 & 1 & 2 & 3 & 1 & 2 & 1 & 3 & 6 & 9 & 3 & 6 & 3 & 6 & 12 & 6 & 10 \\
1 & 1 & 1 & 1 & 2 & 2 & 2 & 3 & 3 & 4 & 3 & 3 & 3 & 6 & 6 & 9 & 6 & 6 & 12 & 10 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 4 & 4 & 4 & 4 & 4 & 4 & 10 & 10 & 10 & 20
\end{pmatrix}$$

Table 7.4.3 Trivariate cubic $\int BB^{\mathsf{T}} d\Omega$

See Tables 7.4.1,7.4.2,7.4.3 for several other cases that are of interest.

From the above derivation, E_B is a quadratic polynomial function of $\boldsymbol{b}$

$$E_B(\boldsymbol{b}) = \boldsymbol{b}^{\mathsf{T}}\mathbf{K}_B\boldsymbol{b}$$

where

$$\mathbf{K}_B = \sum_{ij} \beta_{ij}\mathbf{K}_B^{ij} \tag{7.4.24}$$

$\mathbf{K}_B$, a $m \times m$ matrix is called the *bending matrix* of the patch.

The partial derivatives of H_{ij} is given as

$$\begin{aligned}
\frac{\partial}{\partial b_k}H_{ij} &= \boldsymbol{J}_i{}^{\mathsf{T}}\frac{\partial}{\partial b_k}\nabla^2 S(\alpha))\boldsymbol{J}_j & (7.4.25)\\
&= B^{\mathsf{T}}\left(\boldsymbol{J}_i{}^{\mathsf{T}}\begin{pmatrix} \mathcal{D}^k_{\alpha_1\alpha_1}{}^{\mathsf{T}} & \mathcal{D}^k_{\alpha_1\alpha_2}{}^{\mathsf{T}} & \mathcal{D}^k_{\alpha_1\alpha_3}{}^{\mathsf{T}} \\ \mathcal{D}^k_{\alpha_1\alpha_2}{}^{\mathsf{T}} & \mathcal{D}^k_{\alpha_2\alpha_2}{}^{\mathsf{T}} & \mathcal{D}^k_{\alpha_2\alpha_3}{}^{\mathsf{T}} \\ \mathcal{D}^k_{\alpha_1\alpha_3}{}^{\mathsf{T}} & \mathcal{D}^k_{\alpha_2\alpha_3}{}^{\mathsf{T}} & \mathcal{D}^k_{\alpha_3\alpha_3}{}^{\mathsf{T}} \end{pmatrix}\boldsymbol{J}_j\right) & (7.4.26)\\
&= B^{\mathsf{T}}\left(\boldsymbol{J}_i{}^{\mathsf{T}}\nabla^2\mathcal{D}^k{}^{\mathsf{T}}\boldsymbol{J}_j\right) & (7.4.27)
\end{aligned}$$

where

$$\nabla^2\mathcal{D}^k = \begin{pmatrix} \mathcal{D}^k_{\alpha_1\alpha_1} & \mathcal{D}^k_{\alpha_1\alpha_2} & \mathcal{D}^k_{\alpha_1\alpha_3} \\ \mathcal{D}^k_{\alpha_1\alpha_2} & \mathcal{D}^k_{\alpha_2\alpha_2} & \mathcal{D}^k_{\alpha_2\alpha_3} \\ \mathcal{D}^k_{\alpha_1\alpha_3} & \mathcal{D}^k_{\alpha_2\alpha_3} & \mathcal{D}^k_{\alpha_3\alpha_3} \end{pmatrix}$$

and $D^k_{\alpha_1\alpha_1}$ denotes the k-th row of $D_{\alpha_1\alpha_1}$. Similarly, $D^k_{\alpha_1\alpha_2}$ and $D^k_{\alpha_2\alpha_3}$, etc..

$$\begin{aligned}
\frac{\partial}{\partial b_k}\left(\int H_{ij}^2 d\Omega\right) &= \int \frac{\partial}{\partial b_k}H_{ij}^2 d\Omega \\
&= \int 2H_{ij}\frac{\partial}{\partial b_k}H_{ij}d\Omega & (7.4.28)\\
&= \int 2\boldsymbol{b}^{\mathsf{T}}\left(\boldsymbol{J}_i{}^{\mathsf{T}}\nabla^2\mathcal{D}\boldsymbol{J}_j\right)BB^{\mathsf{T}}\left(\boldsymbol{J}_i{}^{\mathsf{T}}\nabla^2\mathcal{D}^k\boldsymbol{J}_j\right)d\Omega & (7.4.29)
\end{aligned}$$

$$= \quad 2\boldsymbol{b}^\top \left(\boldsymbol{J}_i{}^\top \nabla^2 \mathcal{D} \boldsymbol{J}_j\right) \int BB^\top d\Omega \left(\boldsymbol{J}_i{}^\top \nabla^2 \mathcal{D}^k \boldsymbol{J}_j\right) \qquad (7.4.30)$$

7.4.2 Stretching Energy

The stretching energy is given in terms of ∇F, the first derivative of F. Let vector

$$\boldsymbol{G} = \nabla F$$

The stretching energy is formulated as

$$E_{strch} = \int_\Omega \boldsymbol{G}^\top \boldsymbol{\Phi} \boldsymbol{G} d\Omega$$

where, in the trivariate case,

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_{11} & \phi_{12} & \phi_{13} \\ \phi_{12} & \phi_{22} & \phi_{23} \\ \phi_{13} & \phi_{23} & \phi_{33} \end{pmatrix}$$

Similar to the bending energy model, as $[p_1 p_2 p_3]$ is parallel to XOY for a 3-sided patch, or $[p_1 p_2]$ and $[p_3 p_4]$ are parallel to XOY for a 4-sided patch, the stretching in Z direction is omitted. Namely,

$$\phi_{13} = \phi_{23} = \phi_{33} = 0 \ .$$

And we may set

$$\phi_{11} = \phi_{22}$$

so that the stretching energy is invariant of the orientation of the XOY frame.

In the bivariate case,

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_{11} & \phi_{12} \\ \phi_{12} & \phi_{22} \end{pmatrix}$$

with $\phi_{12} = \phi_{22} = 0$, providing that $[p_1 p_2]$ is parallel to X axis.

We layout the formulation in the trivariate case.

as

$$\begin{aligned}
\boldsymbol{G} &= \nabla F = \mathbf{J}^{\mathsf{T}} \nabla S && (7.4.31) \\
&= \mathbf{J}^{\mathsf{T}} \begin{pmatrix} \boldsymbol{b}^{\mathsf{T}} \mathcal{D}_{\alpha_1} B \\ \boldsymbol{b}^{\mathsf{T}} \mathcal{D}_{\alpha_2} B \\ \boldsymbol{b}^{\mathsf{T}} \mathcal{D}_{\alpha_3} B \end{pmatrix} && (7.4.32) \\
&= \boldsymbol{b}^{\mathsf{T}} \left(\mathbf{J} \nabla \mathcal{D} \right) B && (7.4.33) \\
G_i &= \boldsymbol{J}_i{}^{\mathsf{T}} \nabla S \qquad i = 1, 2, 3, && (7.4.34) \\
&= \boldsymbol{b}^{\mathsf{T}} \left(\boldsymbol{J}_i{}^{\mathsf{T}} \nabla \mathcal{D} \right) B && (7.4.35)
\end{aligned}$$

we have

$$\begin{aligned}
\int \boldsymbol{G}^{\mathsf{T}} \boldsymbol{\Phi} \boldsymbol{G} d\Omega &= \int \sum_{i=1}^{3} \sum_{j=1}^{3} \phi_{ij} G_i G_j d\Omega && (7.4.36) \\
&= \sum_{i=1}^{3} \sum_{j=1}^{3} \phi_{ij} \boldsymbol{b}^{\mathsf{T}} \left(\boldsymbol{J}_i{}^{\mathsf{T}} \nabla \mathcal{D}^{\mathsf{T}} \left(\int B B^{\mathsf{T}} d\Omega \right) \nabla \mathcal{D} \boldsymbol{J}_j \right) \boldsymbol{b} && (7.4.37) \\
&= \boldsymbol{b}^{\mathsf{T}} \mathbf{K}_L \boldsymbol{b} && (7.4.38)
\end{aligned}$$

where

$$\mathbf{K}_L = \sum_{i=1}^{3} \sum_{j=1}^{3} \phi_{ij} \left(\boldsymbol{J}_i{}^{\mathsf{T}} \nabla \mathcal{D}^{\mathsf{T}} \left(\int B B^{\mathsf{T}} d\Omega \right) \nabla \mathcal{D} \boldsymbol{J}_j \right)$$

Hence E_L is a quadratic polynomial function of $\boldsymbol{b}$.

$$E_L = \boldsymbol{b}^{\mathsf{T}} \mathbf{K}_L \boldsymbol{b}$$

Matrix $\mathbf{K}_L$, a $m \times m$ matrix, is called the *stretching matrix* of the patch.

The surface energy is a weighted sum of the bending energy and the stretching energy.

$$E_S = E_B + E_L = \boldsymbol{b}^{\mathsf{T}} (\mathbf{K}_B + \mathbf{K}_L) \boldsymbol{b} = \boldsymbol{b}^{\mathsf{T}} \mathbf{K}_S \boldsymbol{b}$$

where matrix $\mathbf{K_S}$ is called the *surface stiffness matrix*.

7.4.3 Curve Network Energy

The minimum variation network approach has been explored by many researchers for surface optimization. It has been proven to yield good results. We adopt a similar approach in our scheme, based on the observation that a hyperenergy over an A-spline curve is easier to control. We define a curve network consisting of the boundary curves of each A-patch, which are actually A-splines. The curve network energy is given as

$$E_C = E_{C,B} + E_{C,L} = \boldsymbol{b}^{\mathsf{T}}(\mathbf{K}_{C,B} + \mathbf{K}_{C,L})\boldsymbol{b} = \boldsymbol{b}^{\mathsf{T}}\mathbf{K}_C\boldsymbol{b}$$

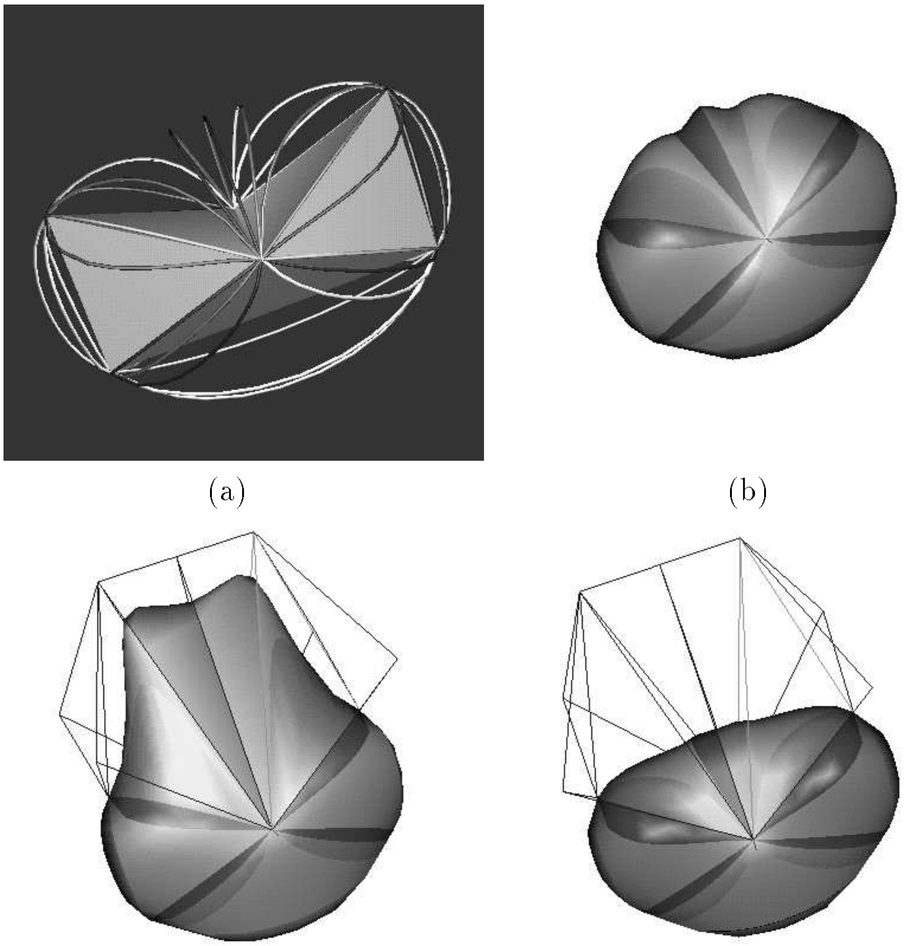
where $E_{C,B}$ is the curve bending energy and $E_{C,L}$ is the curve stretching energy. $\mathbf{K}_C = \mathbf{K}_B + \mathbf{K}_L$ is called the *curve stiffness matrix.*

Figure 7.4.4 shows the role of the curve network in optimization of A-patch surface interpolating the vertexes of a simple nonconvex object (shown in Figure 7.4.4 (a)). An optimization without curve network energy model could result in undesirable configuration (Figure 7.4.4 (c)). Figure 7.4.5 shows that an A-patch surface smoothing a prism is optimized by minimizing a weighted sum of surface energy and curve network energy.

7.4.4 Refinement of the Hyperenergy Model

An obvious problem of the hyperenergy is that, the size of the domain simplex, which is rather independent to the shape of the implicit polynomial, plays a role in the surface optimization. We propose a progressive subdivision scheme to downplay the bad effect.

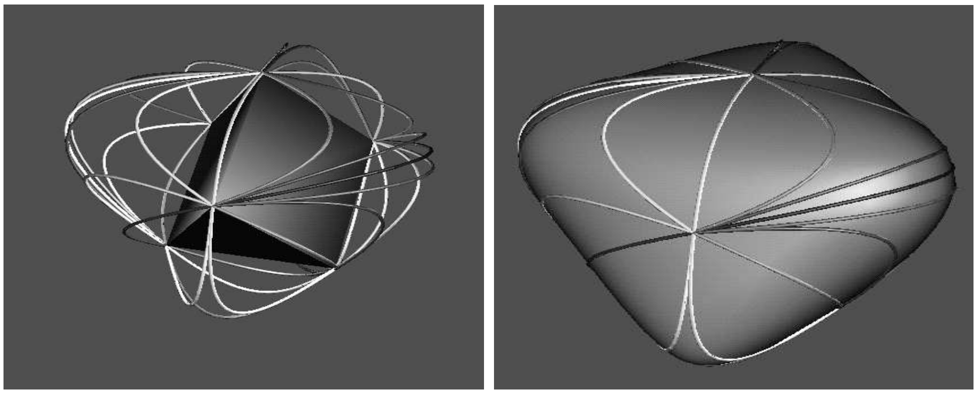
Instead of optimize the surface over the whole domain simplex, we systematically subdivide the simplex into small sub-simplex. And the hyperenergy is minimized only over the ones that the zero contour passes through. One iteration of optimization may shift the zero contour out of one sub-simplex into another. Hence before the next iteration, the list of non-empty sub-simplex is updated by checking the coefficients sub-polynomials.



(a) (b)

(c) (d)

Figure 7.4.4 Curve network energy (a) The solid and the curve network. (b) Initial configuration. (c) Global optimization without curve network energy (d) Global optimization with curve network energy



(a) (b)

Figure 7.4.5 An Optimized smooth interpolation of a prism

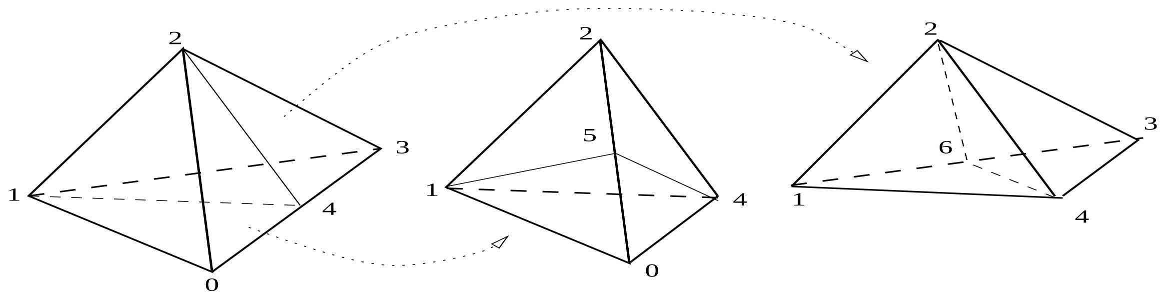


Figure 7.4.6 Two steps of edge subdivision on a tetrahedron

The edge de Casteljeu subdivision used by Peters [103] is a suitable scheme for our purpose. The edge de Casteljeu subdivides a simplex (for example, a triangle or a tetrahedron) into two by evaluation of the polynomial at the midpoint point of one edge of the simplex. The subdivision is repeated until certain degree of refinement is reached. The new vertex is added in the following add-and-shift way. For a m dimensional simplex $V = [v_0, v_1, ..., v_m]$ in the jth iteration, a new vertex is introduced as

$$w_i = \frac{v_i + v_j}{2}, \qquad i = j \bmod m$$

And V is split into

$$\begin{aligned} V_1 &= [v_0, v_1, ..., w_i, v_{i+1}, ..., v_m] \\ V_2 &= [v_0, v_1, ..., w_i, v_i, ..., v_{m-1}] \end{aligned}$$

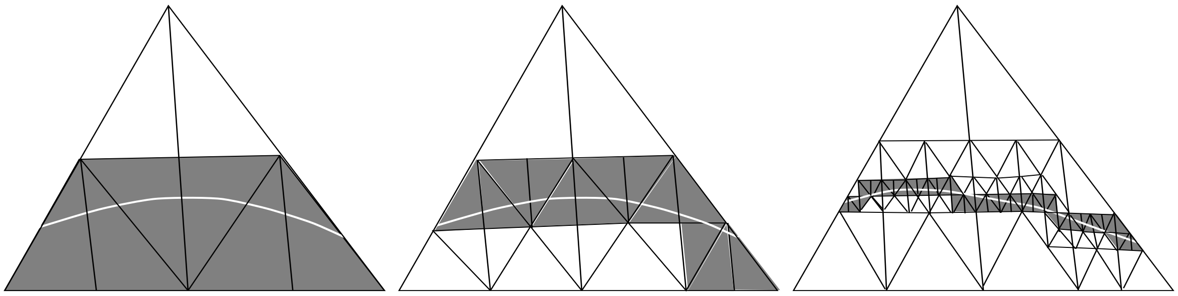
Figure 7.4.6 shows two iteration of subdivisions.

Such a subdivision is very stable as only averaging is involved. The coefficients of a subpolynomial can be expressed as the coefficients of the old polynomial times a matrix with just nonnegative elements. Such a matrix is called an *edge subdivision matrix*.

Figure 7.4.7 shows a refinement in bivariate simplex.

7.4.5 External Energy: Algebraic Distance Approximation

The external energy model is designated to model the influences external objects on the surface. A simple model we propose here is, given three dimensional points



(a) (b) (c)

Figure 7.4.7 Progressive refinement of the hyperenergy

near to surface, specifically, within the domain of an A-patch, the external energy with respect to these points is the sum of the squares of the algebraic distances between the points and the surface, where the algebraic distance of a point $\boldsymbol{p}$ is given as the value of the polynomial at $\boldsymbol{p}$.

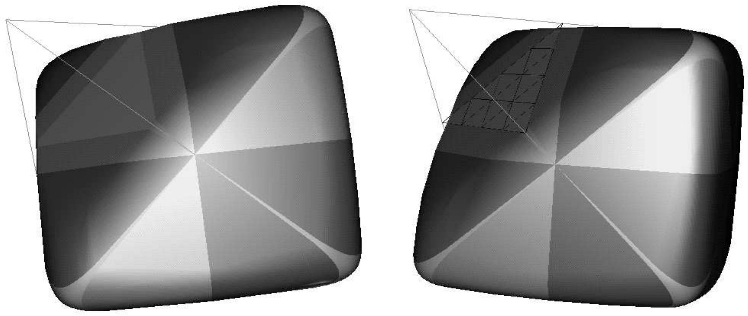
Bajaj, Ihm and Warren [24] use implicit surface to interpolate and least squares approximate given data. They use algebraic distance as simple substitute to measure the distance between a point $\boldsymbol{p}$ and a surface $f(\boldsymbol{x}) = 0$. The algebraic distance between a point $\boldsymbol{p}$ and a surface $f(\boldsymbol{x}) = 0$ is the value of $f(\boldsymbol{p})$ provided that $f(\boldsymbol{x}) = 0$ is normalized. The normalization used in the paper is $\boldsymbol{b}^\mathsf{T}\boldsymbol{b} = 1$, where $\boldsymbol{b}$ is the weights of $f(\boldsymbol{x})$. In the A-patch scheme, as some of nonzero weights are given according to continuity data at the vertices of the triangulation, no normalization is needed.

The external energy is thus formulated as follows

$$E_E = \sum_i \sum_{j=0}^{n} (\boldsymbol{b}^\mathsf{T} \boldsymbol{B}(\boldsymbol{\alpha}_j))^2 = \boldsymbol{b}^\mathsf{T} \mathbf{K}_E \boldsymbol{b}$$

where i is the index of an A-patch and j is the index of an auxiliary point within the domain simplex. Matrix $\mathbf{K}_E$ is called the *external matrix*.

Figure 7.4.8 shows a corner of a rounded cube is attracted to a plane by minimization of the external energy.



(a) (b)

Figure 7.4.8 Chopping a corner: least squares approximation of a plane

7.4.6 The Hyperenergy Model

We have given the formulation of the surface energy, the curve network energy and the external energy.

Let $\mathbf{K} = \mathbf{K}_S + \mathbf{K}_C + \mathbf{K}_E$. The hyperenergy optimization problem is given as

$$\begin{aligned} \text{Minimize} \quad & E = \boldsymbol{b}^{\mathsf{T}}\mathbf{K}\boldsymbol{b} \\ \text{subject to} \quad & \boldsymbol{b}\mathbf{C}(\mathbf{p}) = \mathbf{o} \\ & \boldsymbol{b}\mathbf{S} \geq \mathbf{o} \end{aligned}$$

It can be shown that $\mathbf{K}$ is symmetric and semi-definite. The problem hence can be solved with Lagrangian multipliers and the active set method.

7.5 Energy Optimization Algorithms

Given an energy model formulated as a quadratic programming with inequalities, we can solve a single huge but sparse quadratic programming problem to optimize the surface. However, we favor an iterative solution which locally optimizes the surface separately and repeatedly, until the iteration converges.

We identify three kinds of A-patch grouping to local optimization. Given an interpolant simplicial hull built on a triangulation $\mathcal{T}$,

- a *face group* consists of a pair of face patches for the same triangular face of $\mathcal{T}$, and the edge patches adjacent to them;

- an *edge group* consists of two face groups that share an edge of $\mathcal{T}$;
- a *vertex group* consists of face groups incident to a vertex of $\mathcal{T}$.

Local optimization can be done in different sizes, within a face group, an edge group, or a face group.

For a face group, adjustable parameters include all the weights except the number ⓪ weights are adjustable, along with the ⑧ weights of the adjacent edge patches of the neighboring face group. All, or a subset of this parameters can be chosen to adjust for optimization. We identify 3 modes in a C^1 cubic A-patch optimization session.

- The final four mode: the adjustable parameters are the top four weights, number ④ and ⑤ weights in a face patch, the number ⑥ through ⑨ weights in an edge patch in the group, and the number ⑧ weights in the adjacent edge patch of the group.
- The elite eight mode: besides the adjustable parameter of the final four mode, the number ② and ③ weights are included into the optimization. Now at a face tetrahedron, eight weights are adjustable.
- The sweet seventeen mode: The number ① weights are included.

Modification of these weights may affect the weights of neighboring face group via the continuity constraints. Specifically, a change in an elite eight mode affects the ③ of edge-adjacent face patch; the change in a sweet seventeen affects the ① weights of all the vertex-adjacent face patch. These weights of the non-group patches are included into the adjustable set. The sum of the energy of each patch that has at least one adjustable parameter is the objective function to be minimized.

An edge group, or a vertex group involves more patch into one local optimization. In general, A local optimization involves the weights of a group of patches adjacent to each other. It could be all the patches that cover a feature of a geometric model, for example, the handle of a teapot.

A difficult task, however, for global or local optimization, is to exclude dependent equality constraints. As the simplexes are connected to each other at vertex, edge, and face, some of the continuity constraints become depend on each other. Redundant ones need to be excluded before including to the optimization process. Typically, there is a circular dependency around a vertex of $\mathcal{T}$, e.g. $\boldsymbol{p}_2$ in Figure 3.5.1, or an edge, e.g. $[\boldsymbol{p}_2, \boldsymbol{p}_3]$ of the same figure. One constraint is dropped to remove redundancy. If a Clough-Tocher spit is done to a face patch, or a pair of face patch, there is a circular redundant systems around the splitting vertex, and the splitting edge(s).

If an interpolant simplicial hull is subdivided into groups, one can loop through the list of the group, minimizes the energy of one group at a time, until the solution converges, or a maximum number of loops is reached.

7.6 Distributed Optimization of Elastic A-patch

The optimization algorithms in section 7.5 can be executed in a distributed fashion.

7.6.1 Coloring Approach

We first construct a planar graph G from the topology of the simplicial hull. Each vertex $V \in \mathcal{G}$ represents a group to be optimized and each edge $E \in \mathcal{G}$ represents adjacency between the groups. For a vertex group optimization, $\mathcal{G}_v$ is the triangulation consisting of the bottom face of each face tetrahedron. For a face group optimization, $\mathcal{G}_f$ is the dual graph of $\mathcal{G}_v$, where a vertex in $\mathcal{G}_f$ is a face in $\mathcal{G}_v$ and an edge connects two vertices in $\mathcal{G}_f$ if their counterpart faces in $\mathcal{G}_v$ are adjacent to each other.

After G is constructed, we compute a partition $\mathcal{P}_i$, $(i = 1, 2, ...)$ of the vertices of G where vertices in the same partition $\mathcal{P}_i$ are not adjacent to each other. This is the classical four color theorem.

Now in each iteration, all the vertices in one partition $\mathcal{P}_i$ are distributed to the servers to be optimized. When all the optimization in $\mathcal{P}_i$, we move on to the next partition $\mathcal{P}_{i+1}$ and so on.

7.6.2 Token Approach

The performance can be improved by allowing vertices that are in different partitions but do not share an edge to be optimized at the same time. Each edge is assigned a token. We then have the following rules.

1. A vertex cannot be optimized unless it acquires all the tokens of its incident edges.

2. Tokens are released from the vertex after an optimization.

3. A vertex has to acquire all the tokens needed, simultaneously. Tokens cannot be acquired one at a time.

4. Despite the first time, a vertex cannot be optimized unless some of its neighboring vertices releases tokens.

This scenario is actually the dining philosopher problem where each philosopher needs to grab two chopsticks at the same time in order to dine – (the starvation problem).

7.7 Conclusion

Several elastic models using A-patches have been proposed, each of which has its own advantages and shortcomings. Our preliminary results show that A-patch is a better prototype for free-form design in the sense that it is geometrically richer than the parametric spline counterpart, and furthermore has a simple energy model. Besides the traditional energy model adapted from theory of elasticity, we give several different simplified models that take advantage of the A-patch formulation.

While promising perspectives are pointed out, lots of problems remain unanswered. For example there is no theoretical analysis about how good the models simulate the “real” objects. With the hyperenergy model, the choice of the domain affects the optimal configuration.

A disadvantage of hyperenergy model is due to the interpolant simplicial hull. An elastic A-patch has fixed points that are pinned to the vertices of the simplicial hull. These fixed points in general may not be semantically meaningful in an elastic model, and hence should be change freely. However, modification of the simplicial hull, which can be described by the change of the Jacobian matrix $\mathbf{J}$'s, would introduce high non-linearity into the formulations. Based on this observation, a non-interpolant simplicial hull (see Section 1.2 of Chapter 1), where surface is not pinned to the simplicial hull, may be more suitable for the hyperenergy model.

A final interesting thought is that, whether we can modify the hyperenergy model to be an energy model of a volumetric representation.

8. SUMMARY

8.1 Conclusion

We devise the A-patch technique to model piecewise C^1 and C^2 surfaces of arbitrary topology using implicit polynomial patches. The technique produces implicit surfaces that are free of unwanted branches and singularities. We propose schemes that interpolate or approximate scattered data in three dimensional space. The constraints of a piecewise C^1 cubic A-patch surface consists of linear equalities and linear inequalities over the coefficients of the trivariate polynomials defined in an interpolant simplicial hull. The C^2 quintic scheme involves instead subdivisions and nonlinear inequalities. In both case, the weight (coefficient) setting algorithms enforce the constraints locally.

Based on the functional properties of A-patches, we develop several ray shooting algorithms that quickly triangulate the surface patches. An animating version of the ray shooting algorithms updates the triangulation quickly to capture small changes in the surface configuration. With the fast (re-)triangulation schemes, interactive schemes are proposed to modify the shape of the surface by changing the simplicial mesh or the coefficients of the Bernstein-Bézier polynomials defined in the simplexes.

We propose two kinds of elastic surface models that are built on top of the A-patch representation. A curvature based energy model formulates the surface energy as the integral of the square sum of the two principal curvatures over the zero contouring surface. Nonlinear optimization techniques are required to minimize the energy. The hyperenergy is a simplified energy model we then propose to achieve interactivity. The hyperenergy is formulated as the integral of a weighted square sum of the first and second derivatives of the polynomial over the whole domain. Such an energy function is easy to minimize as it can be formulated as a quadratic programming problem with linear inequality constraints for a C^1 A-patch object.

8.2 Future Work

As discussed in previous chapters, there are much future work to be done in related topics. We mention here several other future directions that are natural extension to this thesis.

8.2.1 Dynamic Elastic A-patches and Applications

Chapter 7 proposes several elastic models for the A-patch representation. But a dynamic A-patch model is yet to be developed to model the movement of an elastic object. A dynamic model can be given in a way similar to the hyperenergy model. Similar to the hyperenergy model in Chapter 7, the Lagrangian formula 2.3.46, by which a dynamic system can be described, may be expressed with respect to the polynomial over the domain simplex, instead of the zero contour set.

8.2.2 Continuous Parameterization of A-paches

In Chapter 6, a "ray shooting" parameterization are given for an A-patches. A 3-sided tetrahedron can be mapped into a 3-sided domain when a 4-sided tetrahedron can be mapped into a 4-sided domain. The parameterization provides an efficient way to triangulate an A-patch surface. However, when A-patches are stitched together to form a free form surface, the ray shooting parameterizations do not meet each other with C^1 or higher continuity. A continuous parameterization is essential for some surface analysis or texturing. A re-parameterization, perhaps based on the ray shooting parameterization, is yet to be developed to provide continuous parameterization across patches.

8.2.3 Intersection of A-patches and Space Curves, Boolean Operations and Space Partitioning

The benefits of using implicit representation include that it is easy to compute intersection between surfaces, easy to perform point location classification against a surface. Hence a natural extension to the A-patch technique is to compute and represent efficiently the intersections between A-patches. A space curve can be given by the the intersections between A-patches. One can further develop the boolean operations between A-patches. Once boolean operations are available, the A-patch representation is ideal to be used as a high order extension of the space partitioning tree [94, 97, 135], which has proved useful in virtual environments and solid modeling. Such an extension would dramatically reduce the complexity of the partitioning tree and hence greatly improve the performance.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] P. Alfeld. A bivariate C^2 Clough-Tocher scheme. *Computer Aided Geometric Design*, 1:257–267, 1984.

[2] P. Alfeld. A trivariate Clough-Tocher scheme for tetrahedral data. *Computer Aided Geometric Design*, 1:169–181, 1984.

[3] V. Anupam and C. Bajaj. Collaborative multimedia scientific design in SHASTRA. In *Proceedings of the First ACM International Conference on Multimedia, ACM MULTIMEDIA 93*, pages 447–456. ACM Press, 1993.

[4] V. Anupam, C. Bajaj, A. Burnett, M. Fields, A. Royappa, and D. Schikore. XS: A hareware independent graphics and windows library. Technical Report CSD-TR-91-062, Department of Computer Sciences, Purdue University, August 1991.

[5] C. Bajaj. Surface fitting with implicit algebraic surface patches. In H. Hagen, editor, *Topics in Surface Modeling*, pages 23–52. SIAM Publications, 1992.

[6] C. Bajaj. The emergence of algebraic curves and surfaces in geometric design. In R. Martin, editor, *Directions in Geometric Computing*, pages 1–29. Information Geometers Press, 1993.

[7] C. Bajaj, F. Bernardini, and G. Xu. Adaptive resconstruction of surfaces and surface-on-surface from dense scattered trivariate data. Technical Report Computer Science Technical Report, CS-95-028, Department of Computer Sciences, Purdue University, 1994.

[8] C. Bajaj, F. Bernardini, and G. Xu. Automatic reconstruction of surfaces and scalar fields from 3D scans. In R. Cook, editor, *Annual Conference Series. Proceedings of SIGGRAPH 95*, pages 109–118, August 6-11 1995.

[9] C. Bajaj and J. Chen. SplineX user's manual and programmer's manual. Manuscript.

[10] C. Bajaj, J. Chen, and S. Evans. Distributed modeling and rendering of splines using Ganith and SplineX. Technical Report CSD-TR-93-006, CAPO Report CER-93-04, Department of Computer Sciences, Purdue University, January 1993.

[11] C. Bajaj, J. Chen, R. Holt, and A. Netravali. Energy formulations of A-splines. Manuscript, 1995.

[12] C. Bajaj, J. Chen, and G. Xu. Free form surface design with A-patches. In *Proceedings of Graphics Interface '94, Banff, Canada*, pages 174–181, 1994.

[13] C. Bajaj, J. Chen, and G. Xu. Smooth low degree approximations of polyhedra. Computer Science Technical Report, CSD-TR-94-002, Department of Computer Sciences, Purdue University, 1994.

[14] C. Bajaj, J. Chen, and G. Xu. Free form modeling with C^2 quintic A-patches. Presented at the Fourth SIAM Conference on Geometric Design, September 1995.

[15] C. Bajaj, J. Chen, and G. Xu. Interactive shape control and rapid display of A-patches. In *Proceedings of The First International Workshop on Implicit Surfaces*, Grenoble, France, April 1995.

[16] C. Bajaj, J. Chen, and G. Xu. Interactive shape control and rapid display of A-patches. Manuscript. The full version of [15], September 1995.

[17] C. Bajaj, J. Chen, and G. Xu. Modeling with cubic A-patches. *ACM Transactions on Graphics*, 14(2), April 1995.

[18] C. Bajaj, J. Chen, and G. Xu. Smooth low degree approximations of polyhedra (part 1). Manuscript, September 1995.

[19] C. Bajaj, J. Chen, and G. Xu. Smooth low degree approximations of polyhedra (part 2). Manuscript, September 1995.

[20] C. Bajaj and S. Evans. Splines and geometric modeling. Technical Report CSD-TR-95-056, Department of Computer Sciences, Purdue University, September 1995.

[21] C. Bajaj and I. Ihm. Algebraic surface design with Hermite interpolation. *ACM Transactions on Graphics*, 11(1):61–91, January 1992.

[22] C. Bajaj and I. Ihm. C^1 smoothing of polyhedra with implicit algebraic splines. *SIGGRAPH'92, Computer Graphics*, 26(2):79–88, July 1992.

[23] C. Bajaj and I. Ihm. Smoothing Polyhedra using Implicit Algebraic Splines. *Computer Graphics (SIGGRAPH '92)*, 26(2):79–88, July 1992.

[24] 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, Oct. 1993.

[25] C. Bajaj and G. Xu. A-splines: Local interpolation and approximation using C^k continuous piecewise real algebraic curves. Technical Report CSD-TR-92-095, CAPO-92-44, Department of Computer Sciences, Purdue University, December 1992.

[26] C. Bajaj and G. Xu. Modeling scattered function data on curved surface. In J. Chen, N. Thalmann, Z. Tang, and D. Thalmann, editors, *Fundamentals of Computer Graphics*, pages 19–29, Beijing, China, 1994.

[27] R. E. Barnhill. Smooth interpolation over triangles. In Barnhill and Riesenfeld, editors, *Computer Aided Geometric Design*, pages 45–70. Academic Press, Boston, Massachusetts, 1974.

[28] R. E. Barnhill. Surfaces in computer-aided geometric design: A survey with new results. *Computer Aided Geometric Design*, 2:1–17, 1985.

[29] R. E. Barnhill, J. H. Brown, and I. M. Klucewicz. A new twist in computer aided geometric design. *Computer Graphics and Image Processing*, 8:78–91, 1978.

[30] R. E. Barnhill, G. Farin, L Fayard, and H Hagen. Twists, curvatures and surface interrogation. *Computer Aided Geometric Design*, 20:341–346, 1988.

[31] R. E. Barnhill, K. Opitz, and H. Pottmann. Fat surfaces: a trivariate approach to triangle-based interpolation on surfaces. *Computer Aided Geometric Design*, 9:365–378, 1992.

[32] A. Barr. Dynamic constraints for modeling, 1986. Siggraph Course Notes #23.

[33] J. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, July 1982.

[34] J. Bloomenthal. Polygonization of implicit surfaces. *Computer Aided Geometric Design*, 5(00):341–355, 1988.

[35] J. Bloomenthal and B. Wyvill. Interactive techniques for implict modeling. In *Proceedings of the 1990 Symposium on Interactive 3D Graphics*, pages 109–116, Snowbird, Utah, 1990.

[36] W. Böhm, G. Farin, and J. Kahmann. A survey of curves and surface methods in cagd. *Computer Aided Geometric Design*, 1(1):1–60, 1984.

[37] P. Borrel and A. Rappoport. Simple constrained deformations for geometric modeling and interactive design. *ACM Transactions on Graphics*, 13(2):137–155, April 1994.

[38] E. Catmull and J. Clark. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10(6):350–355, 1978.

[39] G. Celniker and D. Gossard. Deformable curve and surface finite-elements for free-form shape design. *Computer Graphics (SIGGRAPH '91)*, 25(4):257–266, July 1991.

[40] J. Chadwick, D. Haumann, and R. Parent. Layered construction for deformable animated characters. *Computer Graphics (SIGGRAPH '89)*, 23(3):243–252, July 1989.

[41] G. Chaikin. An algorithm for high-speed curve generation. *Computer Graphics and Image Processing*, 3:346–349, 1974.

[42] H. Chiyokura. *Solid Modeling with Designbase.* Addison Wesley, 1988.

[43] H. Chiyokura and F. Kimura. Design of solids with free form surfaces. *Computer Graphics*, 17(3):289–298, 1983.

[44] R. Clough and J. Tocher. Finite element stiffness matrices for the analysis of plate blending. In *Proceedings of the First Conference on Matrix Methods in structural mechanics*, pages 515–545, Dayton, Ohio, 1965. Wright-Patterson AFB AFFDL TR 66-88.

[45] E. Cobb. *Design of Sculptered Surfaces using the B–spline Representation.* PhD thesis, University of Utah, June 1984.

[46] S. Coquillart. Extended free-form deformation: A sculpturing tool for 3D geometric modeling. *Computer Graphics (SIGGRAPH '90)*, 24(4):187–196, August 1990.

[47] W. Dahmen. Smooth piecewise quadratic surfaces. In T. Lyche and L. Schumaker, editors, *Mathematical Methods in Computer Aided Geometric Design*, pages 181–193. Academic Press, Boston, Massachusetts, 1989.

[48] W. Dahmen and T-M. Thamm-Schaar. Cubicoids: modeling and visualization. *Computer Aided Geometric Design*, 10(2):89–108, Apr. 1993.

[49] D. Doo. A subdivision algorithm for smoothing down irregular shaped polyhedrons. In *Proceedings of Interactive Techniques in Computer Aided Design, Bologna*, pages 157–165, 1978.

[50] D. Doo and M. Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer Aided Design*, 10(6):356–360, November 1978.

[51] H. Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer Verlag, 1987.

[52] G. Farin. Triangular Bernstein-Bézier patches. *Computer Aided Geometric Design*, 3(00):83–127, 1986.

[53] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design: a practical guide*. Academic Press, Boston, Massachusetts, second edition, 1990.

[54] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design: a practical guide*. Academic Press, Boston, Massachusetts, third edition, 1993.

[55] G. Farin and H. Hagen. A local twist estimator. In H. Hagen, editor, *Topics in Surface Modeling*, pages 79–84. SIAM, 1992.

[56] I. D. Faux and M. J. Pratt. *Computational geometry for design and manufacture*. Horwood, Chichester, England; Halsted Press, New York, 1979.

[57] D. Forsey and R. Bartels. Hierarchical B-spline refinement. *Computer Graphics (SIGGRAPH '88)*, 22(4):205–212, August 1988.

[58] D. Forsey and L. Wang. Multi–resolution surface approximation for animation. In *Proceedings of Graphics Interface*, pages 192–199, Toronto, May 1993.

[59] R. Goldman, T. Sederberg, and D. Anderson. Vector elimination: A technique for the implicitization, inversion and intersection of planar parametric rational polynomial curves. *Computer Aided Geometric Design*, 1:327–356, 1984.

[60] J. A. Gregory. Smooth interpolation without twist constraints. In R.E. Barnhill and R. F. Riessenfeld, editors, *Computer Aided Geometric Design*, pages 71–87. Academic Press, 1974.

[61] U. Güdükbay and B. Özgüc. Free form solid modeling using deformation. *Computer Graphics*, 14(3/4):491–500, 1990.

[62] B. Guo. *Modeling Arbitrary Smooth Objects with Algebraic Surfaces*. PhD thesis, Computer Science, Cornell University, 1991.

[63] B. Guo. Surface generation using implicit cubics. In N.M. Patrikalakis, editor, *Scientific Visualizaton of Physical Phenomena*, pages 485–530. Springer-Verlag, Tokyo, 1991.

[64] B. Guo. Non-splitting macro patches for implicit cubic spline surfaces. *Computer Graphics Forum*, 12(3):434 – 445, 1993.

[65] B. Guo. Quadric and cubic bitetrahedral patches. *Visual Computer*, 11:253–262, 1995.

[66] M. Hall and J. Warren. Adaptive polygonalization of implicitly defined surfaces. *IEEE Computer Graphics and Applications*, pages 33–42, 1990.

[67] D. Haumann. Modeling the physical behavior of flexible objects, 1987. Siggraph Course Notes #17.

[68] C. Hoffmann and J. Hopcroft. Quadratic blending surfaces. *Computer Aided Design*, 18(00):301–306, 1986.

[69] C. Hoffmann and J. Hopcroft. The potential method for blending surfaces and corners. In G. Farin, editor, *Geometric Modeling : Algorithms and New Trends*, pages 347–366. SIAM, Philadelphia, 1987.

[70] C. Hoffmann and J. Hopcroft. Geometric ambiguities in boundary representations. *Computer Aided Design*, 19(3):141–147, 1988.

[71] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweizer, and W. Stuetzle. Piecewise smooth surface reconstruction. *Computer Graphics*, 28:295–302, 1994.

[72] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics*, 26(2):71–78, 1992.

[73] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimzation. *Computer Graphics*, 27:19–26, Aug 1-6 1993.

[74] J. Hoschek and D. Lasser. *Computer Aided Geometric Design.* A. K. Peters, Wellesley, 1993. Originally in 1989 by B.G. Teubner, Stuttgart. Translated by L. Schumaker.

[75] E. Jou and W. Han. Elastica and minimal-energy splines. In P. Laurent, A. Le Méhauté, and L. Schumaker, editors, *Curves and Surfaces*, pages 1–4, Boston, Massachusetts, 1990. Academic Press. International Conference on Curves and Surfaces, 1990, Chamonix-Mont-Blanc, France.

[76] E. Jou and W. Han. Minimal-energy splines:i. plane curves with angle constraints. *Mathematical Methods in the Applied Sciences*, 13:351–370, 1990.

[77] K. Joy. Utilizing arametric hyperpatch methods for modeling and display of free form solids. In *ACM Symposium on Solid Modeling*, pages 245–254, New York, 1991.

[78] M.T. Kosters. An extension of the potential method to higher–order blendings. In *Proceedings of ACM Symposium mon Solid Modelinlg and CAD/CAM Applications*, pages 329–337, Austin, Texas, June 1991.

[79] M.T. Kosters. High-order implicit blending surfaces of low degree. *Computer Aided Geometric Design*, 8:495–512, 1991.

[80] L. Holmström. Piecewise quadric blending of implicitly defined surfaces. *Computer Aided Geometric Design*, 4:171–189, 1987.

[81] R. A. Liming. *Practical Analytic Geometry with Applications to Aircraft.* Macmillan, New York, 1944.

[82] S. Lodha. *Surface Approximation with Low Degree Patches with Multiple Representations.* PhD thesis, Computer Science, Rice University, Houston, Texas, 1992.

[83] C. Loop. Smooth subdivision based on triangles. Master's thesis, University of Utah, 1987.

[84] C. Loop. A G^1 triangular spline surface of arbitrary topological type. *Computer Aided Geometric Design*, 11:303–330, 1994.

[85] C. Loop. Smooth Spline Surfaces over Irregular Meshes. In *Proceedings of SIGGRAPH'94*, pages 303–310, July 1994.

[86] D. Luenberger. *Linear and Nonlinear Programming.* Addison Wesley, Reading, Massachusetts, second edition, 1984.

[87] S. Mann, C. Loop, M. Lounsbery, D. Meyer, J. Painter, T. DeRose, and K. Sloan. A survey of parametric scattered data fitting using triangular interpolants. In H. Hagen, editor, *Curve and Surface Modeling*, pages 145–172. SIAM, Philadelphia, Pennsylvania, 1992.

[88] A. Middleditch and K. Sears. Blend surfaces for set theoretic volume modeling systems. *Computer Graphics*, 19(00):161–170, 1985.

[89] D. Moore and J. Warren. Approximation of dense scattered data using algebraic surfaces. In *Proceedings of the twenty-fourth Hawaii International Conference on System Sciences*, volume 1, pages 681–690, Kauai, Hawaii, 1991. IEEE, IEEE Computer Society Press, Los Alamitos, California.

[90] H. Moreton. *Minimum Curvature Variation Curves Networks and Surface for Fair Free Form Shape Design.* PhD thesis, Computer Science Division, University of California, Berkeley, California, 1993.

[91] H. Moreton and C. Séquin. Functional optimization for fair surface design. *Computer Graphics (SIGGRAPH '92)*, 26(2):167–176, July 92.

[92] D. Mumford. Elastica and computer vision. In C. Bajaj, editor, *Geometry Modeling*, pages 492–506. Springer Verlag, New York, 1994.

[93] A. Nasri. Surface interpolation on irregular networks with normal conditions. *Computer Aided Geometric Design*, 8:89–96, 1991.

[94] B. Naylor, J. Amanatides, and W. Thibault. Merging bsp trees yields polyhedral set operations. *Computer Graphics*, 24(4), Aug. 1990.

[95] G. Nielson, T. Foley, B. Hamann, and D. Lane. Visualizing and modeling scattered multivariate data. *IEEE Computer Graphics And Applications*, 11:47–55, 1991.

[96] J.C. Owen and A.P. Rockwood. Blending surfaces in solid modeling. In G. Farin, editor, *Geometric Modeling : Algorithms and New Trends*, pages 367–383. SIAM, Philadelphia, 1987.

[97] M Paterson and F. Yao. Efficient binary space partions for hidden-surface removal and solid modeling. *Discrete and Computational Geometry*, 5(5):485–503, 1990.

[98] N. Patrikalakis and G. Kriezis. Representation of piecewise continuous algebraic surfaces in terms of b-splines. *The Visual Computer*, 5:360–374, 1989.

[99] H. Pedersen. Decorating implicit surfaces. In R. Cook, editor, *Annual Conference Series. Proceedings of SIGGRAPH '95*, August 6-11 1995.

[100] J. Peters. Local cubic and bicubic C^1 surface interpolation with linearly varying boundary normal. *Computer Aided Geometric Design*, 7:499–516, 1990.

[101] J. Peters. Smooth interpolation of a mesh of curves. *Constructive Approximation*, pages 221–246, July 1991.

[102] J. Peters. Smooth free-form surface over irregular meshes generalizing quadratic splines. *Computer Aided Geometric Design*, pages 347–361, October 1993.

[103] J. Peters. Evaluation and approximate evaluation of the multivariate Bernstein-Bézier form on a regularly partitioned simplex. *ACM Transactions on Mathematical Software*, 20(4):p 460–480, December 1994.

[104] J. Peters. C^1 surface splines. *SIAM Journal of Numerical Analysis*, 32(2):645–666, April 1995.

[105] J. Peters. Smooth mesh interpolation with cubic patches. *Computer Aided Design*, 22(2):109–120, March 1990.

[106] L. Piegl. On NURBS: A survey. *IEEE Computer Graphics and Applications*, 11(1):55–71, January 1991.

[107] B. Piper. Visually smooth interpolation with triangular Bézier patches. In G. Farin, editor, *Geometric Modeling: Algorithms and New Trends*, pages 221–234. SIAM, Philadelphia, Pennsylvania, 1987.

[108] J. Platt and A. Barr. Constraint methods for flexible models. *Computer Graphics (SIGGRAPH '88)*, 22(4):279–288, August 1988.

[109] H. Pottmann. Interpolation on surfaces using minimum norm networks. *Computer Aided Geometric Design*, 9(1):51–67, May 1992.

[110] M. Powell and M. Sabin. Piecewise quadratic approximations on triangles. *ACM Transactions on Mathematical Software*, 3:316–325, 1977.

[111] M. J. Pratt. Parametric curves and surfaces as used in computer aided design. *The Mathematics of Surfaces*, pages 1–18, 1986.

[112] V. Pratt. Direct least squares fitting of algebraic surfaces. *Computer Graphics*, 21(3):145–152, 1987.

[113] F. Preparata and M. Shamos. *Computational Geometry, An Introduction.* Springer Verlag, 1985.

[114] H. Qin and D. Terzopoulos. Dynamic NURBS swung surfaces for physics-based shape design. *Computer-Aided Design*, 27(2):111–127, 1999.

[115] U. Reif. A unified approach to subdivision algorithms. Technical report, Mathematisches Institut A, Universität Stuttgart, 1992. Preprint 92-16.

[116] U. Reif. Biquadratic G-spline surfaces. Technical report, Mathematisches Institut A, Universität Stuttgart, 1993. Preprint 93-4.

[117] K. Rescorla. C^1 trivariate polynomial interpolation. *Computer Aided Geometric Design*, 4:237–244, 1987.

[118] R. Sarraga. G^1 Interpolation of generally unrestricted Cubic Bezier Curves. *Computer Aided Geometric Design*, 4(00):23–29, 1987.

[119] R. Schmidt. Eine Methode zur Konstruktion von C^1–Flächen zur Interpolation unregelmässig verteilter Daten. In Schempp and Zeller, editors, *Multivariate Approximation II*, pages 343–361. Birkhäuser, Basel and Boston, 1982.

[120] T. Sederberg. Piecewise algebraic surface patches. *Computer Aided Geometric Design*, 2(1-3):53–59, 1985.

[121] T. Sederberg. Techniques for cubic algebraic surfaces, Tutorial part I. *IEEE Computer Graphics and Applications*, 10(4):14–25, July 1990.

[122] T. Sederberg. Techniques for cubic algebraic surfaces, Tutorial part II. *IEEE Computer Graphics and Applications*, 10(5):12–21, Sept. 1990.

[123] T. Sederberg, D. Anderson, and R. Goldman. Implicit representation of parametric curves and surfaces. *Computer Vision, Graphics and Image processing*, 28(00):72–84, 1985.

[124] T. Sederberg, D. Anderson, and R. Goldman. Implicitization, inversion, and intersection of planar rational cubic curves. *Computer Vision, Graphics and Image Processing*, 31(00):89–112, 1985.

[125] T. Sederberg and F. Chen. Implicitization using moving curves and surfaces. In R. Cook, editor, *Annual Conference Series. Proceedings of SIGGRAPH '95*, August 6-11 1995.

[126] T. Sederberg and S. Parry. Free-form deformation of solid geometric models. *Computer Graphics (SIGGRAPH '86)*, 20(4):151–160, August 1986.

[127] L. Shirman and C. Séquin. Local surface interpolation with Bézier patches. *Computer Aided Geometric Design*, 4:279–296, 1988.

[128] D. Story and A. Ball. Design of an n–sided surface patch from Hermite boundary data. *Computer Aided Geometric Design*, 6:111–120, 1989.

[129] D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. *Computer Graphics (SIGGRAPH '88)*, 22(4):269–278, August 1988.

[130] D. Terzopoulos, J. Platt, and K. Fleischer. Elastically deformable models. *Computer Graphics*, 21(4):205–214, 1987.

[131] D. Terzopoulos and H. Qin. Dynamic NURBS with geometric constraints for interactive scuplting. *ACM Transactions on Graphics*, 13(2):103–136, April 1994.

[132] D. Terzopoulos, A. Witkin, and M. Kass. Symmetry-seeking models and 3D object reconstruction. *International Journal of Computer Vision*, pages 211–221, 1987.

[133] J. Thingvold and E. Cohen. Physical modeling with B-spline surfaces for interactive design and animation. *Computer Graphics (SIGGRAPH '90)*, 24(2):129–137, March 1990.

[134] J. van Wijk. Bicubic patches for approximating non-rectangular control-point meshes. *Computer Aided Geometric Design*, 3:1–13, 1986.

[135] G. Vanecek. Brep-index: a multideimensional space partitioning tree. *International Journal of Computational Geometry and Applications*, 1(3):243–261, 1991.

[136] R. Walker. *Algebraic Curves.* Springer Verlag, New York, 1978.

[137] J. Warren. Blending algebraic surfaces. *ACM Transactions on Graphics*, 8(4):263–278, 1989.

[138] J. Warren. Subdivision methods for geometric design. Manuscript, December 1994.

[139] W. Welch and A. Witkin. Variational surface modeling. *Computer Graphics (SIGGRAPH '92)*, 26(2):157–166, July 1992.

[140] A. Witkin, K. Fleischer, and A. Barr. Energy constraints on parameterized models. *Computer Graphics (SIGGRAPH '87)*, 21(4), July 1987.

[141] A. Witkin and P. Heckbert. Using particles to sample and control implicit surfaces. *SIGGRAPH'94, Computer Graphics*, pages 269–277, July 1994.

[142] A. Worsey and G. Farin. An n-dimensional Clough-Tocher element. *Constructive Approximation*, 3(2):99–110, 1987.

[143] B. Wyvill, C. McPheeters, and G. Wyvill. Animating soft objects. *The Visual Computer*, 2(4):235–242, 1986.

APPENDIX

APPENDIX
SPLINEX: A MODELING SYSTEM

Most of the models and schemes presented in this thesis are implemented in the SplineX system [9]. SplineX is an X-11 and XS [4] based application toolkit that manipulates different piecewise geometric patches, with emphasis on curve and surface in Bernstein-Bezier (BB) form. The geometric entities represented in SplineX include implicit polynomial patches, parametric Bézier curve and surfaces, NURBS, as well as points and polyhedra. As a tool for interactive design, SplineX provides users an environment with various handles to model objects with piecewise these geometric entities.

SplineX can be divided into two main groups of subsystems according to their functionality. The first one, residing in a process, consists of a graphical user interface system, visualization system, object editing system and computation distributed rendering system. The second is a distributed rendering subsystem that performs massive computation tasks, such as polygonization of surface patches. It runs in a number of processes, distributed in a networked environment, each of which serves as a server to the first one.

The SplineX system is controlled by a multi-window graphical user interface system, consisting of Motif menus and space widgets embed in the 3D graphical windows, provides convenient methods to manipulate different subsystems. Figure A.1 shows the graphical user interface of SplineX.

SplineX is a development environment with object-oriented features. There two kind of fundamental classes that represent a piecewise curve or surface: the simplex class and the complex class. A complex object consists of piecewise simplex objects. The simplex class has two built-in derived classes, polynomial and polyhedron.

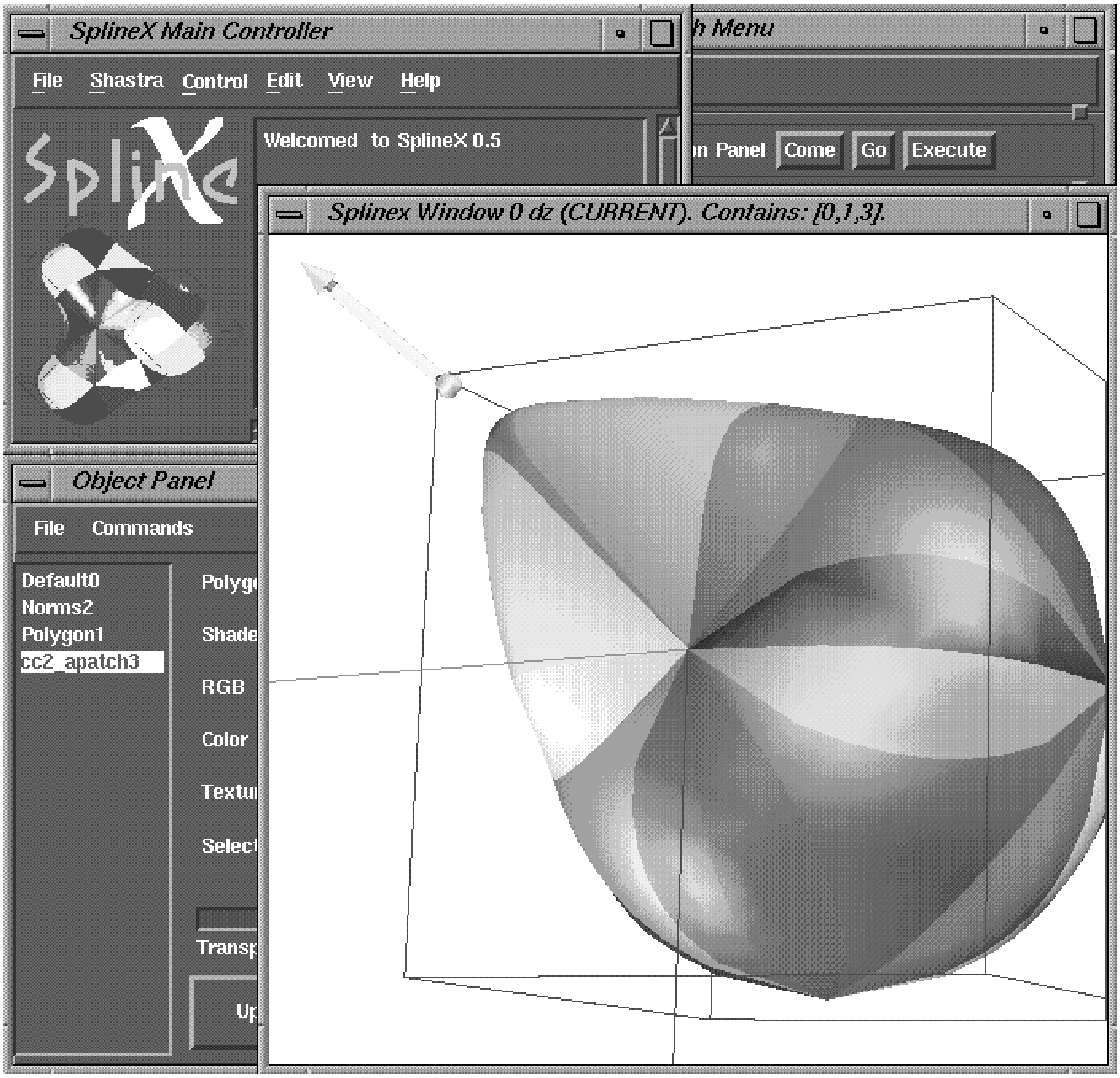


Figure A.1 The graphical user interface of SplineX

The polynomial class has several built-in derived classes: power form polynomial, Bernstein-Bézier polynomial and NURBS. Bernstein-Bézier polynomial is then further subdivide into barycentric, tensor-product and mixed. Then they are further subdivided into polynomial vectors (parametric surface) and implicit polynomials.

A-patch is a derived class of the implicit barycentric polynomial, which receives a good attention in this thesis. It has a derived class of elastic A-patch with the extension of elastic models and corresponding methods. The class of surface on surface (SOS) [26] is another derived class from the class of implicit barycentric polynomial.

SplineX itself is a subsystem is the Shastra system, a distributed collaborative geometric modeling environment. The network connection subsystem connects SplineX with other Shastra subsystems.

VITA

VITA

Jindong Chen (陈劲东) was born on the twenty-fifth of May 1965 in GuangZhou, China. He obtained his senior middle school diploma from the Affiliated Middle School of Southern China Normal University in June 1983. He obtained a bachelor's degree in computer science and engineering from the University of Science and Technology of China in July 1988. He received his master's degree in computer science from the University of Kentucky in August 1990. He joined the Ph.D. program in the Department of Computer Sciences of Purdue University in August 1990 and was awarded the Ph.D. in May 1996. He is a member of ACM and Upsilon Pi Epsilon.

His research interests include computer graphics, geometric modeling, scientific visualization, graphical user interface, networking and distributed systems, solid modeling, computer aided design, and multimedia.