Chapter 4: Geometric Modeling

Chandrajit Bajaj and Andrew Gillette

October 22, 2010

Contents

In geometric design and computer graphics one often uses rational algebraic curves and surfaces because of the advantages obtained from having both the implicit and rational parametric representations [9], [214]. While the rational parametric form of representing a curve allows efficient tracings, ease for transformations and shape control, the implicit form is preferred for testing whether a point is on the given curve, is on the left or right of the curve and is further conducive to the direct application of algebraic techniques. Simpler algorithms are also possible when both representations are available. For example, a straightforward method exists for computing curve - curve and surface - surface intersection approximations when one of the curves, respectively surfaces, is in its implicit form and the other in its parametric form. Global parameterization algorithms exist for implicit algebraic curves of genus zero [2, 1] which allows one to compute this dual representation. A solution to our rational approximation problem yields a rational representation, although approximate, and with all the above advantages for arbitrary genus algebraic plane curves. Perhaps even more important, there are requirements to approximate the algebraic curves in a computer aided geometric design environment. A contour of an algebraic surface, even in its functional form $z = f(x, y)$ is an algebraic curve. The offset of an algebraic curve, even its parametric form, is an algebraic curve either.

1 Smoothing Polygons

Modern applications require the use of curves and surfaces not easily described by a single function, implicitly or parametrically. Hence, pieces of curves and surfaces are stiched together piecewise to

form elaborate shapes and to allow local control of shape parameters. The difficulty in constructing a piecewise function appropriate for modeling is in ensuring that it has as many desirable properties as possible. These properties include low polynomial degree, high continuity degree at join points, close and error-bounded approximation of the control net, and the absence of singularities such as cusps or self-intersections.

Spline curve problems are of two types: approximate and interpolate. The techniques used to solve each kind are slightly different so we state each problem independently in a very general form. Let $\mathbf{p}_0, \ldots, \mathbf{p}_{N-1}$ be a set of N points in the plane i.e. $\mathbf{p}_i = (x_i, y_i) \in \mathbb{R}^2$. We always take indices mod N. Let L denote the **control net** of the $\mathbf{p}_i$, that is, L is the piecewise linear curve formed by connecting $\mathbf{p}_i$ to $\mathbf{p}_{i+1}$ for all i.

Problem 1.1 (Curve Approximation)**.** Find a closed curve C near to L and smoother than L.

Problem 1.2 (Curve Interpolation)**.** Find a closed curve C passing through the $\mathbf{p}_i$ in sequential order and smoother than L.

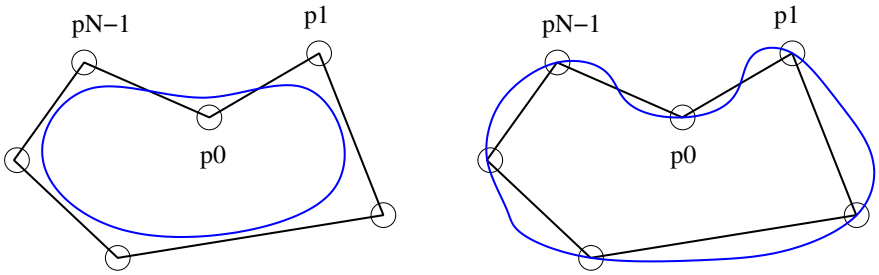


Figure 1: For a set of points $\mathbf{p}_i$, the piecewise linear control net L is shown in black and solutions to curve approximation (Problem 1.1, left) and curve interpolation (Problem 1.2, right) are shown in blue.

We show a simple example of each problem in Figure 1. The problems as stated are significantly underconstrained, for instance, the notions of "near" and "smoother than L" have not been defined. Thus, the problems are open to a wide variety of constrained sub-problems depending on the application context. The distinction between parametric and implicit solutions is easiest to describe in regards to the interpolation problem so we start there.

The parametric solution to Problem 1.2 works as follows. Let $[\mathbf{p}_i, \mathbf{p}_{i+1}]$ denote the linear segment between the two points. Find a set of N functions $f_i : [\mathbf{p}_i, \mathbf{p}_{i+1}] \to \mathbb{R}^2$ such that $f_i(\mathbf{p}_i) = \mathbf{p}_i$ and $f_i(\mathbf{p}_{i+1}) = \mathbf{p}_{i+1}$. This guarantees that the images of the f_i pass through the points p_i and meet with C^0 continuity to form a closed curve C.

To get more than C^0 continuity from a parametric scheme, we can impose the additional constraint that the tangent vector of f_i at $\mathbf{p}_{i+1}$ be a scalar multiple α of the tangent vector of f_{i+1} at $\mathbf{p}_{i+1}$. We call such a curve G^1 **continuous** since the geometry of the curve C will appear to have degree 1 continuity. The term C^1 **continuous** is reserved for the case of $\alpha = 1$, i.e. the parameterizations on adjacent pieces agree not only on the tangent direction of the curve at $\mathbf{p}_{i+1}$ but also its magnitude. This is a more restrictive constraint usually irrelevant to domain and function modeling. We show an example of a portion of a C^1 parametric curve in Figure 2.

The implicit solution to Problem 1.2 requires a set $S_L \subset \mathbb{R}^2$ containing L where the implicit function will be defined. The domain S_L is called the **scaffold** of L and is usually a union of triangles or quadrilaterals attached to L. A typical approach is as follows. Let c_i be a point near $[\mathbf{p}_i, \mathbf{p}_{i+1}]$ and let T_i be the triangle formed with base $[\mathbf{p}_i, \mathbf{p}_{i+1}]$ and distinguished vertex c_i. We discuss details of how to choose such c_i in Section [add ref]. Find a set of N functions $g_i : T_i \to \mathbb{R}$ such that $g_i(\mathbf{p}_i) = g_i(\mathbf{p}_{i+1}) = 0$ and the level set $\{\vec{x} \in \mathbb{R}^2 : g_i(\vec{x}) = 0\}$ is a curve $C_i \subset T_i$ connecting

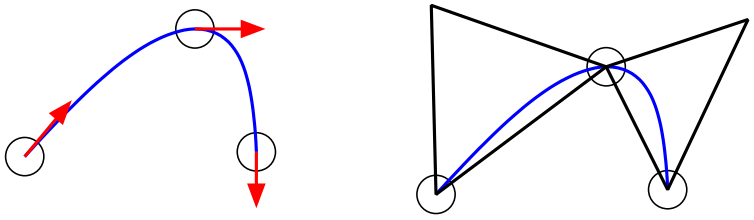


Figure 2: An interpolatory parametric curve (left) is controlled by the tangent vectors at the $\mathbf{p}_i$. An interpolatory implicit curve (right) is controlled by a scaffold S_L of triangles (or other shapes) attached to the control mesh.

$\mathbf{p}_i$ and $\mathbf{p}_{i+1}$. This guarantees that the level set $\{\vec{x} \in \mathbb{R}^2 : \exists j \quad g_j(\vec{x}) = 0\}$ is a closed curve C passing through the points $\mathbf{p}_i$ with C^0 continuity.

To get more than C^0 continuity from an implicit scheme, we can impose the additional constraint that the functions g_i and g_{i+1} have identical power series expansions up to the first k terms at $\mathbf{p}_{i+1}$. If we also use g_i which are at least C^k continuous on T_i, then we produce a level set C which is G^k continuous globally. We show an example of a portion of a C^1 implicit curve and its scaffold S_L in Figure 2.

The advantages and drawbacks of the two approaches to Problem 1.2 are now easy to identify. The parametric schemes allow control of curve shape by modifying the direction of the tangent vectors at the $\mathbf{p}_i$. The construction of a scaffold S_L is not required, although some additional control points are typically used to define the functions f_i. A drawback of G^1 parametric spline solutions to Problem 1.2 is the presence of undesirable topological features such as self-intersections and cusps as well as local behavior deviating far from the linear interpolation. We show some examples in Figure 3. In general, to achieve G^k continuity with parametric curves, the functions f_i must be polynomials of degree $k+1$.

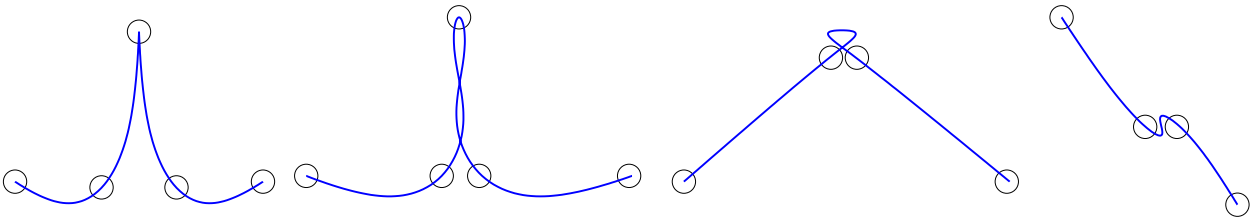


Figure 3: Parametric splines with G^1 continuity may have cusps, self intersections, or undesirable local behavior.

The implicit schemes, on the other hand, can achieve G^k continuity using polynomial functions g_i of degree $\leq \left\lfloor \sqrt{4(k+1)+\frac{9}{4}} - \frac{1}{2} \right\rfloor$. This saves significant computational expense for higher values of k. The most difficult aspect of the implicit scheme is defining g_i so that the zero level set within T_i will be a single connected component and free of singularities. The scaffold S_L aids in the definition as there is a rich theory of splines defined over triangles, quadrilaterals, and other simple polygons which can be leveraged to control the shape and properties of the level sets.

1.1 Smoothing Polygonal Chains

1.1.1 Polygonal Chain Approximation by D_4–Regular Spline Curves

Given an input polygonal chain $\{v_i\}_{i=0}^N$, we use D_4–regular curves to smoothly approximate it, by interpolating the vertices with given first (for G^1 continuity) and the second (for G^2 continuity) order derivatives.

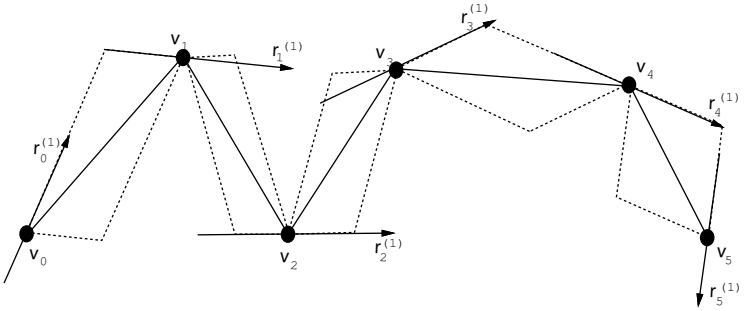


Figure 4: Parallelogram chain.

Step 1. Form a parallelogram chain

For each line segment (edge) of the polygonal chain, construct a parallelogram such that (see Figure 4, where the arrows are tangent vectors): (i) the line segment is one of the diagonals of the parallelogram; (ii) the tangent line of a vertex is contained in the two incident parallelograms.

For a convex edge $[v_{i-1}v_i]$, the corresponding parallelogram can be formed by the four points $p_2,\ v_{i-1},\ p_3, v_i$, where p_2 is the intersection point of the two tangents, $p_3 = v_{i-1} + v_i - p_2$. For a non-convex edge, take one point on each side of the edge such that $p_3 - v_{i-1} = v_i - p_2$. These two points and the endpoints of the edge form the parallelogram.

Assumption 4.1 *For the convex edge $[v_{i-1}v_i]$, the tangent lines $v_{i-1} + sr_{i-1}^{(1)}$ and $v_i + tr_i^{(1)}$ have intersection point at (s^*, t^*) with $s^* > 0$.*

It should be noted that under Assumption 4.1, it is always possible to construct a parallelogram chain, and that this construction is not unique. In the construction of G^1 curves for convex edges, we shall allow p_2 and p_3 to vary along a line (see Figure 5(a) and relation (2) for varying p_2, p_3 that depend on a parameter λ). In other cases, these points are fixed.

Step 2. Construct D_4–Regular Curves

For each parallelogram, construct a D_4–regular curve, such that it interpolates the endpoints of the line segment and has the given first order or second order derivatives. Let $G_{mn}(u, v) = 0$ be the curve defined on $[p_1p_2p_4p_3]$, where p_1 and p_4 are the interpolation points. In the following, we shall determine the minimal m and n, and provide the formulas for computing the coefficients of $G_{mn}(u, v)$ for G^1 and G^2 continuity. These formulas are derived using G^1 and G^2 conditions.

A G^1 Curve Spline Family **A. Convex edge**. Let $[p_1p_4]$ be a convex edge, and $[p_1p_2p_3p_4]$ be the parallelogram. Assume $p_1 = r(a),\ \ p_4 = r(b)$ for some a and b with $a < b$, and assume $\beta_1(a) > \alpha_1(a)$, $\beta_1(b) < \alpha_1(b)$. Take $m = n = 1$.

1. Construction Formulas.

$$b_{00} = b_{11} = 0,\ b_{10} = 1,\ b_{01} = \frac{1-\lambda}{\lambda} \in (-1, 0),\ \lambda > 1, \tag{1}$$

$$p_2 = \lambda p_2' + (1-\lambda)p_3', \quad p_3 = (1-\lambda)p_2' + \lambda p_3', \tag{2}$$

where p_2' is the intersection point of the tangent lines of p_1 and p_4 (see Figure 5(a)), $p_3' = p_1 + p_4 - p_2'$ and λ is a free parameter.

2. Reformulation. Let $p = (p_3' - p_1)s + (p_2' - p_1)t + p_1$. The curve $G_{11}(u, v) = 0$ could be redefined on the smaller parallelogram $[p_1p_2'p_3'p_4]$ as:

$$B_\lambda :\ [4s - (s+t)^2]\lambda^2 - [4s - (s+t)^2]\lambda + s(1-t) = 0. \tag{3}$$

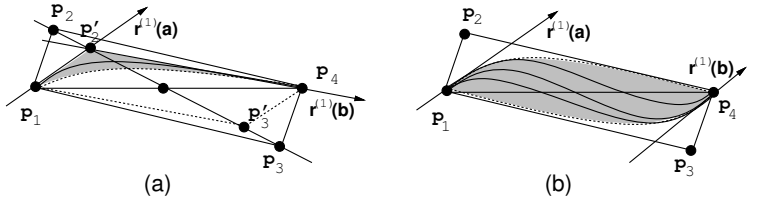


Figure 5: (a). Symmetric parallelogram about the tangent and the curve family for a convex edge. The dotted curve is B_∞. The shaded part is $\mathcal{E}_1$; (b). The curve family for a non-convex edge. The dotted curves are L_0 and L_∞. The shaded part is $\mathcal{E}_2$.

3. Bounding Curves. When $\lambda = 1$, the curve $G_{11}(u, v) = 0$ degenerates to straight lines $s = 0$ (the edge $[p_1 p_2']$) and $t = 1$ (the edge $[p_2' p_4]$), while $\lambda = \infty$, the curve $G_{11}(u, v) = 0$ degenerates to the curve $B_\infty : \ 4s - (s + t)^2 = 0$.

4. Interpolation of an Interior Point. For any given point $p^* = (p_3' - p_1)s^* + (p_2' - p_1)t^* + p_1$ in the interior of the region $\mathcal{E}_1$ enclosed by the curves B_1 and B_∞, there exists a unique $\lambda \in (1, \infty)$, that is

$$\lambda = \frac{1}{2} + \frac{t^* - s^*}{\sqrt{4s^* - (s^* + t^*)^2}}, \tag{4}$$

such that the curve $G_{11}(u, v) = 0$ interpolates the point p^*.

Theorem 1.3. *For a convex edge, there exists a degree (1,1) ($m = n = 1$) D_4–regular curve family $G_{11}(u, v) = 0$, defined by (1)–(2), with a free parameter $\lambda \in (1, \infty)$, in the region $\mathcal{E}_1$ enclosed by the curves B_1 and B_∞. Each curve in the family G^1 interpolates the endpoints of the edge. For any given point p in the interior of $\mathcal{E}_1$, there exists a unique curve, defined by (1)–(2) and (4), in this family that interpolates the point p.*

Note that the curve B_λ defined by (3) on $[p_1 p_2' p_3' p_4]$ is not in the form G_{11}. However, if we transform it into barycentric form on the triangle $[p_1 p_2' p_4]$, then we can show that the curve is D_1–regular on the triangle.

It is obvious that for fixed p_2 and p_3 that satisfy (2), there exists a unique curve $G_{11}(u, v) = 0$ that G^1 interpolates the edge.

Parameterization. From $G_{11}(u, v) = 0$, we obtain the parameterized expression $v = \frac{u}{u - b_{01}(1-u)}, \ u \in [0, 1]$.

B. Non-convex edge . We assume $\beta_1(a) \geq \alpha_1(a)$, $\beta_1(b) \geq \alpha_1(b)$. Take $m = 1, n = 2$. If $\beta_1(a) \leq \alpha_1(a)$, $\beta_1(b) \leq \alpha_1(b)$, take $m = 2, n = 1$.

1. Construction Formulas.

$$b_{00} = b_{12} = 0, \quad b_{10} = 1, \tag{5}$$

$$b_{01} = -\frac{1}{2}\delta \leq 0, \quad b_{11} = -\frac{1}{2}\gamma b_{02} > 0, \tag{6}$$

where $\delta = \frac{\alpha_1(a)}{\beta_1(a)}$, $\gamma = \frac{\alpha_1(b)}{\beta_1(b)}$ and $b_{02} < 0$ is a signed free parameter (see Figure 5(b) for the curve family).

2. Bounding Curves.

$$\begin{aligned} L_0 \quad &: u(1 - v) - \delta(1 - u)v = 0, \\ L_{-\infty} &: \ (1 - u)v - \gamma u(1 - v) = 0. \end{aligned}$$

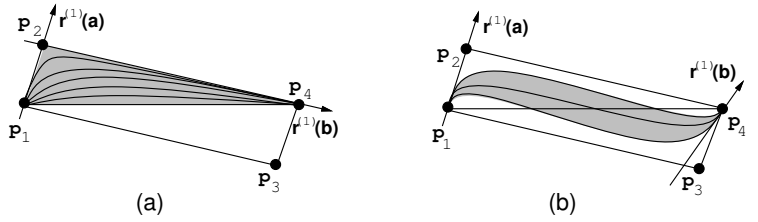


Figure 6: (a). G^2 curve family for a convex edge. The shaded part is $\mathcal{E}_3$; (b). G^2 curve family for a non-convex edge. The shaded part is $\mathcal{E}_4$

3. Interpolation of an Interior Point. For any given point $p = (u, v)^T$ in the interior of the region $\mathcal{E}_2$ enclosed by L_0 and $L_{-\infty}$, take

$$b_{02} = -\frac{(1-v)[u(1-v) - \delta(1-u)v]}{v[(1-u)v - \gamma v(1-v)]}, \tag{7}$$

then the curve determined by b_{02} interpolates the point p.

Theorem 1.4. *For a non-convex edge, there exists a degree (1,2) (or (2,1)) D_4–regular curve family, defined by (5)–(6) with a free parameter $b_{02} \in (0, -\infty)$, in the region $\mathcal{E}_2$ enclosed by L_0 and $L_{-\infty}$, whose members G^1 interpolate the endpoints of the edge. For any given point p in $\mathcal{E}_2$, there exists a unique curve, defined by (5)-(7), in this family that interpolates the point p.*

Parameterization. Since $m = 1, n = 2$, the curve can be expressed in rational parameterized form

$$u = -\frac{b_{01}B_1^2(v) + b_{02}B_2^2(v)}{B_0^2(v) + (b_{11} - b_{01})B_1^2(v) - b_{02}B_2^2(v)}, \quad v \in [0, 1].$$

Shape Control Handles. For the given polygonal chain, the shape control handles are: (i) the direction of tangent vector at each vertex; (ii) an interpolating point p in the region $\mathcal{E}_1$, for convex edges, or $\mathcal{E}_2$, for non-convex edges.

A G^2 Curve Spline Family **A. Convex edge**. Let $[p_1p_4]$ be a convex edge and $[p_1p_2p_3p_4]$ be the parallelogram. Again, we assume $\beta_1(a) > \alpha_1(a)$, $\beta_1(b) < \alpha_1(b)$. Furthermore, we assume that the the parallelogram is constructed so that $\alpha_1(a) = \beta_1(b) = 0$. Now we need to take $m = n = 2$.

1. Construction Formulas.

$$b_{00} = b_{01} = b_{12} = b_{22} = 0, \quad b_{02} = -1, \tag{8}$$

$$b_{10} = \frac{\beta_1(a)^2}{\alpha_2(a)} > 0, \quad b_{21} = -\frac{\alpha_1(b)^2}{\beta_2(b)} > 0, \tag{9}$$

$$4b_{11} = 2b_{10} + 2b_{21} + 1 - b_{20}, \tag{10}$$

where b_{20} is a free parameter (see Figure 6(a) for the curve family).

2. Interpolation of an Interior Point. Parameter b_{20} can be used to interpolate one point $(u, v)^T$ in the interior of the parallelogram with $u < v$. By $G_{22}(u, v) = 0$, we have

$$b_{20} = \frac{B_0^2(u)B_2^2(v) - b_{10}B_1^2(u)B_0^2(v) - [b_{21}B_2^2(u) + b_{11}B_1^2(u)]B_1^2(v)}{B_2^2(u)B_0^2(v)}. \tag{11}$$

3. Reformulation. Let $\alpha_1 = 1 - v$, $\alpha_2 = v - u$, $\alpha_3 = u$. Represent $G_{22}(u, v)$ in the barycentric coordinate form $\tilde{G}_{22}(\alpha_1, \alpha_2, \alpha_3)$ over the triangle $[p_1p_2p_4]$:

$$\tilde{G}_{22}(\alpha_1, \alpha_2, \alpha_3) := \sum_{i+j+k=3} a_{ijk}B_{ijk}^3(\alpha_1, \alpha_2, \alpha_3) \tag{12}$$

with

$$a_{300} = a_{210} = a_{003} = a_{012} = 0, a_{111} = \frac{2b_{10} + 2b_{21} + b_{02} - b_{20}}{6}, \tag{13}$$

$$a_{201} = \frac{2}{3}b_{10},\ a_{102} = \frac{2}{3}b_{21}, a_{120} = a_{021} = \frac{1}{3}b_{02},\ a_{030} = b_{02}. \tag{14}$$

Theorem 1.5. *For a convex edge, say $[p_1p_4]$, there exists a degree (2,2) convex curve family in the triangle $\mathcal{E}_3 = [p_1p_2p_4]$, defined by (8)–(10), with b_{20} as a free parameter. Each member in the family G^2 interpolates the endpoints of the edge. If $b_{20} > 0$, the curve is D_4–regular in the parallelogram $[p_1p_2p_4p_3]$. If $b_{20} \le 0$, the curve, that is re-defined by (12-(14), is D_1–regular on the triangle $[p_1p_2p_4]$. For any given point p in the interior of $\mathcal{E}_3$, there exists a unique curve, defined by (8)–(11), in this family that interpolates the point p.*

B. Non-convex edge. Assume $\beta_1(a) \ge \alpha_1(a)$, $\beta_1(b) \ge \alpha_1(b)$ and the parallelogram is constructed so that $\alpha_1(a) = 0$ or $\alpha_1(b) = 0$. That is, at least one of the tangent lines at p_1 and p_4 coincides with one of the edges of the parallelogram (see Figure 6(b)). Again, we take $m = n = 2$.

1. Construction Formulas.

$$b_{00} = b_{22} = 0, \quad b_{01} = -\delta b_{10}, \quad b_{21} = -\gamma b_{12}, \tag{15}$$

$$4b_{11} = 2(b_{12} + b_{01} + b_{10} + b_{21}) - (b_{02} + b_{20}), \tag{16}$$

$$\begin{aligned} b_{10} = \frac{1}{\Delta}\Big\{&\alpha_1(a)\left[\beta_1(a) - \alpha_1(a)\right]\left[\gamma\beta_2(b) - \alpha_2(b)\right] \\ &+ 2\alpha_1(a)\beta_1(a)\left[\beta_1(b) - \alpha_1(b)\right]^2\Big\}b_{20} \\ &-\frac{1}{\Delta}\Big\{\beta_1(a)\left[\beta_1(a) - \alpha_1(a)\right]\left[\gamma\beta_2(b) - \alpha_2(b)\right]\Big\}b_{02}, \end{aligned} \tag{17}$$

$$\begin{aligned} b_{12} = \frac{1}{\Delta}\Big\{&\alpha_1(b)[\beta_1(b) - \alpha_1(b)]\left[\alpha_2(a) - \delta\beta_2(a)\right] \\ &+ 2\alpha_1(b)\beta_1(b)\left[\beta_1(a) - \alpha_1(a)\right]^2\Big\}b_{02} \\ &-\frac{1}{\Delta}\Big\{\beta_1(b)[\beta_1(b) - \alpha_1(b)]\left[\alpha_2(a) - \delta\beta_2(a)\right]\Big\}b_{20}, \end{aligned} \tag{18}$$

where $\delta = \frac{\alpha_1(a)}{\beta_1(a)}$, $\gamma = \frac{\alpha_1(b)}{\beta_1(b)}$, $\Delta = [\alpha_2(a) - \delta\beta_2(a)][\gamma\beta_2(b) - \alpha_2(b)]$, $b_{02} = -1$ and $b_{20} > 0$ is a free parameter (see Figure 6(b) for the curve family).

2. Bounding Curves. The bounding curves of the curve family are defined by taking $b_{20} = 0$ and $b_{20} = \infty$. Let $G_{22}(u, v, b_{02}, b_{20})$ be defined by (15)–(18). Then $G_{22}(u, v, b_{02}, 0) = b_{02}G_{22}(u, v, 1, 0)$, $G_{22}(u, v, 0, b_{20}) = b_{20}G_{22}(u, v, 0, 1)$. Hence the bounding curves are $G_{22}(u, v, 1, 0) = 0$, $G_{22}(u, v, 0, 1) = 0$.

Theorem 1.6. *For a non-convex edge, we have a one parameter D_4–regular curve family $\{b_{20}G_{22}(u, v, 0, 1) - G_{22}(u, v, 1, 0) = 0 : b_{20} > 0\}$ whose members G^2 interpolate the edge and have only one inflection point. For any given point $p = (u^*, v^*)^T$ in the interior of the region $\mathcal{E}_4$ enclosed by the curves $G_{22}(u, v, 0, 1) = 0$ and $G_{22}(u, v, 1, 0) = 0$ in the parallelogram, there exists a unique curve in the family with $b_{20} = G_{22}(u^*, v^*, 1, 0)/G_{22}(u^*, v^*, 0, 1)$ that interpolates the point p.*

Curve Evaluation and Display. Since $G_{22}(u, v)$ could be expressed as $\sum_{i=0}^{2} B_i(v)B_i^2(u)$ with $B_0(v) < 0, B_2(v) > 0$ on $(0, 1)$, the curve $G_{22}(u, v) = 0$ can be evaluated for each v in $(0, 1)$ by finding the zeros of a quadratic polynomial, here $B_i(v) = \sum_{j=0}^{2} b_{ij}B_j^2(v)$. For the case of a convex edge, it is possible that the quadratic has two zeros in $(0, 1)$, and the correct one is such that $u < v$. For the non-convex edge, the quadratic has exactly one zero in $(0, 1)$.

For intensive evaluation of the curve, the quarterly subdivision process for $G_{22}(u, v)$ on the rectangle $[0, 1] \times [0, 1]$ could be used (see [187]) while discarding those sub-rectangles on which the

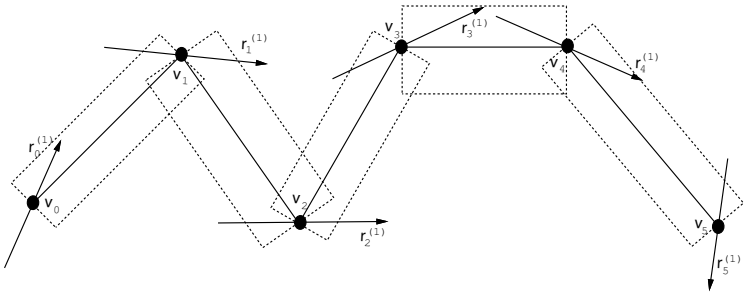


Figure 7: Rectangular chain. The width of the rectangle for edge $[v_{i-1}v_i]$ is $2\epsilon_i$.

subdivision polynomials have only positive or negative coefficients. On each sub-rectangle, a bilinear function, that interpolates function values on the four vertices, is used to evaluate the curve intersection points. It follows from [84] that such a subdivision will have quadratic convergence. For example, ten steps of subdivision will reduce the distance between the polynomial and the BB net to become $(1/2^{10})^2 \approx 10^{-6}$ times the initial distance. By keeping a tree data structure, we achieve a progressive display scheme for our curve splines.

Shape Control Handles. For the given polygonal chain, the shape control handles of the curve are: (i) the direction of tangent vector at each vertex; (ii) the magnitude of the second order derivative vector (related to curvature) at each vertex; (iii) an interpolating point in the region $\mathcal{E}_3$ for convex edges, or $\mathcal{E}_4$ for non-convex edges.

1.1.2 Polygonal Chain Approximation by D_3–Regular Curves

We shall use D_3–regular curve to smoothly approximate the polygon by interpolating the vertices together with the given tangents at the vertices. We could also interpolate second order derivatives at the polygon vertices to achieve G^2 continuity. Here we only detail G^1 continuity. The G^2 construction is very similar. The construction consists of the following two steps:

Step 1. Form a Rectangular Chain

For each line segment (edge) $[v_{i-1}v_i]$ of the polygonal chain, construct a rectangle such that (see Figure 7, where the arrows are tangent vectors) the line segment is in the middle of the rectangle. That is, two edges are parallel to the line segment at an equal distance ϵ_i from it, and the other two edges are orthogonal to the line segment and pass through the endpoints of the line segment. Since the determined curve shall lie within the rectangle, ϵ_i serves as a natural error controller of the approximation. The effect of ϵ_i will be discussed further in section 1.1.2.

Assumption 5.1 *For each edge* $[v_{i-1}v_i]$, $(v_i - v_{i-1})^T r_{i-1}^{(1)} > 0$, $(v_i - v_{i-1})^T r_i^{(1)} > 0$.

Step 2. Construct the D_3-regular Curves

For each rectangle, construct a D_3–regular curve, such that it interpolates the endpoints of the line segment and has given first order derivatives. Let $[p_1p_2p_3p_4]$ be a given rectangle, $v_0 = (p_1 + p_2)/2$, $v_1 = (p_3 + p_4)/2$ be the interpolation points and $r_0^{(1)}, r_1^{(1)}$ be the tangent vectors.

A G^1 Curve Spline Family **A. Convex edge**. Suppose $[v_0v_1]$ be a convex edge. From Assumption 5.1, we have $\alpha_1(a) > 0$, $\alpha_1(b) > 0$. Now assume $\beta_1(a) > 0$, $\beta_1(b) < 0$ (the case $\beta_1(a) < 0$, $\beta_1(b) > 0$ is similar) and take $m = 2, n = 1$.

1. Construction Formulas.

$$b_{00} = 1, \quad b_{21} = -b_{20}, \quad b_{01} = -1, \tag{19}$$

$$b_{10} + b_{11} = 2\alpha = -2\beta b_{20}, \quad b_{20} = -\alpha\beta^{-1} > 0, \tag{20}$$

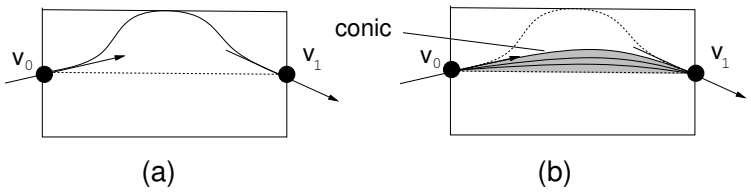


Figure 8: (a). Non-convex curve; (b). Convex curves (the solid curves). The shaded part is $\mathcal{E}_5$

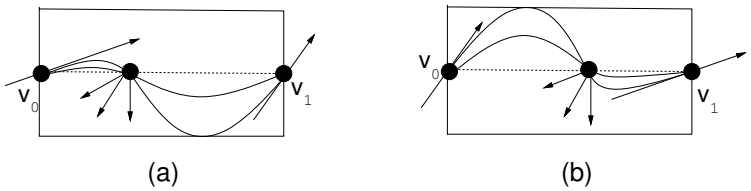


Figure 9: (a) The case $\alpha \le \beta$; (b) The case $\alpha > \beta$.

where $\alpha = \frac{\beta_1(a)}{\alpha_1(a)}$, $\beta = \frac{\beta_1(b)}{\alpha_1(b)}$, b_{11} as a free parameter (see Figure 8(b) for the curve family).
2. Limitations on Free Parameters. To make the curves D_3–regular and convex, we enforce

$$b_{11} < b_{11}^* := \min\left\{\sqrt{-\alpha\beta^{-1}}, \quad -\frac{1}{2} + \alpha\left[1+\beta^{-1}\right]\right\}. \tag{21}$$

Theorem 1.7. *For a convex edge, let $G_{21}(u,v,b_{11})$ be defined by (19)–(20), then we have a convex D_3–regular curve family $\{G_{21}(u,v,b_{11}) = 0 : \ b_{11} < b_{11}^*\}$, whose members G^1 interpolate the endpoints of the edge. For any given point $p = (u^*,v^*)^T$ in the region $\mathcal{E}_5$ enclosed by the curve $G_{21}(u,v,b_{11}^*) = 0$ and the line $v = \frac{1}{2}$ there exists a unique b_{11} satisfying $G_{21}(u^*,v^*,b_{11}) = 0$ such that the curve $G_{21}(u,v,b_{11}) = 0$ interpolates the point p.*

B. Non-convex edge. Assume $\beta_1(a) \ge 0$, $\beta_1(b) \ge 0$. Take $m = 3, n = 1$.
1. Construction Formulas.

$$b_{00} = b_{30} = 1, \quad b_{01} = b_{31} = -1, \tag{22}$$

$$b_{10} + b_{11} = \frac{4}{3}\alpha, \quad b_{20} + b_{21} = -\frac{4}{3}\beta, \tag{23}$$

$$b_{11} + b_{20} = b_{10} + b_{21}, \tag{24}$$

$$b_{10} = b_{20} + \frac{2}{3}(\alpha+\beta), \quad b_{21} = b_{11} - \frac{2}{3}(\alpha+\beta), \tag{25}$$

where $\alpha = \frac{\beta_1(a)}{\alpha_1(a)}$, $\beta = \frac{\beta_1(b)}{\alpha_1(b)}$, b_{20} or b_{11} is a free parameter (see Figure 9 for the curve family).

2. Limitations on Free Parameters. To ensure the curves are D_3–regular and have only one inflection point, we require

$$b_{20} > \max\left\{b_{20}^*, \frac{\alpha-\beta-2\alpha\beta}{3\alpha}, \frac{\beta-\alpha-2\beta^2}{3\beta}\right\} \quad \text{when} \quad \alpha \le \beta, \tag{26}$$

$$b_{11} < \min\left\{b_{11}^*, \frac{\beta-\alpha+2\alpha^2}{3\alpha}, \frac{\alpha-\beta+2\alpha\beta}{3\beta}\right\} \quad \text{when} \quad \alpha > \beta, \tag{27}$$

where b_{20}^* is the largest negative root of $h(b_{20}) = 0$, b_{11}^* is the smallest positive root of $g(b_{11}) = 0$ with

$$h(b_{20}) := 1 + 4b_{10}^3 + 4b_{20}^3 - 3b_{10}^2 b_{20}^2 - 6b_{10}b_{20},$$
$$g(b_{11}) := 1 - 4b_{11}^3 - 4b_{21}^3 - 3b_{11}^2 b_{21}^2 - 6b_{11}b_{21}.$$

3. Interpolation to a Normal. It should be noted that all the curves pass through the same point $(u^*, \frac{1}{2})^T$ with $u^* = \frac{\alpha}{\alpha+\beta}$ (see Figure 9). Since

$$\nabla G_{31}(u^*, \frac{1}{2}) = \left[-\frac{2\alpha\beta}{\alpha+\beta}, -\frac{2(\alpha^3+\beta^3)}{(\alpha+\beta)^3} - \frac{(6b_{20}+4\beta)\alpha\beta}{(\alpha+\beta)^2} \right]^T,$$

by assigning a normal at $(u^*, \frac{1}{2})^T$, the unique b_{20} is determined.

Theorem 1.8. *For a non-convex edge, there exists a D_3–regular curve family $\{G_{31}\ (u, v) = 0\}$ that has the following properties: (i). Each curve in the family G^1 interpolates the edge. (ii). Each curve passes through the point $(u^*, \frac{1}{2})^T$. (iii). There is only one curve in that family that has the given normal at $(u^*, \frac{1}{2})^T$. (iv). The curve $v = \frac{1}{2}$ and the curve given by $b_{20} = b^*_{20}$ (if $\alpha \le \beta$) or $b_{11} = b^*_{11}$ (if $\alpha > \beta$) are the two limit curves of the family.*

Parameterization. Since the curve is defined by $G_{m1}(u, v) = \sum_{i=0}^m b_{i0} B_i^m(u) + v \sum_{i=0}^m (b_{i0} - b_{i1}) B_i^m(u) = 0$, it follows that

$$p = (p_3 - p_1)u - (p_2 - p_1)\frac{\sum_{i=0}^m b_{i0} B_i^m(u)}{\sum_{i=0}^m (b_{i0} - b_{i1}) B_i^m(u)} + p_1, \quad u \in [0, 1].$$

Shape Control Handles. For the given polygonal chain, the shape control handles of the curve are: (i) the direction of the tangent vector at each vertex; (ii) an interpolating point in the region $\mathcal{E}_5$, for convex edges, or a normal at $(u^*, \frac{1}{2})^T$, for non-convex edges.

The Effect of the Size of Rectangle In the construction of rectangles in Step 1 at the beginning of this section, the widths of the rectangles, namely $2\epsilon_i$, are arbitrarily chosen. One may ask: what is the effect of this ϵ_i on the constructed curves for a given edge $[v_{i-1}v_i]$? The conclusion is the following: *The curve family for smaller ϵ_i is a subset of the curve family for larger ϵ_i*, for each of the two cases discussed in section 1.1.2. That is, ϵ_i will not change the shape of the curves but changes the "number" of curves in the family. When $\epsilon_i > 0$ becomes successively smaller, more and more curves are expelled from the curve family, and the remaining curves (still infinitely many) are successively close to edge (see Figure 11(c) and (d)). To prove this conclusion, suppose ϵ_i is magnified by a factor $\theta > 1$, and suppose the notation on the enlarged rectangle is the same as the original one but with an added prime. It is then easy to see that

$$\alpha_1'(l) = \alpha_1(l), \quad \beta_1'(l) = \theta^{-1}\beta_1(l), \quad u = u', \quad v = \theta(v' - \frac{1}{2}) + \frac{1}{2}.$$

Hence

$$B_0^1(v) = \frac{1}{2}(1+\theta)B_0^1(v') + \frac{1}{2}(1-\theta)B_1^1(v'),$$
$$B_1^1(v) = \frac{1}{2}(1-\theta)B_0^1(v') + \frac{1}{2}(1+\theta)B_1^1(v').$$

Substituting these into $G_{m1}(u, v)$, we have

$$G_{m1}(u, v) = \theta G'_{m1}(u', v') = \theta \sum_{i=0}^m \sum_{j=0}^1 b'_{ij} B_i^m(u') B_j^1(v')$$

with

$$b'_{i0} = \frac{(1+\theta)b_{i0} + (1-\theta)b_{i1}}{2\theta}, \quad b'_{i1} = \frac{(1-\theta)b_{i0} + (1+\theta)b_{i1}}{2\theta}.$$

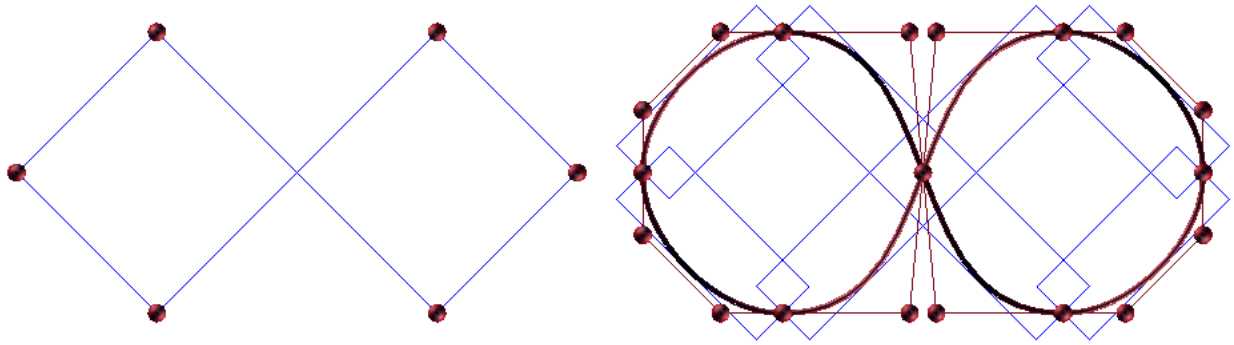
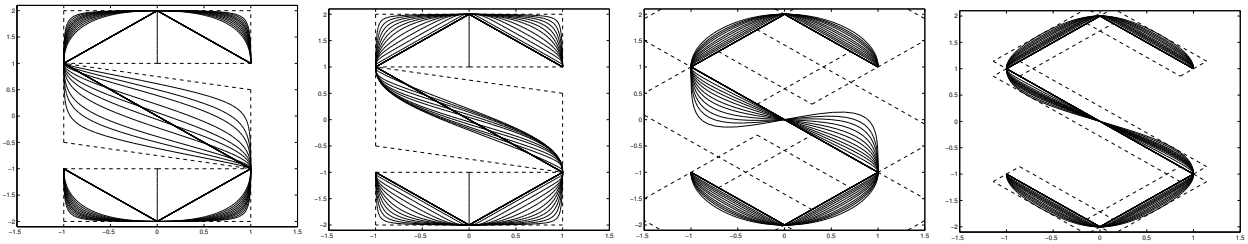


Figure 10: The left figure shows the input polygon. The right shows the G^1 D_4-regular curves and Bézier points interpolating the vertices of the polygon within prescribed bounds.



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

Figure 11: (a) G^1 families on parallelograms. (b) G^2 families on parallelograms. (c) G^1 families on rectangles with $\epsilon_i = 1.0$. (d) G^1 families on rectangles with $\epsilon_i = 0.2$.

Using these relations, we verify that b'_{ij} satisfies all the relations as b_{ij} does. Therefore, curve $G_{m1}(u, v) = 0$ defined on the smaller rectangle is in the curve family defined on the larger rectangle. Note that this statement holds for both the cases of the convex edge and the non-convex edge discussed in section 1.1.2.

Note. In the six spline families we discuss in sections 1.1.1 and 1.1.2, there are four cases with $\min\{m, n\} = 1$. In these cases, rational parametric expressions are easily derived. Hence, for these cases, we have both the implicit form and the parametric form. For example, the G^1 D_3-regular curve could be transformed into parametric rational Bézier curve of degree 4.

The right figure of Figure 10 shows the Bézier points of G^1 D_3-regular curve as well as the rectangle chain for the input polygonal chain (right figure). It is clear that the rectangles enclose more tightly the curve than the convex hull of the Bézier points. Furthermore, the shape of curve is easier to control using its implicit form than using its parametric form, since the implicit form has one free parameter while the rational Bézier of degree 4 has many more degrees of freedom. Also, the parameter change of the rational Bézier form may lead the curve out of the G^1 D_3-regular curve family.

2 Smoothing Polyhedra

2.1 C^1 Continuity and Compatibility Conditions

An *algebraic surface* in $\mathbb{R}^3$ is implicitly defined by a single polynomial equation $f(x, y, z) = \sum c_{ijk} x^i y^j z^k = 0$, where the coefficients c_{ijk} of f are real numbers. The *normal* or *gradient* of

$f(x,y,z)=0$ is the vector $\nabla f=(f_x,f_y,f_z)$. A point $\mathbf{p}=(x_0,y_0,z_0)$ on a surface is a *regular* point if the gradient at $\mathbf{p}$ is not null; otherwise the point is *singular*. An algebraic surface $f(x,y,z)=0$ is *irreducible* if $f(x,y,z)$ does not factor over the field of complex numbers.

An *algebraic space curve* is defined by the common intersection of two or more algebraic surfaces. In geometric design we restrict our consideration to space curve segments and assume they are contained in the intersection of exactly two algebraic surfaces. A *rational parametric space curve* is represented by the triple $\mathbf{G}(s)=(x=G_1(s),y=G_2(s),z=G_3(s))$, where G_1, G_2 and G_3 are rational functions in s. We assume that the curve is only singly defined under the parameterization map, i.e., each triple of values for (x, y, z), corresponds to a single value of s.

The *degree* of an algebraic surface is the number of intersections between the surface and a line, counting complex, infinite and multiple intersections. This degree is also the same as the degree of the defining polynomial. The *degree* of an algebraic space curve is the number of intersections between the curve and a plane, counting complex, infinite and multiple intersections. The degree of an algebraic curve segment given as the intersection curve of two algebraic surfaces is also no larger than the product of the degrees of the two surfaces. Furthermore, the degree of a rational algebraic curve is the same as the maximum degree of the numerator and denominator polynomials in the defining triple of rational functions.

2.1.1 Necessary and Sufficiency Conditions

The followings are definitions and lemmas pertinent to the algorithm for C^1 smoothing of polyhedra:

Definition 2.1. *Let $\boldsymbol{p}=(a,b,c)$ be a point with an associated "normal" $\mathbf{m}=(m_x,m_y,m_z)$ in $\mathbb{R}^3$. An algebraic surface $S: f(x,y,z)=0$ is said to contain $\boldsymbol{p}$ with C^1 continuity if*
(1) $f(\boldsymbol{p})=f(a,b,c)=0$, **(containment condition)**
and
(2) $\nabla f(\mathbf{p})$ *is not zero and* $\nabla f(\mathbf{p})=\alpha\mathbf{m}$, *for some nonzero* α. **(tangency condition)**

Definition 2.2. *Let C be an algebraic space curve with an associated varying "normal" $\mathbf{n}(x,y,z)=(n_x(x,y,z),n_y(x,y,z),n_z(x,y,z))$, defined for all points on C. An algebraic surface $S: f(x,y,z)=0$ is said to contain C with C^1 continuity if*
(1) $f(\boldsymbol{p})=0$ *for all points $\boldsymbol{p}$ of C.* **(containment condition)**
and
(2) $\nabla f(\mathbf{p})$ *is not identically zero and* $\nabla f(\mathbf{p})=\alpha\mathbf{n}(\boldsymbol{p})$, *for some α and for all points $\boldsymbol{p}$ of C.* **(tangency condition)**

Lemma 2.3. *A necessary condition for smoothing a polyhedron with tangent-plane-continuous triangular surface patches is a unique tangent plane at each vertex of the polyhedron.*

Towards sufficiency conditions of C^1 smoothing of polyhedra .....

2.1.2 Compatibility and Non-Singularity Constraints

We first review some basic concepts from differential geometry [96, 180]. A surface $S\subset R^3$ is *regular at a point* $p\subset S$ if there exists a neighborhood $V\subset R^3$ and a map $\mathbf{x}: U\longrightarrow V\cap S$ of an open set U in R^2 onto $V\cap S\subset R^3$ such that $\mathbf{x}(u,v)=(x(u,v),y(u,v),z(u,v))$ is differentiable, homeomorphic, and its differential $d\mathbf{x}_q: R^2\longrightarrow R^3$ is one-to-one for each $q\in U$. A surface S is *regular* if, at each point on S, S is regular. A *tangent vector* to a regular surface S at a point $p\in S$ is the tangent vector $\alpha'(0)$ of a differentiable curve $\alpha:(-\epsilon,\epsilon)\longrightarrow S$ with $\alpha(0)=p$. The plane $T_p(S)$ spanned by all tangent vectors to S at p, is called the *tangent plane* to S at p that is, in fact, a two dimensional vector space. For a regular point $p\in S$, a unit vector which is perpendicular to $T_p(S)$ is called a *unit normal vector* at p. For each $q\in\mathbf{x}(U)$, we define a differentiable field of unit normal vectors $N:\mathbf{x}(U)\longrightarrow R^3$ such that $N(q)=\frac{\mathbf{x}_u\times\mathbf{x}_v}{\|\mathbf{x}_u\times\mathbf{x}_v\|}(q)$, where $\mathbf{x}_u=\frac{\partial\mathbf{x}}{\partial u}$ and $\mathbf{x}_v=\frac{\partial\mathbf{x}}{\partial v}$. The

map $N : S \longrightarrow G$, taking its values in the unit sphere, is called the *Gauss map* of S, where G is a unit sphere. Then the Gauss map is differentiable, and its differential dN_p of N at p is a linear map from $T_p(S)$ to $T_p(S)$. It measures the rate of the normal vector N in a neighborhood of p.

The following lemma provides a condition which must be satisfied when the unit normal vectors of a surface S change in the neighborhood of regular points. Its proof is found in Chapter 3, pp. 140 [96].

Lemma 2.4. *The differential $dN_p : T_p(S) \longrightarrow T_p(S)$ of the Gauss map is a self-adjoint linear map, that is, $(dN_p(w_1), w_2) = (w_1, dN_p(w_2))$ where w_1 and w_2 are two independent tangent vectors at a regular point p, and $(\cdot, \cdot)$ is an inner product of two vectors.*

The symmetry of the linear map dN_p, implied by Lemma 2.4, entails a necessary condition that must be satisfied between tangent vectors and the rates of changes of normal vectors at a *regular* point. It implies that, given two regular curves passing through a regular point on a surface, the unit normal vector must change along each curve satisfying the equality in the lemma.

Consider the problem of tangent-plane-continuous interpolation of two parametric space curves with normal directions, meeting at a point. Let $C_1(u)$ and $C_2(v)$ be two parametric curves with parametrically specified normal directions $N_1(u)$ and $N_2(v)$ such that $C_1(0) = C_2(0) = p$, and $N_1(0)$ and $N_2(0)$ are proportional, that is, the two curves meet at p and they share the same normal direction at the point. We look for a surface S which smoothly interpolates the curves, that is,

- S must contain $C_1(u)$ and $C_2(v)$,
- the normals of tangent planes of S along the curves must coincide with the normals of the curves, and
- S is regular at p.

Suppose that there exist such a surface S. Then, we have a local parametrization $\mathbf{x} : U \longrightarrow V \cap S$ of an open set U in R^2 onto $V \cap S \subset R^3$ for a neighborhood V of p such that

- $\mathbf{x}(0,0) = p$,
- $\mathbf{x}_u = \frac{\partial \mathbf{x}}{\partial u}(0,0) = C_1'(0)$ and $\mathbf{x}_v = \frac{\partial \mathbf{x}}{\partial v}(0,0) = C_2'(0)$, and
- the Gauss map N of S is such that $N(C_1(u)) = \frac{N_1(u)}{\|N_1(u)\|}$ and $N(C_2(v)) = \frac{N_2(v)}{\|N_2(v)\|}$.

Then, by Lemma 2.4, in order for S to be regular at p, it should be that

$$(dN_p(\mathbf{x}_u), \mathbf{x}_v) \;=\; (\mathbf{x}_u, dN_p(\mathbf{x}_v)). \tag{28}$$

By the definition of the differential,

$$\begin{aligned} dN_p(\mathbf{x}_u) &= \frac{dN(C_1(u))}{du} \mid_{u=0} \\ &= \frac{d(\frac{N_1(u)}{\|N_1(u)\|})}{du} \mid_{u=0} \\ &= \frac{N_1'(u) \parallel N_1(u) \parallel - N_1(u) \parallel N_1(u) \parallel'}{\parallel N_1(u) \parallel^2} \mid_{u=0} \\ &= \frac{N_1'(0) \parallel N_1(0) \parallel - N_1(0) \parallel N_1(u) \parallel'_{u=0}}{\parallel N_1(0) \parallel^2}. \end{aligned}$$

Since $(N_1(0), \mathbf{x}_v) = 0$, $(dN_p(\mathbf{x}_u), \mathbf{x}_v) = \frac{(N_1'(0), \mathbf{x}_v)}{\|N_1(0)\|} = \frac{(N_1'(0), C_2'(0))}{\|N_1(0)\|}$. In the same way, we get $(\mathbf{x}_u, dN_p(\mathbf{x}_v)) = \frac{(C_1'(0), N_2'(0))}{\|N_2(0)\|}$. Hence, the equation (28) becomes

$$\frac{(N_1'(0), C_2'(0))}{\| N_1(0) \|} = \frac{(C_1'(0), N_2'(0))}{\| N_2(0) \|}. \tag{29}$$

The above argument implies that enforcing two curves to have the same normal vectors at a common point does not guarantee the regularity of an interpolating surface at the point. The equation (29) is a necessary condition for regularity, indicating that, if the given curves and their normals do not satisfy the equation (29), any smoothly interpolating surface must be singular at p.

Theorem 2.5. *Let $C_1(u)$ and $C_2(v)$ be two parametric curves with parametric normal directions $N_1(u)$ and $N_2(v)$ such that $C_1(0) = C_2(0) = p$, and that $N_1(0)$ and $N_2(0)$ are proportional. Then, any surface S, which interpolates the curves with tangent plane continuity, is singular at p unless $\frac{(N_1'(0), C_2'(0))}{\|N_1(0)\|} = \frac{(C_1'(0), N_2'(0))}{\|N_2(0)\|}$.*

In conclusion we shall not impose the above compatibility constraints on the choice of normals along curves and thereby allow singularities at vertex points...

2.2 Polyhedra Smoothing Algorithm

We present below a sketch of the algorithm to C^1 smooth a simple polyhedron $\mathbb{P}$ with tangent-plane-continuous implicit surface patches.

Algorithm

1. Triangulate each of the non-triangular polygonal faces of the given polyhedron $\mathbb{P}$. Any simple polygon is easily triangulable by adding non-intersecting inner diagonals.

2. Specify a unique "normal" vector at each vertex of $\mathbb{P}$. This provides a unique tangent plane for all patches which shall C^1 interpolate that vertex.

3. Next, construct a curvilinear wire frame by replacing each edge of $\mathbb{P}$ with a curve which C^1 interpolates the end points of the edge and the specified "normals". Any remaining degrees of freedom of the C^1 interpolatory curve are used to select a desired shape of the curve and indirectly thereby a desired shape of the smoothing surface patch.

4. Specify normal vectors at each point along each of the edge curves. This provides the tangent planes for the two incident patches which shall C^1 interpolate the edge curves. If it is required that the individual patches are non-singular at the vertices of $\mathbb{P}$, then the variation of normals along different edge curves incident at the same vertex need also to be made compatible.

5. Finally, C^1 interpolate the three edge curves and curve normals of each face. The remaining degrees of freedom for each individual patch are chosen via weighted least squares to achieve a suitably shaped single-sheeted surface patch. The resulting surface patches yield a globally C^1 smooth curved model for the given polyhedron.

Details of each of the steps 2 to 5 of the algorithm are presented in subsequent sections 2.3, 2.4 and 2.5.

2.3 Wireframe Construction for Implicit Algebraic Splines

2.3.1 Choice of Vertex Normals

The unique "normal" vector assigned to each vertex of the triangulated polyhedron $\mathbb{P}$ can be chosen independently and quite arbitrarily. However the relative directions of each adjacent vertex normal pair affects the degree of the C^1 interpolating edge curve which replaces the straight edges of $\mathbb{P}$. Let the two normal vectors at the two endpoints of an edge be called an *edge-normal-pair*. Certain relative directions of an edge-normal-pair induce an inflection point (zero curvature point) for any C^1 interpolating curve. Since conics do not have inflection points one is then forced to either switch to cubic curves at the least or to artificially split the edge. Splitting an edge in turn induces splitting of the triangular face of $\mathbb{P}$, a case considered for parametric surface patches in [] and for implicit surface patches in []. Here we restrict ourselves to surface fitting without the splitting of any triangular faces of $\mathbb{P}$.

We first derive a necessary and sufficient condition for the relative directions of an edge-normal-pair to allow a C^1 conic interpolation. Here, the interpolation is *strict* in that the curve's normal at the vertex points and the prescribed vertex normal are in the same direction and not opposite. This restriction guarantees the construction wire frames which are free of cusp-like connections. In the following definitions and lemmas we make all of this more precise.

Definition 2.6. *Let $P_0 = (p_0, n_0)$ and $P_1 = (p_1, n_1)$ be an edge-normal-pair. A conic segment $S(P_0, P_1)$ is said to C^1-*interpolate* P_0 and P_1 if there exists a non-degenerate conic curve $f(x, y) = ax^2 + 2hxy + by^2 + 2gx + 2fy + c$ such that*

- *$S(P_0, P_1)$ is a continuous segment of $f(x, y) = 0$,*
- *p_0 and p_1 are the end points of $S(P_0, P_1)$, and*
- *the gradients of $f(x, y) = 0$ at p_0 and p_1 have the same directions as n_0 and n_1, respectively. In other words, $\frac{(\nabla f(p_0), n_0)}{\|\nabla f(p_0)\| \cdot \|n_0\|} = 1$, and $\frac{(\nabla f(p_1), n_1)}{\|\nabla f(p_1)\| \cdot \|n_1\|} = 1$.*

Given a pair $P = ((p_x, p_y), (n_x, n_y))$, we can define $T_P(x, y) = n_x(x - p_x) + n_y(y - p_y) = 0$ which is the equation of the tangent line that passes through (p_x, p_y) and has a normal direction (n_x, n_y). Note that the tangent line $T_P(x, y) = 0$ contain the same direction as (n_x, n_y), and divides a plane into a positive halfspace $\{(x, y) \in \mathbb{R}^2 | T_P(x, y) > 0\}$, and a negative halfspace $\{(x, y) \in \mathbb{R}^2 | T_P(x, y) < 0\}$.

Lemma 2.7. *Let p_0 and p_1 be on a proper conic $f(x, y) = ax^2 + 2hxy + by^2 + 2gx + 2fy + c = 0$. Then, $T_{(p_0, \nabla f(p_0))}(p_1) \cdot T_{(p_1, \nabla f(p_1))}(p_0) > 0$.*

Proof : Without loss of generality, we assume that $p_0 = (0, 0)$, and $p_1 = (1, 0)$. Since $\nabla f(x, y) = (2ax + 2hy + 2g, 2hx + 2by + 2f)$, $\nabla f(0, 0) = (2g, 2f)$ and $\nabla f(1, 0) = (2a + 2g, 2h + 2f)$. Hence, $T_{(p_0, \nabla f(p_0))}(x, y) = 2gx + 2fy$, and $T_{(p_1, \nabla f(p_1))}(x, y) = (2a + 2g)(x - 1) + (2h + 2f)y$. ¿From the containment conditions of the two points, $f(0, 0) = c = 0$, and $f(1, 0) = a + 2g + c = 0$. Then, $T_{(p_0, \nabla f(p_0))}(p_1) \cdot T_{(p_1, \nabla f(p_1))}(p_0) = 2g(-2a - 2g) = 2g(-2(-2g) - 2g) = 4g^2 \geq 0$. If $g = 0$, it follows that $a = c = g = 0$ in which case $f(x, y)$ reduces into two lines. Since we assume that $f(x, y)$ is proper, $g \neq 0$, and we have proven the lemma. □

The geometric interpretation of the inequality $T_{(p_0, \nabla f(p_0))}(p_1) \cdot T_{(p_1, \nabla f(p_1))}(p_0) > 0$ is that p_0 is on the positive (negative) halfspace of T_{P_1} if and only if p_1 is on the positive (negative) halfspace of T_{P_0}. The following theorem shows that this condition is, in fact, a sufficient and necessary condition.

Theorem 2.8. *There exists a conic segment $S(P_0, P_1)$ that smoothly interpolates two pairs $P_0 = (p_0, n_0)$ and $P_1 = (p_1, n_1)$ if and only if $T_{P_0}(p_1) \cdot T_{P_1}(p_0) > 0$.*

Proof : ($\Rightarrow$) Let $f(x,y) = 0$ be a conic for a smoothly containing conic segment. From our definition of smooth interpolation, it follows that $T_{P_0}(p_1) \cdot T_{P_1}(p_0) = T_{(p_0,\nabla f(p_0))}(p_1) \cdot T_{(p_1,\nabla f(p_1))}(p_0)$ which is positive according to Lemma 2.7.
($\Leftarrow$) If $T_{P_0}(p_1) \cdot T_{P_1}(p_0) > 0$, then the conic in $q(x,y) = L(x,y)^2 - \kappa \cdot T_{P_0}(x,y) \cdot T_{P_1}(x,y) = 0$ or $-q(x,y) = 0$ will smoothly interpolate the two pairs where $L(x,y) = 0$ is the line connecting p_0 and p_1, and κ is a constant [201].[1] $\square$

Now, back to the original problem of computing a quadric wire smoothly interpolating two given point and unit normal vector pairs $P_0 = (p_0, n_0)$ and $P_1 = (p_1, n_1)$ in $\mathbb{R}^3$. The concept of the tangent line in a plane is naturally extended to an oriented tangent plane $T_P(x,y,z) = n_x(x - p_x) + n_y(y - p_y) + n_z(z - p_z) = 0$ given $P = ((p_x, p_y, p_z), (n_x, n_y, n_z))$ in 3D space, and this tangent plane divides 3D space into two halfspaces. In fact, we see that the inequality $T_{P_0}(p_1) \cdot T_{P_1}(p_0) > 0$ is also a criterion which determines if a quadric wire can smoothly interpolate two given pairs of points and normal vectors.

Corollary 2.9. *Given two point and unit normal vector pairs $P_0 = (p_0, n_0)$ and $P_1 = (p_1, n_1)$ in 3D space, there exists a quadric wire $W(t) = (C(t), N(t))$, contained in a plane determined by a given plane normal vector npl_{01}, that smoothly interpolates the pairs if and only if $T_{P_0}(p_1) \cdot T_{P_1}(p_0) > 0$.*

Proof : Consider the two pairs P_0 and P_1, their two tangent planes T_{P_0} and T_{P_1}, and the plane H which is defined by npl_{ij}. Then, the intersection lines of H and T_{P_0} and T_{P_1} become the tangent lines in H to which a conic curve must be tangent. That is, the normal vectors of the tangent lines are the projections of the normal vectors of the tangent planes. Note that the positiveness and negativeness of halfspaces are inherited from 3D space to the plane H. Hence, we see that the inequality $T_{P_0}(p_1) \cdot T_{P_1}(p_0) > 0$ holds in 3D space if and only if its 2D version holds in H.

If there exists a conic curve in H, we can find a quadric surface which smoothly interpolates the given pairs, as explained before, and take $W(t)$ from this quadric surface that has the same gradient directions as the given two normal vectors. $\square$

2.3.2 Generation of a Conic Wireframe

First, we give a definition of the *quadric wire*.

Definition 2.10. *Let $C(t) = (\frac{x(t)}{w(t)}, \frac{y(t)}{w(t)}, \frac{z(t)}{w(t)})$ and $N(t) = (\frac{nx(t)}{w(t)}, \frac{ny(t)}{w(t)}, \frac{nz(t)}{w(t)})$ be two triples of quadratic rational parametric polynomials. Then, the pair $W(t) = (C(t), N(t))$ is called a quadric wire if there exists a quadratic surface $q(x,y,z) = 0$ such that $q(C(t)) = 0$ and $\nabla q(C(t))$ is proportional to $N(t)$ for all real t.*

The first step to smoothing a convex polyhedron is to compute a conic curve given two point and unit normal vector pairs (p_0, n_1), (p_1, n_1) and a normal *npl* of a plane such that

1. the computed conic curve passes through p_0 and p_1,
2. its tangents at p_0 and p_1 are perpendicular to n_0 and n_1, respectively, and
3. it is contained in the plane which contains p_0 and p_1, and has the plane normal *npl*.

Especially, we force $W(0) \equiv (p_0, n_1)$ and $W(1) \equiv (p_1, n_1)$, [2] and hence use a segment of $W(t)$, $0 \le t \le 1$. To compute $C(t)$, the normal vectors n_0 and n_1 are projected into the plane P on which $C(t)$ will be. (See Figure 12).

[1] Thanks to J. Yu for pointing this.
[2] By $\equiv$, we mean the points are the same, and the normal vectors are proportional.

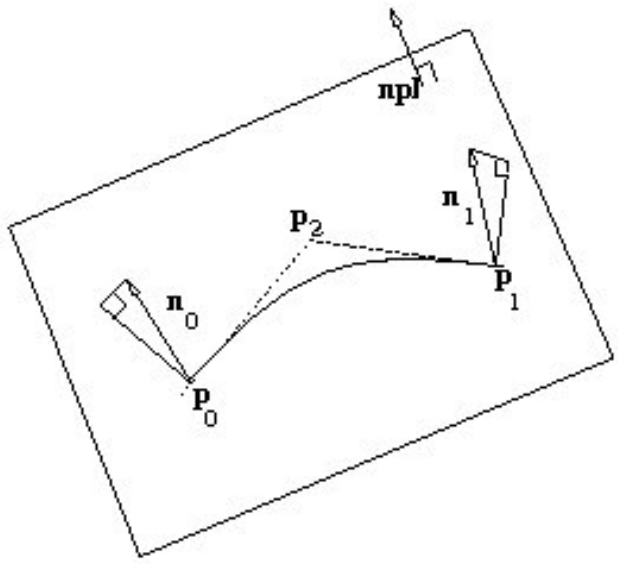


Figure 12: Computation of a Conic Curve

This projection results in a control triangle $p_0 - p_2 - p_1$. Lee [150] presents a compact method for computing a conic curve $C(t)$ from such a control triangle. In his formulation, the conic is expressed in Bernstein-Bézier form :

$$C(t) = \frac{w_0 p_0 (1-t)^2 + 2w_2 p_2 t(1-t) + w_1 p_1 t^2}{w_0(1-t)^2 + 2w_2 t(1-t) + w_1 t^2},$$

where $w_i > 0$, $i = 0, 1, 2$ are shape control parameters. An often used parameterization, called the *rho-conic parameterization*, is given by the special choice $w_0 = w_1 = 1 - \rho$, $w_2 = \rho$, $\rho > 0$. By introducing the parameter ρ, we can control the shape of a conic intuitively. Let $p_{01} = (p_0 + p_1)/2$ be the midpoint of the chord $p_0 p_1$. Then, ρ has a property that $C(0.5) - p_{01} = \rho(p_2 - p_{01})$. From this, we can see that as ρ is increased, the conic gets more curved. In particular, it can be proven that $\rho = 0.5$ for parabola, $0 < \rho < 0.5$ for ellipse and $0.5 < \rho < 1.0$ for hyperbola.

2.3.3 Assigning Normals along Edge curves

Once $C(t)$ is fixed, we find a quadratic surface $q(x, y, z) = 0$ such that $N(t)$, which is a restriction of $\nabla q(x, y, z)$, interpolates n_0 and n_1. Consider a quadratic surface $q(x, y, z) = c_0 x^2 + c_1 y^2 + c_2 z^2 + c_3 xy + c_4 yz + c_5 zx + c_6 x + c_7 y + c_8 z + c_9 = 0$. $q(x, y, z) = 0$ has 10 coefficients, and since dividing the surface by any nonzero coefficient does not change the surface, there are 9 degrees of freedom. The first requirement is that $q(x, y, z) = 0$ must contain the computed conic $C(t)$. Our Hermite interpolation algorithm gives 5 linear equations in terms of the unknowns c_i for the containment requirement. It is obvious that 5 constraints on c_i are required considering the Bezout theorem which says if a conic intersects with a quadratic surface at more than 4 points, the curve is contained in the surface.

Hence, 4 ($= 9-5$) degrees of freedom in choosing c_i are left, and these must be used to interpolate the normal vectors at the two end points. Interpolating n_0 and n_1 at p_0 and p_1, respectively, gives 2 more linear constraints which leaves 2 degrees of freedom in choosing the quadratic surface. But we can see that requiring only one more normal vector at a point on the curve fixes the normal vectors along the whole conic. Consider the gradient vector $\nabla q(x, y, z)$ whose components are linear. Then, the vector function $\nabla q(C(t))$ is a degree 2 polynomial parametric curve in the projective space, and hence, three independent constraints fixes the curve $\nabla q(C(t))$, or the normal vector along $C(t)$.

After we specify one more normal vector at a point on the conic, we obtain a family of quadratic surfaces $q(x, y, z)$ with one degree of freedom where all the surfaces in the family contain $C(t)$, and share the same gradient vectors along $C(t)$. This observation leads to the following lemma :

Lemma 2.11. *Let $W(t) = (C(t), N(t))$ be a quadric wire. Then, the quadratic surfaces which smoothly interpolate $W(t)$ comprises a family of surfaces with one degree of freedom.*

What we do in our implementation in order to fix the normal vector is the following : first, the average $n_{01} = (n_0 + n_1)/2$ is computed, and then n_{01} is projected into a plane which contain $C(0.5)$, and is perpendicular to the tangent vector $C'(0.5)$. Then, we require the projected vector to be the normal vector at $C(0.5)$. Once the normal vectors along $C(t)$ is fixed, we define $N(t)$ to be the vectors.

2.4 Local Interpolatory Patch Generation

2.4.1 Conditions for Low Degree Interpolants

We first compute general degree bounds for interpolatory triangular patches with degree d interpolatory curves.

Definition 2.12. *An augmented triangle is an 9-tuple $T = (p_0, p_1, p_2, n_0, n_1, n_2, npl_{01}, npl_{12}, npl_{20})$ where the points p_i are three vertices of a triangle with the corresponding unit normal vectors n_i, and npl_{ij} is the normal of the plane which will contain the quadric wire made from (p_i, n_i) and (p_j, n_j).*

Definition 2.13. *A quadric triangle is a triple $QT = (W_0(t), W_1(t), W_2(t))$ of quadric wires such that $W_0(1) \equiv W_1(0)$, $W_1(1) \equiv W_2(0)$, and $W_2(1) \equiv W_0(0)$.*

Given an augmented triangle, each quadric wire is computed as described in the foregoing subsection. Now the quadric triangle is to be fleshed using an algebraic surface $f(x, y, z) = 0$. The algebraic surface to be used should be flexible enough to interpolate the three quadric wires smoothly, i.e., with tangent plane continuity. Though higher degree algebraic surfaces provide more flexibility, the number $\binom{n+3}{3}$ of coefficients of a degree n algebraic surface grows dramatically as n increases. Hence, for fast computation and less numerical errors, keeping the degree of a surface in a reasonable range is very important. In the below, we give the low bound of degree which must be used for interpolation of a quadric triangle.

First, let's assume that we use a degree n algebraic surface $f(x, y, z) = 0$ to smoothly interpolate a wire of degree d $W(t) = (C(t), N(t))$. According to the Bezout theorem, $dn + 1$ constraints on the coefficients of f are required for f to contain $C(t)$ which is of degree d. For tangent plane continuity, consider the restricted normal vector $\nabla f(C(t))$. Since the degree of each component of $\nabla f(x, y, z)$ is, at most, $n - 1$, each component of $\nabla f(C(t))$ has the degree $d(n - 1)$. This vector function is, in fact, a degree $d(n - 1)$ parametric polynomial curve in the projective space. Hence $d(n - 1) + 1$ independent constraints are enough to fix the gradient of f along the curve $C(t)$, making $\nabla f(C(t))$ proportional to $N(t)$ which is the requirement of tangent plane continuity.

Lemma 2.14. *Let $W(t) = (C(t), N(t))$ be a degree d wire. For an algebraic surface $f(x, y, z) = 0$ of degree n to smoothly interpolate $W(t)$, at most $2dn - d + 2$ $(= dn + 1 + d(n - 1) + 1)$ independent linear constraints on the f's coefficients must be satisfied.*

For C^1 interpolation of a triangular patch we observe some geometric dependency between the three wires which leads to algebraic dependency. First, since the curvess intersect pairwise, there must be three rank deficiencies between the equations from the containment conditions. [3] Secondly,

[3] This dependency gets more evident to see the Hermite interpolation algorithm. In the algorithm in Section 4.2.1 [26], if we always choose the intersection points for the list L_c of each conic, three equations are generated twice.

at each vertex of the curvlinear triangle, two incident curves automatically determine the normal at the vertex. It is obvious, from the way the curve wire construction, this vector is proportional to the given unit normal vector. So, we see that satisfying the containment conditions for the 3 curves guarantees that any interpolating surface has gradient vectors at the three points as required. This fact implies that, for each conic, there are two rank deficiencies between the linear equations for the containment conditions, and the equations for its tangency condition. [4] Hence, 6 additional rank deficiencies with the previous 3 yield a total of 9 deficiencies.

Lemma 2.15. *Let $QT = (W_0(t), W_1(t), W_2(t))$ be a conic triangle. The rank of the linear system $\mathbf{M_I x} = \boldsymbol{z}$ which is constructed by the Hermite interpolation algorithm for the algebraic surface $f(x, y, z) = 0$ of degree n that smoothly fleshes QT, is at most $12n - 9$.*

Proof. For C^1 of all three wires requires $3(4n - 2 + 2) = 12n$ using lemma 2.14 minus the 9 deficiencies. □

Since $f(x, y, z) = 0$ of degree n has $\binom{n+3}{3}$ coefficients, and the rank of the linear system should be less than the number of coefficients for a nontrivial surface to exist, we see that 5 is the minimum degree required. In the quintic case, there are 56 coefficients (55 degrees of freedom) and the rank is at most 51, which results in a family of interpolating surfaces with at least 4 degrees of freedom in selecting an instance surface from the family.

Even though some special combination of three quadric wires can be interpolated by a surface of degree less than 5, for example, three quadric wires from a sphere, the probability that such spatial dependency occurs, given an arbitrary triple of conics with normals, is infinitesimal. Hence, we can say that 5 is the minimum degree required *with the probability one*.

Lemma 2.16. *Let $QT = (W_0(t), W_1(t), W_2(t))$ be a cubic triangle. The rank of the linear system $\mathbf{M_I x} = \boldsymbol{z}$ which is constructed by the Hermite interpolation algorithm for the algebraic surface $f(x, y, z) = 0$ of degree n that smoothly fleshes QT, is at most $18n - 12$.*

Proof. For C^1 of all three wires requires $3(6n - 3 + 2) = 18n - 3$ using lemma 2.14 minus the 9 deficiencies. □

The minimum degree of the interpolating surface is 7. In the quintic case, there are 120 coefficients (119 degrees of freedom) and the rank is at most 114, which results in a family of interpolating surfaces with at least 5 degrees of freedom in selecting an instance surface from the family.

Lemma 2.17. *Let $QT = (W_0(t), W_1(t), W_2(t))$ be a conic triangle with one edge a cubic curve. The rank of the linear system $\mathbf{M_I x} = \boldsymbol{z}$ which is constructed by the Hermite interpolation algorithm for the algebraic surface $f(x, y, z) = 0$ of degree n that smoothly fleshes QT, is at most $14n - 10$.*

Proof. For C^1 of all three wires requires $2(4n - 2 + 2) + (6n - 3 + 2) = 14n - 1$ using lemma 2.14 minus the 9 deficiencies. □

The minimum degree of the interpolating surface is 6. In the degree 6 case, there are 84 coefficients (83 degrees of freedom) and the rank is at most 74, which results in a family of interpolating surfaces with at least 9 degrees of freedom in selecting an instance surface from the family.

Lemma 2.18. *Let $QT = (W_0(t), W_1(t), W_2(t))$ be a cubic triangle with one edge a conic curve. The rank of the linear system $\mathbf{M_I x} = \boldsymbol{z}$ which is constructed by the Hermite interpolation algorithm for the algebraic surface $f(x, y, z) = 0$ of degree n that smoothly fleshes QT, is at most $16n - 11$.*

[4] Again, for each curve, we can choose point-normal pairs at the two end points. The resulting two linear equations should be linearly dependent on the equations from the containment requirement.

Proof. For C^1 of all three wires requires $(4n-2+2)+2(6n-3+2)=16n-2$ using lemma 2.14 minus the 9 deficiencies. □

The minimum degree of the interpolating surface is 7. In the degree 7 case, there are 120 coefficients (119 degrees of freedom) and the rank is at most 101, which results in a family of interpolating surfaces with at least 18 degrees of freedom in selecting an instance surface from the family.

2.4.2 C^1 Interpolation of a Conic Wireframe

As mentioned previously, each triangular face of a polyhedron is replaced by a triangular patch. To do so, each edge is replaced by a quadric wire forming a wire frame for the polyhedron.

Even though some special combination of three quadric wires can be interpolated by a surface of degree less than 5, for example, three quadric wires from a sphere, the probability that such spatial dependency occurs, given an arbitrary triple of conics with normals, is infinitesimal. Hence, we can say that 5 is the minimum degree required *with the probability one*.

2.5 Surface Selection and Local Shape Control

As a result of smooth interpolation of a quadric triangle QT with a quintic surface, a family of algebraic surfaces $f(x,y,z)=0$ with, at least, 4 degrees of freedom is obtained. The family is expressed as the nontrivial coefficients vectors in the nullspace of $\mathbf{M_I}$. To select a quintic surface from this family, those 4 degrees of freedom must be consumed. Least squares approximation is well suited for this purpose. We can additionally specify a set of points inside the quadric triangle, which approximately describes a desirable surface patch. The final fitting surface can be obtained by consuming the remaining degrees of freedom through least squares approximation to this set of points.

While it is chosen from the family via least squares approximation, the final quintic surface is not always good in the light of geometric modeling. For example, a surface which self-intersects inside the quadric triangle is not practically useful though it approximates the additional points best as well as satisfies the smooth interpolation requirement. Hence, in the approximation step, we need to be careful not to select a surface which is singular inside the quadric triangle. First of all, it is observed that, in general, any surface which smoothly interpolates the quadric triangle, that is, three conics with normal directions, is singular at the three vertices. In Section 2.1.2, we show that just making the normals of the conics consistent at the intersection points is not enough to have a regular surface. In fact, the velocities of changes of the normal vectors at the intersection points affect the regularity of a surface at the points. However, the singularities only at the three vertices, not along the whole curve, does not harm the smooth continuity between surface patches. More serious problem is the singularity of a surface inside a quadric triangle.

2.5.1 Solution of Interpolation and Least-Squares Matrices

The Hermite interpolation algorithm takes as input positional and first derivative (normal) information on points and algebraic space curves. For an algebraic surface $S: f(x,y,z)=0$ of degree n, it produces a homogeneous linear system $\mathbf{M_I x}=\mathbf{z}$, $\mathbf{M_I}\in\mathbb{R}^{n_i\times n_v}$ of n_i equations and n_v unknowns where $\mathbf{x}$ is a vector of the $n_v(=\binom{n+3}{3})$[5] coefficients of S.

Then, the nontrivial solutions in the nullspace of $\mathbf{M_I}$ form a family of all possible algebraic surfaces of degree n, satisfying the given input constraints, whose coefficients are expressed by homogeneous combinations of q free parameters where $q=n_v-r$ is the dimension of the nullspace. Since dividing $f(x,y,z)=0$ by a nonzero number does not change the surface, there are, in fact,

[5]There are $\binom{n+3}{3}$ coefficients in $f(x,y,z)$ of degree n.

$n_v - r - 1$ degrees of freedom in choosing an instance surface from the family. Hence, the rank r of $\mathbf{M_I}$ must be less than the number of the coefficients n_v, should there exist an interpolating surface.

After a family of algebraic surfaces is obtained, we should select an instance surface from the family for geometric design. For this process, called shape control, Bajaj and Ihm [29] proposed to use least squares approximation which leads to the following computational model :

$$\begin{array}{ll} \textit{minimize} & \| \mathbf{M_A x} \|^2 \\ \textit{subject to} & \mathbf{M_I x} = \mathbf{z} \\ & \mathbf{x}^T \mathbf{x} = 1, \end{array}$$

where $\mathbf{M_A} \in \mathbb{R}^{n_a \times n_v}$ is the matrix for least-squares approximation. $\mathbf{M_A}$ is constructed the same as $\mathbf{M_I}$ is, however, there are more linear constraints provided than the remaining degrees $n_v - r - 1$ of freedom such that least squares approximation is applied. For example, we can additionally construct enough number of points or curves around the given input data, which approximately specifies a desirable surface. The final instance surface is obtained by consuming the remaining degrees of freedom through least-squares approximation to the additional data set.

In this article, we use a slightly different model which is :

$$\begin{array}{ll} \textit{minimize} & \| \mathbf{M_A x} - \mathbf{b} \|^2 \\ \textit{subject to} & \mathbf{M_I x} = \mathbf{z}. \end{array}$$

Suppose that $S_0 = \{v_i \in \mathbb{R}^3 | i = 1, \cdots, l\}$ be a set of points which approximately describes a desirable surface patch. Then, we can get a linear system $\mathbf{M_A x} = \mathbf{z}$, where each row of $\mathbf{M_A}$ is obtained from $f(v_i) = 0$. Then the conventional least squares approximation is to minimize $\| \mathbf{M_A x} \|^2$ over the nullspace of $\mathbf{M_I}$. However, our experimentation shows that in many cases, singularities occur inside the quadric triangle. Minimizing $\| \mathbf{M_A x} \|^2$ makes the resulting surface well approximate the set of points, however, this simple algebraic approximation can not prevent the resulting surface from self-intersecting inside the triangle.

To provide more geometric control in least squares approximation, we suggest that contour levels be approximated rather than only the surface itself. In fact, the implicit surface $f(x, y, z) = 0$ is the zero contour of the function $w = f(x, y, z)$. Consider some smooth region of a surface. Since the derivatives of $w = f(x, y, z)$ are well defined in the region, the contour levels behaves well in the proximity of the zero contour. In our scheme, we first generate $S_0 = \{(v_i, n_i) | i = 1, \cdots, l\}$ where v_i are approximating points, and n_i are approximating gradient vectors at v_i. Then, from this set, we construct two more sets $S_1 = \{u_i | u_i = v_i + \alpha n_i, i = 1, \cdots, l\}$, and $S_{-1} = \{w_i | w_i = v_i - \alpha n_i, i = 1, \cdots, l\}$ for some small $\alpha > 0$. Then, we get the least squares system $\mathbf{M_A} = \mathbf{b}$ from three kinds of equations : $f(v_i) = 0$, $f(u_i) = 1$, and $f(w_i) = -1$. These equations give an approximating contour level structure of the function $w = f(x, y, z)$ near the inside of a quadric triangle. We found out that forcing a well behaved contour levels gets rid of self-intersection in the region significantly. We will give a heuristic algorithm for generation of the point-normal set S_0 in the last paragraph of Subsection 2.5.2.

As a result of least squares approximation of the function's contour levels, we lead to the following computational model :

$$\begin{array}{ll} \textit{minimize} & \| \mathbf{M_A x} - \mathbf{b} \|^2 \\ \textit{subject to} & \mathbf{M_I x} = \mathbf{z}, \end{array}$$

where $\mathbf{M_I} \in \mathbb{R}^{n_i \times 56}$ is a Hermite interpolation matrix, and $\mathbf{M_A} \in \mathbb{R}^{n_a \times 56}$ and $b \in \mathbb{R}^{n_a}$ are matrix and vector, respectively, for contour level approximation, and $\mathbf{x} \in \mathbb{R}^{56}$ is a vector containing coefficients of a quintic algebraic surface $f(x, y, z) = 0$.

To find the nullspace of $\mathbf{M_I}$ in a computationally stable manner, the singular value decomposition (SVD) of $\mathbf{M_I}$ is computed [119] where $\mathbf{M_I}$ is decomposed as $\mathbf{M_I} = U \Sigma V^T$ where $U \in \mathbb{R}^{n_i \times n_i}$ and $V \in \mathbb{R}^{56 \times 56}$ are orthonormal matrices, and $\Sigma = diag(\sigma_1, \sigma_2, \cdots, \sigma_s) \in \mathbb{R}^{n_i \times 56}$ is a diagonal

matrix with diagonal elements $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_s \geq 0$ ($s = min\{n_i, 56\}$). It is known that the rank r of $\mathbf{M_I}$ is the number of the positive diagonal elements of Σ, and that the last $56-r$ columns of V span the nullspace of $\mathbf{M_I}$. Hence, the nullspace of $\mathbf{M_I}$ is expressed as :
$\{\mathbf{x} \in \mathbb{R}^{56} \,|\, \mathbf{x} = \sum_{i=1}^{56-r} w_i \mathbf{v}_{r+i},\ \textit{where } w_i \in \mathbb{R},\ \textit{and } \mathbf{v}_j \textit{ is the jth column of } V\}$, or $\mathbf{x} = V_{56-r}\mathbf{w}$ where $V_{56-r} \in \mathbb{R}^{56\times(56-r)}$ is made of the last $56-r$ columns of V, and $\mathbf{w}$ a $(56-r)$-vector. [6] $\mathbf{x} = V_{56-r}\mathbf{w}$ compactly expresses all the quintic surfaces which Hermite-interpolate the three quadric wires.

After substitution for $\mathbf{x}$, we lead to $\| \mathbf{M_A}\mathbf{x} - \mathbf{b} \| = \| \mathbf{M_A}V_{56-r}\mathbf{w} - \mathbf{b} \|$. Then, an orthogonal matrix $Q \in \mathbb{R}^{n_a \times n_a}$ is computed such that

$$Q^T \mathbf{M_A} V_{56-r} = R = \begin{pmatrix} R_1 \\ \mathbf{z} \end{pmatrix}$$

where $R_1 \in \mathbb{R}^{(56-r)\times(56-r)}$ is upper triangular. (This factorization is called a *Q-R factorization* [119]). Now, let

$$Q^T b = \begin{pmatrix} c \\ d \end{pmatrix}$$

where c is the first $56-r$ elements. Then, $\| \mathbf{M_A}V_{56-r}\mathbf{w} - \mathbf{b} \|^2 = \| Q^T\mathbf{M_A}V_{56-r}\mathbf{w} - Q^T\mathbf{b} \|^2 = \| R_1\mathbf{w} - c \|^2 + \| d \|^2$. The solution $\mathbf{w}$ can be computed by solving $R_1\mathbf{w} = c$, from which the final fitting surface is obtained as $\mathbf{x} = V_{56-r}\mathbf{w}$.

2.5.2 Display of the Triangular Algebraic Patch

As implicitly defined algebraic surfaces have become increasingly important in geometric modeling, several algorithms for displaying them have emerged. Implicit algebraic surfaces lend themselves naturally to ray tracing [126]. Sederberg and Zundel [215] uses a scan line display method which offers improvement in speed and correctly displays singularities. Even though both approaches produce images of good qualities, the computational cost is high. Also, the static processes do not allow interactive display of surfaces. On the other hand, polygonization of implicit surfaces [57] can use the capability of the graphics hardware which provides very fast interactive rendering. In [57], Bloomenthal presents a numerical technique that approximates an implicit surface with a polygonal representation. The technique is to surround the implicit surface with an octree, at whose corners the implicit function is sampled to generate polygons. Although, in general, they sample implicit surfaces well, these polygonization methods are not well suited to our purpose which is to draw a implicit triangular surface patch with singular vertices. A major problem is how to isolate only the necessary part or the triangular patch from the whole implicit surface. Clipping surfaces could be added to the polyonization algorithms, however, the current polygonization algorithms do not handle singularities quite well.

In our display routine, we *walk over* implicit quintic surfaces only around the necessary regions producing polygons which approximate triangular patches. Since smooth segments of intersection curves of two algebraic surfaces are well traced [25], and we are to display smooth portions of implicit surfaces, the algebraic space curve tracing routines performs well for the walk-over. Note that though the fleshing quintic surface are usually singular at the three vertices of a quadric triangle, the boundary curves can be traced easily from their parametric equations.

The following simple recursive procedure produces adaptive polygonization of a triangular algebraic surface patch. Let $f(x, y, z) = 0$ be a primary surface whose triangular portion clipped by three planes $h_i(x, y, z) = 0$, $i = 1, 2, 3$ is to be polygonized. Initially, the triangle $T_0 = (P_0, P_1, P_2)$ is a rough approximation of the surface patch. Each boundary curve decided by f and h_i is traced

[6] As mentioned before, in most cases, the rank r of $\mathbf{M_I}$ is 51. However, we keep the variable r because it is possible that there are more dependency between boundary curves and normal vectors though the chances are rare.

producing a digitized linear approximation to the space curve, then the linear approximation is segmented adaptively into a new segment of order 2^d for some given d. (See, for example, [135] for an adaptive segmentation algorithm of space curves.) Then, T_0 is refined into four triangles by introducing the 3 points Q_0, Q_1, and Q_2 where Q_i, $i = 0, 1, 2$ is the center point of each adaptive segmentation of order 2^d. The clipping planes of subdivided triangles can be computed by averaging the normals of the two triangles incident to the edge. Then, each new edge is traced, and then its adaptive linear approximation of order 2^{d-1} is produced. In this way, this new approximation is further refined by recursively subdividing each triangle until some stopping criterion is met.

While the method produces a regular, but adaptive, network of polygons, it might be improved to generate more adaptive polygonization. Rather than subdividing all the triangles up to the same level, each triangle is examined to see if it is already a good approximation to the surface portion it is approximating. It is refined only when the answer is no. Some criterions for such local refinement are suggested in [57]. However, to design an irregular adaptive polygonization algorithm with robust local refinement criterions, is an open problem.

We also use the above recursive subdivision scheme to produce $S_0 = \{(v_i, n_i)|i = 1, \cdots, l\}$.Initially, only the boundary curves are known, and each time a new curve is to be traced in the algorithm, a quadric wire is computed. from the information on the initial and final points, their normals and clipping plane. The generated quadric wire gives approximate curve and normal information, and is traced to generate points and normals. The final polygonal approximation obtained in this way gives a set of points which is used in least squares approximation. We observe that this heuristic method work quite well when the ρ value is in the reasonable range, say, $0.25 \leq \rho \leq 0.75$.

2.5.3 Smoothing a Convex Polyhedron

In Section 2.4, we have described how to compute a quintic triangular algebraic surface patch from a given augmented triangle. A convex polyhedron is smoothed by replacing its faces with the triangular patches meeting each other with tangent plane continuity. For the augmented triangles $T = (p_0, p_1, p_2, n_0, n_1, n_2, npl_{01}, npl_{12}, npl_{20})$ of the faces of a polyhedron, the normal data, i.e., three vertex normals and three edge normals, must be provided as well as the given three vertices. In some applications, the normal data may come with a solid, but, in general, only vertices and their facial information are provided. The vertex normal n_i at each vertex p_i can be computed by averaging the normals of the faces incident to the vertex.

Example 2.19. *Construction of Quadric Wire Frames*

Example 2.20. *Smoothed Polyhedrons Using Quintic Implicit Surfaces*

Each of 32 faces of the polyhedron in Example 2.19 is replaced by a quintic implicit algebraic surface which smoothly fleshes its quadric triangle. The result is the piecewise tangent-plane-continuous quintic algebraic surface meshes which smooth the given polyhedron. As explained before, ellipses, parabolas, and hyperbolas are used as quadric wires for $\rho = 0.4$, 0.5, and 0.75, respectively. □

Problem 2.21. *Given a list of data points $P = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_k\} \in \mathbb{R}^3$ 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 $\{\mathbf{p}_1, \ldots, \mathbf{p}_j\} \in \mathbb{R}^3$ with $j \leq 4$. Then the **convex hull** of these points is defined by $[\mathbf{p}_1\mathbf{p}_2 \cdots \mathbf{p}_j] = \{\mathbf{p} \in \mathbb{R}^3 : \mathbf{p} = \sum_{i=1}^{j} \alpha_i \mathbf{p}_i, \alpha_i \geq 0, \sum_{i=1}^{j} \alpha_i = 1\}$ and the **affine hull** is defined by $\langle \mathbf{p}_1\mathbf{p}_2 \cdots \mathbf{p}_j \rangle = \{\mathbf{p} \in \mathbb{R}^3 : \mathbf{p} = \sum_{i=1}^{j} \alpha_i \mathbf{p}_i, \sum_{i=1}^{j} \alpha_i = 1\}$. The interior of the convex hull $[\mathbf{p}_1\mathbf{p}_2 \cdots \mathbf{p}_j]$ is denoted by $(\mathbf{p}_1\mathbf{p}_2 \cdots \mathbf{p}_j) = \{\mathbf{p} \in \mathbb{R}^3 : \mathbf{p} = \sum_{i=1}^{j} \alpha_i \mathbf{p}_i, \alpha_i > 0, \sum_{i=1}^{j} \alpha_i = 1\}$.

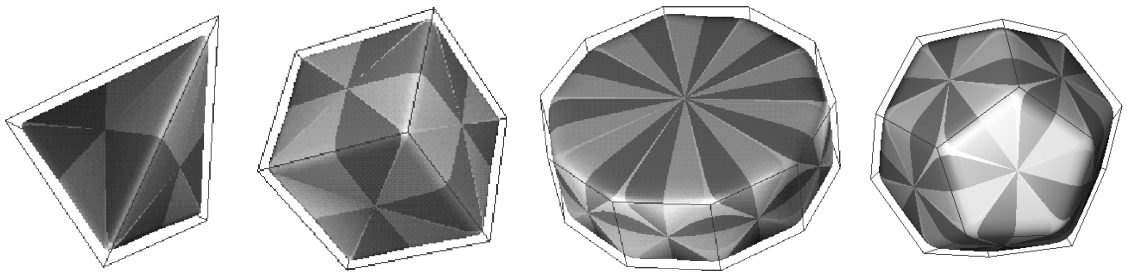


Figure 13: C^1 Smoothing of some regular polyhedra with trihedral corners.

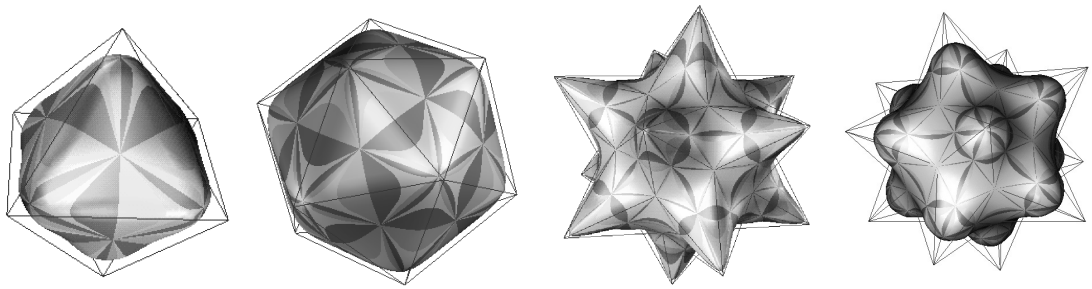


Figure 14: C^1 smoothing of some regular and stellated polyhedra with non-trihedral corners.

Sufficient Conditions of an A-Patch Let $F(\alpha) = \sum_{|\lambda|=n} b_\lambda B_\lambda^n(\alpha)$ be a given polynomial of degree n on the simplex(tetrahedron) $S = \{(\alpha_1, \alpha_2, \alpha_3, \alpha_4)^T \in \mathbb{R}^4 : \sum_{i=1}^4 \alpha_i = 1,\ \alpha_i \geq 0\}$. 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.

Smooth vertices 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 $\lambda_i \geq n-1$.

Smooth edges 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 $\sum_{m=0}^{n-1} b_{me_i+(n-1-m)e_j+e_k} B_m^{n-1}(t)$ and $\sum_{m=0}^{n-1} b_{me_i+(n-1-m)e_j+e_l} B_m^{n-1}(t)$ have no common zero in $[0, 1]$, for distinct i, j, k, l.

If the surface S_F contains a vertex/edge, then it is easy to show by the formulas of directional derivatives(see [108], p. 312) that the surface is smooth there if the smooth vertices/edges conditions above are satisfied.

Definition 3.1. *Three-sided patch.*

Let the surface patch S_F be smooth on the boundary of the tetrahedron S. If any open line segment (e_j, 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 16).

Definition 3.2. *Four-sided patch.*

Let the surface patch S_F be smooth on the boundary of the tetrahedron S. Let (i, j, k, ℓ) be a permutation of (1, 2, 3, 4). If any open line segment (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 16).

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

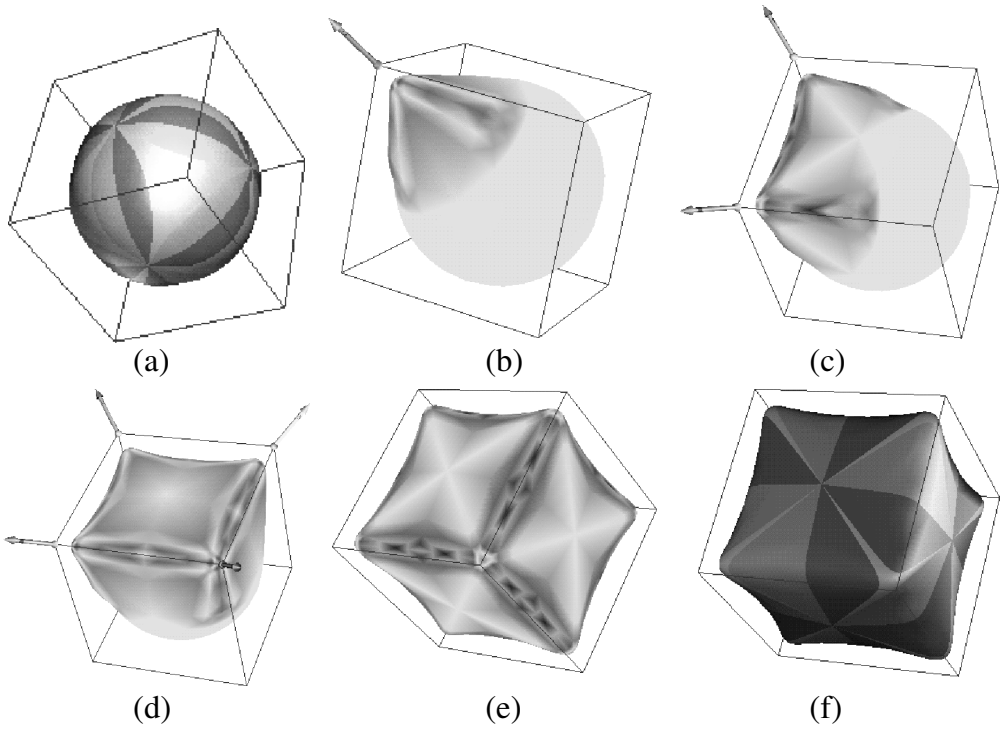


Figure 15: Interactive deformation of a sphere defined by C^2 continuous quintic A-patches. (a) Shading to show patches. (b) Pulled towards one control vertex. Mean curvature texture. (c) Pulled towards two control vertices. Gaussian curvature texture. (d) Pulled towards the four control vertices of a face of the cube. Mean curvature texture. (e) Pulled towards all six faces of the cube. Gaussian curvature texture. (f) Shading to show the final patches.

Theorem 3.2. *Let $F(\alpha) = \sum_{|\lambda|=n} b_\lambda B_\lambda^n(\alpha)$ satisfy the smooth vertices and smooth edges conditions and $j (1 \le j \le 4)$ be a given integer. If there exists an integer $k (0 \le k < n)$ such that*

$$b_{\lambda_1\lambda_2\lambda_3\lambda_4} \ge 0, \qquad \lambda_j = 0, 1, \ldots, k-1, \tag{30}$$

$$b_{\lambda_1\lambda_2\lambda_3\lambda_4} \le 0, \qquad \lambda_j = k+1, \ldots, n \tag{31}$$

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 $m (k < m \le n)$, then S_F is a three-sided j-patch.

Theorem 3.3. *Let $F(\alpha) = \sum_{|\lambda|=n} b_\lambda B_\lambda^n(\alpha)$ satisfy the smooth vertices and smooth edges conditions and (i, j, k, ℓ) be a permutation of (1, 2, 3, 4). If there exists an integer $k (0 \le k < n)$ such that*

$$b_{\lambda_1\lambda_2\lambda_3\lambda_4} \ge 0; \qquad \lambda_i + \lambda_j = 0, 1, \ldots, k-1, \tag{32}$$

$$b_{\lambda_1\lambda_2\lambda_3\lambda_4} \le 0; \qquad \lambda_i + \lambda_j = k+1, \ldots, n \tag{33}$$

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 $m (k < m \le n)$, then S_F is four-sided ij-$k\ell$-patch.

Note. The conditions on the coefficients b_λ in Theorems 3.2 and 3.3 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.

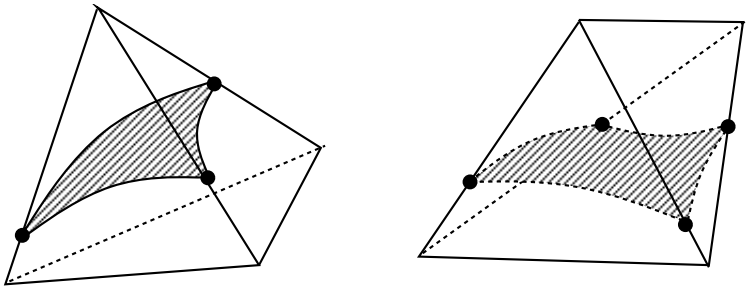


Figure 16: Three-sided and four-sided tetrahedral patches.

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.*

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

Note that a four sided patch may degenerate into a two sided patch. However, we do not need to treat the degenerate patches any different and consider it to be a special four sided patch.

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 $[e_i e_j e_k]$.*

2.6 Normals and the Simplicial Hull

For the given point set $P = \{p_1, \ldots, p_k\} \in \mathbb{R}^3$ and their surface triangulation T, we first construct a normal set $N = \{n_1, \ldots, n_k\} \in \mathbb{R}^3$ for P. That is, for each point p_i, we associate a normal n_i. We will force the constructed surface to interpolate these point 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 nclude (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 [11, 189]. Of course at times the data set can have prespecified 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 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 *positive convex*. If both the numbers are zero then we say it is *zero convex*. A *negative 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 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

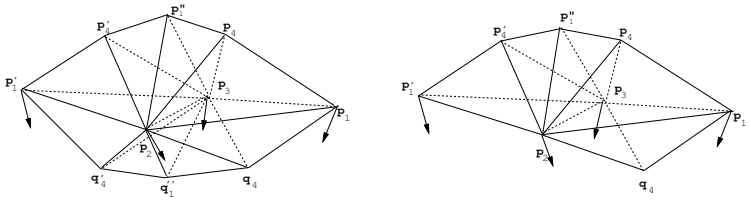


Figure 17: The construction of tetrahedra for adjacent non-convex/non-convex faces and convex/non-convex faces.

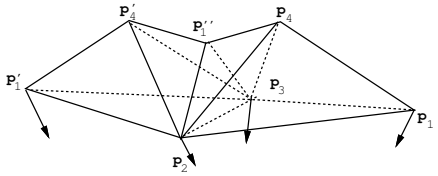


Figure 18: The construction of tetrahedra for adjacent convex/convex faces.

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 4.3. *Simplicial hull.*

A simplicial hull of T, denoted by $\sum$, is a collection of non-degenerate tetrahedra which satisfies:
(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 *non-intersection*.
(4) For each edge there is/are only one/two pair/pairs of common face sharing edge tetrahedra in $\sum$ if the edge is convex/non-convex such that the pair/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 and normals assignment, T there may exist infinitely many simplicial hulls or no simplicial hull may exist. We now describe a scheme for constructing a simplicial hull for the surface triangulation T and prescribed vertex normal assignment. We also enumerate the exceptional configurations where a simplicial hull of T is not possible and then provide a solution for constructing the simplicial hull for a locally modified T.
1. Build 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 farthermost 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. If F is positive convex, then p_4 is chosen on the side opposite to the direction of the normals, and a single face tetrahedron $[p_1p_2p_3p_4]$ is formed. If F is negative convex, then q_4 is chosen on the same side as the normals and again the single face tetrahedron $[p_1p_2p_3q_4]$ is formed. If F is zero convex, no tetrahedron needs to be built. Figure 18 shows the case where both faces are convex and Figure 17 shows the cases where at least one of the two adjacent faces

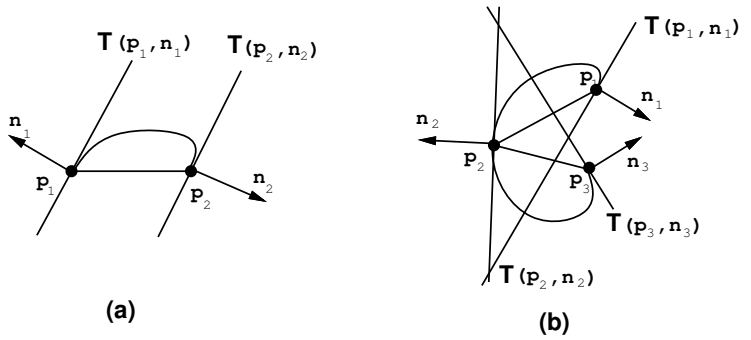


Figure 19: Examples of **(a)** sharp edge and **(b)** sharp vertex.

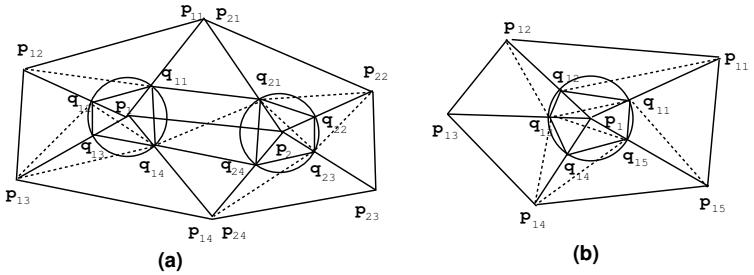


Figure 20: The re-triangulation **(a)** sharp edge **(b)** and sharp vertex

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 19 (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 negatively 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 19 (b).

A heuristic strategy rectify 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 experiment we have perfprmed.

(i) Sharp edge problem. Let $[p_1p_2]$ be a sharp edge(see Figure 20(a)), and let $[p_ip_{ij}]$ $(i = 1, 2;\ j = 1, 2, \cdots k_i)$ 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 lengthes $\|p_i - p_{ij}\|$. The sharper one wants the constructed smooth surface around the edge $[p_1p_2]$, the smaller we take r_i. Let q_{ij} be the intersection points of $S(p_i, r_i)$ and $[p_ip_{ij}]$. 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 20(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 20(b)), and let $[p_1p_{1j}]$

$(j = 1, 2, \cdots k)$ 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 lengthes $\|p_1 - p_{1j}\|$. The sharper one wants the constructed smooth surface around the vertex p_1, the smaller we take r . 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 20(b)) by connecting the adjacent edges of the polygon in the least inner angle.

2. Build Edge Tetrahedra. Let $[p_2p_3]$ be an edge of T and $[p_1p_2p_3]$ and $[p_1'p_2p_3]$ be the two adjacent faces. Let $[p_1p_2p_3p_4]$ and/or $[p_1p_2p_3q_4]$, and $[p_1'p_2p_3p_4']$ and/or $[p_1'p_2p_3q_4']$ be the face tetrahedra built for the faces $[p_1p_2p_3]$ and $[p_1'p_2p_3]$, respectively. Then if the edge $[p_2p_3]$ is non-convex, two pair tetrahedra need to be constructed. The first pair $[p_1''p_2p_3p_4]$ and $[p_1''p_2p_3p_4']$ are between $[p_1'p_2p_3p_4']$ and $[p_1p_2p_3p_4]$. The second pair $[q_1''p_2p_3q_4]$ and $[q_1''p_2p_3q_4']$ are between $[p_1'p_2p_3q_4']$ and $[p_1p_2p_3q_4]$. Here $p_1'' \in (p_4p_4')$ or is above (p_4, p_4'), say

$$p_1'' = \frac{(1-t)}{2}(p_2 + p_3) + \frac{t}{2}(p_4' + p_4), \quad t \geq 1$$

so that p_1'' is above plane $[p_1p_2p_3]$ and plane $[p_1'p_2p_3]$. Similarly, $q_1'' \in (q_4q_4')$ or is below (q_4, q_4'), say

$$q_1'' = \frac{(1-t)}{2}(p_2 + p_3) + \frac{t}{2}(q_4' + q_4), \quad t \geq 1$$

so that q_1'' is below plane $[p_1p_2p_3]$ and plane $[p_1'p_2p_3]$. 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.

2.7 Construction of a C^1 Interpolatory Surface using Cubic A-Patches

Having established a 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

$$f(p_i) = 0, \quad \nabla f(p_i) = n_i, \quad i = 1, 2, \ldots, k \tag{34}$$

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

2.7.1 The Construction of a Piecewise C^1 Cubic Function

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

(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 $F = [p_1p_2p_3]$ and $F' = [p_1'p_2p_3]$ be two adjacent non-convex faces. Then we have double tetrahedra $[p_1p_2p_3p_4]$ and $[p_1p_2p_3q_4]$ for F and double tetrahedra $[p_1'p_2p_3p_4']$ and $[p_1'p_2p_3q_4']$ for F'(see Figure 21). Let

$$V_1 = [p_1p_2p_3p_4], \quad V_2 = [p_1'p_2p_3p_4'], \quad W_1 = [p_1''p_2p_3p_4], \quad W_2 = [p_1''p_2p_3p_4']$$

$$V_1' = [p_1p_2p_3q_4], \quad V_2' = [p_1'p_2p_3q_4'], \quad W_1' = [q_1''p_2p_3q_4], \quad W_2' = [q_1''p_2p_3q_4']$$

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_λ^i, b_λ^i, c_λ^i, and d_λ^i, $i = 1, 2$, respectively. Now we shall determine these coefficients.

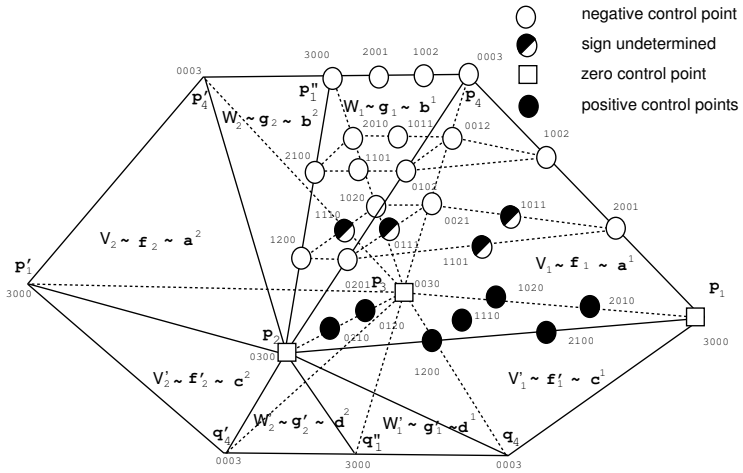


Figure 21: Adjacent tetrahedra, functions and control points for two non-convex adjacent faces

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 2.2):

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

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

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: We have, for $j = 2, 3$

$$\begin{array}{rclrcl} a^1_{2e_j+e_1} &=& \frac{1}{3}(p_1 - p_j)^T n_j, & a^2_{2e_j+e_1} &=& \frac{1}{3}(p'_1 - p_j)^T n_j \\ a^1_{2e_j+e_4} &=& \frac{1}{3}(p_4 - p_j)^T n_j, & a^2_{2e_j+e_4} &=& \frac{1}{3}(p'_4 - p_j)^T n_j, \\ b^1_{2e_j+e_1} &=& \frac{1}{3}(p''_1 - p_j)^T n_j, & d^1_{2e_j+e_1} &=& \frac{1}{3}(q''_1 - p_j)^T n_j, \\ c^1_{2e_j+e_4} &=& \frac{1}{3}(q_4 - p_j)^T n_j, & c^2_{2e_j+e_4} &=& \frac{1}{3}(q'_4 - p_j)^T n_j \end{array} \tag{35}$$

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 $[p_2p_3p_4]$ and $[p_2p_3p'_4]$. Suppose

$$\begin{array}{rclrcl} p''_1 &=& \beta^1_1 p_1 + \beta^1_2 p_2 + \beta^1_3 p_3 + \beta^1_4 p_4, & \beta^1_1 + \beta^1_2 + \beta^1_3 + \beta^1_4 = 1 \\ p''_1 &=& \beta^2_1 p'_1 + \beta^2_2 p_2 + \beta^2_3 p_3 + \beta^2_4 p'_4, & \beta^2_1 + \beta^2_2 + \beta^2_3 + \beta^2_4 = 1 \end{array} \tag{36}$$

   Then, the C^1 conditions require(see Lemma 2.2)

$$b^i_{1\lambda_2\lambda_3\lambda_4} = \beta^i_1 a^i_{1\lambda_2\lambda_3\lambda_4} + \beta^i_2 a^i_{0\lambda_2\lambda_3\lambda_4+0100} + \beta^i_3 a^i_{0\lambda_2\lambda_3\lambda_4+0010} + \beta^i_4 a^i_{0\lambda_2\lambda_3\lambda_4+0001} \tag{37}$$

   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 (37) for $\lambda_2\lambda_3\lambda_4 = 110$ will be treated later.

2. Interface at $[p_2p_3p''_1]$. Let

$$p''_1 = \mu_1 p_4 + \mu_2 p'_4 + \mu_3 p_2 + \mu_4 p_3, \quad \mu_1 + \mu_2 + \mu_3 + \mu_4 = 1 \tag{38}$$

then C^1 conditions require

$$b^i_{\lambda_1\lambda_2\lambda_3 0+1000} = \mu_1 b^1_{\lambda_1\lambda_2\lambda_3 1} + \mu_2 b^2_{\lambda_1\lambda_2\lambda_3 1} + \mu_3 b^i_{\lambda_1\lambda_2\lambda_3 0+0100} + \mu_4 b^i_{\lambda_1\lambda_2\lambda_3 0+0010} \tag{39}$$

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 (37).

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 (37)–(39) 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

$$d^i_{1110} = \gamma^i_1 a^i_{1110} + \gamma^i_2 a^i_{0210} + \gamma^i_3 a^i_{0120} + \gamma^i_4 c^i_{0111} \tag{40}$$

$$d^i_{1110} = \eta_1 c^1_{0111} + \eta_2 c^2_{0111} + \eta_3 a^i_{0210} + \eta_4 a^i_{0120} \tag{41}$$

where the coefficients γ_i and η_i are defined by

$$\begin{array}{rcll} q_1'' &=& \gamma^1_1 p_1 + \gamma^1_2 p_2 + \gamma^1_3 p_3 + \gamma^1_4 q_4, & \gamma^1_1 + \gamma^1_2 + \gamma^1_3 + \gamma^1_4 = 1 \\ q_1'' &=& \gamma^2_1 p_1' + \gamma^2_2 p_2 + \gamma^2_3 p_3 + \gamma^2_4 q_4', & \gamma^2_1 + \gamma^2_2 + \gamma^2_3 + \gamma^2_4 = 1 \\ q_1'' &=& \eta_1 q_4 + \eta_2 q_4' + \eta_3 p_2 + \eta_4 p_3, & \eta_1 + \eta_2 + \eta_3 + \eta_4 = 1 \end{array} \tag{42}$$

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

$$\begin{array}{rcll} q_4 &=& \alpha^1_1 p_1 + \alpha^1_2 p_2 + \alpha^1_3 p_3 + \alpha^1_4 p_4, & \alpha^1_1 + \alpha^1_2 + \alpha^1_3 + \alpha^1_4 = 1 \\ q_4' &=& \alpha^2_1 p_1' + \alpha^2_2 p_2 + \alpha^2_3 p_3 + \alpha^2_4 p_4', & \alpha^2_1 + \alpha^2_2 + \alpha^2_3 + \alpha^2_4 = 1 \end{array} \tag{43}$$

Then we have

$$c^i_{0111} = \alpha^i_1 a^i_{1110} + \alpha^i_2 a^i_{0210} + \alpha^i_3 a^i_{0120} + \alpha^i_4 a^i_{0111} \tag{44}$$

Now we treat the equations (37), (39), (40), (41) and (44). It follows from (37), (39), (40) and (41) that

$$\mu_1 a^1_{0111} + \mu_2 a^2_{0111} + \mu_3 a^i_{0210} + \mu_4 a^i_{0120} = \beta^i_1 a^i_{1110} + \beta^i_2 a^i_{0210} + \beta^i_3 a^i_{0120} + \beta^i_4 a^i_{0111} \tag{45}$$

$$\eta_1 c^1_{0111} + \eta_2 c^2_{0111} + \eta_3 a^i_{0210} + \eta_4 a^i_{0120} = \gamma^i_1 a^i_{1110} + \gamma^i_2 a^i_{0210} + \gamma^i_3 a^i_{0120} + \gamma^i_4 c^i_{0111} \tag{46}$$

Therefore, (44)–(46) 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 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

$$\begin{array}{rcll} p_4 &=& \theta^1_1 p_1 + \theta^1_2 p_2 + \theta^1_3 p_3 + \theta^1_4 p_1', & \theta^1_1 + \theta^1_2 + \theta^1_3 + \theta^1_4 = 1 \\ p_4' &=& \theta^2_1 p_1 + \theta^2_2 p_2 + \theta^2_3 p_3 + \theta^2_4 p_1', & \theta^2_1 + \theta^2_2 + \theta^2_3 + \theta^2_4 = 1 \\ q_4 &=& \vartheta^1_1 p_1 + \vartheta^1_2 p_2 + \vartheta^1_3 p_3 + \vartheta^1_4 p_1', & \vartheta^1_1 + \vartheta^1_2 + \vartheta^1_3 + \vartheta^1_4 = 1 \\ q_4' &=& \vartheta^2_1 p_1 + \vartheta^2_2 p_2 + \vartheta^2_3 p_3 + \vartheta^2_4 p_1', & \vartheta^2_1 + \vartheta^2_2 + \vartheta^2_3 + \vartheta^2_4 = 1 \end{array} \tag{47}$$

then we can derive from (45) and (46) that

$$a^i_{0111} = \theta^i_1 a^1_{1110} + \theta^i_2 a^i_{0210} + \theta^i_3 a^i_{0120} + \theta^i_4 a^2_{1110} \tag{48}$$

$$c^i_{0111} = \vartheta^i_1 a^1_{1110} + \vartheta^i_2 a^i_{0210} + \vartheta^i_3 a^i_{0120} + \vartheta^i_4 a^2_{1110} \tag{49}$$

If the edge $[p_2p_3]$ is nonnegative (or non-positive) convex, a^i_{1110}(or c^i_{1110}) are free and equation (49) (or (48)) is removed, since we do not need the function g_1' and g_2'(or g_1 and g_2). The free parameters a^i_{1110}(or c^i_{1110}) may be determined by approximating a quadratic(see §6 or [87]).

b. Both faces are convex.

(b1). Both faces are nonnegative (or non-positive) 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 non-positive 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 §6). In particular, $a^i_{1110} = 0$ (or $c^i_{1110} = 0$) if the face is zero convex.

3. We now need only (48) for unknowns a^1_{0111} and a^2_{0111} if the edge $[p_2p_3]$ is nonnegative convex, or (49) for unknowns c^1_{0111} and c^2_{0111} if the edge $[p_2p_3]$ is non-positive 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 need to only 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. $a^1_{1110} \geq 0$ and $c^2_{1110} \leq 0$ can be determined freely(see §6).

3. we need only have (44) for $i = 2$ and (48) 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. 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 non-positive convex.

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

3. For the treatment of equations (44)–(46), we need only have (44) for $i = 2$ and (48) for unknowns a^1_{0111}, a^2_{0111} and c^2_{0111} if the edge $[p_2p_3]$ is nonnegative convex, or solve (44) for $i = 2$ and (46) for unknowns c^1_{0111}, c^2_{0111} and a^2_{0111} if the edge $[p_2p_3]$ is non-positive convex(see Theorem 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 (45) and (46) are singular. However, the system (44)–(46) are still solvable(see Theorem 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 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 $[p'_1p_2p_3]$ into three subtriangles at their center points w and w' (a 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.

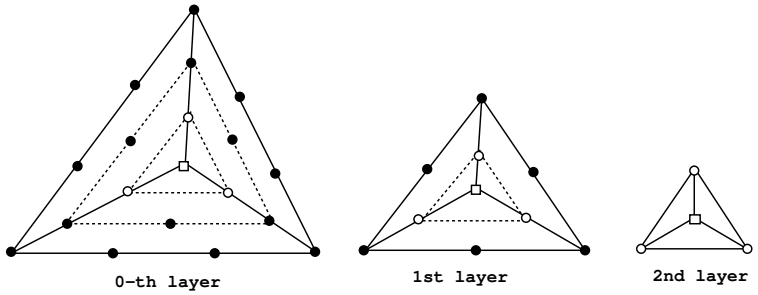


Figure 22: Control points of 0th, 1st and 2nd layers

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 22), 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 22), the control points labeled $\circ$ and $\square$ are similarly determined as the 0-th layer. For the 2nd layer (see Figure 22), the control points $\circ$ are arbitrarily chosen and $\square$ is determined by the coplanar condition. Finally, the 3rd layer coefficient is free.

2.7.2 The Solvability of the Related System

Concerning the solvability of the system (44)–(46) and its sub-system, we have the following result.

Theorem 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 21. (i) The system (44)–(46) has four independent equations. If (p_1, p_1', p_2, p_3) is affine independent, then (45) and (46) 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

$$x_k = \phi^k_1 x_i + \phi^k_2 x_j + \phi^k_3 a^1_{0210} + \phi^k_4 a^1_{0120}, \quad k \neq i, j \tag{50}$$

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

2.8 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 q_4 and q_4' lie(see Figure 21). Under this assumption, it follows from Definition 4.1 and equation (35) that, the control points on the edge, say a^i_{0210}, a^i_{0120} on edge $[p_2p_3]$(see Figure 21), are non-negative if the edge is non-negative convex, and non-positive if the edge is non-positive convex. Now we can divide all the control points into 7 groups called layers. The 0-th layer are the control points that are "on" the faces of T. The 1st layer is next to the 0-th layer but opposite to the normal direction, followed by the 2nd and 3rd layers. Next to the 0-th layer and on the same side as the normal, is the -1st layer, then the -2nd and -3rd layers. Now we show that, we can make all the control points on the 2nd and 3rd layers negative and the control points on the -2nd and -3rd layers positive.

For the face-tetrahedra, it is always possible to make the 2nd and 3rd layers control points negative, because these control points are free under the C^0 condition. For the control points on the edge-tetrahedra, it follows from (37) that the 2nd and 3rd layers control points can be negative only if the 2nd layer control points on the neighbor face-tetrahedra are small enough. This is achieved since β_4^i in (37) is positive(see the proof of Proposition 5.3 for details). Similarly, the control points on the $-$2nd and $-$3rd layers can be chosen to be positive. 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 vertices condition is obviously satisfied. If the surface contains the edge $[p_2p_3]$(see Figure 21), then since a^i_{1110}(or a^i_{0111}) is freely chosen, the smooth edges condition is easily satisfied(see the proof of Proposition 5.3). 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 5.2. *If the face $[p_1p_2p_3]$ is non-negative convex, then the control points can be determined so that the surface over V_1 is a three-sided 4-patch.*

Proposition 5.3. *If the edge $[p_2p_3]$ is non-negative convex, then the control points can be determined such that the surface over W_1 is a four-sided 14-23-patch.*

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 Clough-Tocher split). The coefficients are specified as before by regarding w as p_1(see Figure 21).

Proposition 5.4. *If the above subdivision procedure above 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.*

These propositions guarantee that the surface constructed are single sheeted.

2.9 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 normal is 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([11, 87]).

By least squares approximation of the coefficients of a quadric ([87]), one can derive that

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

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

$$a_\lambda = \frac{1}{3}\sum_{i=1}^{4} \lambda_i x_{\lambda - e_i}, \quad |\lambda| = 3, \quad \lambda_4 > 1 \tag{51}$$

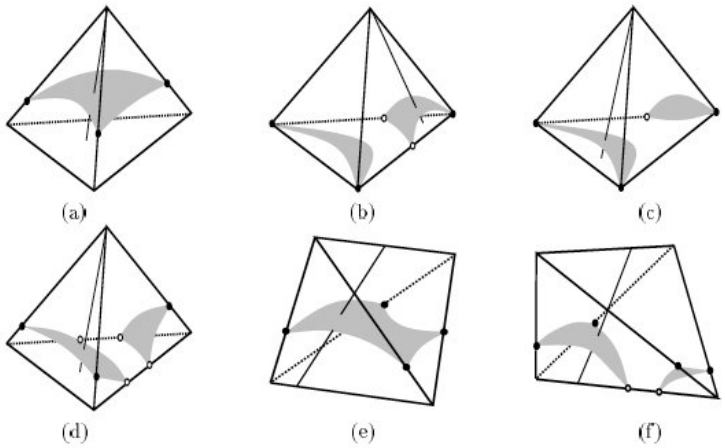


Figure 23: Interactive shape control.

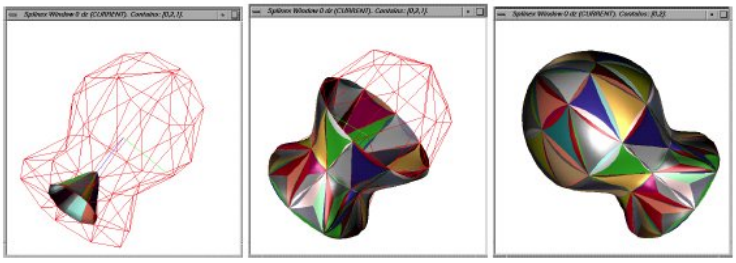


Figure 24: Interactive shape control.

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

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

In the same way, b_{2001} can be determined. Therefore, under the C^1 conditions, we can define two sets of control points $\{a^s_\lambda\}$ and $\{a^q_\lambda\}$ over V_1, where $\{a^s_\lambda\}$ is yielded from the single sheeted consideration(see Proposition 5.2–5.6), and $\{a^q_\lambda\}$ comes from approximating a simple(quadratic) surface. Note that the surface defined by $\{a^s_\lambda\}$ above may not be desirable in shape, while the surface defined by $\{a^q_\lambda\}$ 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 $\{a^{(i)}_\lambda\} = \{(1-t_i)a^q_\lambda + t_i a^s_\lambda\}, \quad i = 0, 1, \cdots, m$ selecting the single sheeted surface defined by $\{a^{(i)}_\lambda\}$ 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 Figures 13, 14, and 15 and in the figures below.

2.10 Curvilinear Patch Construction

2.10.1 Constructions of Wire Frames

For each edge of the triangulation, we shall construct a space curve and a normal function(for G^1 smoothness) such that the curve interpolates the end points of the edge and has the given normal, and the normal function interpolates the given normals and is orthogonal to the tangent of the curve. The normals at these vertices are defined by the original surface normals. However, at the

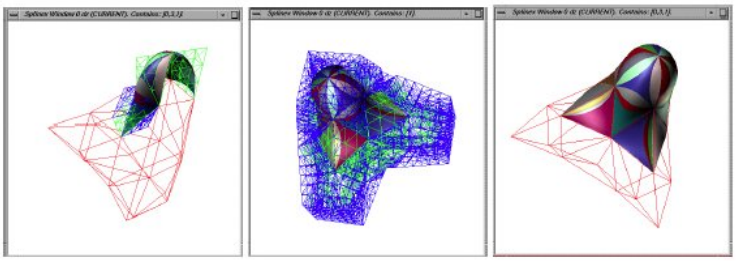


Figure 25: Interactive shape control.

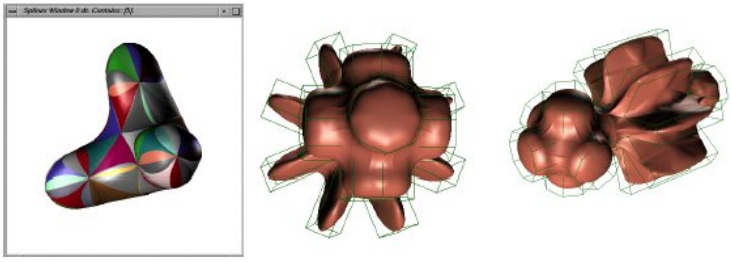


Figure 26: Interactive shape control.

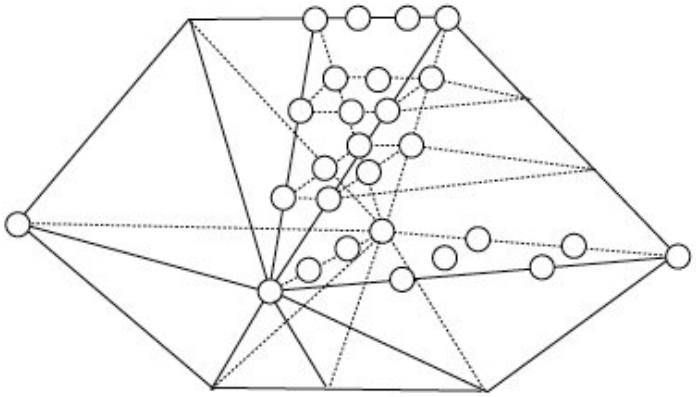


Figure 27: Interactive shape control.

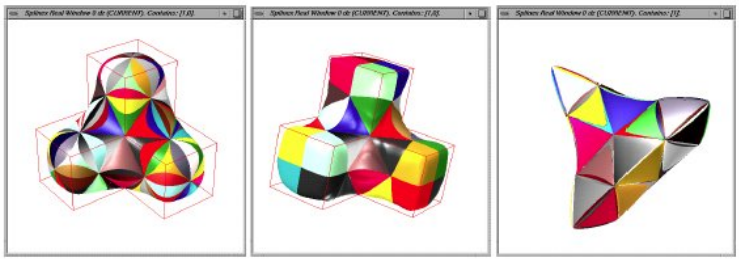


Figure 28: Interactive shape control.

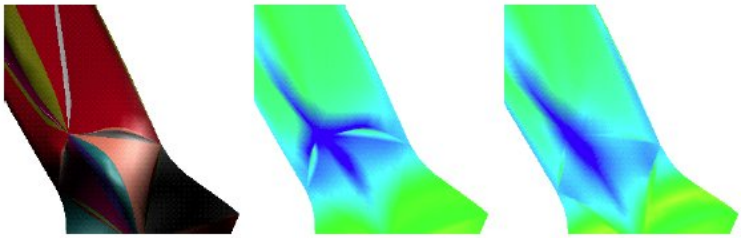


Figure 29: Interactive shape control.

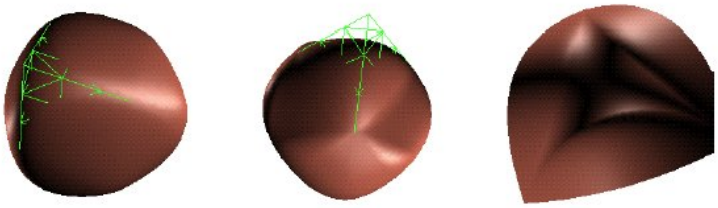


Figure 30: Interactive shape control.

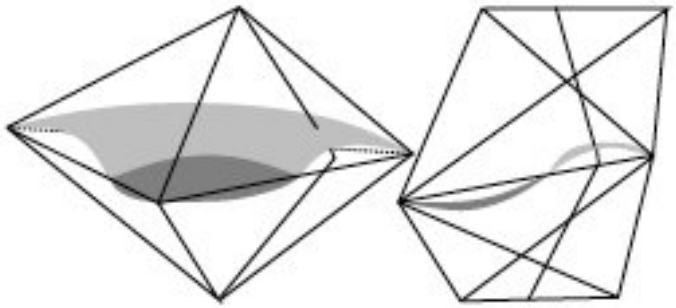


Figure 31: Interactive shape control.

singular points of the surface, the normals are not defined. Hence the space curve and the normal function will not have normal conditions there. In the following, the construction of the wire frame on an edge is considered in different cases according to having two normals, one normal and no normal:

Problem 1. *Given an edge* $[p_0, p_1]$ *and two normals* n_0 n_1 *(i) find a space curve* $C(t) = [X(t), Y(t), Z(t)]^T$ *such that*

$$C(0) = p_0, \qquad c(1) = p_1 \tag{52}$$

$$n_0^T C'(0) = 0, \qquad n_1^T C'(1) = 0 \tag{53}$$

and (ii) find a normal function $n(t)$ *on* $C(t)$ *such that*

$$n(0) = n_0, \qquad n(1) = n_1 \tag{54}$$

$$n^T(t) C'(t) \equiv 0, \qquad t \in [0, 1]. \tag{55}$$

This is the general case that happens when we handle the smooth part of the surface.

Problem 2. *Given an edge* $[p_0, p_1]$ *and one normal* n_0 *or* n_1*, (i) find a space curve* $C(t) = [X(t), Y(t), Z(t)]^T$ *such that (52) holds and*

$$n_0^T C'(0) = 0, \quad \text{or} \quad n_1^T C'(1) = 0 \tag{56}$$

and (ii) find a normal function $n(t)$ *on* $C(t)$ *such that*

$$n(0) = n_0, \quad \text{or} \quad n(1) = n_1 \tag{57}$$

and (55) holds.

This problem rises when an edge has a smooth end point and a singular end point.

Problem 3. *Given an edge* $[p_0, p_1]$*, (i) find a space curve* $C(t) = [X(t), Y(t), Z(t)]^T$ *such that (52) holds and (ii) find a normal function* $n(t)$ *on* $C(t)$ *such that (55) holds.*

This problem rises when an edge has two singular end-points.

Conic Space Curve With Minimum Energy Solution of Problem 1(i). Let $C(t) = At^2 + Bt + C$, with A, B, $C \in I\!R^3$. Then by (52), we have $C = p_0$, $A = p_1 - p_0 - B$. From (53), it follows that

$$[n_0, n_1]^T B = \begin{bmatrix} 0 \\ 2n_1^T(p_1 - p_0) \end{bmatrix} \tag{58}$$

A. If n_0, n_1 are linearly dependent, we must have

$$n_1^T(p_1 - p_0) = 0, \tag{59}$$

otherwise, equation (58) has no solution. If (59) is true, we take $A = 0$, $B = p_1 - p_0$, then equations (52)–(53) are satisfied.

B. If n_0, n_1 are linearly independent, then equation (58) has many solutions. Let $n_2 = n_0 \times n_1 / ||n_0 \times n_1||$, where $\times$ denotes cross product and $||\cdot||$ denotes the Euclidean norm. Then B can be expressed as $B = \alpha n_2 + [n_0, n_1]\beta$, $\beta \in I\!R^2$. From equation (58), we have

$$[n_0, n_1]^T [n_0, n_1]\beta = \begin{bmatrix} 0 \\ 2n_1^T(p_1 - p_0) \end{bmatrix} \tag{60}$$

That is, β is determined uniquely by (60) and α is arbitrary. For simplicity, denote

$$B = n_3 + \alpha n_2 \quad with \quad n_3 = [n_0, n_1]\beta$$
$$A = p_1 - p_0 - B = n_4 - \alpha n_2$$

Now we take α, such that the energy of the curve $C(t)$ is minimal: $\int_0^1 ||C'(t)||^2 dt = \min$. Since $C'(t) = 2At + B$,

$$\begin{aligned} \int_0^1 ||C'(t)||^2 dt &= \tfrac{4}{3}A^T A + 2A^T B + B^T B \\ &= \tfrac{1}{3}n_2^T n_2 \alpha^2 - \tfrac{2}{3}n_4^T n_2 \alpha + \tfrac{4}{3}n_4^T n_4 + 2n_4^T n_3 + n_3^T n_3 \end{aligned}$$

From $\frac{d}{d\alpha}\int_0^1 ||C'(t)||^2 dt = 0$, we get the α that minimize the energy: $\alpha = n_4^T n_2 = (p_1 - p_0)^T n_2$. Therefore

$$B = [n_0, n_1]\beta + n_2(p_1 - p_0)^T n_2$$

Lemma 5.1. *If n_0, n_1 are linearly independent, the space curve interpolation problem 1(i) by conic has unique minimum energy solution.*

Solution of Problem 2(i). Suppose we are given a normal n_0 at p_0, we shall construct $C(t) = At^2 + Bt + C$ such that the curve is in the plane $span(n_0, p_1 - p_0)$ spanned by n_0 and $p_1 - p_0$.

If $n_0^T(p_1 - p_0) \neq 0$, then as before, $C = p_0,\ A = p_1 - p_0 - B$ and $n_0^T B = 0$. Furthermore, B is in the plane $span(n_0, p_1 - p_0)$. These requirements lead to

$$B = \alpha n_2, \quad with \quad n_2 = n_0(p_1 - p_0)^T n_0 - (p_1 - p_0)n_0^T n_0$$

where α is parameter that is determined by minimizing the energy of the curve that leads to $\alpha = (p_1 - p_0)^T n_2 / n_2^T n_2$. If $n_0^T(p_1 - p_0) = 0$, we take $A = 0,\ B = p_1 - p_0,\ \ C = p_0$.

Similarly, if we are given a normal n_1 at p_1, then if $n_1^T(p_1 - p_0) \neq 0$, we have

$$B = 2(p_1 - p_0) + \alpha n_2, \ \ C = p_0, \quad A = p_1 - p_0 - B$$

with

$$\alpha = (p_0 - p_1)^T n_2 / n_2^T n_2, \quad n_2 = n_1(p_0 - p_1)^T n_1 - (p_0 - p_1)n_1^T n_1$$

Again, if $n_1^T(p_1 - p_0) = 0$, we take $A = 0,\ B = p_1 - p_0,\ \ C = p_0$.

Therefore, problem 2(i) always have conic solution.

Solution of Problem 3(i). Now we simply take $C(t)$ be a linear curve. That is $A = 0,\ B = p_1 - p_0,\ \ C = p_0$, Therefore, problem 3(i) always have linear solution.

Normal Function on Conic Space Curve Solution of Problem 1(ii). Let

$$n(t) = (Dt + E)/(1 + wt),$$

be the normal function, where $D,\ E \in I\!R^3,\ w \in I\!R$. Then by (54) we have

$$E = n_0, \qquad D = n_1 - n_0 + wn_1$$

Since the numerator of $n^T(t)C'(t)$ is a polynomial of degree 2 in t and $n^T(t)C'(t) = 0$ when $t = 0$ and $t = 1$, (55) holds if there is another point in [0,1] such that (55) holds. Take $t = \frac{1}{2}$ then by $n^T(\frac{1}{2})C'(\frac{1}{2}) = 0$, we have $1 + w = -\dfrac{n_0^T C'(1)}{n_1^T C'(0)}$. Since $C'(0) = B$, $C'(1) = 2A + B = 2(p_1 - p_0) - B$, it follows from (58) that

$$1 + w = -\frac{n_0^T(p_1 - p_0)}{n_1^T(p_1 - p_0)}$$

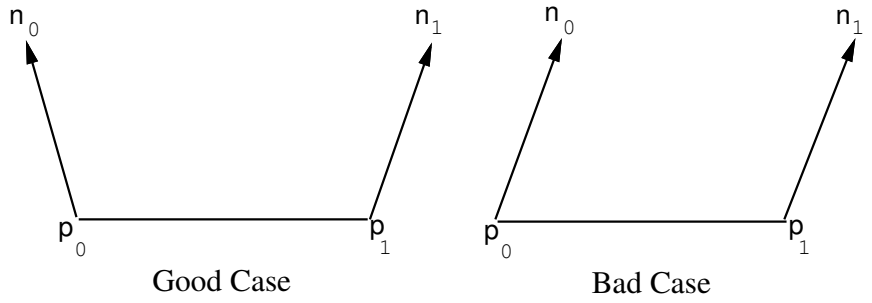


Figure 32: The normal pattern.

The good w should make $1 + w > 0$, i.e., $n(t)$ has no pole in [0,1]. This requires that $n_0^T(p_1 - p_0)$ and $n_1^T(p_1 - p_0)$ have opposite signs(see Fig. 2.1).

Lemma 5.2. *If $n_0^T(p_1 - p_0)/n_1^T(p_1 - p_0) < 0$, there exist unique linear rational normal function $n(t)$ on $C(t)$ such that (54) and (55) are satisfied.*

Solution of Problem 2(ii). Now we are given only one normal, say, n_0 at p_0. We specify a normal n_1 at p_1 by taking n_1 to be the normal of $C(t)$ at p_1 such that n_1 in the plane $span(n_0, p_1 - p_0)$ and point to the same side of the edge $[p_0, p_1]$ as n_0. This normal is uniquely defined and the condition in Lemma 2 is satisfied if $n_0^T(p_1 - p_0) \neq 0$. Hence the results above can be used. If $n_0^T(p_1 - p_0) = 0$, the constant normal function $n(t) = n_0$ satisfies the required condition.

Solution of Problem 3(ii). This case happen when the edge is on a singular curve. Now we can not expect that the constructed surface is smooth. Here the normal function will be constructed sereral times according to which triangle the edge belongs to. For a specified triangle that contains the edge, we take the normals at the end points of the edge to be the other edges curves normals at the corresponding points. Then the normal function is defined to be the linear function that has the two normals at the end points. Since an edge on singular curve of the original surface will be shared by several triangles, the normal function will be defined as many times as the triangles. However, the space curve on this edge is defined uniquely. Therefore, the surface constructed here is continuous but not smooth.

Cubic Space Curve With Minimum Energy Since conic space curve does not always exist for problem 1, we may use cubic space curves instead. Let $C(t) = At^3 + Bt^2 + Ct + D$. We determine A, B, C, $D \in I\!R^3$ such that

$$C(0) = p_0, \quad C(1) = p_1, \quad C(1/2) = p_2, \tag{61}$$

$$n_0^T C'(0) = 0, \quad n_1^T C'(1) = 0 \tag{62}$$

From (61)

$$D = p_0, \quad A = p_1 - p_0 - B - C, \quad B = 8(p_2 - p_0) - (p_1 - p_0) - 3C$$

So we need to determine C. It follows from (62) that

$$[n_0, n_1]^T C = \begin{bmatrix} 0 \\ 4n_1^T(2p_2 - p_1 - p_0) \end{bmatrix} \tag{63}$$

A. If $n_0,\ n_1$ are linearly dependent, we must choose p_2 such that

$$n_1^T p_2 = \frac{1}{2} n_1^T (p_1 + p_0) \tag{64}$$

Let n_3, n_4 satisfy $n_0^T n_3 = n_0^T n_4 = n_3^T n_4 = 0$ and $||n_3|| = ||n_4|| = 1$. Then the solution of (63) can be expressed as $C = \alpha n_3 + \beta n_4$. It is not difficult to calculate that when $\alpha = (4p_2 - p_1 - 3p_0)^T n_3$, $\beta = (4p_2 - p_1 - 3p_0)^T n_4$, the energy $\int_0^1 ||C'(t)||^2 dt$ of the curve $C(t)$ achieves minimal. We can take

$$n_4 = (2p_2 - p_1 - p_0)/||2p_2 - p_1 - p_0||, \quad n_3 = n_1 \times n_4$$

Then

$$\alpha = (p_1 - p_0)^T n_3, \quad \beta = 2||2p_2 - p_1 - p_0|| + (p_1 - p_0)^T n_4$$

Theorem 5.3. *If n_0, n_1 are linearly dependent, then if p_2 satisfies (64) and*

$$(p_2 - p_0)^T n_1 \neq 0, \quad \det[p_2 - p_1, p_2 - p_0, n_1] \neq 0$$

then the matrix $[A, B, C]$ is nonsingular.

Proof. Since $n_1^T n_3 = n_0^T n_4 = n_3^T n_4 = 0$, we have $\alpha = (p_1 - p_0)^T n_3 \neq 0$. Otherwise we are lead to $(p_2 - p_0)^T n_3 = 0$ and then $[p_2 - p_1, p_2 - p_0, n_1]^T n_3 = 0$. This contradicts the nonsingularity of $[p_2 - p_1, p_2 - p_0, n_1]$ and $n_3 \neq 0$. Hence $[A, B, C] \cong [p_2 - p_1, p_2 - p_0, C] \cong [n_4, p_2 - p_0, n_3]$. Since $n_4, p_2 - p_0$ and n_1 are linearly independent by the assumption of the theorem, n_3 can be expressed as $n_3 = an_4 + b(p_2 - p_0) + cn_1$. Then by timing n_3 on this equality we know that $b \neq 0$ and then by timing n_1 on the same equality we get $c \neq 0$. Therefore the matrix $[n_4, p_2 - p_0, n_3]$ is nonsingular.

B. If n_0, n_1 are linearly independent, then equation (63) has many solutions. Let $n_2 = n_0 \times n_1$, $||n_2|| = 1$. Then C can be expressed as $C = \alpha n_2 + [n_0, n_1]\beta$, $\beta \in I\!R^2$. where β is determined uniquely by $[n_0, n_1]^T [n_0, n_1]\beta = \begin{bmatrix} 0 \\ 4n_1^T(2p_2 - p_1 - p_0) \end{bmatrix}$, and α, which make the energy of the curve $C(t)$ to be minimal, is $\alpha = (4p_2 - p_1 - 3p_0)^T n_2$.

Normal Function on Cubic Space Curve Let the normal function $n(t)$ be in the form $n(t) = Et^2 + Ft + G$ that satisfies

$$n(0) = n_0, \quad n(1) = n_1, \quad n^T(t)C'(t) \equiv 0, \quad t \in [0, 1]. \tag{65}$$

Since $n^T(t)C'(t)$ is a polynomial of degree 4 and it vanishes at $t = 0$ and $t = 1$, we need to choose three points, say $t = 1/4, 1/2, 3/4$, such that (65) holds. Since $G = n_0$, $E = n_1 - n_0 - F$, we have, for unknow vector F, the following equations

$$\begin{cases} C'(1/4)^T(1/16(n_1 - n_0 - F) + 1/4F + n_0) = 0 \\ C'(1/2)^T(1/4(n_1 - n_0 - F) + 1/2F + n_0) = 0 \\ C'(3/4)^T(9/16(n_1 - n_0 - F) + 3/4F + n_0) = 0 \end{cases}$$

The coefficient matrix of this equation is equivalent to the matrix $[A, B, C]$. Hence the equation has unique solution iff the matrix $[A, B, C]$ is invertible. If the the matrix is singular, one can solve the equation by the least square approximation.

2.10.2 Parametric Surface Patches Interpolation

Suppose we are given a triangular wire frame $C_i(t)$, $i = 0, 1, 2$ and furthermore normal functions $n_i(t)$, $i = 0, 1, 2$, such that $p_1 = C_0(0) = C_2(1)$, $p_2 = C_0(1) = C_1(0)$, $p_3 = C_1(1) = C_2(0)$, and $C_i^T(t)n_i(t) = 0$. We want to construct a parametric patch $X(u, v) = [x(u, v)\ y(u, v)\ z(u, v)]$. that covers the given wire frame (for G^0 continuity) and further has the given normal (for G^1 continuity) on the wire frame, where u, v, w are barycentric coordinate systems with $w = 1 - u - v$.

2.10.3 G^0 Interpolation

Covering conic wire frame

Since the degree of the space curve is 2, we choose one more point on each edge of the triangle in addition to the vertices. Let $p_{i+4} = C_i(\frac{1}{2}), i = 0, 1, 2$. Then we have six points $(p_i, i = 1, \ldots, 6)$. Now find a polynomial P_2 of degree 2 such that

$$\begin{array}{lll} P_2(V_i) = p_i, & & i = 1, 2, 3 \\ P_2(\frac{V_1+V_2}{2}) & = & p_4 \\ P_2(\frac{V_2+V_3}{2}) & = & p_5 \\ P_2(\frac{V_3+V_1}{2}) & = & p_6 \end{array} \tag{66}$$

The coefficient matrix of (66) $\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{4} & 0 & \frac{1}{2} & 0 & 0 & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{2} & 0 & \frac{1}{4} & 0 & 0 \end{bmatrix}$ is nonsingular, hence equation (66) has an unique solution.

Covering cubic wire frame

Since the degree of the space curve now is 3, we need to choose two more points on each edge of the triangle in addition to the vertices. Let $p_{i+4} = C_i(\frac{1}{3})$, $p_{i+7} = C_i(\frac{2}{3}), i = 0, 1, 2$. Then we have nine points $(p_i, i = 1, \ldots, 9)$. Now find a polynomial P_3 of degree 3 such that

$$\begin{array}{rcl|rcl} P_3(V_i) & = & p_i, \quad i = 1, 2, 3 & & & \\ P_3(\frac{V_1+2V_2}{3}) & = & p_4, & P_3(\frac{2V_1+V_2}{3}) & = & p_7 \\ P_3(\frac{V_2+2V_3}{3}) & = & p_5, & P_3(\frac{2V_2+V_3}{3}) & = & p_8 \\ P_3(\frac{V_3+2V_1}{3}) & = & p_6, & P_3(\frac{2V_3+V_1}{3}) & = & p_9 \end{array}$$

Since P_3 has ten coefficients, one more equation $P_3(V) = p$ is needed, where $V = (u, v, w)$ is any given point inside the triangle and $p \in I\!R^3$ that can be used to control the shape of the patch.

It is easy to check that the resulted system of linear equations has nonsingular coefficient matrix for any V inside the triangle.

G^1 Interpolation We now want to construct the surface patch such that it has the given normal on the wire frame. The composite surface constructed in this way is therefore tangent plane continuous (G^1).

Covering conic wire frame

The patch is defined in the following form

$$P(u, v, w) = P_2(u, v, w) + uvwP_1(u, v, w)$$

where P_2 is a BB-form polynomial of degree 2 that covers the conic wire frame. P_1 is a rational function in the form of

$$P_1(u, v, w) = \frac{uvP_w + uwP_v + vwP_u}{uv + uw + vw} \tag{67}$$

and P_u, P_v and P_w are polynomial of degree 0 to be determined such that G^1 continuity is guaranteed.

Since variable t in $C_i(t)$ and $n_i(t)$ can be changed into $(1 - t)$, we may assume, without loss of generality, that t is increased from 0 to 1 when point goes from (0,1,0) to (0,0,1), from (0,0,1)

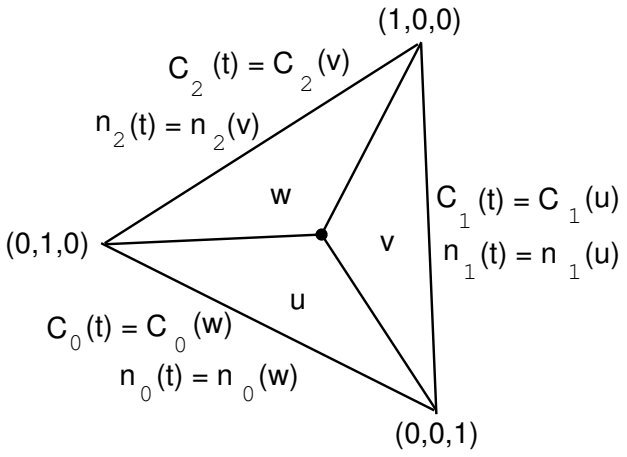


Figure 33: The split triangular wireframe.

to (1,0,0) and from (1,0,0) to (0,1,0) (see Figure 33). Hence variable of C_0 and n_0 is w, C_1 and n_1 is u, C_2 and n_2 is v. Now we determine P_1 such that tangent plane determined by the span of $(\frac{\partial P}{\partial u}, \frac{\partial P}{\partial v})$ are orthogonal to normal functions $n_i(t)$ on $C_i(t)$

A. when $u = 0$

$$\frac{\partial P}{\partial u} = \frac{\partial P_2}{\partial u} - \frac{\partial P_2}{\partial w} + vwP_1 \tag{68}$$

$$\frac{\partial P}{\partial v} = \frac{\partial P_2}{\partial v} - \frac{\partial P_2}{\partial w} = \frac{d}{dw}(P_2(0, 1-w, w) = -C_0'(w) \tag{69}$$

B. when $v = 0$

$$\frac{\partial P}{\partial u} = \frac{\partial P_2}{\partial u} - \frac{\partial P_2}{\partial w} = \frac{d}{du}P_2(u, 0, 1-u) = C_1'(u) \tag{70}$$

$$\frac{\partial P}{\partial v} = \frac{\partial P_2}{\partial v} - \frac{\partial P_2}{\partial w} + uwP_1 \tag{71}$$

C. when $w = 0$

$$\frac{\partial P}{\partial u} = \frac{\partial P_2}{\partial u} - \frac{\partial P_2}{\partial w} - uvP_1 \tag{72}$$

$$\frac{\partial P}{\partial v} = \frac{\partial P_2}{\partial v} - \frac{\partial P_2}{\partial w} - uvP_1 \tag{73}$$

and further

$$\frac{\partial P}{\partial v} - \frac{\partial P}{\partial u} = \frac{\partial P_2}{\partial v} - \frac{\partial P_2}{\partial u} = \frac{d}{dv}(P_2(1-v, v, 0)) = C_2'(v) \tag{74}$$

By the definition of $n_i(t)$, we have $C_i'(t)^T n_i(t) = 0$, so we need to have by (68), (71) and (72)

$$\left(\frac{\partial P_2}{\partial u} - \frac{\partial P_2}{\partial w} + (1-w)wP_u\right)^T n_0(w) = 0 \qquad u = 0 \tag{75}$$

$$\left(\frac{\partial P_2}{\partial v} - \frac{\partial P_2}{\partial w} + u(1-u)P_v\right)^T n_1(u) = 0 \qquad v = 0 \tag{76}$$

$$\left(\frac{\partial P_2}{\partial u} - \frac{\partial P_2}{\partial w} - (1-v)vP_w\right)^T n_2(v) = 0 \qquad w = 0 \tag{77}$$

Consider firstly the left side of Equation (75). Since

(i) for $u = w = 0$, $v = 1$, it follows from (69) and (74) that

$$\begin{aligned} \frac{\partial P_2}{\partial u} - \frac{\partial P_2}{\partial w} &= \left(\frac{\partial P_2}{\partial v} - \frac{\partial P_2}{\partial w}\right) - \left(\frac{\partial P_2}{\partial v} - \frac{\partial P_2}{\partial u}\right) \\ &= -C_0'(0) - C_2'(1) \end{aligned}$$

(ii) for $u = v = 0$, $w = 1$ it follows from (70) that $\frac{\partial P_2}{\partial u} - \frac{\partial P_2}{\partial w} = C_1'(0)$, (75) holds for $w = 0$ and $w = 1$. Similarly, we can show that (76) and (77) hold for end points of the unit interval.

Therefore, (75)–(77) is equivalent to

$$\left\{ \begin{array}{rcl} P_u(t)^T n_0(t) &=& a \\ P_v(t)^T n_1(t) &=& b \\ P_w(t)^T n_2(t) &=& c \end{array} \right. \tag{78}$$

where a, b, and c are constants. The left side of (78) are polynomials of degree 1. At point $u = v = w = \frac{1}{3}$, we give a vector p of P_1 that controls the shape of the patch. Then we have the following equation.

$$\begin{bmatrix} n_0^T(0) & & & \\ n_0^T(1) & & & \\ & n_1^T(0) & & \\ & n_1^T(1) & & \\ & & n_2^T(0) & \\ & & n_2^T(1) & \\ I & I & I & \end{bmatrix} \begin{bmatrix} P_u \\ \\ P_v \\ \\ P_w \end{bmatrix} = \begin{bmatrix} a \\ a \\ b \\ b \\ c \\ c \\ 3p \end{bmatrix}$$

In order to study the singularity of the coefficient matrix, we assume $n_0^T(1) = n_1^T(0)$, $n_1^T(1) = n_2^T(0)$, $n_1^T(1) = n_0^T(0)$. Then by some elimination we know that the coefficient matrix is nonsingular if the matrix $[n_0, n_1, n_2]$ is nonsingular. Therefore we have

Lemma 6.1. If $[n_0, n_1, n_2]$ is nonsingular, the G^1 interpolation problem with one control point has an unique solution.

Note. One way to choose the control value p at the middle point is to take $p = 0$.

The condition that the matrix $[n_0, n_1, n_2]$ is nonsingular in Lemma 4.1 can be relaxed as that the vectors n_0, n_1 and n_2 are pairwise independent. In this case, equation (78) can be solved separately with one degree of freedom left that can be used to control the shape at point $u = v = w = \frac{1}{3}$ in the least square sense.

Covering cubic wire frame

The patch now is defined in the following form

$$P(u, v, w) = P_3(u, v, w) + uvwP_1(u, v, w)$$

where P_3 is a BB-form polynomial of degree 3 that covers the cubic wire frame. P_1 is a rational function in the same form of (67).

Parallel to the case of covering conic wire frame, we are lead to a system(see (78)) of equations

$$\left\{ \begin{array}{rcl} P_u^T n_0(t) &=& a(t) \\ P_v^T n_1(t) &=& b(t) \\ P_w^T n_2(t) &=& c(t) \end{array} \right.$$

where the normal functions $n_i(t)$ and $a(t)$, $b(t)$, and $c(t)$ are polynomials of degree 2. In fact, this system can be solved separately for P_u, P_v and P_w. Each equation has unique solutions iff the coefficient vectors of the corresponding normal function are linearly independent. In practice, we can solve these equations by least square approximations.

2.11 C^1 Modeling with Hybrid Multiple-Sided A-patches

We consider the problem of constructing a smooth interpolatory surface from a surface discretization $\mathcal{L}$ by piecewise implicit surface patches. The discretization $\mathcal{L}$ of the surface consists of triangles, quadrilaterals and pentagons. The constructed surface passes through the vertices of the discretization and has the given normals at the vertices. This solution uses piecewise rational functions defined on a hull that consists of tetrahedra and pyramids

Several approaches to using implicit surface representation in modeling geometric objects have been proposed in papers (see for examples, ([12], [18], [87], [122], [154], [213]). Most of the schemes use various simplicial hulls over surface triangulation and polynomial functions (see [13], [18], [87], [122], [124]). They in general consist of the following three steps: **a**. Generate a normal for each vertex of $\mathcal{L}$ which will also be the normal of the constructed smooth surface at the vertex. **b**. Build a surrounding simplicial hull $\sum$ (consisting of a series of tetrahedra) of the triangulation. **c**. Construct a piecewise trivariate polynomial F within that simplicial hull, and use the zero contour of F to represent the surface. Dahmen [85] first proposed an approach for constructing a simplicial hull of $\mathcal{L}$. In this approach, for each face $[p_ip_jp_k]$ of $\mathcal{L}$, two points u_{ijk} and v_{ijk} off each side of the face are chosen and two tetrahedra $[p_ip_jp_ku_{ijk}]$ and $[p_ip_jp_ku_{ijk}]$ (called face tetrahedra) are constructed. For each edge of $\mathcal{L}$, two tetrahedra (called edge tetrahedra) are formed that blend the neighboring face tetrahedra. The collection of these tetrahedra contains the tangent plane near the vertices and have no self-intersection. Since such simplicial hulls are nontrivial to construct for arbitrary triangulation, several improvements have been made in later publications to overcome the difficulties (see [18], [87], [122], [124]). For the construction of the surface within $\sum$, Dahmen [85] used six quadric patches for each face tetrahedron and four quadric patches for each edge tetrahedron. Guo [122] uses a Clough-Tocher split to subdivide each face tetrahedron of the simplicial hull, hence utilizing six cubic patches per face of $\mathcal{L}$. The edge tetrahedra are subdivided into two. Dahmen and Thamm-Schaar [87] do not split the face tetrahedra, but the edge tetrahedra is split. All of these papers provided heuristics to overcome the multiple-sheeted and singularity problem of the implicit patches. Since the multi-sheeted property may cause the constructed surface to be disconnected, Bajaj et al [18] constructed A-patches that were guaranteed to be nonsingular, connected and single sheeted within each tetrahedron. Xu et al [238] use rational functions in constructing F so that the edge patches and convex face patches do not need to be split.

All the works mentioned above construct smooth implicit surface patches for the given surface triangulation. The more general parametric spline fitting problem of constructing a mesh of finite elements that interpolate or approximate multivariate data is discussed in [22]. One approach to creating multi–sided patches has been by introducing base points into rational parametric functions. Base points are parameter values for which the homogeneous coordinates (x,y,z,w) are mapped to (0,0,0,0) by the rational parameterization. Gregory's patch is defined using a special collection of rational basis functions that evaluate to $0/0$ at vertices of the parametric domain and thus introduce base points in the resulting parameterization. Warren uses base points to create parameterizations of four-, five-, and six–sided surface patches using rational Bézier surfaces defined over triangular domains. Setting a triangle of weights to zero at one corner of the domain triangle produces a four–sided patch that is the image of the domain triangle. [155, 156] present generalizations of biquadratic and bicubic B-spline surfaces that are capable representing surfaces of arbitrary topology by placing restrictions on the connectivity of the control mesh, relaxing C^1 continuity to G^1 (geometric) continuity, and allowing n-sided finite elements. This generalized view considers the spline surface to be a collection of possibly rational polynomial maps from independent n-sided polygonal domains, whose union possesses continuity of some number of geometric invariants, such as tangent planes. This more general view allows patches to be sewn together to describe free form surfaces in more complex ways.

We shall construct 3,4,5 sided A-patches from rational functions. That is F is a piecewise

rational function defined on a hull that consists of tetrahedra and pyramids. The construction method of F on edge tetrahedra is the same as our earlier scheme [238]. However, the method of face patch construction is new. Although the modeling function F is rational in form, it is evaluated as easy as cubic (see section 4 and 5). Furthermore, the surface constructed has plane recovery property. That is, if the normals at the vertices of a face are perpendicular to the face, then the surface coincides with the face. Having this feature is important since many geometric objects have planar portion. Even further, the surface constructed could recover quadratics.

2.11.1 Bases

Let p_1 and p_2 be two different points in $\mathbb{R}^3$. We use $[p_1p_2]$ to denote the line segment that has end-points p_1 and p_2. Let $p_1, \cdots, p_k$ be k ($k > 3$) different points in $\mathbb{R}^3$. Then we use $\langle p_1 \cdots p_k \rangle$ to denote the polygon consisting of $[p_1p_2], \cdots, [p_{k-1}p_k]$, $[p_kp_1]$. Further, we use the following notations

$$
\begin{aligned}
&[p_1p_2p_3] = \{p = \alpha_1p_1 + \alpha_2p_2 + \alpha_3p_3 : \ \alpha_i \in [0,1], \alpha_1 + \alpha_2 + \alpha_3 = 1\} \\
&[p_1p_2p_3p_4] = \{p = \alpha_1p_1 + \alpha_2p_2 + \alpha_3p_3 + \alpha_4p_4 : \ \alpha_i \in [0,1], \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 = 1\} \\
&(p_1p_2p_3p_4) = \{p = t[sp_1 + (1-s)p_2] + (1-t)[sp_3 + (1-s)p_4] : \ (s,t) \in [0,1]^2\} \\
&[p_0p_1 \cdots p_4] = \{p = up_0 + (1-u)q : \ u \in [0,1], \ q \in (p_1p_2p_3p_4)\}
\end{aligned}
$$

That is, $[p_1p_2p_3]$ is a triangle, $(p_1p_2p_3p_4)$ is ruled surface, $[p_1p_2p_3p_4]$ is a tetrahedron and $[p_0p_1p_2p_3p_4]$ is a pyramid.

1. BB Form on Tetrahedra. The trivariate polynomials defined in a tetrahedron are expressed in Bernstein-Bézier (BB) form. Let $q_1, q_2, q_3, q_4 \in \mathbb{R}^3$ be affine independent. Then any $p \in \mathbb{R}^3$ could be written as

$$
p = (x, y, z)^T = \sum_{i=1}^{4} \alpha_i q_i, \qquad \sum_{i=1}^{4} \alpha_i = 1 \tag{79}
$$

$\alpha = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)^T$ is the barycentric coordinate of p. Any polynomial $F_T(p)$ of degree n then can be expressed as BB form over $[q_1q_2q_3q_4]$ as

$$
F_T(p) = \sum_{i+j+k+l=n} a_{ijkl} \ B^n_{ijkl}(\alpha) \tag{80}
$$

where $B^n_{ijkl}(\alpha) = \frac{n!}{!j!k!l!} \ \alpha_1^i\alpha_2^j\alpha_3^k\alpha_4^l$ is the Bernstein polynomial. Lemma 2.1 in the following gives conditions of C^1 join of two BB form polynomials defined on two adjacent tetrahedra.

Lemma 2.22. ([108]) *Let $F_T(p) = \sum_{i+j+k+l=n} \ a_{ijkl} \ B^n_{ijkl}(\alpha)$ and $G_T(p) = \sum_{i+j+k+l=n} \ b_{ijkl} \ B^n_{ijkl}(\alpha)$ be two polynomials defined on two tetrahedra $[q_1q_2q_3q_4]$ and $[q_1'q_2q_3q_4]$, respectively. Then*
(i) F_T and G_T are C^0 at the face $[q_2q_3q_4]$ iff $a_{0jkl} = b_{0jkl}$ for any $j + k + l = n$
(ii) F_T and G_T are C^1 at the face $[q_2q_3q_4]$ iff they are C^0 and

$$
b_{1,j,k,l} = \beta_1 a_{1,j,k,l} + \beta_2 a_{0,j+1,k,l} + \beta_3 a_{0,j,k+1,l} + \beta_4 a_{0,j,k,l+1}, \quad j + k + l = n - 1 \tag{81}
$$

where $\beta = (\beta_1, \beta_2, \beta_3, \beta_4)^T$ is defined by the relation $q_1' = \beta_1 q_1 + \beta_2 q_2 + \beta_3 q_3 + \beta_4 q_4, \quad |\beta| = 1$.

Degree Elevation ([108]). A polynomial $\sum_{i+j+k+l=n-1} \ a_{ijkl} \ B^{n-1}_{ijkl}(\alpha)$ of degree $n-1$ could be written as a polynomial $\sum_{i+j+k+l=n} \ b_{ijkl} \ B^n_{ijkl}(\alpha)$ of degree n with

$$
b_{ijkl} = \frac{i}{n} a_{i-1,jkl} + \frac{j}{n} a_{i,j-1,kl} + \frac{k}{n} a_{ij,k-1,l} + \frac{l}{n} a_{ijk,l-1} \tag{82}
$$

2. BB Form on Pyramid. The BB form polynomial of degree n on a pyramid $[p_0p_1p_2p_3p_4]$ is defined by

$$F_P(x,y,z) := f(u,s,t) := \sum_{i=0}^{n}\sum_{j=0}^{n-i}\sum_{k=0}^{n-i} b_{ijk}B_i^n(u)B_j^{n-i}(s)B_k^{n-i}(t) \tag{83}$$

where $(x,y,z) \in [p_0p_1p_2p_3p_4]$ and $(u,s,t) \in [0,1]^3$ are related by

$$(x,y,z)^T = up_0 + (1-u)\{t[sp_1 + (1-s)p_2] + (1-t)[sp_3 + (1-s)p_4]\}. \tag{84}$$

Since transform (84) is not linear, a polynomial in (u,s,t) may not be written as a polynomial in (x,y,z). However, a polynomial in (x,y,z) of total degree n could always be written as the same degree polynomial in (u,s,t). Since transform (84) is not invertible at the point p_0, where $(1,s,t)$ map to p_0 for any (s,t), the polynomial F_P may not be smooth at the point p_0 even though F_P is any time differentiable in the local system (u,s,t). Fortunately, we do not use the smoothness of F_P at p_0. The following theorem gives the conditions of C^k join between two polynomials that are defined on an adjacent tetrahedron and pyramid , respectively.

Theorem 2.23. *Let $F_T(x,y,z)$ and $F_P(x,y,z)$ be defined as (80) and (83) on $[q_1q_2q_3q_4]$ and $[p_0p_1p_2p_3p_4]$, respectively, with $q_2 = p_2, q_3 = p_4$ and $q_4 = p_0$. Then F_T and F_P are C^K join on the interface $[q_2q_3q_4] \setminus \{q_4\}$ if and only if*

$$b_{ijk} = \sum_{\substack{i_1+i_2+j_1+j_2+j_3+k_1+k_2+k_3+l_1+l_2 = n-i \\ i_1+i_2+j_1+j_2+k_1+k_2+l_1+l_2 = j \\ i_1+j_1+k_1+l_1+j_3 = k}} \frac{C_{j_3,k-j_3}^{k} C_{k_3,i_2+j_2+k_2+l_2}^{i_2+j_2+k_2+l_2+k_3}}{C_{j_3+k_3,n-i-j_3-k_3}^{n-i}}$$
$$*a_{i_1+i_2,j_1+j_2+j_3,k_1+k_2+k_3,l_1+l_2+i} B_{i_1,j_1,k_1,l_1}^{k-j_3}(a) B_{i_2,j_2,k_2,l_2}^{j-k+j_3}(b) \tag{85}$$

for $j = 0,\cdots,K$, $i = 0,\cdots,n-j$, and $k = 0,\cdots,n-i$, where $a = (a_1,a_2,a_3,a_4)^T$ and $b = (b_1,b_2,b_3,b_4)^T$ are defined by

$$p_1 = \sum_{i=1}^{4} a_i q_i, \quad \sum_{i=1}^{4} a_i = 1, \quad p_3 = \sum_{i=1}^{4} b_i q_i, \quad \sum_{i=1}^{4} b_i = 1 \tag{86}$$

Taking $K = 1$ in Theorem 2.23, we have the following corollary.

Corollary 2.24. *F_T and F_P that are defined in Theorem 2.23 are C^1 at $[q_2q_3q_4] \setminus \{q_4\}$ iff*

$$\begin{aligned} b_{i,0,k} &= a_{0,k,n-i-k,i}, \quad i = 0,\cdots,n, \quad k = 0,\cdots,n-i \\ b_{i,1,k} &= \frac{k}{n-i}\sum_{l=1}^{4} a_l a_{e_l+(0,k-1,n-i-k,i)} + \frac{n-i-k}{n-i}\sum_{l=1}^{4} b_l a_{e_l+(0,k,n-i-k-1,i)}, \\ &\qquad i = 0,\cdots,n-1, \quad k = 0,\cdots,n-i \end{aligned} \tag{87}$$

where $e_l \in \mathbb{R}^4$ is the unit vector in the l-th direction, $l = 1,\cdots,4$.

Degree Elevation. A polynomial $\sum_{i=0}^{n-1}\sum_{j=0}^{n-i-1}\sum_{k=0}^{n-i-1} a_{ijk}\ B_i^{n-1}(u)B_j^{n-i-1}(s)B_k^{n-i-1}(t)$ of degree $n-1$ could be written as a polynomial $\sum_{i=0}^{n}\sum_{j=0}^{n-i}\sum_{k=0}^{n-i} b_{ijk}\ B_i^n(u)B_j^{n-i}(s)B_k^{n-i}(t)$ of degree n with

$$\begin{aligned} b_{ijk} = \frac{i}{n}a_{i-1,jk} &+ \frac{jk}{n(n-i)}a_{i,j-1,k-1} + \frac{(n-i-k)j}{n(n-i)}a_{i,j-1,k} \\ &+ \frac{(n-i-j)k}{n(n-i)}a_{ij,k-1} + \frac{(n-i-j)(n-i-k)}{n(n-i)}a_{ijk} \end{aligned} \tag{88}$$

3. C^1 of Cubics around an Edge. Let $q_1, q_2, \cdots, q_k$ be the given points around a line segment $[p_2p_3]$ such that q_{i-1} and q_{i+1} lie on different sides of the plane $[q_ip_2p_3]$ and all the tetrahedra $[q_iq_{i+1}p_2p_3]$ enclose the edge $[p_2p_3]$ as interior Hence the five points $q_{i-1}, q_i, q_{i+1}, p_2, p_3$ are related by either

$$q_i = \alpha_1^i q_{i-1} + \alpha_2^i q_{i+1} + \alpha_3^i p_2 + \alpha_4^i p_3, \quad \sum_{j=0}^{4} \alpha_j^i = 1 \tag{89}$$

if $q_{i-1}, q_{i+1}, p_2, p_3$ are affine independent, or

$$0 = \alpha_1^i q_{i-1} + \alpha_2^i q_{i+1} + \alpha_3^i p_2 + \alpha_4^i p_3, \quad \sum_{j=0}^{4} \alpha_j^i = 0 \tag{90}$$

if $q_{i-1}, q_{i+1}, p_2, p_3$ are affine dependent, where $\alpha_1^i \neq 0$ and $\alpha_2^i \neq 0$. Let F_i be the cubic polynomial in BB-form on the tetrahedron $[q_iq_{i+1}p_2p_3]$ that satisfy C^0 condition. Let x_i be the Bézier coefficients on the center of $[q_ip_2p_3]$. Then the C^1 condition at the interface $[q_ip_2p_3]$ is either

$$x_i = \alpha_1^i x_{i-1} + \alpha_2^i x_{i+1} + \alpha_3^i b_2 + \alpha_4^i b_3 \tag{91}$$

if $q_{i-1}, q_{i+1}, p_2, p_3$ are affine independent, or

$$0 = \alpha_1^i x_{i-1} + \alpha_2^i x_{i+1} + \alpha_3^i b_2 + \alpha_4^i b_3 \tag{92}$$

otherwise, where b_2 and b_3 are the Bézier coefficients on the edge $[p_2p_3]$. Then we have

Theorem 2.25. *If the tetrahedra around the edge $[p_2p_3]$ enclose the edge, then (i) there are $k-2$ independent equations among the k C^1 conditions (91)–(92) with k unknowns around the edge; (ii) any two adjacent equations of them can be deleted; (iii) any two unknowns x_m, x_n can be chosen as free parameters if q_m, q_n, p_2, p_3 are affine independent.*

For the open case we can treat it as closed case with the first and last equations being deleted. Hence we have by Theorem 2.25 that

Corollary 2.26. *If the tetrahedra around the edge $[p_2p_3]$ do not enclose the edge, then (i) the $k-2$ C^1 conditions (91)–(92) with k unknowns around the edge is independent; (ii) any two unknowns x_m, x_n can be chosen as free parameters if q_m, q_n, p_2, p_3 are affine independent.*

Theorem 2.27. *Let $\Delta = \cup_{i=1}^k [q_iq_{i+1}p_2p_3]$. Let $S_3^1(\Delta)$ be the collection of functions that are C^1 on Δ and cubics on each tetrahedron of Δ. Then*

$$\dim S_3^1(\Delta) = 4k + 10$$

if $[q_ip_2p_3]$, $i = 1, \cdots, k$, lie on at least three different planes, and

$$\dim S_3^1(\Delta) = 4k + 12$$

if $[q_ip_2p_3]$, $i = 1, \cdots, k$, lie on two different planes.

Note that the index of q_{i+1} is out off the range $1, \cdots, k$ when $i = k$. We assume that it is modulo by k. This convention is used throughout the sectionwithout indication.

4. Miscellaneous. If a trivariate function F could be expressed as Bernstein polynomial form on a line segment $[p_1p_2]$. That is, $F|_{[p_1p_2]}(p) = \sum_{i=0}^n b_i B_i^n(t)$ with $p = (1-t)p_1 + tp_2$. Then

$$b_0 = F(p_1), \quad b_1 = b_0 + \frac{1}{n}(p_2 - p_1)^T \nabla F(p_1) \tag{93}$$

2.11.2 Finite Element Hull

Suppose we are given a surface discretization $\mathcal{L}$ consisting of triangles, quadrilaterals and pentagons with attached normal on each vertex. We assume that the surface is double sided and all the normals on the vertices point to one side of the discretization. We call this side as positive. The other side is negative. Since we do not assume the vertices of any quadrilateral or pentagon are coplanar, we do not call the quadrilateral or pentagon as face, but *polygon*.

Let $[p_ip_j]$ be an edge of $\mathcal{L}$, if $(p_j - p_i)^T n_j \ (p_i - p_j)^T n_i \geq 0$ and at least one of $(p_j - p_i)^T n_j$ and $(p_i - p_j)^T n_i$ is positive, then we say the edge $[p_ip_j]$ is *positive convex* (see [18]). If both the numbers are zero then we say it is *zero convex*. The *negative convex* edge is similarly defined. If $(p_j - p_i)^T n_j \ (p_i - p_j)^T n_i < 0$, then we say the edge is *non-convex*. Let F be a polygon of $\mathcal{L}$. If all its edges are nonnegative (positive or zero) convex and at least one of them is positive convex, then we say the polygon is *positive convex*. If all its edges are zero convex then we label the polygon as *zero convex*. The *negative convex* polygon is similarly defined. All the other cases are labeled as *non-convex*.

Let $\mathcal{L} = \mathcal{L}_{non-zero} \cup \mathcal{L}_{zero}$, where $\mathcal{L}_{non-zero}$ and $\mathcal{L}_{zero}$ are the collections of non-zero convex polygons and zero convex polygons of $\mathcal{L}$, respectively.

Now we construct a *finite-element-hull*, denoted as $\mathcal{H}$, that consists of tetrahedra and pyramids on $\mathcal{L}_{non-zero}$ such that each polygon of $\mathcal{L}_{non-zero}$ is contained in $\mathcal{H}$ and tangent plane at each vertex of $\mathcal{L}_{zero}$ is contained in $\mathcal{H}$.

a. Build Tetrahedra for Convex Triangle. Let $\langle p_1p_2p_3\rangle$ be a convex triangular polygon of $\mathcal{L}_{non-zero}$. Let $c = (p_1 + p_2 + p_3)/3$, n be the normal of face $[p_1p_2p_3]$ that points to the positive side of $\mathcal{L}$. Then choose a *top vertex* u if the polygon is positive convex or a *bottom vertex* v if the polygon is negative convex as follows: $u = c + tn$, or $v = c - tn$. Then *positive face tetrahedron* $[up_1p_2p_3]$ or *negative face tetrahedron* $[vp_1p_2p_3]$ are formed where $t > 0$ is a properly chosen number such that the tangent planes at the vertices are contained in the tetrahedra constructed.

b. Build Pyramids for Convex Quadrilaterals. Let $\langle p_1p_2p_3p_4\rangle$ be a convex quadrilateral of $\mathcal{L}_{non-zero}$. Let $c = (p_1 + p_2 + p_3 + p_4)/4$, n be the normal of the ruled surface $(p_1p_2p_3p_4)$ at c that points to the positive side of $\mathcal{L}$. Then choose a *top vertex* u if the polygon is positive convex or a *bottom vertex* v if the polygon is negative convex as follows: $u = c + tn$, or $v = c - tn$. Then *positive face pyramid* $[up_1p_2p_3p_4]$ or *negative face pyramid* $[vp_1p_2p_3p_4]$ are formed where $t > 0$ is a properly chosen number such that the tangent planes at the vertices are contained in the pyramids constructed.

c. Build Tetrahedra for Non-convex Polygons. Let $\langle p_1p_2\cdots p_k\rangle$ $(3 \leq k \leq 5)$ be a non-convex polygon of $\mathcal{L}_{non-zero}$. Let $c = (p_1 + \cdots + p_k)/k$. Define a normal n as the average of the normals of the triangle faces $[p_ip_{i+1}c]$, $i = 1, \cdots, k$. Then both top vertex u and bottom vertex v are chosen as $u = c + tn$, or $v = c - tn$, and k tetrahedra $[uvp_ip_{i+1}]$, $i = 1, \cdots, k$, are formed. Here $t > 0$ is defined so that the tangent planes at the vertices are contained in the tetrahedra constructed.

d. Build Tetrahedra for Edges. Let $[p_1p_2]$ be an edge of $\mathcal{L}$ where F_l and F_r are the two adjacent polygons in $\mathcal{L}_{non-zero}$. If the top vertices u_l and u_r exist for F_l and F_r , respectively, then the *positive edge tetrahedron* is $[u_lu_rp_1p_2]$. Similarly, the *negative edge tetrahedron* $[v_lv_rp_1p_2]$ is constructed if the bottom vertices v_l and v_r exist for F_l and F_r , respectively.

2.11.3 C^1 Modeling of Surface by Rational A-patches

In this section, we shall construct a piecewise C^1 rational function F over $\mathcal{H}$ whose zero contour $\{p : \ F(p) = 0\}$ possesses a separate subset S such that $S \cup \mathcal{L}_{zero}$ (i) passes through the vertices of $\mathcal{L}$, (ii) has the given normal at each vertex, and (iii) is a smooth surface. We further require that the function F has quadratic recovery property.

Modeling Functions First we give the forms of the modeling functions over the finite elements. The parameters in these functions will be specified later in this section.

1. Function on tetrahedron for a convex triangular polygon. Let $\langle p_1p_2p_3\rangle \in \mathcal{L}_{non-zero}$ be any one convex triangular polygon. If the positive face tetrahedron $[up_1p_2p_3]$ exists, we define for the indices of the coefficients)

$$\begin{aligned} F|_{[up_1p_2p_3]} &= \sum_{i+j+k+l=3} t_{ijkl}B^3_{ijkl}(\alpha) \\ &+ \frac{t^{(3)}_{0111}\alpha_2\alpha_3 + t^{(2)}_{0111}\alpha_2\alpha_4 + t^{(1)}_{0111}\alpha_3\alpha_4}{\alpha_2\alpha_3+\alpha_2\alpha_4+\alpha_3\alpha_4}B^3_{0111}(\alpha) \end{aligned} \tag{94}$$

$F|_{[vp_1p_2p_3]}$ is similarly defined if the negative face tetrahedron $[vp_1p_2p_3]$ exists. In the following we only give the expressions of the functions on positive elements. The functions on the negative elements are in the same forms. To distinguish the difference, we place a tilde on the corresponding coefficients.

2. Function on pyramid for convex quadrilateral. Let $\langle p_1p_2p_3p_4\rangle$ be a convex quadrilateral of $\mathcal{L}$ and $[up_1p_2p_3p_4]$ be the pyramid. Then define for the indices of the coefficients)

$$\begin{aligned} F|_{[up_1p_2p_3p_4]} &= \sum_{i=0}^{3}\sum_{j=0}^{3-i}\sum_{k=0}^{3-i} p_{ijk}B^3_i(u)B^{3-i}_j(s)B^{3-i}_k(t) \\ &+ \left[\left(p^{(l)}_{022}w^{(l)}_{lb} + p^{(b)}_{022}w^{(b)}_{lb}\right)B^3_2(s) + \left(p^{(r)}_{012}w^{(r)}_{rb} + p^{(b)}_{012}w^{(b)}_{rb}\right)B^3_1(s)\right]B^3_0(u)B^3_2(t) \\ &+ \left[\left(p^{(l)}_{021}w^{(l)}_{lt} + p^{(t)}_{021}w^{(t)}_{lt}\right)B^3_2(s) + \left(p^{(r)}_{011}w^{(r)}_{rt} + p^{(t)}_{011}w^{(t)}_{rt}\right)B^3_1(s)\right]B^3_0(u)B^3_1(t) \\ &+ \left(p^{(l)}_{111}w_l + p^{(r)}_{111}w_r + p^{(t)}_{111}w_t + p^{(b)}_{111}w_b\right)B^3_1(u)B^2_1(s)B^2_1(t) \end{aligned} \tag{95}$$

where

$$\begin{aligned} w^{(l)}_{lb} &= \frac{v_l}{v_l+v_b}, \quad w^{(b)}_{lb} = \frac{v_b}{v_l+v_b}, \quad w^{(r)}_{rb} = \frac{v_r}{v_r+v_b}, \quad w^{(b)}_{rb} = \frac{v_b}{v_r+v_b}, \\ w^{(l)}_{lt} &= \frac{v_l}{v_l+v_t}, \quad w^{(t)}_{lt} = \frac{v_t}{v_l+v_t}, \quad w^{(r)}_{rt} = \frac{v_r}{v_r+v_t}, \quad w^{(t)}_{rt} = \frac{v_t}{v_r+v_t}, \\ w_l &= \frac{v_l}{v_l+v_r+v_t+v_b}, \qquad w_r = \frac{v_r}{v_l+v_r+v_t+v_b}, \\ w_t &= \frac{v_t}{v_l+v_r+v_t+v_b}, \qquad w_b = \frac{v_b}{v_l+v_r+v_t+v_b}, \end{aligned}$$

with

$$\begin{aligned} v_l(s,t) &= s^2t^2(1-t)^2, \qquad v_r(s,t) = (1-s)^2t^2(1-t)^2 \\ v_t(s,t) &= s^2(1-s)^2(1-t)^2, \qquad v_b(s,t) = s^2(1-s)^2t^2 \end{aligned}$$

3. Function on tetrahedra for non-convex 3,4-sided polygon and 5-sided polygon. Let $\langle p_1\cdots p_K\rangle$ be a non-convex polygon, $3 \le K \le 5$. Then K tetrahedra $[uvp_ip_{i+1}]$ have been constructed. On each tetrahedron, a cubic is used.

$$F|_{[uvp_mp_{m+1}]} = \sum_{i+j+k+l=3} t^{(m)}_{ijkl}B^3_{ijkl}(\alpha), \quad m = 1,\cdots,K. \tag{96}$$

4. Function on edge tetrahedron. Let $[p_1p_2]$ be a non-zero-convex edge of $\mathcal{L}$ and $[u_lu_rp_1p_2]$ be the positive edge tetrahedron. Then define for the indices of the coefficients)

$$\begin{aligned} F|_{[u_lu_rp_1p_2]} &= \sum_{i+j+k+l=3} e_{ijkl}B^3_{ijkl}(\alpha) \\ &+ \frac{e^{(l)}_{1101}\alpha_1 + e^{(r)}_{1101}\alpha_2}{\alpha_1+\alpha_2}B^3_{1101}(\alpha) + \frac{e^{(l)}_{1110}\alpha_1 + e^{(r)}_{1110}\alpha_2}{\alpha_1+\alpha_2}B^3_{1110}(\alpha) \end{aligned} \tag{97}$$

Construction of Rational A-patches Now we shall determine the parameters of F step by step

Total Algorithm. *Specifying the weights*
Step 1. In order to have the surface constructed contains the vertices of $\mathcal{L}$, we take the number 1 weights to be zero.
Step 2. The number 2 weights are determined by formula (93) from the normals.
Step 3. The number 3 weights in triangle interfaces are defined by interpolating the perpendicular directional derivative. For example, the number 3 weight on the triangle interface $[p_1p_2u]$ is defined by interpolating the directional derivative $\frac{1}{2}\left[\frac{(u-p_j)^T(p_1-p_2)}{\|p_1-p_2\|^2}\ (u\ -\ p_1) + \frac{(u-p_1)^T(p_2-p_1)}{\|p_1-p_2\|^2}(u-p_2)\right]^T$ (n_1+n_2) at the point $\frac{1}{2}(p_1+p_2)$ where the direction is in the face $[p_1p_2u]$ and perpendicular to the edge $[p_1p_2]$. We can derive that

$$t_{1110} = \frac{1}{2}\left[t_{1200} + t_{1020} + \alpha(u,p_1,p_2)t_{0210} + (1-\alpha(u,p_1,p_2))t_{0120}\right]$$

where $\alpha(u,p_1,p_2)$ is given by

$$\alpha(u,p_1,p_2) = \frac{[2(u-p_2)+(u-p_1)]^T(p_1-p_2)}{\|p_1-p_2\|^2}$$

The three rational coefficients $t^{(1)}_{0111}$, $t^{(2)}_{0111}$ and $t^{(3)}_{0111}$ are defined as above by interpolating directional perpendicular derivatives at the mid-point of the edge $[p_2p_3]$, $[p_1p_3]$ and $[p_1p_2]$, respectively. Set $t_{0111} = \frac{1}{3}\left(t^{(1)}_{0111} + t^{(2)}_{0111} + t^{(3)}_{0111}\right)$ and then reset the values of $t^{(1)}_{0111}$, $t^{(2)}_{0111}$ and $t^{(3)}_{0111}$ by reducing the value t_{0111}.

The number 3 weights, which are the coefficients of rational terms, on the quadrilaterals are determined by C^1 condition (81) or (87). Then set the corresponding polynomial coefficients as

$$p_{012} = \frac{1}{2}\left(p^{(b)}_{012} + p^{(r)}_{012}\right),\quad p_{011} = \frac{1}{2}\left(p^{(t)}_{011} + p^{(r)}_{011}\right),$$
$$p_{022} = \frac{1}{2}\left(p^{(l)}_{022} + p^{(b)}_{022}\right),\quad p_{021} = \frac{1}{2}\left(p^{(l)}_{021} + p^{(t)}_{021}\right).$$

Then reset the rational coefficients by reducing the value of corresponding polynomial coefficients.
Step 4. The remaining weights on the finite elements are specified by the following sub-algorithms.

Sub-Algorithm 1. *Compute the weights on convex face tetrahedra*

The number 4 and 5 weights are free We assign the function value $F(u)$ and gradient $\nabla F(u)$ as parameters. Then $p_{3000} = F(u)$ and, by (93),

$$t_{2000+e_{i+1}} = F(u) + \frac{1}{3}(p_i-u)^T\nabla F(u),\quad i=1,2,3.$$

The use of degrees of freedom.

Parameters $F(u)$ and $\nabla F(u)$ could be used to control the shape interactively. The default choice is we make the polynomial part of F defined by (94) approximate a quadratic. It follows from (82), we have a linear system with 14 unknowns and 20 equations. Since the coefficient matrix of the system is not full rank, we add a set of equations by making F approximate a linear function. Solving this system in the least square sense, we get the parameters.

Sub-Algorithm 2. *Compute the weights on convex pyramid*

The four number 4 weights and one number 5 weight are free We assign the function value $F(u)$ and gradient $\nabla F(u)$ as parameters. Then $p_{300} = F(u)$ and, by (93),

$$p_{211} = F(u) + \frac{1}{3}(p_1-u)^T\nabla F(u),\quad p_{201} = F(u) + \frac{1}{3}(p_2-u)^T\nabla F(u),$$
$$p_{210} = F(u) + \frac{1}{3}(p_3-u)^T\nabla F(u),\quad p_{200} = F(u) + \frac{1}{3}(p_4-u)^T\nabla F(u).$$

Note that defining the number 4 and 5 weights in this way reduces the degrees of freedom from five to four. The gain of this degree reduction is that the function defined by (95) is guarantee to be C^1 at u.

Now we consider the computation of coefficients $p^{(l)}_{111}, p^{(r)}_{111}, p^{(t)}_{111}$ and $p^{(b)}_{111}$ of rational terms. These coefficients could be computed separately. Suppose the pyramid considered is $[u_l p_1 p_2 p_3 p_4]$ and u_r is the top vertex of the element adjacent to the interface $[u_l p_2 p_4]$. Then $p^{(l)}_{111}$ is computed as follows. Let

$$p_1 = \alpha_1 u_l + \alpha_2 u_r + \alpha_3 p_2 + \alpha_4 p_4, \quad p_3 = \beta_1 u_l + \beta_2 u_r + \beta_3 p_2 + \beta_4 p_4.$$

Then by Corollary 2.24, we have

$$\begin{aligned} p_{112} &= \alpha_1 e_{2010} + \alpha_2 e_{1110} + \alpha_3 e_{1020} + \alpha_4 e_{1011} = \alpha_1 p_{201} + \alpha_2 e_{1110} + \alpha_3 p_{102} + \alpha_4 p_{101}, \\ p_{110} &= \beta_1 e_{2001} + \beta_2 e_{1101} + \beta_3 e_{1011} + \beta_4 e_{1002} = \beta_1 p_{200} + \beta_2 e_{1101} + \beta_3 p_{101} + \beta_4 p_{100}, \end{aligned}$$

and

$$\begin{aligned} p^{(r)}_{111} &= \frac{1}{2}\left(\alpha_1 e_{2001} + \alpha_2 e_{1101} + \alpha_3 e_{1011} + \alpha_4 e_{1002} + \beta_1 e_{2010} + \beta_2 e_{1110} + \beta_3 e_{1020} + \beta_4 e_{1011}\right) \\ &= \frac{1}{2}\left[\left(\beta_1 - \frac{\alpha_1\beta_2}{\alpha_2}\right) p_{201} + \left(\alpha_1 - \frac{\alpha_2\beta_1}{\beta_2}\right) p_{200} + \frac{\beta_2}{\alpha_2} p_{112} + \frac{\alpha_2}{\beta_2} p_{110}\right. \\ &\quad \left. + \left(\beta_3 - \frac{\alpha_3\beta_2}{\alpha_2}\right) p_{102} + \left(\alpha_3 - \frac{\alpha_2\beta_3}{\beta_2} + \beta_4 - \frac{\alpha_4\beta_2}{\alpha_2}\right) p_{101} + \left(\alpha_4 - \frac{\alpha_2\beta_4}{\beta_2}\right) p_{100}\right]. \end{aligned}$$

Other coefficients of rational terms are similarly computed. Set $p_{111} = \frac{1}{4}(p^{(l)}_{111} + p^{(r)}_{111} + p^{(t)}_{111} + p^{(b)}_{111})$ and then reset the $p^{(l)}_{111}, p^{(r)}_{111}, p^{(t)}_{111}$ and $p^{(b)}_{111}$ by reducing their values by p_{111}.

The use of degrees of freedom.

Parameters $F(u)$ and $\nabla F(u)$ could be used to control the shape interactively. The default choice is we make the polynomial part of F defined by (95) approximate a quadratic. Let $\sum_{i+j+k+l=n} a_{ijkl} B^n_{ijkl}(\alpha)$ be a polynomial of degree n over the tetrahedron $[up_1p_2p_3]$. Then we could express it as a polynomial $\sum_{I=0}^{n}\sum_{J=0}^{n-I}\sum_{K=0}^{n-I} B^n_I(u) B^{n-I}_J(s) B^{n-I}_K(t)$ of degree n over the pyramid $[up_1p_2p_3p_4]$. Similar to the proof of Theorem 2.23, we can derive that

$$b_{IJK} = \sum_{l=I}^{n} \sum_{i+j+k=n-l} a_{ijkl} c^{IJK}_{ijk} \tag{98}$$

with

$$c^{IJK}_{ijk} = \sum_{\lambda=\max\{0,K-j,J-k\}}^{\min\{i,K,J\}} \frac{C^{n-I}_{\lambda,K-\lambda,J-\lambda,n-I-J-K+\lambda}}{C^{n-I}_{K,n-I-K} C^{n-I}_{J,n-I-J}} B^{n-I-J-K+\lambda}_{i-\lambda,j-K+\lambda,k-J+\lambda,L-I}(a_1,a_2,a_3,a_4),$$

where (a_1,a_2,a_3,a_4) is defined by $p_4 = a_1 u + a_2 p_1 + a_3 p_2 + a_4 p_4$, $\sum_{i=1}^4 a_i = 1$. Hence, a quadratic over $[up_1p_2p_3]$ could be expressed as a polynomial of degree 3 over $[up_1p_2p_3p_4]$ using (98) first and then (88). Approximating this quadratic by the polynomial part of F defined by (95) we lead to a linear system with 14 unknowns (10 for the coefficients of the quadratic, 4 for $F(u)$ and $\nabla F(u)$) and 30 equations. Solving this system in the least square sense, we get the parameters.

Sub-Algorithm 3. *Compute the weights on tetrahedra for non-convex polygon.*

Consider a $K-$sided polygon $\langle p_1 \cdots p_K\rangle$ for $3 \le K \le 5$. The number 1,2,3 weights have been determined. The other weights labeled as $\bullet$ are defined by the C^1 condition. Under the C^0 condition

$$t^{(s)}_{ij0k} = t^{(s+1)}_{ijk0}, \quad i+j+k=3, \quad s = 1,2,\cdots,K, \tag{99}$$

there are $3(K+1)+1$ weights undefined. From Theorem 2.27 we know that the dimension of $S_3^1(\Delta)$ is $4K+10$. Since the function under construction interpolates positions and gradients at the vertices p_i for $i=1,\cdots K$, and interpolates two directional derivatives at the midpoints of the edges $[p_ip_{i+1}]$, that is, it satisfies $6K$ interpolation conditions, the remaining degree of freedom is $10-2K$. Now we take $F(v)$ and $\nabla F(v)$ as free parameters and express other weights in terms of these parameters and derive a system of $2K-6$ equations that the parameters $F(v)$ and $\nabla F(v)$ satisfy. That is, we express

$$t_{ijkl}^{(s)} = \alpha_{ijkl}^{(s)} F(v) + \beta_{ijkl}^{(s)} \nabla F(v) + \gamma_{ijkl}^{(s)}, \quad \alpha_{ijkl}^{(s)}, \gamma_{ijkl}^{(s)} \in \mathbb{R}, \ \beta_{ijkl}^{(s)} \in \mathbb{R}^3.$$

It is obvious that for the number 1,2,3 weights $t_{ijkl}^{(s)}$, $\alpha_{ijkl}^{(s)}=0$, $\beta_{ijkl}^{(s)}=0$ and $\gamma_{ijkl}^{(s)} = t_{ijkl}^{(s)}$. It follows from (93) that

$$t_{1200}^{(1)} = F(v) + \frac{1}{3}(u-v)^T \nabla F(v), \quad t_{0210}^{(s)} = F(v) + \frac{1}{3}(p_s - v)^T \nabla F(v), \quad s = 1, \cdots, K.$$

That is, $\alpha_{0210}^{(s)} = 1$, $\beta_{0210}^{(s)} = \frac{1}{3}(p_s - v)^T$, $\gamma_{0210}^{(s)} = 0$ and $\alpha_{1200}^{(1)} = 1$, $\beta_{1200}^{(1)} = \frac{1}{3}(u-v)^T$, $\gamma_{1200}^{(1)} = 0$. Let

$$u = \alpha_1^{(s)} p_s + \alpha_2^{(s)} p_{s+1} + \alpha_3^{(s)} p_{s-1} + \alpha_4^{(s)} v, \quad s = 1, \cdots, K.$$

Then

$$t_{1110}^{(s)} = \alpha_1^{(s)} t_{0120}^{(s)} + \alpha_2^{(s)} t_{0111}^{(s)} + \alpha_3^{(s)} t_{0111}^{(s-1)} + \alpha_4^{(s)} t_{0210}^{(s)}, \quad s = 1, \cdots, K. \tag{100}$$

Since $t_{0120}^{(s)}$, $t_{0111}^{(s)}$, and $t_{0111}^{(s-1)}$ are all number 2 and 3 weights, we have

$$t_{1110}^{(s)} = \alpha_1^{(s)} \gamma_{0120}^{(s)} + \alpha_2^{(s)} \gamma_{0111}^{(s)} + \alpha_3^{(s)} \gamma_{0111}^{(s-1)} + \alpha_4^{(s)} (\frac{1}{3}(p_s - v)^T \nabla F(v) + F(v))$$

Hence we have the same form expression for $\alpha_{1110}^{(s)}, \beta_{1110}^{(s)}$ and $\gamma_{1110}^{(s)}$. For example,

$$\begin{aligned} \alpha_{1110}^{(s)} &= \alpha_1^{(s)} \alpha_{0120}^{(s)} + \alpha_2^{(s)} \alpha_{0111}^{(s)} + \alpha_3^{(s)} \alpha_{0111}^{(s-1)} + \alpha_4^{(s)} \alpha_{0210}^{(s)} = \alpha_4^{(s)}, \\ \beta_{1110}^{(s)} &= \alpha_1^{(s)} \beta_{0120}^{(s)} + \alpha_2^{(s)} \beta_{0111}^{(s)} + \alpha_3^{(s)} \beta_{0111}^{(s-1)} + \alpha_4^{(s)} \beta_{0210}^{(s)} = \tfrac{1}{3} \alpha_4^{(s)} (p_s - v)^T. \end{aligned}$$

Furthermore, we have

$$t_{2100}^{(s)} = \alpha_1^{(s)} t_{1110}^{(s)} + \alpha_2^{(s)} t_{1110}^{(s+1)} + \alpha_3^{(s)} t_{1110}^{(s-1)} + \alpha_4^{(s)} t_{1200}^{(s)}, \tag{101}$$

and

$$\begin{aligned} \alpha_{2100}^{(s)} &= \alpha_1^{(s)} \alpha_{1110}^{(s)} + \alpha_2^{(s)} \alpha_{1110}^{(s+1)} + \alpha_3^{(s)} \alpha_{1110}^{(s-1)} + \alpha_4^{(s)} \alpha_{1200}^{(s)} \\ &= \alpha_1^{(s)} \alpha_4^{(s)} + \alpha_2^{(s)} \alpha_4^{(s+1)} + \alpha_3^{(s)} \alpha_4^{(s+1)} + \alpha_4^{(s)}, \\ \beta_{2100}^{(s)} &= \alpha_1^{(s)} \beta_{1110}^{(s)} + \alpha_2^{(s)} \beta_{1110}^{(s+1)} + \alpha_3^{(s)} \beta_{1110}^{(s-1)} + \alpha_4^{(s)} \beta_{1200}^{(s)} \\ &= \tfrac{1}{3} [\alpha_1^{(s)} \alpha_4^{(s)} (p_s - v)^T + \alpha_2^{(s)} \alpha_4^{(s+1)} (p_{s+1} - v)^T \\ &+ \alpha_3^{(s)} \alpha_4^{(s+1)} (p_{s-1} - v)^T + \alpha_4^{(s)} (u - v)^T]. \end{aligned}$$

In the case of a planar polygon, $\alpha_4^{(s)} = \alpha_4^{(s+1)}$ for all s and we get

$$\begin{aligned} \alpha_{2100}^{(s)} &= \alpha_4^{(s)} (\alpha_1^{(s)} + \alpha_2^{(s)} + \alpha_3^{(s)} + 1) \\ &= \alpha_4^{(s)} (2 - \alpha_4^{(s)}) \\ \beta_{2100}^{(s)} &= \tfrac{1}{3} \alpha_4^{(s)} [\alpha_1^{(s)} (p_s - v)^T + \alpha_2^{(s)} (p_{s+1} - v)^T + \alpha_3^{(s)} (p_{s-1} - v)^T + (u - v)^T] \\ &= \tfrac{1}{3} \alpha_4^{(s)} [-v^T (\alpha_1^{(s)} + \alpha_2^{(s)} + \alpha_3^{(s)} + 1) + \alpha_1^{(s)} p_s^T + \alpha_2^{(s)} p_{s+1}^T + \alpha_3^{(s)} p_{s-1}^T + u^T] \\ &= \tfrac{1}{3} \alpha_4^{(s)} [-v^T (2 - \alpha_4^{(s)}) + u^T - \alpha_4^{(s)} v^T + u^T] \\ &= \tfrac{2}{3} \alpha_4^{(s)} (u - v)^T. \end{aligned}$$

It follows from Theorem 2.25 that (101) has $K-2$ independent equations and they define the same weight $t^{(1)}_{2100} = t^{(2)}_{2100} = \cdots = t^{(K)}_{2100}$. Therefore, we have the following equations for $F(v)$ and $\nabla F(v)$

$$\alpha^{(s)}_{2100} F(v) + \beta^{(s)}_{2100} \nabla F(v) + \gamma^{(s)}_{2100} = \alpha^{(s+1)}_{2100} F(v) + \beta^{(s+1)}_{2100} \nabla F(v) + \gamma^{(s+1)}_{2100}, \tag{102}$$

for $s = 1, \cdots K-3$, Similarly, we have

$$t^{(s)}_{2010} = \alpha^{(s)}_1 t^{(s)}_{1020} + \alpha^{(s)}_2 t^{(s)}_{1011} + \alpha^{(s)}_3 t^{(s+2)}_{1011} + \alpha^{(s)}_4 t^{(s)}_{1110}, \quad s = 1, \cdots, K, \tag{103}$$

$$t^{(s)}_{3000} = \alpha^{(s)}_1 t^{(s)}_{2010} + \alpha^{(s)}_2 t^{(s+1)}_{2010} + \alpha^{(s)}_3 t^{(s+2)}_{2010} + \alpha^{(s)}_4 t^{(s)}_{2100}, \quad s = 1, \cdots, K, \tag{104}$$

and $F(v)$ and $\nabla F(v)$ satisfy the following equations

$$\alpha^{(s)}_{3000} F(v) + \beta^{(s)}_{3000} \nabla F(v) + \gamma^{(s)}_{3000} = \alpha^{(s+1)}_{3000} F(v) + \beta^{(s+1)}_{3000} \nabla F(v) + \gamma^{(s+1)}_{3000}, \tag{105}$$

for $s = 1, \cdots K-3$. Hence all the weights are defined and all $\alpha^{(s)}_{ijkl}$, $\beta^{(s)}_{ijkl}$ and $\gamma^{(s)}_{ijkl}$ can be computed from (100)–(105).

The use of freedoms. Interactive shape control by giving $F(v)$ and $\nabla F(v)$ under the restrictions (102) and (105). The default choice is to make the K cubics approximate quadratics. By using the degree elevation formula, we need to solve the following equations

$$\frac{i}{3} t^{(s)}_{i-1,jkl} + \frac{j}{3} t^{(s)}_{i,j-1,kl} + \frac{k}{3} t^{(s)}_{ij,k-1,l} + \frac{l}{3} t^{(s)}_{ijk,l-1} - \alpha^{(s)}_{ijkl} F(v) - \beta^{(s)}_{ijkl} \nabla F(v) = \gamma^{(s)}_{ijkl}, \tag{106}$$

for $i+j+k+l = 3$, $\quad s = 1, \cdots, K$, in the least squares sense for the unknowns $t^{(s)}_{ijkl}$, $F(v), \nabla F(v)$ under the C^0 condition (99) for the K cubics and C^0 condition

$$t^{(m)}_{ijk0} = t^{(m+1)}_{ij0k}, \quad i+j+k = 2, \quad m = 1, 2, 3$$

and the constraints (102) and (105). System (106) has $(4K+3)$ unknowns $t^{(s)}_{ijkl}$ for $i+j+k+l = 2$ and 4 unknowns $F(v), \nabla F(v)$, and has $10K+4$ equations.

Sub-Algorithm 4. *Compute the weights on edge tetrahedra*

Suppose the edge tetrahedron considered is $[u_l u_r p_1 p_2]$ The weights e_{1110} and e_{1101} are set to zero. The number 6 coefficients $e^{(l)}_{1110}, e^{(r)}_{1110}, e^{(l)}_{1101}$ and $e^{(r)}_{1101}$ are determined by the C^1 condition. If the right neighbor, that is adjacent to $[u_r p_1 p_2]$, of the edge tetrahedron is tetrahedron $[u_r p_1 p_2 p_3]$, and if we express $u_l = \alpha_1 u_r + \alpha_2 p_1 + \alpha_3 p_2 + \alpha_4 p_3$ with $\sum_{i=1}^4 \alpha_i = 1$, we have

$$e^{(r)}_{1110} = \alpha_1 f_{2100} + \alpha_2 f_{1200} + \alpha_3 f_{1110} + \alpha_4 f_{1101},$$
$$e^{(r)}_{1101} = \alpha_1 f_{2001} + \alpha_2 f_{1101} + \alpha_3 f_{1011} + \alpha_4 f_{1002}.$$

If the right neighbor of $[u_l u_r p_1 p_2]$ is pyramid $[u_r p_1 p_2 p_3 p_4]$, then let

$$u_l = \alpha_1 u_r + \alpha_2 p_1 + \alpha_3 p_2 + \alpha_4 p_3, \quad u_l = \beta_1 u_r + \beta_2 p_1 + \beta_3 p_2 + \beta_4 p_4$$

with $\sum_{i=1}^4 \alpha_i = \sum_{i=1}^4 \beta_i = 1$. Then we have

$$e^{(r)}_{1110} = \alpha_1 p_{211} + \alpha_2 p_{122} + \alpha_3 p_{112} + \alpha_4 p_{121},$$
$$e^{(r)}_{1101} = \beta_1 p_{201} + \beta_2 p_{112} + \beta_3 p_{102} + \beta_4 p_{101}.$$

The weights $e^{(l)}_{1110}$ and $e^{(l)}_{1101}$ are similarly computed.

Theorem 4.1. *For the given discretization $\mathcal{L}$ of a surface with a built finite-element hull $\mathcal{H}$ on it, the surface defined by the union of all edge A-patches, face A-patches and zero convex faces of $\mathcal{L}$*

interpolates the vertices of the discretization and has the normals at the vertices, and it is smooth and topologically equivalent to $\mathcal{L}$.

The scheme proposed above makes the constructed surface have the plane recovery property. Even further, the scheme can recover quadratic. That is if the normal at the vertices of a polygon are extracted from a quadratic surface $Q(p) = 0$ that passes through the vertices of the polygon, and furthermore if the free weights are defined by approximating a quadratic, then $F(p) = Q(p)$. Similarly, if the normals at the vertices of an edge and the vertices of the two adjacent polygons are extracted from a quadratic surface $Q(p) = 0$ that passes through these vertices, and if the free weights on the neighbor polygon elements are defined by approximating a quadratic surface, then $F(p) = Q(p)$ on the edge tetrahedron.

The proof of the quadratic recovery property is based on the following facts: (a). F interpolates function values and first order partial derivatives of Q at the vertices, and F interpolates directional derivatives of Q in any directions that perpendicular to edges at the mid-points of the edges. (b). The free weights are defined by the degree elevation formula. (c). The rational function is degenerate to zero. The detailed discussion needs to distinguish the cases when the polygon is convex or non-convex. We omit the detail here.

2.11.4 Evaluate the Surfaces

Since the patches for edges and convex triangles are defined in the same way as in [238], we can evaluate these patches using the scheme in [238]. In the following, we ignore these cases.

A. Evaluate the Triangular Face A-patch

For each triangular nonconvex polygon in $\mathcal{L}_{no-zero}$, we shall produce a piecewise triangular approximation for the surface patch $F = 0$. Let $\langle p_1p_2p_3\rangle \in \mathcal{L}_{no-zero}$ be one triangular polygon and u and v be top and bottom vertices. Let N be a given positive number, which represents the resolution of the piecewise approximation. Then the piecewise triangular approximation is defined by the naive connection of the points $s_{xyz}(x + y + z = N,\ x, y, z \geq 0)$. Here s_{xyz} is the intersection point of the surface $F = 0$ with the polygonal line $[uq_{xyz}] \cup [q_{xyz}v]$, where $q_{xyz} = \frac{x}{N}p_1 + \frac{y}{N}p_2 + \frac{z}{N}p_3$ and the intersection point is computed by solving the cubic polynomial equation $F\left((1-t)q_{xyz} + tu\right) = 0$ if $F(q_{xyz}) \leq 0$ or solving a similar equation $F\left((1-t)q_{xyz} + tv\right) = 0$ if $F(q_{xyz}) > 0$, where the required root is the minimal one.

Since q_{xyz} is in one of the tetrahedra $[uvp_ip_{i+1}]$ with $i = 1, \cdots, 3$, q_{xyz} could be expressed in the following form:

$$q = \beta_1^{(i)}u + \beta_2^{(i)}v + \beta_3^{(i)}p_i + \beta_4^{(i)}p_{i+1} \tag{107}$$

Then we can derive, from (96), that

$$F((1-t)q + tu) = \sum_{s=0}^{3}\left(\sum_{\lambda_1=s}^{3} C_{\lambda_1 s}^{(i)}\right) B_s^3(t) \tag{108}$$

with $C_{\lambda_1 s}^{(i)} = \sum_{\lambda_2+\lambda_3+\lambda_4=3-\lambda_1} t_{\lambda_1\lambda_2\lambda_3\lambda_4}^{(i)} B_{\lambda_1-s,\lambda_2\lambda_3\lambda_4}^{3-s}(\beta_1^{(i)}, \beta_2^{(i)}, \beta_3^{(i)}, \beta_4^{(i)})$. Similarly, we have

$$F((1-t)q + tv) = \sum_{s=0}^{3}\left(\sum_{\lambda_2=s}^{3} \tilde{C}_{\lambda_2 s}^{(i)}\right) B_s^3(t) \tag{109}$$

with $\tilde{C}_{\lambda_2 s}^{(i)} = \sum_{\lambda_1+\lambda_3+\lambda_4=3-\lambda_2} t_{\lambda_1\lambda_2\lambda_3\lambda_4}^{(i)} B_{\lambda_1\lambda_2-s,\lambda_3\lambda_4}^{3-s}(\beta_1^{(i)}, \beta_2^{(i)}, \beta_3^{(i)}, \beta_4^{(i)})$.

B. Evaluate the Quadrilateral Face A-patch

For each quadrilateral polygon in $\mathcal{L}_{no-zero}$, we shall produce a piecewise quadrilateral approximation for the surface patch. Let N be two given positive numbers, which represent the resolution

of the piecewise approximation and it should have the same value as above, let $\langle p_1p_2p_3p_4\rangle$ be a quadrilateral of $\mathcal{L}_{no-zero}$ and u and v(if exist) be the top and bottom vertices of $\langle p_1p_2p_3p_4\rangle$. Then the piecewise quadrilateral approximation is defined by connecting the points $p_{xy}(x = 0, \cdots, N; y = 0, \cdots, N)$. Here p_{xy} is the intersection point of the polygonal line $[uq_{xy}] \cup [q_{xy}v]$ and the surface $F = 0$, where

$$q_{xy} = \frac{y}{N}\left[\frac{x}{N}p_1 + \frac{N-x}{N}p_2\right] + \frac{N-y}{N}\left[\frac{x}{N}p_3 + \frac{N-x}{N}p_4\right] \tag{110}$$

and the intersection point is computed by solving the cubic equation $F((1-t)q_{xy} + tu) = 0$ if $F(q_{xy}) \leq 0$ or solving a similar equation $F((1-t)q_{xy} + tv) = 0$ if $F(q_{xy}) > 0$. Again, we use the minimal root.

If the polygon is convex, (95) gives explicit expression for $F((1-t)q_{xy} + tu)$.

If the polygon is non-convex, q_{xy} is in one of the tetrahedra $[uvp_ip_{i+1}]$ with $i = 1, \cdots, 4$. Using (110), q_{xy} could be expressed as (107), and (108) and (109) could be used again.

C. Evaluate the Pentagon Face A-patch

Let $\langle p_1 \cdots, p_5\rangle$ be a 5-sided polygon and u and v be the top and bottom vertices. Then The pentagon face A-patch is evaluated by evaluating 5 patches defined by $F|_{[uvp_ip_{i+1}]}(p) = 0$ for $i = 1, \cdots 5$. Then the piecewise triangular approximation of $F_{[uvp_ip_{i+1}]}(p) = 0$ is defined by connecting the points $s^{(i)}_{xyz}(x + y + z = N,\ x, y, z \geq 0)$. Here $s^{(i)}_{xyz}$ is the intersection point of the polygonal line $[uq^{(i)}_{xyz}] \cup [q^{(i)}_{xyz}v]$ and the surface $F = 0$, where

$$q^{(i)}_{xyz} = \frac{x}{N}p_i + \frac{y}{N}p_{i+1} + \frac{z}{N}c, \quad c = \frac{1}{5}(p_1 + \cdots + p_5).$$

The intersection point can be computed by solving the cubic polynomial equation $F((1-t)q^{(i)}_{xyz} + tu) = 0$ if $F(q^{(i)}_{xyz}) \leq 0$ or solving a similar equation $F((1-t)q^{(i)}_{xyz} + tv) = 0$ if $F(q^{(i)}_{xyz}) > 0$, where the required root is the minimal one. Express $q^{(i)}_{xyz}$ as (107), and (108) and (109) could be used to define the cubic equations.

2.12 Adaptive Model Reconstruction by Triangular Prism A-Patches

Various approaches of using implicit surface representation–the zero contour of trivariate function– in modeling geometric objects or reconstructing the image to scattered data have been described in some papers(see for examples, [18], [87], [122],[154] and [213]). However, since the implicit surface could have multiple sheets, could have singularities and is not easy to evaluate, effective and easy used schemes are still under investigation. Starting from a triangulation(this is often the preprocessing stage of surface construction) of a unknown surface, we construct an implicit surface that interpolates the vertices of the triangulation. The constructed surfaces are G^0 at the the edges that are labeled as sharp, and G^1 smooth(tangent plane continuous) elsewhere, and respects the topology of the triangulation. We assume the surface triangulation is double sided so that we could label one side as positive and other as negative.

A class of successful approaches of using implicit surface representation in interpolating a surface triangulation $\mathcal{T}$ with normals consists of the following two steps: **a**. Build a surrounding simplicial hull $\sum$(consisting of a series of tetrahedra) of the triangulation. **b**. Construct a piecewise trivariate polynomial F on that simplicial hull, and use the zero contour of F to represent the surface. Dahmen [85] propose firstly an approach for constructing such a simplicial hull of $\mathcal{T}$. In this approach, for each face, two tetrahedra are constructed. For each edge of $\mathcal{T}$, two tetrahedra are formed that blending the neighboring face tetrahedra. For the construction of the surface over $\sum$, Dahmen [85] use six quadric patches for each face tetrahedron and four quadric patches for each edge tetrahedron. Guo[122] used a Clough-Tocher split and subdivided each face tetrahedron of the simplicial hull, hence utilizing six cubic patches per face of $\mathcal{T}$ and four cubic patches per edge. Dahmen and

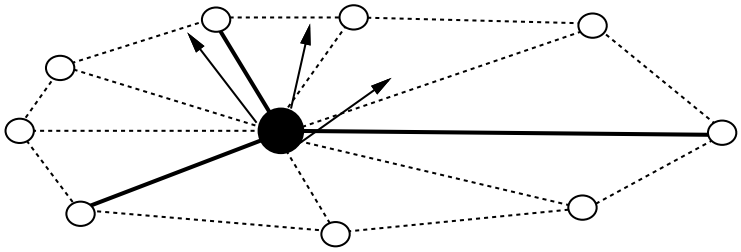


Figure 34: Grouping the triangles by the sharp edges(think lines) and assigning a normal for each group.

Thamm-Schaar [87] do not split the face tetrahedra, but the edge tetrahedra is split. All of these papers provided heuristics to overcome the multiple-sheeted and singularity problem of the implicit patches. In Bajaj Chen and Xu[18], their A-patches are guaranteed to be nonsingular and single sheeted within each tetrahedron. They use two surface patches for each face and four patches for each edge.

Instead of using tetrahedra, we use prisms based on the idea of fat surface introduced by Barnhill, Opitz and Pottmann in [50]. The pipeline of the construction is as follows:

1. Compute the face normals and oriented them such that they point to the positive side of the surface triangulation.
2. For each edge, compute dihedral angle $\theta = \pi - \theta_1$ for the two incident faces. If $\theta > \alpha$, then this edge is labeled as sharp edge. Here θ_1 is the angle between the two faces' normals and α is a threshold values for control the sharp feature.
3. Estimate normals at each vertex of the triangulation.
4. Decimate the mesh if it is considered to be too dense.
5. For each triangle of $\mathcal{T}$, construct an irregular triangular prism such that its three edges pass through the vertices of the triangle and contain the given normal, respectively.
6. Define a trivariate function F on the union of the prisms, such that its zero contour in each prism passes through the corresponding triangle vertices and has the given normals.
7. Display the surface patches $F = 0$.

Hence, each triangle of the triangulation corresponds to one triangular surface patch. Since there is no surface patch corresponding to edge and there is no splitting of the triangle, the number of surface patches are significantly reduced comparing with the earlier implicit approaches. Another attractive feature of the method is that evaluating one point of the surface is almost equivalent to solving a linear equation. Therefore, the surface could be easily and quickly displayed.

Step 1 and 2 are strait forward. For the normal estimation at a vertex, we need to distinguish the cases of sharp vertex or non-sharp vertex. If there exist sharp edges incident to the vertex, then we say the vertex is sharp, otherwise, it is non-sharp. For non-sharp vertices, we have implemented three schemes to compute the normals. The first one is weighted averaging the normals of the faces that are adjacent to the vertex. The weight we chosen are the area of the triangles. The second scheme we use is the limit normal of Loop's subdivision surface. The third method is the least square fitting the vertices around by a quadratic function, and use the normal of the quadratic at the vertex as the approximation. All these methods works well, but the results has little difference.

If the surface is fair, all them are very similar. If the surface is bumpy, the third method tend to keep the detail, while the second tend filter out some noisy.

For the sharp vertex, the triangles around the vertex are divided into some groups by the sharp edges(see Figure 34). For each group, we assign a single normal for the vertex. This normal is computed by the weighted averaging approach mention above. In the construction of surface patch for one triangle, there is only one normal used for one vertex of the triangle. This normal is vertex normal if the vertex non-sharp, otherwise the normal is group's normal to which the triangle belongs. When we mention vertex normal of a triangle in the following, we mean this normal. In the following we do not address the normal estimation problem.

Instead of using geometric error based decimation scheme of a triangulation, we have developed a normal variation based decimation scheme in order to capture the detail structures. This scheme will be reported elsewhere. Hence we omit it here. We mention the step here because it is in our implementation of the scheme and most of the examples given in the last section are produced from the decimated mesh.

2.12.1 Construction of the Triangular Surface Patches

Suppose we are given a space surface triangulation $\mathcal{T}$ of the point set $\mathcal{V} = \{v_l\}_{l=1}^N$. For each triangle, say $[v_iv_jv_k]$, we have three normals $N_{ijk}^{(i)}, N_{ijk}^{(j)}, N_{ijk}^{(k)}$ for the vertices v_i, v_j, v_k, respectively. Our aim is to construct a triangular surface patch for the triangle such that the surface patch passes through the three vertices of the triangle and has the given normal at the vertices and further, the composite surface is G^1 smooth except at the sharp edges where it is G^0. The construction involves two steps: Construct prism hull(step 5). Construction a trivariate function on the hull(step 6).

Construct Prism Hull For the given triangulation $\mathcal{T}$, the prism hull, denoted as $\mathcal{D}$, is a collection of prisms, that could be regarded as expansion of $\mathcal{T}$ in both the positive and negative directions of $\mathcal{T}$. To describe how each of the triangles expand, we need to specify a direction at each vertex along which the triangles are extrude. At the non-sharp vertex, this direction is specified as the normal at the vertex. At a sharp vertex, say v_i, there are several normals as described before. We choose the extrude direction, denoted as N_i, as the average of all the face normals. Then the prism hull is build as follows.

Let $[v_iv_jv_k]$ be a triangle of $\mathcal{T}$. Then define a prism D_{ijk} as

$$D_{ijk} := \{p : \ p \in \Delta_{ijk}(\lambda), \quad \lambda \in I_{ijk}\}$$

where $\Delta_{ijk}(\lambda) = \{p \in I\!R^3 : \ p = p_{ijk}(b_1, b_2, b_3, \lambda), \quad b_i \geq 0\}$ is a triangle for fixed λ with

$$p_{ijk}(b_1, b_2, b_3, \lambda) = b_1v_i(\lambda) + b_2v_j(\lambda) + b_3v_k(\lambda), \quad b_1 + b_2 + b_3 = 1 \tag{111}$$

and $v_l(\lambda) = v_l + \lambda n_l$, $n_l = N_l/\|N_l\|$, $l = i, j, k$; and I_{ijk} is a maximal open interval such that $0 \in I_{ijk}$ and for any $\lambda \in I_{ijk}$, the points $v_i(\lambda)$, $v_j(\lambda)$ and $v_k(\lambda)$ are not collinear and the three normals N_i, N_j and N_k point to the same side of the plane $P_{ijk}(\lambda) = \{p : \ p = p_{ijk}(b_1, b_2, b_3, \lambda), \ b_l \in I\!R\}$. We call D_{ijk} as *irregular triangular prism* since the edges $v_l(\lambda)$, $\lambda \in I_{ijk}$, $l = i, j, k$, are not parallel, and we call (b_1, b_2, b_3, λ) as D_{ijk} coordinate of p if $p = b_1v_i(\lambda) + b_2v_j(\lambda) + b_3v_k(\lambda)$.

Construction of the boundary curves, functions and gradients Let N_i be the normal of vertex v_i and $N_{ijk}^{(i)}$ be the normals that is attached to the vertex v_i of the triangle $[v_iv_jv_k]$. Then we make the assumption:

Assumption. *For each vertex v_i, $N_i^T N_{ijk}^{(i)} > 0$ for any triangle $[v_iv_jv_k] \in \mathcal{T}$.*

In order to define a at least C^0 function over $\mathcal{D} = \sum_{i,j,k} D_{ijk}$, we adjust the length of normal N_{ijk}, so that

$$N_i^T N_{ijk}^{(i)} = \|N_i\|^2 \tag{112}$$

Let $[v_iv_j]$ be an edge of the triangulation and $[v_iv_jv_k]$ and $[v_iv_jv_l]$ be the adjacent triangles of the edge. Then we define

$$N_{ij}^{(i)} = \frac{1}{2}\left(N_{ijk}^{(i)} + N_{ijl}^{(i)}\right), \quad N_{ij}^{(j)} = \frac{1}{2}\left(N_{ijk}^{(j)} + N_{ijl}^{(j)}\right)$$

In the following, we use the notations:

$$\begin{array}{lll} F_i = F|_{V_i}, & \nabla F_i = \nabla F|_{V_i}, & v_i \in \mathcal{V} \\ F_{ij} = F|_{H_{ij}}, & \nabla F_{ij} = \nabla F|_{H_{ij}}, & [v_iv_j] \in \partial\mathcal{T} \\ F_{ijk} = F|_{D_{ijk}}, & \nabla F_{ijk} = \nabla F|_{D_{ijk}}, & [v_iv_jv_k] \in \mathcal{T} \end{array}$$

where $V_i = \{v_i(\lambda) : \ \lambda \in (-\infty, \infty)\} \cap \mathcal{D}$, $H_{ij} = \{h_{ij}(t,\lambda) : \ h_{ij}(t,\lambda) = (1-t)v_i(\lambda) + tv_j(\lambda), \quad t \in [0,1], \ \lambda \in (-\infty,\infty)\} \cap \mathcal{D}$ and $\partial\mathcal{T}$ denotes all the edges of the triangulation $\mathcal{T}$. For the construction of F_{ijk}, there is no restriction on the interval I_{ijk}. However, I_{ijk} should be as large as possible, such that the surface $F_{ijk} = 0$ is contained in D_{ijk}. The problem of determining the largest I_{ijk} is considered in section 5. In this section, we mainly construct the function F_{ijk}. This will be done by firstly constructing function values and gradients on the edges and faces of D_{ijk} and then using a transfinite triangle interpolation to extend the function to the interior of D_{ijk}.

On the edges V_l, $l = i, j, k$, of D_{ijk}, the function value is defined by

$$F(v_l(\lambda)) = \|N_l\|\lambda = n_l^T N_{ijk}^{(l)}\lambda, \quad l = i, j, k, \quad n_l = N_l/\|N_l\| \tag{113}$$

On the boundary face H_{lm} of D_{ijk}, the function value is defined by cubic Hermite interpolation along the line segment $[v_l(\lambda)v_m(\lambda)] = \{h_{lm}(t,\lambda) : t \in [0,1]\}$. That is, interpolates the function values $F(v_l(\lambda))$, $F(v_m(\lambda))$ and the derivatives

$$[v_m(\lambda) - v_l(\lambda)]^T N_{lm}^{(l)}, \quad [v_m(\lambda) - v_l(\lambda)]^T N_{lm}^{(m)} \tag{114}$$

at the end points $v_l(\lambda)$ and $v_m(\lambda)$, respectively, plus an additional free quartic term. This leads to

$$F(h_{lm}(t,\lambda)) = \phi_{lm}(t) + \psi_{lm}(t)\lambda, \quad t \in [0,1], \quad \lambda \in I_{ijk} \tag{115}$$

with

$$\phi_{lm}(t) = (v_m - v_l)^T N_{lm}^{(l)} t(1-t)^2 + (v_l - v_m)^T N_{lm}^{(m)} t^2(1-t) + \theta_{lm} t^2(1-t)^2 \tag{116}$$

$$\begin{aligned} \psi_{lm}(t) = {} & n_l^T N_{lm}^{(l)}(1-t)^3 + (n_m + 2n_l)^T N_{lm}^{(l)}\ t(1-t)^2 + \\ & + (n_l + 2n_m)^T N_{lm}^{(m)}\ t^2(1-t) + n_m^T N_{lm}^{(m)} t^3 \end{aligned} \tag{117}$$

where θ_{lm} is a free parameter that is used to control the shape of the boundary curve:

$$C_{lm}(t, \theta_{lm}) := v_l + t(v_m - v_l) + \lambda(t)[n_l + t(n_m - n_l)] \tag{118}$$

which will be the boundary of our constructed surface patch in D_{ijk}, where

$$\lambda(t) = -\frac{\phi_{lm}(t)}{\psi_{lm}(t)} = \lambda_1(t)\theta_{lm} + \lambda_0(t)$$

satisfies the equation $F(h_{lm}(t,\lambda)) = 0$. For any θ_{lm}, the curve $C_{lm}(t,\theta_{lm})$ passes through the vertices v_l and v_m and perpendicular to the normal n_l and n_m at the vertices.

When θ_{lm} increase(or decrease), the boundary curve $C_{lm}(t,\theta_{lm})$ goes away from the line segment $[v_lv_m]$ in the normal(or opposite normal) direction. The default choice of this parameter could be to take $\theta_{lm} = 0$ or to make some energy to be minimal. For example,

$$\int_0^1 \|C''_{lm}(t,\theta_{lm})\|dt = \min \tag{119}$$

or

$$\int_0^1 k(t,\theta_{lm})dt = \min \tag{120}$$

where $k(t,\theta_{lm})$ is the curvature of $C_{lm}(t,\theta_{lm})$, or

$$\max_{t\in[0,1]} k(t,\theta_{lm}) = \min \tag{121}$$

To avoid produce bumpy surface, we prefer to use (121).

In order to define the gradients on the boundary faces H_{lm}, we take three different directions in $I\!R^3$ as follows:

$$d_1 = v_m(\lambda) - v_l(\lambda), \quad d_2 = (1-t)n_l + tn_m, \quad d_3 = d_1 \times d_2 \tag{122}$$

It is easy to see that

$$D_{d_1}F(h_{lm}(t,\lambda)) = \frac{\partial F(h_{lm}(t,\lambda))}{\partial t} = \phi'_{lm}(t) + \psi'_{lm}(t)\lambda \tag{123}$$

$$D_{d_2}F(h_{lm}(t,\lambda)) = \frac{\partial F(h_{lm}(t,\lambda))}{\partial \lambda} = \psi_{lm}(t) \tag{124}$$

We artificially define

$$D_{d_3}F(h_{lm}(t,\lambda)) = (1-t)N^{(l)}_{ijk} + tN^{(m)}_{ijk}$$

From the equations

$$d_i^T \nabla F_{lm} = D_{d_i}F_{lm}, \quad i = 1,2,3$$

we have

$$\nabla F_{lm} = \left\{[d_1,d_2,d_3]^T\right\}^{-1}[D_{d_1}F_{lm}, D_{d_2}F_{lm}, D_{d_3}F_{lm}]^T$$

Since

$$[d_1,d_2,d_3]^T[d_1+\alpha d_2, d_2+\beta d_1, d_3] = \begin{bmatrix} \|d_1\|^2+\alpha d_1^T d_2 & 0 & 0 \\ 0 & \|d_2\|^2+\beta d_1^T d_2 & 0 \\ 0 & 0 & \|d_3\|^2 \end{bmatrix}$$

where $\alpha = -d_1^T d_2/\|d_2\|^2$, $\beta = -d_1^T d_2/\|d_1\|^2$, and $\|d_1\|^2\|d_2\|^2 - (d_1^T d_2)^2 = \|d_3\|^2$, we have

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

Hence

$$\nabla F_{lm} = \frac{[d_1\|d_2\|^2 - d_2(d_1^T d_2)]D_{d_1}F_{lm} + [d_2\|d_1\|^2 - d_1(d_1^T d_2)]D_{d_2}F_{lm} + d_3 D_{d_3}F_{lm}}{\|d_1\|^2\|d_2\|^2 - (d_1^T d_2)^2} \tag{125}$$

which could be written as

$$\nabla F(h_{lm}(t,\lambda)) = \frac{P_5(t) + Q_5(t)\lambda + R_5(t)\lambda^2}{p_2(t) + q_1(t)\lambda + r_0(t)\lambda^2}$$

where P_5, Q_5, R_5 are polynomials of degree 5 and p_2, q_1, r_0 are polynomials of degree 2, 1, 0, respectively.

Construction of the function in the interior Having the function values and gradients on the boundary of D_{ijk}, we can now apply any transfinite triangular interpolant over the triangle $\Delta_{ijk}(\lambda)$ to construct the function F_{ijk}. We use the side-vertex scheme defined by Theorem 3.1 in [170] with some variations. The implementations show that the direct application of the scheme (3.9) in [170] leads to bad shaped surface. Hence we alter the equation (3.6) in [170] by introducing additional term. The following is the modified scheme for a typical triangle $[v_1v_2v_3]$:

$$F(p_{123}(b_1,b_2,b_3,\lambda)) = \frac{\sum_{i=1}^{3}\prod_{j=1,j\neq i}^{3} b_j^2\ D_i(b_1,b_2,b_3,\lambda)}{\sum_{i=1}^{3}\prod_{j=1,j\neq i}^{3} b_j^2} + E(b_1,b_2,b_3,\lambda) \tag{126}$$

where $D_i(b_1,b_2,b_3,\lambda)$ are defined by interpolating function values and derivative at $v_i(\lambda)$ and $\frac{b_j}{1-b_i}v_j(\lambda)+\frac{b_k}{1-b_i}v_k(\lambda)$:

$$\begin{aligned} D_i(b_1,b_2,b_3,\lambda) &= (1+2b_i)(1-b_i)^2F(h_{jk}(S_i,\lambda)) - b_i(1-b_i)[b_je_k^T(\lambda)+b_ke_j^T(\lambda)]\nabla F(h_{jk}(S_i,\lambda)) \\ &\quad + b_i^2(3-2b_i)N_i^Tn_i\lambda + b_i^2[b_je_k^T(\lambda)+b_ke_j^T(\lambda)]N_i + b_i^2(1-b_i)(b_j\theta_{ij}+b_k\theta_{ki}) \end{aligned} \tag{127}$$

where $(i,j,k)\in\{(1,2,3),(2,3,1),(3,1,2)\}$, $e_k(\lambda)=v_j(\lambda)-v_i(\lambda)$, $e_j(\lambda)=v_k(\lambda)-v_i(\lambda)$, $S_i=\frac{b_k}{b_j+b_k}$.

$$E(b_1,b_2,b_3,\lambda) = b_1^2b_2^2b_3^2\left[\sum_{l_1+l_2+l_3=l}(c_{l_1l_2l_3}+\lambda w_{l_1l_2l_3})B^l_{l_1l_2l_3}(b_1,b_2,b_3)\right]$$

is free. That is, $E(b_1,b_2,b_3,\lambda)$ has no influence on the function value and the first order partials of F at the boundary of D_{ijk} since it has the factor $b_1^2b_2^2b_3^2$. This term can be used to fit data in the volume D_{ijk} to get a better approximation. If there are no data available, $E(b_1,b_2,b_3,\lambda)$ could be taken as zero. The function $B^l_{l_1l_2l_3}(b_1,b_2,b_3)$ in $E(b_1,b_2,b_3,\lambda)$ is the well-known Bernstein-Bézier polynomial on a triangle.

Theorem. *The function F defined by $F|_{D_{ijk}}=F_{ijk}$ is C^1 over $\sum_{ijk}D_{ijk}$ everywhere except at the sharp edges where it is C^0 and it interpolates the function values $N^Tn_l\lambda$ at $v_l(\lambda)$, $l=1,\cdots,N$. At non-sharp vertex $v_l(\lambda)$, it interpolates also the gradient N_l.*

2.12.2 Optimized the Shape of the Surface Patch

In the construction of the function F, the term $E(b_1,b_2,b_3,\lambda)$ is free that could be used to control the shape of the surface. We choose this term so that the maximal value of the Gaussian curvature of the surface is minimal. Since the influence of $E(b_1,b_2,b_3,\lambda)$ is local, that is, this term in one triangle does not influent the shape of the surface patches defined on the other triangles. The optimization could be computed triangle by triangle. To reduce the computation cost, we minimize the Gaussian curvature on finite number of points, and the second derivatives are computed by divided difference of the first order derivatives.

2.12.3 Evaluation of the Surface Patch and Error Computation

Evaluation of the Surface Patch It is easy to see that the function F_{ijk} is C^1 within D_{ijk}. Hence we need only to consider the continuity of F on the face H_{lm}. First we show that the function is C^0.

On the edge V_l the function value is uniquely defined by (113), hence the function is continuous there. On the face H_{lm}, the function values defined by (115) use the values on the edge and

derivatives defined by (114). These values are edge dependent. That is, For the two triangles that share the edge, the function values for the two function at the common face H_{lm} are the same. Hence F is continuous.

Now we show the function is C^1 on the non-sharp edges. Let $[v_iv_jv_k]$ be any triangle of $\mathcal{T}$. Then for each (b_1, b_2, b_3), $b_i \geq 0$, $\sum b_i = 1$, determine $\lambda_{min} = \lambda_{min}(b_1, b_2, b_3)$ such that

$$\lambda_{min} = \min\{\lambda : F(p_{ijk}(b_1, b_2, b_3, \lambda)) = 0\} \tag{128}$$

Then the surface point is defined by $p = p_{ijk}(b_1, b_2, b_3, \lambda_{min})$. Since F is C^1 over $\sum D_{ijk}$, the constructed surface is G^1 smooth(tangent plane continuous). The main task here is to compute λ_{min} for each (b_1, b_2, b_3) with $b_i \geq 0$. It follows from (127) that $D_i(b_1, b_2, b_3, \lambda)$ is a rational function of λ in the form

$$F_0 + F_1\lambda + \frac{N_0 + N_1\lambda + N_2\lambda^2}{D_0 + D_1\lambda + D_2\lambda^2}$$

Hence $\phi(\lambda) := F(p_{ijk}(b_1, b_2, b_3, \lambda))$ is a rational function in λ. The minimal zero of $\phi(\lambda)$ is the required λ_{min}.

Although $\phi(\lambda) = 0$ is a nonlinear equation, the computation shows that $\phi(\lambda)$ is nearly a linear function. Hence, starting from $\lambda = 0$, find an interval with opposite signs of $\phi(\lambda)$ by searching, then using linear interpolation, an approximate minimal zero of $\phi(\lambda)$ is obtained by one or two time iterations.

Let $\tilde{F}_{ijk}(b_1, b_2, \lambda) = F(p_{ijk}(b_1, b_2, 1 - b_1 - b_2, \lambda))$. Then the surface normal is computed by the following equation

$$\nabla F_{ijk} = \left\{[v_1(\lambda) - v_3(\lambda),\ v_2(\lambda) - v_3(\lambda),\ b_1n_i + b_2n_j + b_3n_k]^T\right\}^{-1} \left[\frac{\partial \tilde{F}_{ijk}}{\partial b_1},\ \frac{\partial \tilde{F}_{ijk}}{\partial b_2},\ \frac{\partial \tilde{F}_{ijk}}{\partial \lambda}\right]^T \tag{129}$$

It should be noted that the 3×3 matrix in (129) is nonsingular if the points $v_1(\lambda), v_2(\lambda)$ and $v_3(\lambda)$ are not collinear and the vector $b_1n_i + b_2n_j + b_3n_k$ is not parallel to the plane $P_{ijk}(\lambda)$.

Error Computation After the decimation step of our algorithm(step 4). The points that are not the vertices of the decimated mesh are grouped into the volume D_{ijk} for the triangle $[v_iv_jv_k]$. The surface patch for this triangle provide an approximation to these points. Now we consider the computation of the approximation error. Let $p_{ijk}^{(l)},\ l = 1, 2, \cdots, m_{ijk}$ be the initial input points that in the volume D_{ijk}. The error bound is computed by the following steps:

a. For each point $p_{ijk}^{(l)}$, compute the volume coordinate $(b_1^{(l)}, b_2^{(l)}, b_3^{(l)}, \lambda^{(l)})$.

b. For $(b_1^{(l)}, b_2^{(l)}, b_3^{(l)})$ compute $\lambda_{min}(b_1^{(l)}, b_2^{(l)}, b_3^{(l)})$ as the (128)

Then error of the point $p_{ijk}^{(l)}$ to the surface is bounded by $|\lambda^{(l)} - \lambda_{min}(b_1^{(l)}, b_2^{(l)}, b_3^{(l)})|$, since

$$\begin{aligned}
\|p_{ijk}^{(l)} - p_{ijk}(b_1^{(l)}, b_2^{(l)}, b_3^{(l)}, \lambda_{min}(b_1^{(l)}, b_2^{(l)}, b_3^{(l)})\| &\leq b_1^{(l)}\|v_i(\lambda^{(l)}) - v_i(\lambda_{min}(b_1^{(l)}, b_2^{(l)}, b_3^{(l)})\| \\
&+b_2^{(l)}\|v_j(\lambda^{(l)}) - v_j(\lambda_{min}(b_1^{(l)}, b_2^{(l)}, b_3^{(l)})\| + b_3^{(l)}\|v_k(\lambda^{(l)}) - v_k(\lambda_{min}(b_1^{(l)}, b_2^{(l)}, b_3^{(l)})\| \\
&\leq |\lambda^{(l)}) - \lambda_{min}(b_1^{(l)}, b_2^{(l)}, b_3^{(l)})|(b_1^{(l)}\|n_1\| + b_2^{(l)}\|n_2\| + b_3^{(l)}\|n_3\| \\
&= |\lambda^{(l)} - \lambda_{min}(b_1^{(l)}, b_2^{(l)}, b_3^{(l)})|
\end{aligned}$$

2.12.4 The Condition of the Triangulation

In this section, we will give conditions on the triangulation $\mathcal{T}$ under which the function F and its gradient ∇F are well defined.

Definition(see definition 3.1 of [50]). *A triangle $\Delta_{ijk}(\lambda)$ is called non-degenerate, if its vertices are not collinear, the normal vectors n_i, n_j, n_k are not parallel to its plane $P_{ijk}(\lambda)$ and point to the same side of this plane.*

Theorem(see Theorem 4 of [50]). *Let $[v_i v_j v_k]$ be a non-degenerate triangle of $\mathcal{T}$ with respect to the normal n_i, n_j, n_k. Consider the real numbers $\lambda_1, \cdots, \lambda_s (s \leq 6)$ that solve one of the following three quadratic equations:*

$$\det(n_l, v_j(\lambda) - v_i(\lambda), v_k(\lambda) - v_i(\lambda)) = 0, \quad l = i, j, k$$

and define $a := max(-\infty, \{\lambda_i : \lambda_i < 0\}),\ b := min(\infty, \{\lambda_i : \lambda_i > 0\})$. Then $I_{ijk} = (a, b)$.

Theorem 4.2. *Let $[v_i v_j v_k]$ be a non-degenerate triangle of $\mathcal{T}$. Then both the function F_{ijk} and its gradient ∇F_{ijk} are well defined on D_{ijk}.*
Proof. Let $\lambda \in I_{ijk}$. Since n_i, n_j, n_k are not parallel to the plane $P_{ijk}(\lambda)$ and point to the same side of the plane

$$\det(n_l, v_j(\lambda) - v_i(\lambda), v_k(\lambda) - v_i(\lambda)) \neq 0, \quad l = i, j, k$$

and have the same sign. Hence for any $b_i \geq 0,\ b_j \geq 0,\ b_k \geq 0,\ b_i + b_j + b_k = 1$

$$\sum_{l=i,j,k} b_l \det\left(n_l, v_j(\lambda) - v_i(\lambda), v_k(\lambda) - v_i(\lambda)\right) = \det\left(\sum_{l=i,j,k} b_l n_l, v_j(\lambda) - v_i(\lambda), v_k(\lambda) - v_i(\lambda)\right) \neq 0$$

That is, the vector $\sum_{l=i,j,k} b_l n_l$ is not in parallel to the plane $P_{ijk}(\lambda)$. Hence the 3×3 matrix in (129) is nonsingular. This implies that the gradient ∇F_{ijk} is well defined if $\frac{\partial \tilde{F}_{ijk}}{\partial b_1}$, $\frac{\partial \tilde{F}_{ijk}}{\partial b_2}$, $\frac{\partial \tilde{F}_{ijk}}{\partial \lambda}$ are well defined. This is true if ∇F_{lm}, $(l, m) \in \{(i, j), (j, k), (k, i)\}$ are well defined. Hence we need to verify that the denominator of ∇F_{lm} is positive.

Since the vector $\sum_{l=i,j,k} b_l n_l$ is of course not parallel to the boundary of the triangle $\Delta_{ijk}(\lambda)$. Hence, the non-zero vectors d_1 and d_2 defined in (116) and (117) are not parallel. Therefore, the denominator of ∇F_{lm}, which is $\|d_1\|^2\|d_2\|^2 - (d_1^T d_2)^2$, is positive. $\Diamond$

2.12.5 Approximation by Parametric Rational Bezier

In many applications of surface modeling, to have both the implicit and parametric representations is important. In this section, we are intend to provide rational Bezier form represention for the A-patch defined in the prism.

Triangular Rational Bezier From the construction of the surface patches in section 4, we know that the surface patch in each volume could be expressed in the parametric form with $\Delta = \{(b_1, b_2, b_3) :\ \ b_i \geq 0;\ \ b_1 + b_2 + b_3 = 1\}$ be the parametric domain. However, this parametric form has no close form representaion and in general, no Bezier representaion for the patch exists. Hence, we appear to rational Bezier form approximation instead of exact conversion. With the increase of the degree of rational Bezier, the error of the approximation will decrease. Let d be the degree of the rational Bezier patch, the approximant is obtained as follows:

1. Generate a degree $d-1$ functional Bezier form approximation $\tilde{\lambda}(b_1, b_2, b_3)$ of $\lambda(b_1, b_2, b_3)$. This is a classical polynomial aprroximation problem on a triangle. To obtain global C^0 approximation, we generate this approximant in steps: **a**. Compute Bezier coefficients of $\tilde{\lambda}(b_1, b_2, b_3)$ on the three boundaries by interpolating $\lambda(b_1, b_2, b_3)$. bf b. The inner coefficients are defined by the least square fitting:

$$\int\int_{\Delta} \|\tilde{\lambda}(b_1, b_2, b_3) - \lambda(b_1, b_2, b_3)\ dS\| = min$$

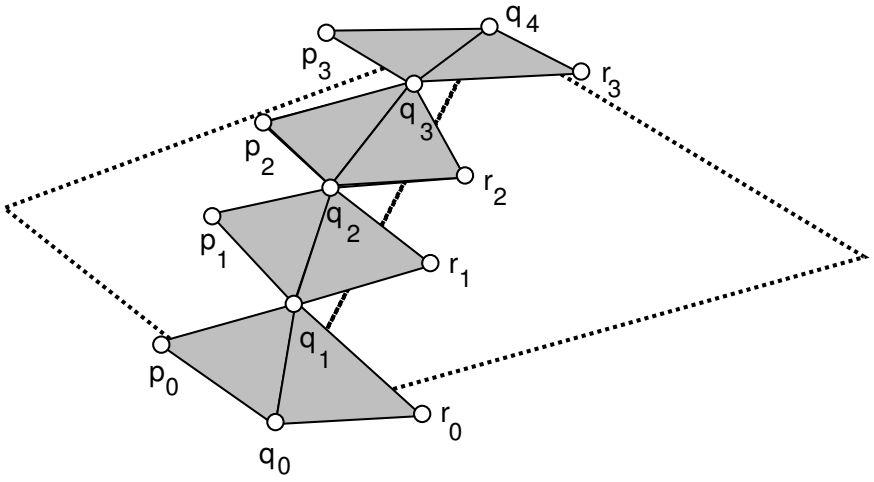


Figure 35: Related Bézier coefficients by G^1 continuity for $d = 4$.

The integration above is computed on regularly subdivided triangles and on each sub-triangle, a 6-points numerical quadrature rule(see [54], page 35) is employed.

2. Generate a G^0 degree d parametric form Bezier representation by

$$P_{ijk}^{(0)}(b_1, b_2, b_3) = b_1 v_i(\tilde{\lambda}(b_1, b_2, b_3)) + b_2 v_j(\tilde{\lambda}(b_1, b_2, b_3)) + b_3 v_k(\tilde{\lambda}(b_1, b_2, b_3))$$

The collection of $P_{ijk}^{(0)}$ define a continous(not smooth) parametric surface.

3. Generate a G^1 degree n parametric form Bezier coefficients. Let $P_{ijk}^{(0)}(b_1, b_2, b_3) = \sum_{l+m+n=d} \mathbf{b}_{lmn}^{(0)} B_{lmn}^{d}(b_1, b_2,$
Then we adjust the coefficients $\mathbf{b}_{lmn}^{(0)}$ for $m \leq 1$, or $n \leq 1$ or $l \leq 1$ so that the G^1 continuous condition

$$\begin{aligned} &\frac{i}{d}[[\alpha_1 p_{i-1} + (1-\alpha_1) r_{i-1}] - [[\beta_1 q_{i-1} + (1-\beta_1) q_i]] + \\ &(1 - \frac{i}{d})[[\alpha_0 - p_i + (1-\alpha_0) r_i] - [[\beta_0 q_i + (1-\beta_0) q_{i+1}]] = 0, \quad i = 0, 1, \cdots, d \end{aligned} \tag{130}$$

given by Farin in [108](see page 334-339), is satisfied. Using Farin's notation q_i represent the Bezier coefficients on the boundary (see Figure 35), r_i and p_i represent the Bezier coefficients near the boundary coefficients on two adjacent triangles.

The condition (130) is applied as follows: For fix q_0, q_1, p_0 and r_0, soving α_0 and β_0 from the equation (130) for $i = 0$. Similarly, α_1 and β_1 is sovled from (130) for $i = d$ and fixed q_3, q_4, p_3 and r_3. After $\alpha_0, \alpha_1, \beta_0$ and β_i are defined, the other equations ($i = 1, \cdots, d-1$) are used to solve other coefficients. For our problem, q_0 and q_4 are known, that are the vertices. To have the equation (130) for $i = 0$ have unique solution we need to adjuast the coeeficients q_1, p_0 and r_0, so that the four points q_0, q_1, p_0 and r_0 are coplaner. We adjust these coeeficients so that they lie on the boundaries of the prisms considered. For example, q_1 is adjusted as

$$q_1 + t((d-1)n_0 + n_4)/d$$

where n_0 and n_4 are the given normals at q_0 and q_4, respectively. For $d \geq 4$, the system (126) for $i = 1, \cdots, d-1$ is under-determined. It has $d-1$ equations and $3d-7$ unknowns. We solve this system by approximating the corresponding coefficients of $P_{ijk}^{(0)}$ that are defined in the last step. This solving stratige leads to a restricted least square problem

$$\begin{cases} MX - B = 0, \\ \|X - C\|^2 = min \end{cases}$$

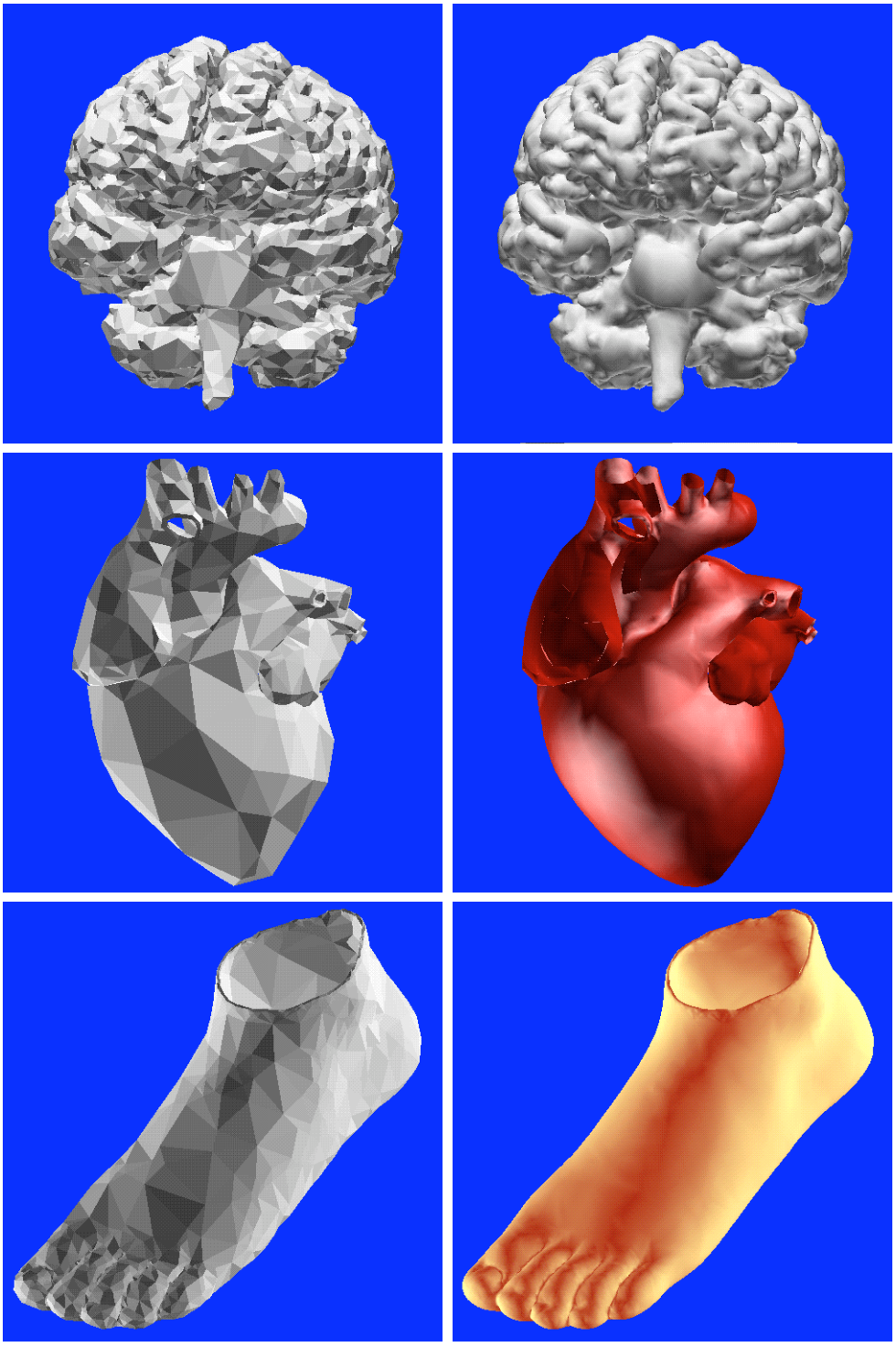


Figure 36: Adaptive feature of the reconstruction: The flat parts use less patches than the curved parts

where X is a vectors consists of unknowns. B is the left-handed side. C consists of the corresponding known coefficients of $P^{(0)}_{ijk}$. We solve this problem by singular-valued decomposition of the matrix M. The details are omitted.

4. Produce rational Bezier represention

$$P^{(1)}_{ijk}(b_1,b_2,b_3)=\frac{\sum_{l+m+n=d+2}\mathbf{b}_{lmn}B^d_{lmn}(b_1,b_2,b_3)}{\sum_{l+m+n=d+2}w_{lmn}B^d_{lmn}(b_1,b_2,b_3)},\quad \mathbf{b}_{lmn}\in\mathbb{R}^3,\quad w_{lmn}\in\mathbb{R} \tag{131}$$

It is should be noted that, solving (126) for each edge, the coefficients at the corner, that is p_1, r_1, p_2 and r_2 (they are called twist term) are doubly determined, we take their average as the required value. However, such defined twist term will not lead to G^1 surface. To satisfy the G^1 condition (126), a rational term

$$\frac{(b_1b_3\mathbf{b}^{(1)}_{d-2,1,1}+b_1b_2\mathbf{b}^{(2)}_{d-2,1,1})B^d_{d-2,1,1}+(b_1b_2\mathbf{b}^{(2)}_{1,d-2,1}+b_2b_3\mathbf{b}^{(0)}_{1,d-2,1})B^d_{1,d-2,1}+(b_2b_3\mathbf{b}^{(0)}_{1,1,d-2}+b_1b_3\mathbf{b}^{(1)}_{1,1,d-2})B^d_{1,1,d-2}}{b_2b_3+b_1b_3+b_1b_2}$$

is added to the Bezier part, where $\mathbf{b}^{(1)}_{d-2,1,1}$ is defined by G^1 condition on the edge 1 and minus the corresponding average value, similarly for others. The degree d Bezier plus the rational term could be written in the rational form (131). It should be note that the vertices of the triangle are base point of the surface. To eleminate these base points, we pertube the denorminator into $b_2b_3+b_1b_3+b_1b_2+\epsilon(b_1^2+b_2^2+b_3^2)$ and the numerator

$$\begin{aligned}
&(b_1b_3\mathbf{b}^{(1)}_{d-2,1,1}+b_1b_2\mathbf{b}^{(2)}_{d-2,1,1}+\frac{\epsilon}{2}b_1^2(\mathbf{b}^{(1)}_{d-2,1,1}+\mathbf{b}^{(2)}_{d-2,1,1})+\frac{\epsilon}{2}b_2^2\mathbf{b}^{(2)}_{d-2,1,1}++\frac{\epsilon}{2}b_3^2\mathbf{b}^{(1)}_{d-2,1,1})B^d_{d-2,1,1}\\
&+(b_1b_2\mathbf{b}^{(2)}_{1,d-2,1}+b_2b_3\mathbf{b}^{(0)}_{1,d-2,1}+\frac{\epsilon}{2}b_1^2\mathbf{b}^{(2)}_{1,d-2,1}+\frac{\epsilon}{2}b_2^2(\mathbf{b}^{(2)}_{1,d-2,1}+\mathbf{b}^{(0)}_{1,d-2,1})+\frac{\epsilon}{2}b_3^2\mathbf{b}^{(0)}_{1,d-2,1})B^d_{1,d-2,1}\\
&+(b_2b_3\mathbf{b}^{(0)}_{1,1,d-2}+b_1b_3\mathbf{b}^{(1)}_{1,1,d-2}+\frac{\epsilon}{2}b_1^2\mathbf{b}^{(1)}_{1,1,d-2}+\frac{\epsilon}{2}b_2^2\mathbf{b}^{(0)}_{1,1,d-2}+\frac{\epsilon}{2}b_3^2(\mathbf{b}^{(0)}_{1,1,d-2}+\mathbf{b}^{(1)}_{1,1,d-2}))B^d_{1,1,d-2}
\end{aligned}$$

where $\epsilon>0$ is a small number which chosen so that the function value is not affected by this ϵ for fixed word-length computation. Hence, the pertubation has no infflunce on the patch in practice.

Tensor Rational Bezier Since many CAGD system support tensor product form spline function but not the triangular form Bezier, it would be convienient to convert the rational Bezier into tensor form. For instance, in IGES specification, there is no triangular Bezier form. This conversion is achieved by introducing a nonlinear transfom to convert the triangle, say $[p_1p_2p_3]$ into the unit square $[0,1]\times[0,1]$. Let

$$\begin{cases} b_1=(1-t)(1-s)\\ b_2=(1-t)s\\ b_3=t \end{cases}$$

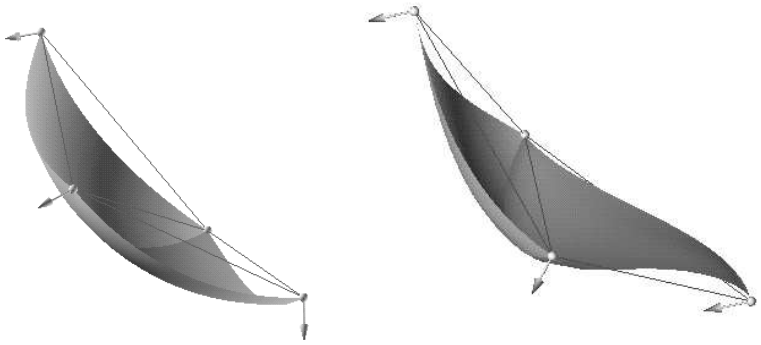


Figure 37: Left: Two convex faces. Right: Convex face and non-convex face

Then substitute (b_1, b_2, b_3) into Bezier form we could a tensor product form as follows.

$$\begin{aligned}
F(b_1, b_2, b_3) &= \sum_{i+j+k=n} \beta_{ijk} \frac{n!}{i!j!k!} b_1^i b_2^j b_3^k \\
&= \sum_{k=0}^{n} \sum_{i+j=n-k} \beta_{ijk} \frac{n!}{i!j!k!} b_1^i b_2^j b_3^k \\
&= \sum_{k=0}^{n} B_k^n(t) \sum_{i+j=n-k} \beta_{ijk} B_j^{n-k}(s) \\
&= \sum_{k=0}^{n} B_k^n(t) \sum_{j=0}^{n-k} \beta_{n-k-j,jk} B_j^{n-k}(s) \\
&= \sum_{k=0}^{n} B_k^n(t) \sum_{j=0}^{n} \tilde{\beta}_{kj} B_j^n(s)
\end{aligned}$$

where $\sum_{j=0}^{n} \tilde{\beta}_{kj} B_j^n(s)$ is derived from $\sum_{j=0}^{n-k} \beta_{n-k-j,jk} B_j^{n-k}(s)$ by degree elevation.

Examples In the study of using implicit surface patches for modeling a surface triangulation, we have been constantly seeking for the approaches of using one triangular patch for each face of

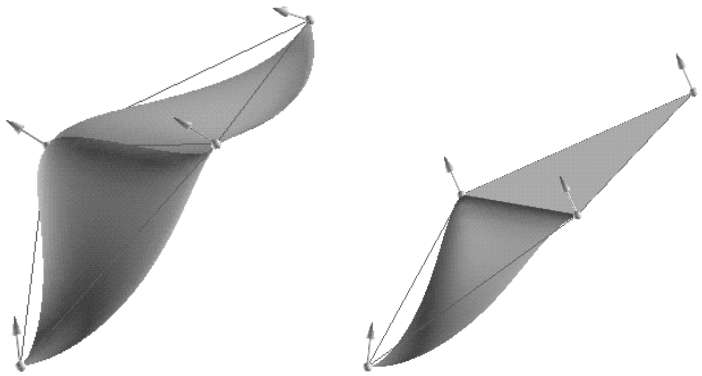


Figure 38: Left: Two non-convex faces. Right: Zero-convex face and non-convex face

the triangulation. The technique presented in has ours wish fulfilled. Comparing with the earlier scheme of A-patch in [85, 87, 122, 18], the present scheme use much less number of patches. For example, from Euler formula $v + f - e = 2$ for a triangulation of a closed surface with genus zero, we have $f = 2v - 4$, $e = 3v - 6$, where v, e and f represent the the numbers of vertices, edges and faces, respectively. Since the early approaches uses two or four pieces of surface patch for each edge, and one or two pieces of patch for each face. Hence the sum of the patch number is at least $8v - 16$ and as large as $2(8v - 16)$. Therefore, the ratio of patch numbers is in the range $[4 : 1, 8 : 1]$.

Furthermore, the proposed approach has the following features:

1. It is adaptive.
2. Could model sharp feature.
3. Error is easy to compute.

The implementation of the test show another attractive feature of the approach. That is, the evaluation of the surface is quit easy. It is approximately to solve a linear equation for evaluating one surface point. More importantly, the constructed surface patches have nice shape.

Figure 5.1-5.4 show the different join configuration of two patches. In Figure 5.1, the given two faces $[p_1p_2p_3]$ and $[p_1p_2p_4]$ are convex(see [18] for the definition of the convexity of an edge or face), where $p_1 = (-2, 0, 0)$, $p_2 = (2, 0, 0)$, $p_3 = (0, 4, -1)$ and $p_4 = (0, -4, -1)$. The corresponding normals are chosen as $n_1 = (-1, 0, 1)$, $n_2 = (1, 0, 1)$, $n_3 = (0, 1, 1)$ and $n_4 = (0, -1, 1)$. In Figure 5.2, the vertices are the same as in Figure 5.1. The normal n_3 n_4 are replaced by $(0, 1, 1.5)$ and $(0, 1, 1.5)$. Hence face $[p_1p_2p_3]$ is convex and face $[p_1p_2p_4]$ is non-convex. In Figure 5.3, the vertices are the same as before, but the normals are chosen as $n_1 = n_2 = (1, 0, 1)$, $n_3 = (0, -1, 1.5)$, and $n_4 = (0, 1, 1.5)$. Hence the two faces are non-convex and the common edge $[p_1p_2]$ is also non-convex. Figure 5.4 shows a triangular patch join a plane smoothly, where $p_1 = (-2, 0, 0)$, $p_2 = (2, 0, 0)$, $p_3 = (0, 4, -1)$ and $p_4 = (0, -4, 0)$. The normals are chosen as $n_1 = n_2 = n_4 = (0, 0, 1)$ and $n_3 = (0, -1, 1.5)$. Hence, the face $[p_1p_2p_3]$ is non-convex and the face $[p_1p_2p_4]$ is zero convex. As an application of the scheme in surface reconstruction, Figure 5.6 and 5.8 show the composite G^1 smooth surfaces constructed from the triangulation shown in Figure 5.5 and 5.7, respectively.

3 Filling Holes and Blending

3.1 G^1 Spline Surface Construction By Geometric Partial Differential Equations

A surface satisfying a geometric partial differential equation (GPDE) is referred to as a GPDE surface. GPDE surfaces are favorable in the areas of computer graphics and computer aided geometric design since they often possess certain optimal properties. For instance, the GPDE

surface which is a steady state solution of mean curvature flow has minimal surface area and the Willmore surface which is a steady state solution of Willmore flow has minimal total squared mean curvature. However, the construction of GPDE surfaces is not a trivial task, because GPDEs are highly nonlinear in general. In recent years, the constructions of discrete GPDE surfaces (typically using triangular meshes) have been studied intensively using divided difference-like discretization techniques or even finite element methods. It is worth noting that there are only very few published results on construction techniques for continuous GPDE surfaces, using Bernstein-Bézier surface, B-spline surfaces and NURBS surfaces.

As a subset of NURBS surfaces and an extension of Bernstein-Bézier surfaces, the class of B-spline surfaces has become the most popular for shape design and geometric modeling. B-spline surfaces are widely used in the fields of computer graphics, animation and computer added design (CAD) and computer added manufacturing (CAM). The aim of this work is to establish an efficient, reliable and mathematically correct method for constructing GPDE B-spline surfaces with specified G^1 boundary conditions, using mixed finite element methods.

Divided difference-like techniques are generally more efficient and easier to implement. However, they lack rigorous convergence analysis. In contrast, the finite element methods, while computationally more intensive, have sound and well established mathematical theory. More importantly, finite element method can handle G^1 boundary condition much more naturally than the divided difference-like method. In divided difference-like method, the length of the tangent vectors on the surface boundaries have to be taken into account, while in the finite element method presented the lengths of the tangent vectors have no effect on the constructed surfaces, and hence is our solution method of choice.

3.1.1 Preliminaries and Notations

This section introduce the necessary background material and notations, including definitions and relations amongst a few geometric differential operators, curvatures and spline functions.

Differential Geometry of Parametric Surface Let $\mathcal{S} := \{\mathbf{x}(u^1, u^2) \in \mathbb{R}^3 : (u^1, u^2) \in \mathcal{D} \subset \mathbb{R}^2\}$ be a parametric surface. For simplicity, we assume it is sufficiently smooth and orientable. Let $g_{\alpha\beta} = \langle \mathbf{x}_{u^\alpha}, \mathbf{x}_{u^\beta} \rangle$ and $b_{\alpha\beta} = \langle \mathbf{n}, \mathbf{x}_{u^\alpha u^\beta} \rangle$ be the coefficients of the first and the second fundamental forms of $\mathcal{S}$ with

$$\mathbf{x}_{u^\alpha} = \frac{\partial \mathbf{x}}{\partial u^\alpha},\ \mathbf{x}_{u^\alpha u^\beta} = \frac{\partial^2 \mathbf{x}}{\partial u^\alpha \partial u^\beta},\ \alpha, \beta = 1, 2,$$
$$\mathbf{n} = (\mathbf{x}_u \times \mathbf{x}_v)/\|\mathbf{x}_u \times \mathbf{x}_v\|, \quad (u, v) := (u^1, u^2),$$

where $\langle \cdot, \cdot \rangle$, $\| \cdot \|$, $\cdot \times \cdot$ stand for inner product, Euclidean norm and cross product in $\mathbb{R}^3$ respectively. Let

$$[\, g^{\alpha\beta} \,] = [\, g_{\alpha\beta} \,]^{-1}, \quad [\, b^{\alpha\beta} \,] = [\, b_{\alpha\beta} \,]^{-1}, \quad g = \det[\, g_{\alpha\beta} \,], \quad b = \det[\, b_{\alpha\beta} \,].$$

Curvatures. To introduce the notions of mean curvature and Gaussian curvature, we use the concept of *Weingarten map* or *shape operator* (see [95]). The shape operator of surface $\mathcal{S}$ is a self-adjoint linear map on the tangent space $T_{\mathbf{x}}\mathcal{S} := \text{span}\{\mathbf{x}_u, \mathbf{x}_v\}$ defined by

$$\mathcal{W} : T_{\mathbf{x}}\mathcal{S} \to T_{\mathbf{x}}\mathcal{S},$$

such that

$$\mathcal{W}(\mathbf{x}_u) = -\mathbf{n}_u, \quad \mathcal{W}(\mathbf{x}_v) = -\mathbf{n}_v. \tag{132}$$

We can easily represent this linear map by a 2×2 matrix $S = [\, b_{\alpha\beta}\,][\, g^{\alpha\beta}\,]$. In particular,

$$[\mathbf{n}_u,\ \mathbf{n}_v] = -[\mathbf{x}_u,\ \mathbf{x}_v]S^T \tag{133}$$

is valid. The eigenvalues k_1 and k_2 of S are *principal curvatures* of $\mathcal{S}$ and their arithmetic average and product are the *mean curvature* H and the *Gaussian curvature* K, respectively. That is

$$H = \frac{k_1 + k_2}{2} = \frac{\mathrm{tr}(S)}{2}, \qquad K = k_1 k_2 = \det(S).$$

Let $\mathbf{H} = H\mathbf{n}$ and $\mathbf{K} = K\mathbf{n}$, they are referred to as mean curvature vector and Gaussian curvature vector, respectively. Now we introduce a few geometric differential operators.

Tangential gradient opertor. Suppose $f \in C^1(\mathcal{S})$ then the tangential gradient operator ∇ acting on f is defined as

$$\nabla f \ = \ [\mathbf{x}_u,\ \mathbf{x}_v][\, g^{\alpha\beta}\,][f_u,\ f_v]^T \in \mathbb{R}^3. \tag{134}$$

For a vector-valued function $\mathbf{f} = [f_1, \cdots, f_k]^T \in C^1(\mathcal{S})^k$, we define its gradient by

$$\nabla \mathbf{f} = [\nabla f_1, \cdots, \nabla f_k] \in \mathbb{R}^{3\times k}.$$

It is easy to see that

$$\nabla \mathbf{x} \ = \ [\mathbf{x}_u,\ \mathbf{x}_v][g^{\alpha\beta}][\mathbf{x}_u,\ \mathbf{x}_v]^T, \tag{135}$$

$$\nabla \mathbf{n} \ = \ -[\mathbf{x}_u,\ \mathbf{x}_v][g^{\alpha\beta}]S[\mathbf{x}_u,\ \mathbf{x}_v]^T, \tag{136}$$

and both $\nabla\mathbf{x}$ and $\nabla\mathbf{n}$ are symmetric 3×3 matrices.

The second tangential operator. Let $f \in C^1(\mathcal{S})$. Then the second tangential operator $\Diamond$ acting on f is defined as

$$\Diamond f = [\mathbf{x}_u, \mathbf{x}_v][h_{\alpha\beta}][f_u, f_v]^T \in \mathbb{R}^3. \tag{137}$$

where

$$[h_{\alpha\beta}] = \frac{1}{g}\begin{bmatrix} b_{22} & -b_{12} \\ -b_{12} & b_{11} \end{bmatrix}. \tag{138}$$

The third tangential operator. Let $f \in C^1(\mathcal{S})$. Then the third tangential operator $\oslash$ acting on f is defined as

$$\oslash f = [\mathbf{x}_u, \mathbf{x}_v][\, g^{\alpha\beta}\,]S[f_u, f_v]^{\mathrm{T}} \in \mathbb{R}^3.$$

The three tangential operators introduced above are in the tangent space $T_{\mathbf{x}}\mathcal{S}$, they are linearly dependent. In fact, we have the following Lemma.

Lemma 3.1. *For any function $f \in C^1(\mathcal{S})$, we have*

$$2H\nabla f = \oslash f + \Diamond f. \tag{139}$$

Proof. From the definitions of the three tangential operators we know that if the equality

$$2H[g^{\alpha\beta}] = [g^{\alpha\beta}]S + [h_{\alpha\beta}]. \tag{140}$$

holds, then (139) is obviously valid. From a direct calculation, (140) could be easily derived.

Divergence operator. Suppose $\mathbf{v}$ is a smooth vector field on surface $\mathcal{S}$, then the divergence operator div_s acting on $\mathbf{v}$ is defined as

$$\mathrm{div}_s(\mathbf{v}) = \frac{1}{\sqrt{g}}\left[\frac{\partial}{\partial u}, \frac{\partial}{\partial v}\right]\left[\, \sqrt{g}\,[\, g^{\alpha\beta}\,]\,[\mathbf{x}_u, \mathbf{x}_v]^T\mathbf{v}\,\right]. \tag{141}$$

Laplace-Beltrami operator. Let $f \in C^2(\mathcal{S})$. Then the Laplace-Beltrami operator (LBO) Δ acting on f is defined as (see [95], p. 83)

$$\Delta f = \mathrm{div}_s(\nabla f).$$

Obviously, Δ is a second order differential operator. It is well known that LBO relates to the mean curvature vector via the equation: $\Delta \mathbf{x} = 2\mathbf{H}$.

Lemma 3.2 (Riemannian Divergence Theorem, see [66], p. 142)**.** *Let $\mathcal{S}$ be an orientable surface, Ω a subregion of $\mathcal{S}$ with a piecewise smooth boundary $\partial\Omega$. Let $\mathbf{n}_c \in T_{\mathbf{x}}\mathcal{S}$ ($\mathbf{x} \in \partial\Omega$) be the outward unit normal along the boundary $\partial\Omega$. Then for any given C^1 vector field $\mathbf{v}$ on $\mathcal{S}$, we have*

$$\int_\Omega \mathrm{div}_s(\mathbf{v})\mathrm{d}A = \int_{\partial\Omega} \langle \mathbf{v}, \mathbf{n}_c\rangle \mathrm{d}s.$$

Theorem 3.3 (Green's formula for LBO)**.** *Let $\mathcal{S}$ be an orientaable surface, Ω a subregion of $\mathcal{S}$ with a piecewise smooth boundary $\partial\Omega$. Let $\mathbf{n}_c \in T_{\mathbf{x}}\mathcal{S}$ ($\mathbf{x} \in \partial\Omega$) be the outward unit normal along the boundary $\partial\Omega$. Then for a given C^1 smooth vector field $\mathbf{v}$ on $\mathcal{S}$, we have*

$$\int_\Omega [\langle \mathbf{v}, \nabla f\rangle + f\,\mathrm{div}(\mathbf{v})]\mathrm{d}A = \int_{\partial\Omega} f\langle \mathbf{v}, \mathbf{n}_c\rangle \mathrm{d}s. \tag{142}$$

Proof. Taking $\mathbf{v}$ as $f\mathbf{v}$ in the Riemannian divergence theorem, we immediately obtain (142).

Spline surface There are several equivalent approaches to define spline functions, including divided difference of truncated power function (see [82, 210]), the blossoming method (see [194]) and Cox-Door's recursive formulas (see [81, 91]). We adopt the approach of recursive formulas as these are the easiest to program.

Definition. Given a positive integer m, nonnegative integer k and a knot sequence

$$u_0 \le \cdots \le u_i \le u_{i+1} \le u_{i+2} \le \cdots \le u_{m+2k}.$$

$U = \{u_0, \cdots, u_{m+2k}\}$ is referred to as knot vector. Then B-splines basis functions are defined as follows

$$\begin{cases} N_{i,0}(u) = \begin{cases} 1, & \text{for } \ u \in [u_i, u_{i+1}), \\ 0, & \text{otherwise,} \end{cases} \qquad i = 0, 1, \cdots, m+2k-1, \\ N_{i,k}(u) = \dfrac{u-u_i}{u_{i+k}-u_i} N_{i,k-1}(u) + \dfrac{u_{i+k+1}-u}{u_{i+k+1}-u_{i+1}} N_{i+1,k-1}(u), \quad i = 0, 1, \cdots, m+k-1, \\ \text{Assume } \ \dfrac{0}{0} = 0 \end{cases} \tag{143}$$

where i is the index of $N_{i,k}(u)$, k is the degree.

Spline surface. For given positive integers m, n and a nonnegative integer k, and knot vectors

$$U = \{u_0, \cdots, u_{m+2k}\}, \quad V = \{v_0, \cdots, v_{n+2k}\},$$

the degree k four-sided spline surface is defined as

$$\mathbf{x}(u,v) = \sum_{i=0}^{m+k-1} \sum_{j=0}^{n+k-1} \mathbf{p}_{ij} N_{i,k}(u) N_{j,k}(v), \quad (u,v) \in \Omega := [0,1]^2,$$

where $\mathbf{p}_{ij} \in \mathbb{R}^3$ are called control points of surface $\mathbf{x}(u,v)$. If $i = 0$ or m, $j = 0$ or n, $\mathbf{p}_{ij}$ is called boundary control points. Other control points are called inner control points. In order to have the spline surface be at least C^2 smooth, we take $k \ge 3$.

3.1.2 Construction GPDE Spline Surfaces

This section is devoted to the details of the construction of B-spline GPDE surfaces, including the formulation of GPDEs, and their variational forms.

GPDEs and their Variational Forms To construct G^1 smooth GPDE B-spline surface patch, we use three fourth order equations, namely surface diffusion flow (SDF), Willmore flow (WF) and quasi-surface flow (QSDF).

Surface Diffusion Flow

$$\frac{\partial \mathbf{x}}{\partial t} = -2\Delta H \mathbf{n}. \tag{144}$$

This flow is introduced by Mullins in 1957 (see [168]), to describe the interface motion law of growing crystal.

Willmore flow

$$\frac{\partial \mathbf{x}}{\partial t} = -\left[\Delta H + 2H(H^2 - K)\right]\mathbf{n}. \tag{145}$$

Quasi-surface diffusion flow

$$\frac{\partial \mathbf{x}}{\partial t} = -\Delta^2 \mathbf{x}. \tag{146}$$

This flow was introduced in [239], and is used in surface design. It is known that tangent flow on a surface does not alter the surface shape (see [105]). Hence if we remove the tangential flow portion of (146), we obtain a geometric flow

$$\frac{\partial \mathbf{x}}{\partial t} = -2(\Delta H - 4H^3 + 2HK)\mathbf{n}.$$

If $\mathcal{S}$ is a closed surface and A stands for its area, then by Green's formula we obtain (see [77, 203] for the change rates of the surface area and the enclosed volume of the evolved surface)

$$\frac{\mathrm{d}}{\mathrm{d}t}A(t) = -2\int_{\mathcal{S}(t)} \left[\|\nabla H\|^2 + 2H^2(2H^2 - K)\right] \mathrm{d}A \leq 0.$$

Hence, Quasi-surface diffusion flow is area diminishing. The shrinkage stops when $H \equiv 0$.

Next we present variational form formulations for the GPDE (144)–(146). The detailed derivations are given in the appendix.

The mixed variational form of (144) is: Find $(\mathbf{x}, \mathbf{y}) \in H^2(\mathcal{S})^3 \times H^1(\mathcal{S})^3$ such that

$$\begin{cases} \displaystyle\int_{\mathcal{S}} \frac{\partial \mathbf{x}}{\partial t}\phi \ \mathrm{d}A + 2\int_{\mathcal{S}} \left[\phi \oslash \mathbf{y} - \mathbf{n}(\nabla\phi)^{\mathrm{T}}\nabla\mathbf{y}\right]\mathbf{n} \ \mathrm{d}A = \mathbf{0}, & \forall \phi \in H_0^1(\mathcal{S}), \\ \displaystyle\int_{\mathcal{S}} \mathbf{y}\psi \ \mathrm{d}A + \frac{1}{2}\int_{\mathcal{S}} (\nabla\mathbf{x})^{\mathrm{T}}\nabla\psi \ \mathrm{d}A - \frac{1}{2}\int_{\partial\mathcal{S}} \mathbf{n}_c\psi \ \mathrm{d}s = \mathbf{0}, & \forall \psi \in H^1(\mathcal{S}), \\ \mathcal{S}(0) = \mathcal{S}_0, \ \ \partial\mathcal{S}(t) = \Gamma, \ \ \mathbf{n}_c(\mathbf{x}) = \mathbf{n}_c^{(\Gamma)}(\mathbf{x}), \ \forall \mathbf{x} \in \Gamma, \end{cases} \tag{147}$$

where $\mathbf{n}_c^{(\Gamma)}$ is the given co-normal on the boundary curve Γ. Similarly, the mixed variational form of Willmore flow (145) can be writen as: Find $(\mathbf{x}, \mathbf{y}) \in H^2(\mathcal{S})^3 \times H^1(\mathcal{S})^3$ such that

$$\begin{cases} \displaystyle\int_{\mathcal{S}} \frac{\partial \mathbf{x}}{\partial t}\phi \ \mathrm{d}A + \int_{\mathcal{S}} \left[\phi \oslash \mathbf{y} - \mathbf{n}(\nabla\phi)^{\mathrm{T}}\nabla\mathbf{y}\right]\mathbf{n} \ \mathrm{d}A \\ \qquad\qquad + \displaystyle\int_{\mathcal{S}} 2\mathbf{n}(H^2 - K)\phi\mathbf{n}^{\mathrm{T}}\mathbf{y} \ \mathrm{d}A = \mathbf{0}, & \forall \phi \in H_0^1(\mathcal{S}), \\ \displaystyle\int_{\mathcal{S}} \mathbf{y}\psi \ \mathrm{d}A + \frac{1}{2}\int_{\mathcal{S}} (\nabla\mathbf{x})^{\mathrm{T}}\nabla\psi \ \mathrm{d}A - \frac{1}{2}\int_{\partial\mathcal{S}} \mathbf{n}_c\psi \ \mathrm{d}s = \mathbf{0}, & \forall \psi \in H^1(\mathcal{S}), \\ \mathcal{S}(0) = \mathcal{S}_0, \ \partial\mathcal{S}(t) = \Gamma, \ \ \mathbf{n}_c(\mathbf{x}) = \mathbf{n}_c^{(\Gamma)}(\mathbf{x}), \ \forall \mathbf{x} \in \Gamma. \end{cases} \tag{148}$$

Finally, the mixed variational form of the quasi-surface diffusion flow is: Find $(\mathbf{x}, \mathbf{y}) \in H^2(\mathcal{S})^3 \times H^1(\mathcal{S})^3$ such that

$$\begin{cases} \displaystyle\int_{\mathcal{S}} \frac{\partial \mathbf{x}}{\partial t}\phi \, \mathrm{d}A - 2\int_{\mathcal{S}} (\nabla \mathbf{y})^{\mathrm{T}} \nabla \phi \, \mathrm{d}A = \mathbf{0}, & \forall \phi \in H_0^1(\mathcal{S}), \\ \displaystyle\int_{\mathcal{S}} \mathbf{y}\psi \, \mathrm{d}A + \frac{1}{2}\int_{\mathcal{S}} (\nabla \mathbf{x})^{\mathrm{T}} \nabla \psi \, \mathrm{d}A - \frac{1}{2}\int_{\partial \mathcal{S}} \mathbf{n}_c \psi \, \mathrm{d}s = \mathbf{0}, & \forall \psi \in H^1(\mathcal{S}), \\ \mathcal{S}(0) = \mathcal{S}_0, \ \partial\mathcal{S}(t) = \Gamma, \ \ \mathbf{n}_c(\mathbf{x}) = \mathbf{n}_c^{(\Gamma)}(\mathbf{x}), \ \forall \mathbf{x} \in \Gamma. \end{cases} \tag{149}$$

In section 3.1.2, systems (147)–(149) are numerically solved using mixed finite element methods.

Construction Steps of GPDE B-Spline Surfaces Problem Description. *Given four boundary curves and the cross tangents on the curves, we need to construct a four-sided B-spline surface* $\mathbf{x}(u,v) = \sum_{i=0}^{m+k-1}\sum_{j=0}^{n+k-1} \mathbf{p}_{ij} N_{i,k}(u) N_{j,k}(v)$, $(u,v) \in [0,1]^2$ *which interpolates the given boundary curves with the given tangents, and satisfies a specific GPDE in* $(0,1)^2$

The construction steps are outlined as follows. The details of each step are given in subsequent subsections.

1. **Construct initial inner control points**: The initial inner control points can be arbitrarily given. However, to have a fast convergence of the evolution process, good initial values are necessary. We use the Coons interpolation technique [80] using the boundary control points to construct inner control points (see section 3.1.2).
2. **Evolve the control points**: Use a specific GPDE to evolve the inner control points, till steady state solution is achieved. The evolution includes the following steps (see section 3.1.2):
   (a) Set a temporal step-size τ.
   (b) Discretize the specific GPDE in the spatial direction using a mixed finite element method (see section 3.1.2), to yield a system of nonlinear ordinary differential equations (ODEs).
   (c) Discretize the system of ODEs in the temporal direction using a semi-implicit scheme (see section 3.1.2), to yield a linear system.
   (d) Solve the linear system using an iterative approach, to yield a new approximate solution of the inner control points.
   (e) Check the termination conditions, if they are satisfied, stop the evolution, otherwise go back to step (b).

Construction of Initial Inner Control Points Good initial inner control points yield a more efficient evolution process. We utilize the construction technique of Coons surface patch (see [80]) from the boundary curves, to calculate our initial inner control points. The construction results are

$$\mathbf{p}_{ij}^{(0)} = \mathbf{p}_{ij}^{(u)} + \mathbf{p}_{ij}^{(v)} - \mathbf{p}_{ij}^{(uv)}, \quad i = 1, \cdots, m+k-2, \ \ j = 1, \cdots, n+k-2,$$

where (if $k = 3$),

$$\begin{aligned} \mathbf{p}_{ij}^{(u)} &= (1-\alpha_i)\mathbf{p}_{0j} + \alpha_i \mathbf{p}_{m+k-1,j}, \\ \mathbf{p}_{ij}^{(v)} &= (1-\beta_j)\mathbf{p}_{i0} + \beta_j \mathbf{p}_{i,n+k-1}, \\ \mathbf{p}_{ij}^{(uv)} &= (1-\alpha_i)(1-\beta_j)\mathbf{p}_{00} + \alpha_i(1-\beta_j)\mathbf{p}_{m+k-1,0} \\ &\quad + (1-\alpha_i)\beta_j \mathbf{p}_{0,n+k-1} + \alpha_i\beta_j \mathbf{p}_{m+k-1,n+k-1}, \end{aligned}$$

$$\alpha_i = \begin{cases} \frac{1}{3m}, & i = 1, \\ \frac{i-1}{m}, & i = 2, \cdots, m, \\ 1 - \frac{1}{3m}, & i = m+1, \end{cases} \qquad \beta_j = \begin{cases} \frac{1}{3n}, & j = 1, \\ \frac{j-1}{n}, & j = 2, \cdots, n, \\ 1 - \frac{1}{3n}, & j = n+1. \end{cases}$$

Construction of Inner Control Points For ease of description, we reorder the basis functions and control points into 1-dimensional arrays. First, we order the inner basis and control points as follows:

$$\begin{aligned} \phi_{(i-1)(n+k-2)+j-1}(u,v) &= N_{i,k}(u)B_{j,k}(v), \quad i = 1, \cdots, m+k-2,\ j = 1, \cdots, n+k-2, \\ \mathbf{x}_{(i-1)(n+k-2)+j-1} &= \mathbf{p}_{ij}, \qquad i = 1, \cdots, m+k-2,\ j = 1, \cdots, n+k-2. \end{aligned}$$

Then we order the basis and control points on the boundary. Starting with the indices $(0, 0)$, the basis functions and control points at the surface boundary are arranged in clockwise order. Using this ordering of the basis functions and control points, the spline surface patch is represented as

$$\mathbf{x}(u,v) = \sum_{j=0}^{n_0} \mathbf{x}_j \phi_j(u,v) + \sum_{j=n_0+1}^{n_1} \mathbf{x}_j \phi_j(u,v), \tag{150}$$

where

$$n_0 = (m+k-2)(n+k-2) - 1, \quad n_1 = (m+k)(n+k) - 1.$$

The mean curvature vector of the surface is represented approximately as

$$\mathbf{H}(u,v) = \sum_{j=0}^{n_1} \mathbf{h}_j \phi_j(u,v), \quad \mathbf{h}_j \in \mathbb{R}^3. \tag{151}$$

The coefficients $\mathbf{x}_j$ in the first term of (150) are unknowns, while the coefficients $\mathbf{x}_j$ in the second term are given. All the coefficients in (151) have to be calculated.

Spatial Discretizations **Spatial Discretizations of SDF and WF**. Substituting (150) and (151) into (147) and (148), and taking the test functions ϕ as $\phi_i (i = 0, \cdots, n_0)$, ψ and $\phi_i (i = 0, \cdots, n_1)$, and finally noting that $\frac{\partial \mathbf{x}_j(t)}{\partial t} = \mathbf{0}$ if $j > n_0$, we obtain the following matrix representations of (147) and (148):

$$\begin{cases} M_{n_0}^{(1)} \frac{\partial X_{n_0}(t)}{\partial t} + L_{n_1}^{(1)} H_{n_1}(t) = \mathbf{0}, \\ M_{n_1}^{(2)} H_{n_1}(t) + L_{n_1}^{(2)} X_{n_1}(t) = B, \end{cases} \tag{152}$$

where

$$\begin{aligned} X_j(t) &= [\mathbf{x}_0^{\mathrm{T}}(t), \cdots, \mathbf{x}_j^{\mathrm{T}}(t)]^{\mathrm{T}} \quad \in \mathbb{R}^{3(j+1)}, \\ H_{n_1}(t) &= [\mathbf{h}_0^{\mathrm{T}}(t), \cdots, \mathbf{h}_{n_1}^{\mathrm{T}}(t)]^{\mathrm{T}} \quad \in \mathbb{R}^{3(n_1+1)}, \\ B &= [\mathbf{b}_0^{\mathrm{T}}, \cdots, \mathbf{b}_{n_1}^{\mathrm{T}}]^{\mathrm{T}} \quad \in \mathbb{R}^{3(n_1+1)}, \\ M_{n_0}^{(1)} &= (m_{ij})_{ij=0}^{n_0,n_0}, \qquad M_{n_1}^{(2)} = (m_{ij})_{ij=0}^{n_1,n_1}, \\ L_{n_1}^{(1)} &= \left(l_{ij}^{(1)}\right)_{ij=0}^{n_0,n_1}, \qquad L_K^{(2)} = \left(l_{ij}^{(2)}\right)_{ij=0}^{n_1,K}, \end{aligned}$$

and

$$m_{ij} = \mathrm{I}_3 \int_{\mathcal{S}} \phi_i \phi_j \, \mathrm{d}A, \quad \mathrm{I}_3 \int_{\mathcal{S}} \phi_i \phi_j \, \mathrm{d}A, \tag{153}$$

$$\begin{aligned}
l_{ij}^{(1)} =& 2 \int_{\mathcal{S}} \left[\phi_i \oslash \phi_j - \mathbf{n} (\nabla \phi_i)^{\mathrm{T}} \nabla \phi_j \right] \mathbf{n}^{\mathrm{T}} \, \mathrm{d}A \ \text{ for SDF}, \\
l_{ij}^{(1)} =& \int_{\mathcal{S}} \left[\phi_i \big[\oslash \phi_j + 2\mathbf{n}(H^2 - K) \phi_j \right] - \mathbf{n} (\nabla \phi_i)^{\mathrm{T}} \nabla \phi_j \big] \mathbf{n}^{\mathrm{T}} \mathrm{d}A \ \text{ for WF}, \\
l_{ij}^{(2)} =& \frac{1}{2} \mathrm{I}_3 \int_{\mathcal{S}} \left[(\nabla \phi_i)^{\mathrm{T}} \nabla \phi_j \right] \, \mathrm{d}A, \\
\mathbf{b}_i =& \frac{1}{2} \int_{\Gamma} \mathbf{n}_c \phi_i \, \mathrm{d}s.
\end{aligned} \tag{154}$$

Moving the terms related to the known vertices $\mathbf{x}_{n_0+1}, \cdots, \mathbf{x}_{n_1}$ in equation(152) to the equations's right-hand side, we rewrite (152) as

$$\begin{cases}
M_{n_0}^{(1)} \frac{\partial X_{n_0}(t)}{\partial t} + L_{n_1}^{(1)} H_{n_1}(t) = \mathbf{0}, \\
M_{n_1}^{(2)} H_{n_1}(t) + L_{n_0}^{(2)} X_{n_0}(t) = B^{(2)}.
\end{cases} \tag{155}$$

Note that, matrices $M_{n_0}^{(1)}$ and $M_{n_1}^{(2)}$ are symmetric and positive definite. The integrals in defining the matrix elements are computed using Gaussian quadrature formulas over rectangles. The knots and weights of the Gaussian quadrature formulas can be found in [3, 240].

Spatial Discretizations of QSDF. Substituting (150) and (151) into (149), taking the test functions ϕ as $\phi_i (i = 1, \cdots, n_0)$ and ψ as $\phi_i (i = 1, \cdots, n_1)$, and finally noting that $\frac{\partial \mathbf{x}_j(t)}{\partial t} = \mathbf{0}$ if $j > n_0$, we obtain the following matrix form of (149):

$$\begin{cases}
M_{n_0}^{(3)} \frac{\partial \mathcal{X}_{n_0}(t)}{\partial t} + L_{n_1}^{(3)} \mathcal{H}_{n_1}(t) = \mathbf{0}, \\
M_{n_1}^{(4)} \mathcal{H}_{n_1}(t) + L_{n_1}^{(4)} \mathcal{X}_{n_1}(t) = \mathbf{B},
\end{cases} \tag{156}$$

where

$$\begin{aligned}
\mathcal{X}_j(t) &= [\mathbf{x}_0(t), \cdots, \mathbf{x}_j(t)]^{\mathrm{T}} && \in \mathbb{R}^{(j+1)\times 3}, \\
\mathcal{H}_{n_1}(t) &= [\mathbf{h}_0(t), \cdots, \mathbf{h}_{n_1}(t)]^{\mathrm{T}} && \in \mathbb{R}^{(n_1+1)\times 3}, \\
\mathbf{B} &= [\mathbf{b}_0, \cdots, \mathbf{b}_{n_1}]^{\mathrm{T}} && \in \mathbb{R}^{(n_1+1)\times 3} \\
M_{n_0}^{(3)} &= (m_{ij})_{ij=0}^{n_0,n_0}, && M_{n_1}^{(4)} = (m_{ij})_{ij=0}^{n_1,n_1}, \\
L_{n_1}^{(3)} &= \left(l_{ij}^{(3)} \right)_{ij=0}^{n_0,n_1}, && L_K^{(4)} = \left(l_{ij}^{(4)} \right)_{ij=0}^{n_1,K},
\end{aligned}$$

and

$$\begin{aligned}
m_{ij} =& \int_{\mathcal{S}} \phi_i \phi_j \, \mathrm{d}A, \\
l_{ij}^{(3)} =& -2 \int_{\mathcal{S}} (\nabla \phi_j)^{\mathrm{T}} \nabla \phi_i \, \mathrm{d}A, \\
l_{ij}^{(4)} =& \frac{1}{2} \int_{\mathcal{S}} (\nabla \phi_j)^{\mathrm{T}} \nabla \phi_i \, \mathrm{d}A, \\
\mathbf{b}_i =& \frac{1}{2} \int_{\Gamma} \mathbf{n}_c \phi_i \, \mathrm{d}s.
\end{aligned} \tag{157}$$

Moving the known terms in (156) to the equation's right-hand side, we obtain

$$\begin{cases} M_{n_0}^{(3)} \frac{\partial \mathcal{X}_{n_0}(t)}{\partial t} \ + \ L_{n_1}^{(3)} \mathcal{H}_{n_1}(t) = \mathbf{0}, \\ M_{n_1}^{(4)} \mathcal{H}_{n_1}(t) \ + \ L_{n_0}^{(4)} \mathcal{X}_{n_0}(t) = B^{(4)}. \end{cases} \tag{158}$$

Matrices $M_{n_0}^{(3)}$ $M_{n_1}^{(4)}$ are symmetric and positive definite. It should be pointed out that the size of the matrices in (155) are much larger than those in (158). But the right-handed side of (158) has three columns.

Boundary Conditions In the boundary integrals (154) and (157), $\mathbf{n}_c$ is the co-normal of the surface, it is infeasible to compute these co-normals from the previous approximation, since they do not satisfy the given boundary condition. The right way is to replace $\mathbf{n}_c$ with $\mathbf{n}_c^{(\Gamma)}$. That is

$$\mathbf{b}_i = \frac{1}{2} \int_\Gamma \mathbf{n}_c^{(\Gamma)} \phi_i \ \mathrm{d}s.$$

A good point to note is that the integral above does not involve the length of the tangent vector.

Temporal Direction Discretization We consider only the temporal direction discretization of SDF and WF. The temporal direction discretization of QSDF is similar. Suppose we have approximate solutions $X_{n_0}^{(k)} = X_{n_0}(t_k)$ and $H_{n_1}^{(k)} = H_{n_1}(t_k)$ at $t = t_k$. We obtain approximate solutions $X_{n_0}^{(k+1)}$ and $H_{n_1}^{(k+1)}$ at $t = t_{k+1} = t_k + \tau^{(k)}$ using a semi-implicit Euler scheme. Specifically, we use the following approximation

$$\frac{X_{n_0}(t_{k+1}) - X_{n_0}(t_k)}{\tau^{(k)}} \approx \frac{\partial X_{n_0}}{\partial t}.$$

The matrices $M^{(1)}$, $M^{(2)}$, $L^{(1)}$ and $L^{(2)}$ in (155) are computed using the surface data at $t = t_k$. This yields a linear system with $X_{n_0}^{(k+1)}$ and $H_{n_1}^{(k+1)}$ as unknowns:

$$\begin{bmatrix} M_{n_0}^{(1)} & \tau^{(k)} L_{n_1}^{(1)} \\ L_{n_0}^{(2)} & M_{n_1}^{(2)} \end{bmatrix} \begin{bmatrix} P_{n_0}^{(k+1)} \\ H_{n_1}^{(k+1)} \end{bmatrix} = \begin{bmatrix} \tau^{(k)} B^{(1)} + M_{n_0}^{(1)} P_{n_0}^{(k)} \\ B^{(2)} \end{bmatrix}$$

Though the matrices $M^{(1)}$ and $M^{(2)}$ are symmetric and positive definite, the total matrix is neither symmetric nor positive definite. However the coefficient matrix of this system is highly sparse, hence a stable iterative method for its solution is desirable. We use Saad's iterative method, namely GMRES (see [199]), to solve our sparse linear system. The numerical tests show that this iterative method works very well.

3.1.3 Implementation and Experimental Results

This section presents some of our experimental results including our results on approximation errors, our results of smoothness at the surface boundaries as well as illustrative examples of surface modeling.

Numerical Test of the Convergence Our numerical examples demonstrate that our GPDE solution is convergent. To illustrate this goal, we select some surface models which are the exact solution of certain GPDEs.

Taking $N = m + 2$, $m = n$, we compute the approximation errors of the evolved surface and the exact solution for $N = 8$. For $\tau = 0.001$, Table 1 lists the maximal errors of the discretized solutions and exact solution for $i = 0, 1, \cdots, 10$, where i is the index of the forward steps in the

temporal direction. Additionally, we compute the maximal errors of the steady state solutions of the used geometric PDE and the exact solution for $N = 3, 4, \cdots$. The computation results are given in Table 2. The initial surfaces are constructed by averaging the linear interpolations in the u and v directions. We do not use the Coons interpolation as suggested in section 3.1.2 to determine the initial surfaces simply because the initial surfaces constructed are too accurate for these geometric models. To show the strength of our approach, we rather choose a poor initial surfaces.

Example 3.4. *Let $\mathcal{S}$ be a sphere. Then $\Delta H = 0$, $H^2 - K = 0$. Hence $\mathcal{S}$ is a steady state solution of SDF and WF. We use six four-sided spline surface patches to approximate the sphere and compute the maximal errors of our six GPDE surfaces and the exact sphere. The compution results are listed in the second and third columns of Tables 1 and 2.*

Example 3.5. *Let $\mathcal{S}$ be a torus, which is formed by rotating a circle in the xy-plane with center $[0, R, 0]^{\mathrm{T}}$ and radius r around the y-axis:*

$$\mathbf{x}(u, v) = [r\sin(u), (R + r\cos(u))\sin(v), (R + r\cos(u))\cos(v)]^{\mathrm{T}}.$$

If $R/r = \sqrt{2}$, then $\mathcal{S}$ is a steady state solution of WF. The fourth column of Tables 1 and 2 gives the maximal approximation errors.

Example 3.6. *Let $\mathcal{S}$ be a cylinder. Then its mean curvature is a nonzero constant and its Gaussian curvature is zero. Hence, $\mathcal{S}$ is a steady state solution of the SDF. The fifth column of Tables 1 and 2 gives the maximal approximation errors.*

Example 3.7. *Let $\mathcal{S}$ be a minimal surface defined by the following equation*

$$\mathbf{x}(u, v) = [u,\ h(u)\cos(v),\ h(u)\sin(v)]^T, \tag{159}$$

where

$$h(u) = a\cosh(u/a + b), \quad a = 1.5, \quad b = 0.$$

Since the mean curvature of this surface is zero (but the Gaussian curvature is nonzero), $\mathcal{S}$ is a steady solution of SDF, QSDF and WF. The sixth, seventh and eighth columns of Tables 1 and 2 give the maximal approximation errors.

Table 1: The maximal errors of evolved spline surface for $N = 8$

i	Sphere-SD	Sphere-WM	Torus-WM	Sylinder-SD	Minimal-SD	Minimal-WM	Miniml-QSD
0	0.063509	0.063509	0.120674	0.103557	0.058233	0.058233	0.058233
1	0.052931	0.052803	0.101407	0.088043	0.010384	0.010475	0.007606
2	0.034853	0.034703	0.069213	0.062165	0.003105	0.003170	0.001538
3	0.021780	0.021653	0.045283	0.041831	0.001989	0.002068	0.001182
4	0.013532	0.013439	0.029703	0.027913	0.001647	0.001736	0.001179
5	0.008420	0.008357	0.019641	0.018657	0.001486	0.001585	0.001156
6	0.005250	0.005208	0.013076	0.012501	0.001388	0.001492	0.001130
7	0.003279	0.003252	0.008752	0.008390	0.001318	0.001425	0.001105
8	0.002051	0.002035	0.005884	0.005634	0.001266	0.001375	0.001080
9	0.001286	0.001276	0.003974	0.003781	0.001226	0.001336	0.001055

From the numerical results in Table 1 we can see that the maximal errors are monotonically decreasing as i increases. This shows that the proposed method is effective and reliable. The results in Table 2 further show that the approximation is very accurate.

Table 2: Maximal asymptotic errors of spline surface

N	Sphere-SD	Sphere-WM	Torus-WM	Sylinder-SD	Minimal-SD	Minimal-WM	Minimal-QSD
3	2.836e-04	2.852e-04	3.732e-03	5.906e-03	1.667e-03	1.734e-03	1.639e-03
4	3.173e-05	3.173e-05	7.197e-04	1.281e-03	1.363e-03	1.461e-03	1.566e-03
5	9.925e-05	1.037e-04	5.110e-04	5.854e-04	1.001e-03	1.141e-03	9.161e-04
6	3.524e-05	3.590e-05	2.076e-04	2.382e-04	1.360e-03	1.404e-03	8.859e-04
7	5.992e-05	6.525e-05	4.993e-04	3.637e-04	1.200e-03	1.300e-03	1.017e-03
8	3.537e-05	3.446e-05	3.001e-04	3.630e-04	9.168e-04	1.006e-03	7.773e-04
9	4.983e-05	4.944e-05	3.417e-04	2.844e-04	1.110e-03	1.244e-03	1.033e-03
10	3.998e-05	3.991e-05	2.554e-04	2.590e-04	9.843e-04	1.033e-03	9.958e-04
11	7.603e-05	7.943e-05	2.147e-04	2.571e-04	1.133e-03	1.249e-03	8.425e-04
12	7.163e-05	7.213e-05	1.939e-04	2.438e-04	1.001e-03	1.094e-03	7.688e-04

Visualization of Smoothness of Test Examples To visually examine the constructed B-spline surfaces satisfy the specified G^1 boundary condition, we present a few example figures. In Figs 39-42, the first figures show the boundary curves and the outer surfaces. The aim of showing the outer surfaces is to present normal or tangent information. The second figures show the outer surfaces and the constructed initial surfaces. As in section 3.1.3, the initial surfaces are constructed by averaging the linear interpolations in the u and v directions. The third figures give the the outer surfaces and the GPDE B-spline surfaces with $N = 8$.

The given boundary curves and and outward tangents are computed from the exact surfaces to be approximated. In Fig 39, the exact surface to be approximated is a minimal surface defined by (159), the constructed B-spline surface is on the domain $\{[x, y]^T \in \mathbb{R}^2 : x \in [0, 1],\ y \in [0, 1]\}$.

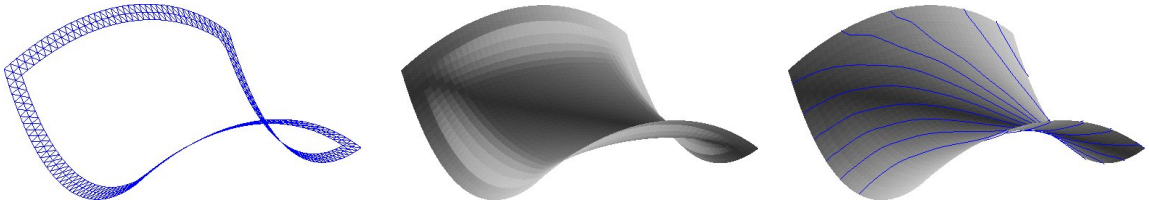


Figure 39: The first figure shows the inut boundary curves and the out-side surfaces. The second figure shows the outer surfaces and constructed initial surfaces. The third figure gives the the outer surfaces and GPDE spline surface using QSDF.

In Fig 40, the given surface to be approximated is a cylinder defined by

$$(y - \frac{1}{2})^2 + z^2 = \frac{1}{2}.$$

The constructed B-spline surface is corresponding to the domain $\{[x, y]^T \in \mathbb{R}^2 : x \in [0, 1],\ y \in [0, 1]\}$.

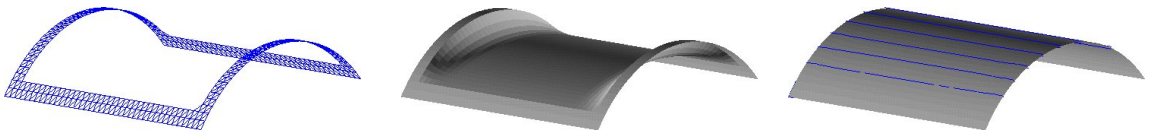


Figure 40: The first figure shows the inut boundary curves and the out-side surfaces. The second figure shows the outer surfaces and constructed initial surfaces. The third figure gives the the outer surfaces and GPDE spline surface using QSDF.

In Fig 41, the given surface is defined by

$$\mathbf{x}(u, v) = \left[u,\ v,\ \frac{1.25 + \cos(5.4v)}{6 + 6(3u - 1)^2}\right]^T.$$

The constructed B-spline surface is corresponding to the domain $\{[u,v]^T \in \mathbb{R}^2 : u \in [0,1],\ v \in [0,1]\}$.

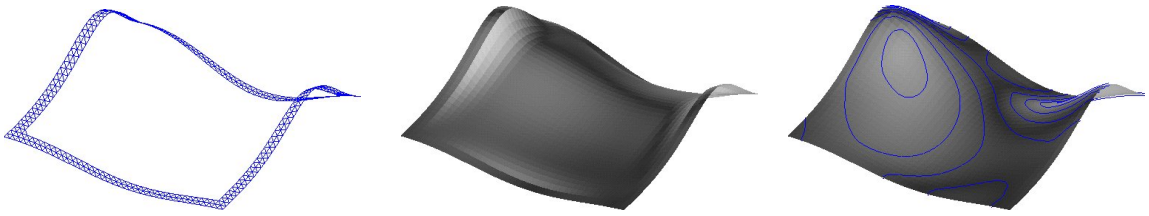


Figure 41: The first figure shows the inut boundary curves and the out-side surfaces. The second figure shows the outer surfaces and constructed initial surfaces. The third figure gives the the outer surfaces and GPDE spline surface using QSDF.

In Fig 42, the given surface to be approximated is

$$\mathbf{x}(u,v) = \left[u,\ v,\ e^{-\frac{81}{16}\left[(u-0.5)^2+(v-0.5)^2\right]}\right]^T.$$

The constructed surface is corresponding to the domain $\{[u,v]^T \in \mathbb{R}^2 : u \in [0,1],\ v \in [0,1]\}$.

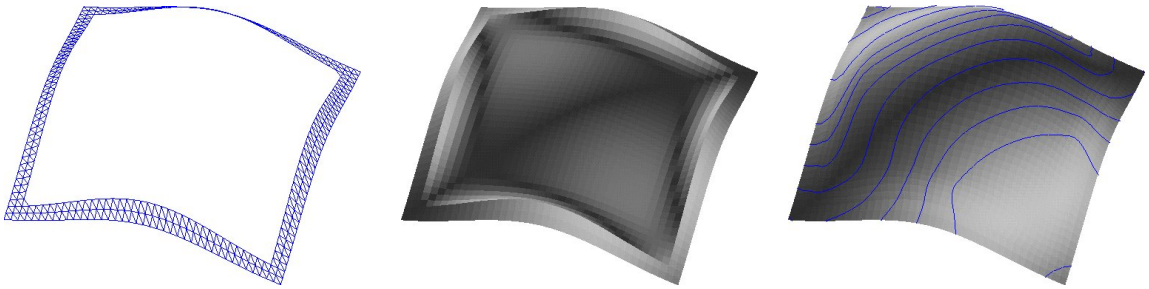


Figure 42: The first figure shows the inut boundary curves and the out-side surfaces. The second figure shows the outer surfaces and constructed initial surfaces. The third figure gives the the outer surfaces and GPDE spline surface using QSDF.

The surface meshes in these figures are generated by sampling uniformly the corresponding surfaces on the domain $[-2h, 1+2h] \times [-2h, 1+2h]$, where h is taken as $\frac{1}{64}$. When $[x,y]^{\mathrm{T}} \in [0,1] \times [0,1]$, we sample on the constructed surfaces. If $[x,y]^{\mathrm{T}} \notin [0,1] \times [0,1]$, we sample the given exact surfaces. The aim of such a sampling strategy is to show how smooth the outer surfaces join with the inner surfaces at the boundaries.

The last figures in Figs 39–42 clearly show that the constructed surfaces join the outer surfaces very smoothly. Hence G^1 condition is well satisfied.

3.1.4 Derivation of Variational Forms

Now we derive the variational form formulation for the equations (144). Let $H^1(\mathcal{S})$ be the Sobolev space on the surface $\mathcal{S}$, $H_0^1(\mathcal{S})$ a subspace of $H^1(\mathcal{S})$ consisting of the functions with compact support. Using Green's formulas, the systems (144) and (145) can be rewritten as weak forms. Let $y = H$, and $\phi \in H_0^1(\mathcal{S})$ be a test function. Then we have

$$\begin{aligned}
\int_{\mathcal{S}} \mathbf{n}\phi\Delta y \mathrm{d}A &= -\int_{\mathcal{S}} [\nabla(\mathbf{n}\phi)]^{\mathrm{T}}\nabla y \ \mathrm{d}A + \int_{\partial\mathcal{S}} \mathbf{n}\phi(\nabla y)^{\mathrm{T}}\mathbf{n}_c \mathrm{d}s \\
&= -\int_{\mathcal{S}} \left[\phi\nabla\mathbf{n}\nabla y + \mathbf{n}(\nabla\phi)^{\mathrm{T}}\nabla y\right] \mathrm{d}A \\
&\overset{(a)}{=} \int_{\mathcal{S}} \left[\phi \oslash \mathbf{x}\nabla y - \mathbf{n}(\nabla\phi)^{\mathrm{T}}\nabla y\right] \mathrm{d}A \\
&= \int_{\mathcal{S}} \left[\phi \oslash y - \mathbf{n}(\nabla\phi)^{\mathrm{T}}\nabla y\right] \mathrm{d}A, \qquad (160)
\end{aligned}$$

where the validity of (a) is owing to the equality

$$\nabla \mathbf{n} + \oslash \mathbf{x} = \mathbf{0}.$$

Let

$$\mathbf{y} = \mathbf{n}y = \mathbf{H}(\mathbf{x}),$$

Then it is easy to derive that

$$\nabla y = (\nabla \mathbf{y})\mathbf{n}, \quad \oslash y = (\oslash \mathbf{y})\mathbf{n}.$$

Substituting this into (160), we have

$$\int_{\mathcal{S}} \mathbf{n}\phi \Delta y \mathrm{d}A = \int_{\mathcal{S}} \left[\phi \oslash \mathbf{y} - \mathbf{n}(\nabla\phi)^{\mathrm{T}} \nabla \mathbf{y}\right] \mathbf{n} \mathrm{d}A.$$

On the other hand, for $\psi \in H^1(\mathcal{S})$, using Green's formula, we have

$$\begin{aligned} \int_{\mathcal{S}} \mathbf{H}\psi \, \mathrm{d}A &= \frac{1}{2} \int_{\mathcal{S}} \Delta \mathbf{x} \psi \, \mathrm{d}A \\ &= -\frac{1}{2} \int_{\mathcal{S}} (\nabla \mathbf{x})^{\mathrm{T}} \nabla \psi \, \mathrm{d}A + \frac{1}{2} \int_{\partial \mathcal{S}} (\nabla \mathbf{x})^{\mathrm{T}} \mathbf{n}_c \psi \, \mathrm{d}s \\ &= -\frac{1}{2} \int_{\mathcal{S}} (\nabla \mathbf{x})^{\mathrm{T}} \nabla \psi \, \mathrm{d}A + \frac{1}{2} \int_{\partial \mathcal{S}} \mathbf{n}_c \psi \, \mathrm{d}s. \end{aligned}$$

Therefore, the mixed variational form of (144) is (147).

The derivation of the mixed variational form for (145) and (146) are similar. We omit the details.

3.2 Discrete Surface Modeling Using PDEs

We use various nonlinear partial differential equations to efficiently solve several surface modelling problems, including surface blending, N-sided hole filling and free-form surface fitting. The nonlinear equations used include two second order flows, two fourth order flows and two sixth order flows. These nonlinear equations are discretized based on discrete differential geometry operators. The proposed approach is simple, efficient and gives very desirable results, for a range of surface models, possibly having sharp creases and corners.

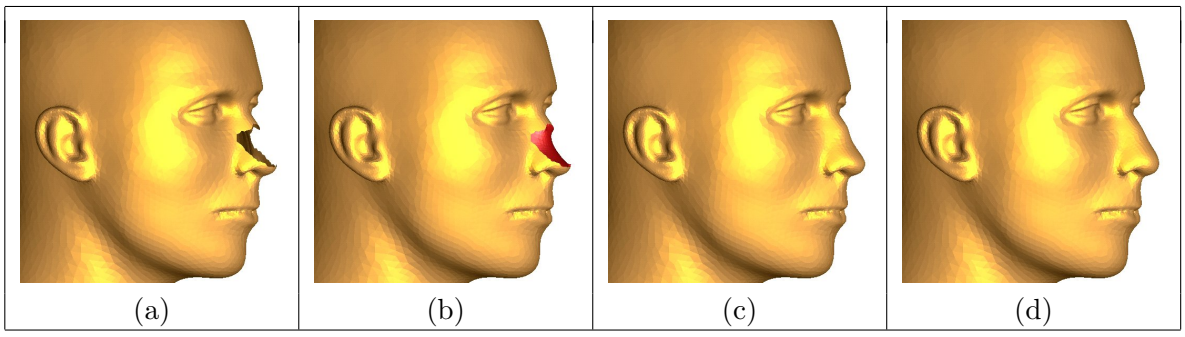


Figure 43: (a) shows a head mesh with a hole around the nose. (b) shows an initial filler construction of the nose with a piece of minimal surface. (c) the filler surface, after 30 iteration, generated using fourth order flow ($k = 2$ in (169)) with time step size 0.0002. (d) the filler surface, after 20 iteration, generated using sixth order flow ($k = 3$ in (169)) with time step size 0.00002.

Introduction We use various partial differential equations (PDE) to solve several surface modelling problems. The PDEs we use include the mean curvature flow, the averaged mean curvature flow, two fourth order (surface diffusion flow and quasi surface diffusion flow) and even higher order flows. All these equations are nonlinear and the geometry is intrinsic, i.e., the PDEs do not depend upon any particular parameterization. The problems we solve include surface blending, N-sided hole filling and free-form surface fitting with high order boundary continuity.

For the problems of surface blending and N-sided hole filling, we are given triangular surface meshes of the surrounding area. Triangular surface patches need to be constructed to fill the openings enclosed by the surrounding surface mesh and interpolate the hole boundary with some specified order of continuity. For the free-form surface fitting problem, we are possibly given a set of points, or a wire frame of curves that defines an outline of the desired shape, or even some surface patches. We construct a surface which interpolates the points or curves or the boundaries of the patches with specified order of continuity. The free-form surface fitting problem is the most general, including the surface blending and N-sided hole filling problems, as its special cases.

Our twofold strategy for solving these problems is as follows: First we construct an initial triangular surface mesh ("filler") using any of a number of automatic or semi-automatic free-form modelling techniques One may also interactively edit this "filler" to meet the weak assumptions for an initial solution shape. This "filler" may be bumpy or noisy, and in general this "filler" does not satisfy the smoothness boundary conditions, though it may roughly characterize the shape of the surface to be constructed. Second we deform the initial mesh by solving a suitable flow PDE. Unlike most of the previous free-form modelling techniques, our approach solves high-order boundary continuity constraints without any prior estimation of normals or derivative jets along the boundary. The solution of the PDE is time dependent. We consider two possibilities for the time span of the evolution. One is a short time evolution, where we require the solution to respect to the initial shape or geometry (see Fig. 49). The other is a long time evolution, where the initial filler provides a topological structure, and what we look for is a stable solution state of the flow (see Fig. 43 and Fig. 46). We focus our attention on these twofold solutions of PDEs with boundary continuity constraints, rather than the construction of initial filler mesh. In section 3.2.4, we present automatic approaches for constructing the initial filler mesh, and our preferred choice.

Main Results. We use second order flows (mean curvature flow and averaged mean curvature flow) for G^0 continuity, fourth order flows for G^1 continuity and sixth order flows for G^2 continuity in each of several surface modelling problems. The proposed approach is simple and easy to implement. It is general, solves several surface modelling problems in the same manner, and gives very desirable results for a range of complicated free-form surface models, possibly having sharp features and corners. Furthermore, it avoids the estimation of normals or tangents or curvatures on the boundaries.

Partial Differential Equation Models Let $\mathcal{M}$ be a smooth surface and $p \in \mathcal{M}$ be the surface point. The general form of the geometric flows we consider is in the following form (see [227])

$$\frac{\partial p}{\partial t} = V(p,t),$$

where $V(p,t) \in \mathbb{R}^3$ represents a velocity field. We shall focus our attention on using two classes velocity fields, one is curvature driven velocity field in the normal direction, the other is the higher order Laplace-Beltrami operators acting on surface point p.

Geometric Partial Differential Equations We now describe several geometric PDE models we use.

Let $\mathcal{M}_0$ be a compact closed immersed orientable surface in $\mathbb{R}^3$. A curvature driven geometric evolution consists of finding a family $\{\mathcal{M}(t) : t \geq 0\}$ of smooth closed immersed orientable surfaces

in $\mathbb{R}^3$ which evolve according to the flow equation

$$\frac{\partial p}{\partial t} = N(p)V_n(k_1, k_2, p), \quad \mathcal{M}(0) = \mathcal{M}_0. \tag{161}$$

Here $p(t)$ is a surface point on $\mathcal{M}(t)$, $V_n(k_1, k_2, p)$ denotes the normal velocity of $\mathcal{M}(t)$, which depends on the principal curvatures k_1, k_2 of $\mathcal{M}(t)$, $N(p)$ stands for the unit normal of the surface at $p(t)$. We identify the surface point p and surface normal $N(p)$ as 3×1 matrices (column vectors). Hence, the arithmetic operations of these quantities are regarded matrix operations. The product of a scalar $a \in \mathbb{R}$ and a matrix M is written as either aM or Ma.

Let $A(t)$ denote the area of $\mathcal{M}(t)$, $V(t)$ denote the volume of the region enclosed by $\mathcal{M}(t)$. Then it has been shown that (see [230], Theorem 4)

$$\frac{dA(t)}{dt} = \int_{\mathcal{M}(t)} V_n H d\sigma, \quad \frac{dV(t)}{dt} = \int_{\mathcal{M}(t)} V_n d\sigma, \tag{162}$$

where $H = \frac{1}{2}(k_1 + k_2)$ is the mean curvature of $\mathcal{M}(t)$.

1. Mean Curvature Flow (see [98, 229])

Taking $V_n = -H = -\frac{1}{2}(k_1 + k_2)$ in (161), we obtain the mean curvature flow PDE:

$$\frac{\partial p}{\partial t} = -N(p)H(p), \quad \mathcal{M}(0) = \mathcal{M}_0. \tag{163}$$

It follows from (162) that

$$\frac{dA(t)}{dt} = -\int_{\mathcal{M}(t)} H^2 d\sigma. \tag{164}$$

(164) implies that the mean curvature flow is area reducing.

2. Averaged Mean Curvature Flow (see [107, 134, 203])

In (161), if we take $V_n = h(t) - H(t)$, where $h(t) = \int_{\mathcal{M}(t)} H d\sigma / \int_{\mathcal{M}(t)} d\sigma$, then we have the averaged mean curvature flow PDE:

$$\frac{\partial p}{\partial t} = N(p)[h(t) - H(p)], \quad \mathcal{M}(0) = \mathcal{M}_0. \tag{165}$$

The existence proof of the global solutions to this flow can be found in Huiskens' paper [134]. It follows from (162) that

$$\frac{dA(t)}{dt} = \int_{calM(t)} (hH - H^2) d\sigma = \int_{\mathcal{M}(t)} [hH - H^2 - h(h - H)] d\sigma = -\int_{\mathcal{M}(t)} (h - H)^2 d\sigma \leq 0, \tag{166}$$

since obviously $\int_{M(t)} h(h - H) = h(h \int_{M(t)} d\sigma - \int_{M(t)} H d\sigma) = 0$. On the other hand, the second equation of (162) implies that

$$\frac{dV(t)}{dt} \;=\; h(t) \int_{\mathcal{M}(t)} d\sigma - \int_{\mathcal{M}(t)} H d\sigma = 0.$$

Hence the averaged mean curvature flow is volume preserving and area shrinking. The area shrinking stops if $H \equiv h$.

3. Surface Diffusion Flow (see [205])

If we take $V_n = \Delta H$, we get the so-called surface diffusion flow PDE:

$$\frac{\partial p}{\partial t} = N(p)\Delta H(p), \quad \mathcal{M}(0) = \mathcal{M}_0, \tag{167}$$

where $\Delta := \Delta_{\mathcal{M}}$ is Laplace-Beltrami operator which acts on functions defined on surface $\mathcal{M}(t)$. The existence and uniqueness of solutions for this flow is given in [106]. From (162) and Green's formula we have

$$\frac{d}{dt}A(t) = \int_{\mathcal{M}(t)} \Delta H H d\sigma = -\int_{\mathcal{M}(t)} |\nabla H|^2 d\sigma \leq 0,$$
$$\frac{d}{dt}V(t) = \int_{\mathcal{M}(t)} \mathrm{div}(\nabla H) d\sigma = -\int_{\mathcal{M}(t)} \nabla H \nabla(1) d\sigma = 0,$$

where ∇ stands for the (tangential) gradient operator (see [96], pages 101-102) acting on differential functions defined on the surface $\mathcal{M}$. Hence, the surface diffusion flow is area shrinking, but volume preserving. The area stops shrinking when the gradient of H is zero. That is, $\mathcal{M}$ is a surface with constant mean curvature.

4. Higher order Geometric Flows

$$\frac{\partial p}{\partial t} = (-1)^{k+1} N(p) \Delta^k H(p), \quad \mathcal{M}(0) = \mathcal{M}_0. \tag{168}$$

Using Green formula, we have

$$\int_{\mathcal{M}(t)} \Delta^k H d\sigma = \int_{\mathcal{M}(t)} \Delta(\Delta^{k-1} H) d\sigma = \int_{\mathcal{M}(t)} \nabla(\Delta^{k-1} H) \nabla(1) d\sigma = 0.$$

Hence, the flow (168) is volume preserving if $k \geq 1$ from the second equation of (162).

Remark 2.1. We should note that the area/volume preserving/shrinking properties for the flows mentioned above are valid for closed surfaces. In our application of these flows, these properties may not be true since the surfaces always have fixed boundaries. For a open surface with fix boundary, the volume $V(t)$ could be defined as the directional volume between $\mathcal{M}(0)$ and $\mathcal{M}(t)$. It is easy to see that the volume preserving property for the averaged mean curvature flow is still valid. But for the higher order flow (168) ($k \geq 1$), this property is no longer valid, because a term related to the boundary does not vanish when Green's formula is used. For our modelling problems, volume preservation is not a desirable property (see Fig. 43 and 46).

Remark 2.2. In [205], Schneider and Kobbelt use elliptic equation $N(p)\Delta H(p) = 0$, while we use several time dependent parabolic type equations. In our approach, we have a progressive process starting from an initial value, so that a family of solutions is obtained. Such an approach is very desirable if the initial value is an approximation of the required solution.

Quasi Geometric Partial Differential Equations Now we generalize the heat equation on a surface to the following higher order flows:

$$\frac{\partial p}{\partial t} = (-1)^{k+1} \Delta^k p, \quad \mathcal{M}(0) = \mathcal{M}_0, \quad k > 0. \tag{169}$$

Since $\Delta p = -2H(x)N(p)$, it is easy to see that (169) is the mean curvature flow when $k = 1$ (up to a factor 2). But since $(\Delta^k H)N \neq \Delta^k(HN)$ in general, (169) is different from the flow (168). To distinguish the difference between (168) and (169), we call (169) as a quasi geometric PDE.

The experiments conducted show that flows (169) sometimes behave better than the geometric flows mentioned above for our geometry modelling problems. However, the theoretical analysis on the existence and stability of their solutions is currently unavailable.

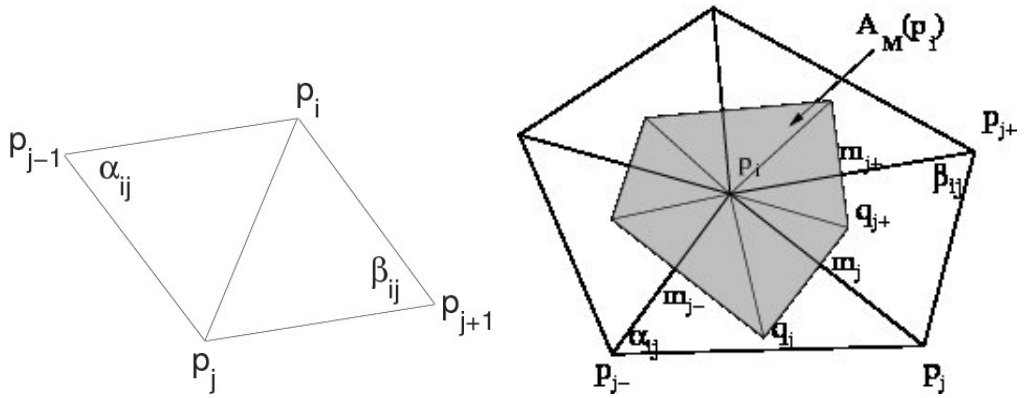


Figure 44: Left: The definition of the angles α_{ij} and β_{ij}. Right: The definition of the area $A_M(p_i)$.

Solution of the PDEs There are basically two classes of approaches for solving a PDE on any domain. One approach is based on finite divided differences, the other is based on finite elements . The approach we adopt is based on finite divided differences. Since we are dealing with differential equations over 2-manifolds in $\mathbb{R}^3$, the classical finite divided differences will be replaced by discretized differential geometric operators over surfaces. Section 3.2.1 deals with discretized geometric differential operators. Next in Section 3.2.2 we detail how the boundary conditions are respected. Discretizations of the PDEs in the spatial direction are described in section 3.2.3 and 3.2.4. Semi-implicit discretization in the time domain is considered in section 3.2.4. Other issues, such as mesh regularization and initial mesh construction, are addressed in section 3.2.5.

Discretized Laplace-Beltrami Operator One of the fundamental problems in solving our PDEs is the discretization of the Laplace-Beltrami operator. On a triangular surface mesh, several discretized approximations of the operator have been proposed. We adopt the discretization developed by Meyer et al in [162]. A comparative research about the various discretized Laplace-Beltrami operators is conducted in [237]. It has been shown that the scheme of Meyer et al's is better for discretizing our PDEs. Let f be a smooth function on a surface, then Δf is approximated over a triangular mesh M by

$$\Delta f(p_i) \approx \frac{1}{A_M(p_i)} \sum_{j \in N_1(i)} \frac{\cot \alpha_{ij} + \cot \beta_{ij}}{2} [f(p_j) - f(p_i)], \tag{170}$$

where $N_1(i)$ is the index set of 1-ring of neighbor vertices of vertex p_i, α_{ij} and β_{ij} are the triangle angles shown in Fig 44 (Left). $A_M(p_i)$ is the area for vertex p_i as shown in Fig 44 (Right), where q_j is the circumcenter point for the triangle $[p_{j-1}p_jp_i]$ if the triangle is non-obtuse. If the triangle is obtuse, q_j is chosen to be the midpoint of the edge opposite to the obtuse angle. Since $\Delta p = -2H(p)N(p)$ (see [230], page 151), we have

$$(\Delta p)_{p=p_i} = -2H(p_i)N(p_i) \approx \frac{1}{A_M(p_i)} \sum_{j \in N_1(i)} \frac{\cot \alpha_{ij} + \cot \beta_{ij}}{2} (p_j - p_i). \tag{171}$$

This gives an approximation of the mean curvature normal (see [162]). The higher order Laplace-Beltrami operators are discretized recursively as

$$\Delta^k f(p_i) = \Delta(\Delta^{k-1} f)(p_i) = \frac{1}{A_M(p_i)} \sum_{j \in N_1(i)} \frac{\cot \alpha_{ij} + \cot \beta_{ij}}{2} [\Delta^{k-1} f(p_j) - \Delta^{k-1} f(p_i)] \tag{172}$$

with $\Delta^0 f(p_i) = f(p_i)$. Note that $\Delta^k f(p_i)$ involves function values on a k-ring of neighboring vertices of p_i.

Handling of Boundary Conditions 1). Natural Boundary Conditions for Blending and Hole Filling

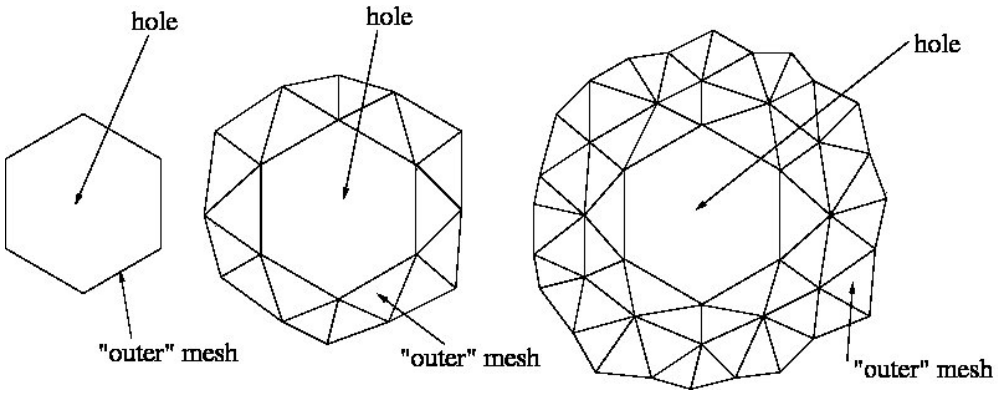


Figure 45: Left: The involved vertices of the"outer" mesh for a G^0 boundary condition. The "outer" mesh is just the boundary of the hole. Middle: The involved vertices of the "outer" mesh for a G^1 boundary condition. Right: The involved vertices of the "outer" mesh for a G^2 boundary condition.

By the natural boundary conditions, we mean that no continuity conditions are specified at the boundary points, but the continuity is implied by the "outer" mesh incident to the boundary of the hole (see Fig 45). Such a treatment for boundary condition is suitable for both the blending problem and the N-sided hole filling problem, since the "outer" mesh always exists in such problems.

Let g_i be the order of continuity at a boundary point p_i, $g = \max g_i$. Then we can use the order $2g$ flow $\frac{\partial p}{\partial t} = (-1)^{g+1}\Delta^g H(p)N(p)$ for constructing the triangular surface patch with G^{g_i} continuity at the boundary vertex p_i. $\Delta^g H$ is discretized recursively: $\Delta^g H = \Delta(\Delta^{g-1}H)$. At a boundary vertex p_i, $\Delta^k H(p_i)$ is evaluated according to the following rule:

Evaluation Rule at Boundary. *$\Delta^k H(p_i)$ is evaluated recursively by formulas 175 and 176 if $k \leq g_i$, otherwise $\Delta^k H(p_i)$ is set to zero and the recursion stops.*

Note that even for an inner vertex p_j, the recursive definition may make $\Delta^k H(p_j)$ involve the evaluation of a lower order Laplace-Beltrami operator on the boundary. In general, the recursive evaluation of $\Delta^k H(p_i)$ at p_i (for either p_i being an inner or an outer vertex) involves $k+1$-ring neighbor vertices of p_i. Some of them may be inner vertices, and the remaining are outer vertices. The inner vertices are treated as unknowns in the discretized equations and the outers are incorporated into the right-hand side.

2). Natural Boundary Conditions for Free-Form Surface Filling

In the free-form surface filling problem, we are given a wireframe of curves (edges) and we wish to flesh the wireframe with surface patches that contain the curves as boundary with pre-specified order of continuity. At each of the intersection points of the patches, an order of continuity is pre-specified and the evaluation rule mentioned above is applied. For each inner point, a discretized linear equation is generated using the operator discretization (176). These linear equations for different patches are collected together and solved simultaneously. Note that one linear equation may involve inner vertices of several patches. However, if the continuity order at each boundary point is zero, any equation corresponding to an inner vertex does not involve inner vertices of other patches.

Remark 3.1. Schneider and Kobbelt in [205] use Moreton and Sequin's least square fitting of the second fundamental form relative to a local parameterization to estimate the required data on the boundary. These estimations of the boundary derivative data are based on incomplete information. Hence, the estimated data maybe not reliable. Our approach is based on the identity

$\Delta_{\mathcal{M}} p = -2H(p)N(p)$. Hence, we do not need to estimate boundary derivative data, such as normals, tangents or curvatures. Furthermore, the boundary conditions are treated in the same way for equations with different orders.

Spatial Discretization of Quasi Geometric Flows Let us consider first the discretization of (169) in the spatial direction for $k = 1, 2, 3$. Let $P = [p_1, \cdots, p_m]^T \in \mathbb{R}^{m\times 3}$, $\Delta P = [\Delta p_1, \cdots, \Delta p_m]^T \in \mathbb{R}^{m\times 3}$, where $p_1, \cdots, p_m$ are all the unknown vertices to be determined in each of our modelling problems. Then (171) could be written in matrix form:

$$\Delta P = -(\mathcal{D}\mathcal{W})P + B^{(1)}, \tag{173}$$

where $\mathcal{D} = \mathrm{diag}[\frac{1}{2A(p_1)}, \cdots, \frac{1}{2A(p_m)}]$ is a diagonal matrix, $\mathcal{W} = \{w_{ij}\}_{i,j=1}^m$ with

$$w_{ij} = \begin{cases} \sum_{k\in N_1(i)} \cot\alpha_{ik} + \cot\beta_{ik}, & i = j, \\ -(\cot\alpha_{ij} + \cot\beta_{ij}), & i \neq j, \quad i \in N_1(j), \quad j \in N_1(i), \\ 0, & \text{otherwise.} \end{cases}$$

Furthermore, $\mathcal{W}$ is a sparse, symmetric and positive definite matrix (see [205]). The constant term $B^{(1)} \in \mathbb{R}^{m\times 3}$ is obtained from the boundary conditions. It follows from (173) that

$$\Delta^2 P = (\mathcal{D}\mathcal{W}\mathcal{D}\mathcal{W})P + B^{(2)}, \tag{174}$$

where $B^{(2)} \in \mathbb{R}^{m\times 3}$ is obtained from the boundary conditions. Again, $\mathcal{W}\mathcal{D}\mathcal{W}$ is a sparse, symmetric and positive definite matrix. In general,

$$\Delta^k P = (-1)^k(\mathcal{D}\mathcal{W})^k P + B^{(k)},$$

and the matrix for $\mathcal{D}^{-1}(\mathcal{D}\mathcal{W})^k$ is also sparse, symmetric and positive definite.

Spatial Direction Discretization of Geometric Flows Let

$$\omega_{ij} = \begin{cases} \sum_{k\in N_1(i)} \frac{\cot\alpha_{ik}+\cot\beta_{ik}}{2A_M(p_i)}, & i = j, \\ -\frac{\cot\alpha_{ij}+\cot\beta_{ij}}{2A_M(p_i)}, & i \neq j, \quad i \in N_1(j), \quad j \in N_1(i), \\ 0, & \text{otherwise,} \end{cases}$$

and $N(i) = N_1(i) \cup \{i\}$. Then we have

$$N(p_i)H(p_i) \approx \frac{1}{2} \sum_{j\in N(i)} \omega_{ij} p_j. \tag{175}$$

The higher order Laplace-Beltrami operators acting on H are discretized recursively as

$$\Delta^k H(p_i) = \Delta(\Delta^{k-1}H)(p_i) \approx -\sum_{j\in N(i)} \omega_{ij}\Delta^{k-1}H(p_j) \tag{176}$$

with

$$\Delta^0 H(p_i) = H(p_i) \approx \frac{1}{2} \sum_{j\in N(i)} \omega_{ij} N(p_i)^T p_j. \tag{177}$$

Note that $\Delta^k H(p_i)$ involves values of the mean curvature on a k-ring of neighboring vertices of p_i.

Using (175)–(177) and the evaluation rule at the boundary, we can write $N(p_i)\Delta^k H(p_i)$ as the following form:

$$N(p_i)\Delta^k H(p_i) \approx (-1)^k \sum_{j\in J_0} \omega_{ij}^{(k)} p_j + B_i^{(k)}, \quad \omega_{ij}^{(k)} \in \mathbb{R}^{3\times 3}, \quad B_i^{(k)} \in \mathbb{R}^3,$$

where J_0 is the index set of the (unknown) vertices to be determined, $B_i^{(k)}$ comes from boundary condition. To be more specific, let J denote the index set of the mesh M, J_k be the union of J_0 and the index set of the boundary vertices where C^k condition is specified. Then

$$\begin{aligned} N(p_i)H(p_i) &\approx \frac{1}{2}\sum_{j\in N(i)}\omega_{ij}p_j = \frac{1}{2}\sum_{j\in N(i)\cap J_0}\omega_{ij}p_j + \frac{1}{2}\sum_{j\in N(i)\cap\{J\setminus J_0\}}\omega_{ij}p_j \\ &= \sum_{j\in J_0}\omega_{ij}^{(0)}p_j + B_i^{(0)}, \end{aligned} \tag{178}$$

where $\omega_{ij}^{(0)} = \frac{1}{2}\omega_{ij}I_3$ for $j \in N(i)\cap J_0$, $\omega_{ij}^{(0)} = 0$ otherwise, $B_i^{(0)} = \frac{1}{2}\sum_{j\in N(i)\cap\{J\setminus J_0\}}\omega_{ij}p_j$. Similarly,

$$\begin{aligned} N(p_i)\Delta H(p_i) &\approx -N(p_i)\sum_{j\in N(i)}\omega_{ij}H(p_j) = -N(p_i)\sum_{j\in N(i)\cap J_1}\omega_{ij}H(p_j) \\ &= -N(p_i)\sum_{j\in N(i)\cap J_1}\omega_{ij}N(p_j)^TN(p_j)H(p_j) \\ &\approx -\sum_{j\in N(i)\cap J_1}\omega_{ij}N(p_i)N(p_j)^T\left[\sum_{k\in J_0}m_{jk}^{(0)}p_k + B_j^{(0)}\right] \\ &= -\sum_{j\in J_0}\omega_{ij}^{(1)}p_j + B_i^{(1)}, \end{aligned} \tag{179}$$

$$\begin{aligned} N(p_i)\Delta^2 H(p_i) &\approx -N(p_i)\sum_{j\in N(i)}\omega_{ij}\Delta H(p_j) = -N(p_i)\sum_{j\in N(i)\cap J_2}\omega_{ij}\Delta H(p_j) \\ &\approx N(p_i)\sum_{j\in N(i)\cap J_2}\omega_{ij}\sum_{k\in N(j)\cap J_1}\omega_{jk}H(p_k) \\ &\approx \sum_{j\in N(i)\cap J_2}\sum_{k\in N(j)\cap J_1}\omega_{ij}\omega_{jk}N(p_i)N(p_k)^T\left[\sum_{l\in J_0}m_{kl}^{(0)}p_l + B_k^{(0)}\right] \\ &= \sum_{j\in J_0}\omega_{ij}^{(2)}p_j + B_i^{(2)}. \end{aligned} \tag{180}$$

(178)–(180) are used to discretize the right-handed side of (168) for $k = 0, 1, 2$. The discretization of $N(p_i)\Delta^k H(p_i)$ for $k > 2$ is recursively calculated using (176) and boundary conditions.

Time Discretization Given an approximate solution $\{p_i^{(n)}\}_{i=1}^m$ of the order $2k$ PDE at t_n for all the inner vertices, we construct an approximate solution $\{p_i^{(n+1)}\}_{i=1}^m$ for the next time step $t_{n+1} = t_n + \tau^{(n)}$ by using a semi-implicit Euler scheme. That is, we replace the derivative $\frac{\partial p}{\partial t}$ with $[p(t_{n+1}) - p(t_n)]/\tau^{(n)}$, and the quantities w_{ij} in (173), ω_{ij} and $N(p_i)$ in (175)–(177), $h(t)$ in (165) are computed using the previous result at t_n. Normals $N(p_i)$ are computed from Loop's subdivision surface (see [41] for detail). Such a treatment yields a linear system of equations with the inner vertices as unknowns. Let $\mathcal{P}^{(n+1)} = [(p_1^{(n+1)})^T, \cdots, (p_m^{(n+1)})^T]^T \in \mathbb{R}^{3m}$. The linear system for the geometric flows can be written as the matrix form

$$[I + \tau^{(n)}\mathcal{W}^{(k)}]\mathcal{P}^{(n+1)} = \mathcal{B}^{(k)}, \quad \mathcal{W}^{(k)} = \{\omega_{ij}^{(k)}\}, \quad \mathcal{B}^{(k)} \in \mathbb{R}^{3m}. \tag{181}$$

The matrix $\mathcal{W}^{(k)} \in \mathbb{R}^{3m\times 3m}$ is highly sparse, hence an iterative method for solving such a linear system is desirable. We use Saad's iterative method [199], named GMRES, to solve the system. The experiment shows that this iterative method works very well.

Let $P^{(n+1)} = [p_1^{(n+1)}, \cdots, p_m^{(n+1)}]^T \in \mathbb{R}^{m\times 3}$. The linear system for the flows (169) can be written as the matrix form

$$[I + \tau^{(n)}(\mathcal{D}\mathcal{W})^{k+1}]P^{(n+1)} = B^{(k)}, \quad \text{or} \quad W^{(k)}P^{(n+1)} = \mathcal{D}^{-1}B^{(k)} \tag{182}$$

where $B^{(k)} \in \mathbb{R}^{m\times 3}$, $W^{(k)} = \mathcal{D}^{-1} + \tau^{(n)}\mathcal{W}(\mathcal{D}\mathcal{W})^k \in \mathbb{R}^{m\times m}$ is a highly sparse, symmetric and positive definite matrix, and hence we use a conjugate gradient iterative method with diagonal preconditioning to solve the system.

Note that for the same size problem, the size of coefficient matrix in (181) is three times larger than that of coefficient matrix in (182). Furthermore, the matrix $W^{(k)}$ in (182) is symmetric and positive definite. The matrix in (181) is not. We also note that the discretization of (169) does not involve the computation of the surface normals.

Remark 3.2. It is well known that the condition of the linear system arising from the proposed semi-implicit discretization behaves like $O(1 + \tau^{(n)}h^{-2k})$, where h is the minimal edge length of the mesh. Hence, if the mesh to be evolved is very irregular, the resulting system will be ill-conditioned. In such a case, a small time step size is required to make an iterative solver converge. Such a problem is relieved by the mesh regularization treatment (see section 3.6). On the other hand, more advanced iterative method, such as multi-grid techniques based on a hierarchical mesh representation (see [146]) or algebraic multi-grid techniques, could be used to accelerate the iteration process. In the current implementation, these techniques are not incorporated.

Upper-bound of time step. It is known that several surface evolutions (e.g. the mean curvature flow (see [98, 229]) and the surface diffusion flow (see [46])) may develop singularities. For our geometric modelling problems, suppose we have a topologically correct initial surface mesh construction and we look for solutions that have the same topology as the initial mesh. Hence, we require that our solution is within the time period in that no singularity occurs. Therefore, we shall determine the time step $\tau^{(n)}$ so that t_n should not go beyond the time moment when the singularity occurs. Let $L(p_i^{(n)}, M(t_n))$ be the spatial discretization of $V(p,t)$ at vertex $p_i^{(n)}$ over the mesh $M(t_n)$. Then from the approximate equality

$$\|p_i^{(n+1)} - p_i^{(n)}\| = \tau^{(n)}\|L(p_i^{(n)}, M(t_n))\|$$

and the requirement

$$\|p_i^{(n+1)} - p_i^{(n)}\| \le \frac{1}{2}\min_{j\in N_1(i)}\|p_j^{(n)} - p_i^{(n)}\| \tag{183}$$

we determine an upper-bound for $\tau^{(n)}$ as follows

$$\tau^{(n)} \le B_n := \frac{1}{2}\min_{1\le i\le m}\left\{\frac{\min_{j\in N_1(i)}\|p_j^{(n)} - p_i^{(n)}\|}{\|L(p_i^{(n)}, M(t_n))\|}\right\}.$$

Requirement (183) guarantees that no vertex-collision happens. When the singularity is nearly to occur, the upper-bound B_n will approach to zero. Hence the evolution cannot move beyond the singular point for time.

Remark 3.3. When the singularity is nearly to occur, the upper-bound B_n will approach to zero. This will be a very low efficiency process. So a threshold value ϵ_0 should be put on the minimal B_n. If the determined B_n is smaller than the threshold value, we terminate the evolution process (see (186)–(187)).

Other Important Issues 1. Mesh Regularization

The surface motion by the geometric PDEs described in section 3.2 may cause a very irregular (nonuniform) distribution of the mesh vertices. Hence, introducing a regularization mechanism in

the evolution process is necessary. Since *the tangential displacement does not influence the geometry of the deformation, just its parameterization* (see [105]), we also add a tangential displacement to the motion. Hence, the general form of our geometric evolution problem could be written as

$$\frac{\partial p}{\partial t} = V(p,t) + V_t(p)T(p), \quad \mathcal{M}(0) = \mathcal{M}_0, \tag{184}$$

where $T(p)$ is a tangent direction at the surface point p, $V_t(p)$ is the tangential velocity. In the process of numerical solution of equation (184), $V_t(p)T(p)$ is chosen as

$$\mathcal{U}_0(p_i^{(n)}) - \Big(\mathcal{U}_0(p_i^{(n)}), N(p_i^{(n)})\Big)\, N(p_i^{(n)}) \tag{185}$$

where $\mathcal{U}_0(p_i^{(n)}) = \frac{1}{card(N_1(i))} \sum_{j \in N_1(i)} (p_j^{(n)} - p_i^{(n)})$, N is the surface normal computed from the limit surface of Loop's subdivision. This discretization of $V_t(p)T(p)$ is very similar to the one given by Ohtake et al. [178], which is $\mathcal{U}_0(p_i^{(n)}) - \Big(\mathcal{U}_0(p_i^{(n)}), N(p_i^{(n)})\Big)\mathcal{U}_0(p_i^{(n)})$. The difference is that our displacement is in the tangent plane. In (185), $\mathcal{U}_0(p_i^{(n)})$ could be replaced by $\mathcal{U}_0(p_i^{(n+1)})$ to use as many of the new values as possible, and still yield a linear system. However, such a treatment destroys the symmetric property of the coefficient matrix. The tangential motion (185) is also used by Wood et al [233] and Ohtake et al [179].

2). Stopping Criteria

We need to determine the minimal iteration number n, so that the evolution procedure stops at $t = t_n$. The following two criteria are used

$$\|M(t_n) - M(0)\| \geq \epsilon_1 \quad \text{or} \quad B_n < \epsilon_0 \tag{186}$$

$$\|M(t_{n+1}) - M(t_n)\| / \tau^{(n)} \leq \epsilon_2 \quad \text{or} \quad B_n < \epsilon_0 \tag{187}$$

where ϵ_i are given control constants, B_n is the determined upper-bound for $\tau^{(n)}$. Criterion (186) is for short time evolution, where we require $M(n\tau^{(n)})$ near $M(0)$. Criterion (187) is for long time evolution, where we are looking for a stable status of the solution. Condition $B_n < \epsilon_0$ is imposed for avoiding dead-loop around the singular point of time.

3). Construction of Initial Surface Mesh

To provide an initial solution to the geometric evolution problem, we need to construct an initial triangular surface mesh ("filler") for each opening using any of a number of automatic or semi-automatic free-form surface construction techniques. One can also interactively edit this "filler" to meet the weak assumptions for an initial solution shape.

Since the opening to be filled could be topologically complicated, we solve the problem in two steps. In the first step we fit each opening by an implicit algebraic surface or spline which interpolates or approximates the boundary data. The approach we used is the one developed by Bajaj et al [27, 29, 39]. In this approach, the data to be interpolated or approximated could be points or curves (even with normals). For ours, the boundary data are always points. Of course, this approach may not guarantee to produce topologically correct surfaces. If this happens, we break the opening into several parts by inserting a few curves (polygons) and then repeat the surface fitting for each part until we achieve a reasonable shape for the "filler".

After the algebraic surface is obtained, a triangulation step is employed. Since this triangulation should be consistent with the boundary polygon of the opening, we adopted the expansion technique developed in [39]. Using this approach, we triangulate the surfaces starting from the boundary of the opening.

Remark 3.4. Comparing with finite element approach, the finite difference approach described above is easy to implement and it treats the equations with different orders in a uniform fashion. In the finite element approach, one has to make efforts to derive a variational form for each of the

PDEs. For higher order flows, hybrid method is used in general, such an approach will introduce much more unknowns, and therefore the resulted linear system is much larger. For example, in order to use finite element method (linear element) for the surface diffusion flow, Bänsch et al [46] split the PDE into a system of four equations.

Comparative Examples In this section, we give several examples to show how the PDEs are used to solve different problems in a uniform fashion. We also compare the effects of flows (168) and (169). All the figures produced by the fourth and sixth flows are generated using (169), except for the figures of the second row of Fig. 46 and third row of Fig. 48. These figures are produced using the flow (168). When we compare the effects of (168) and (169), we use the same number of iterations but double time step size for (168) because the factor 2 in the relation $\Delta p = -2HN$.

1). Comparison of the Flows

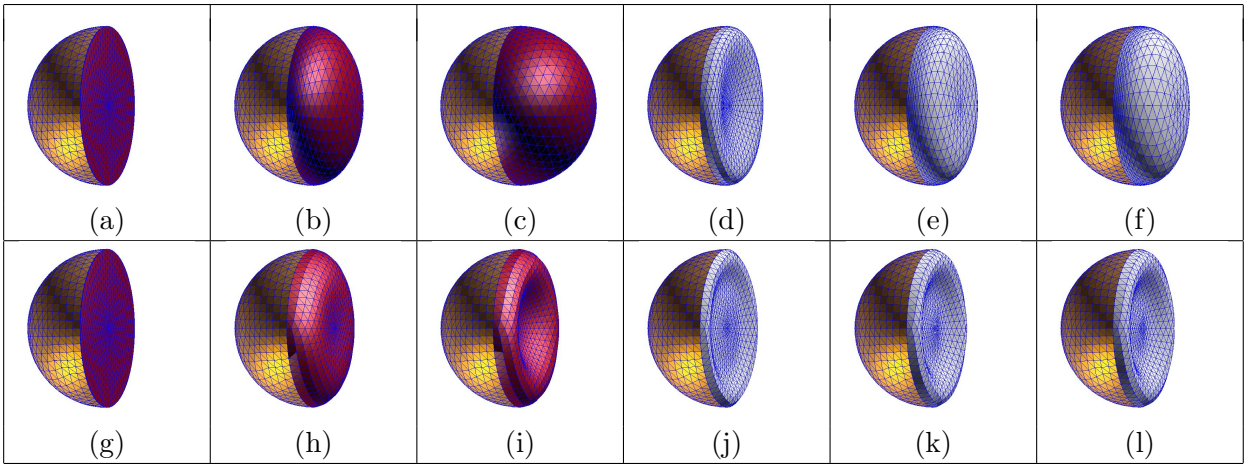


Figure 46: The first and second row show the results of (169) and (168), respectively. (a) (same as (g)) The input semi-sphere (left part) with an initial planar triangulation of the disk opening. The mean curvature flow does not change the disk (initial mesh). (b) The result of fourth order flow after 10 iteration with $\tau^{(n)} = 0.1$. (c) The result of the sixth order flow after 10 iteration with $\tau^{(n)} = 0.01$. (d), (e) and (f) show three intermediate results of the sixth order flow with $\tau^{(n)} = 0.001$, and 1, 6 and 10 iterations, respectively. (h) The result of the surface diffusion flow after 10 iteration with $\tau^{(n)} = 0.2$. (i) The result of the sixth order flow (168) after 10 iteration with $\tau^{(n)} = 0.02$. (j), (k) and (l) show three intermediate results of the sixth order flow (168) with $\tau^{(n)} = 0.002$, and 1, 6 and 10 iterations, respectively.

The first three figures of the first row of Fig. 46 show the long time evolution solutions of the mean curvature flow, the fourth order flow, and the sixth order flow (169) for the input semi-sphere with an initial construction of the opening, a triangulated disk. The mean curvature flow does not change the disk. Figures (b) and (c) are the results after 10 iterations with $\tau^{(n)} = 0.1$ and $\tau^{(n)} = 0.001$, respectively. Further iterations do not have a significant change on the shape of the solution surface. The fourth and sixth order flows yield convex surfaces and the smoothness is clearly observed. Also notice that the sixth order flow recovers the sphere accurately. The last three figures show three intermediate results of the sixth order flow. The second and third figures of the second row of Fig. 46 show the evolution solutions of the surface diffusion flow and sixth order flows (168) for the input semi-sphere with an initial construction of the opening. Figure (h) and (i) are produced using the same number of iterations as (b) and (c), respectively, and double time step sizes. Again, the last three figures show three intermediate results of the sixth order flow. Comparing with the figures of the first row, the geometric flows change the surface shape in a much slower rate.

Remark 4.1. We have pointed that the geometric flows (168) have volume preserving properties

for a closed surface. However, for an open surface with fixed boundary, the volume preserving properties are not guaranteed. Figures (h) and (i) show that the volume preserving property is not valid.

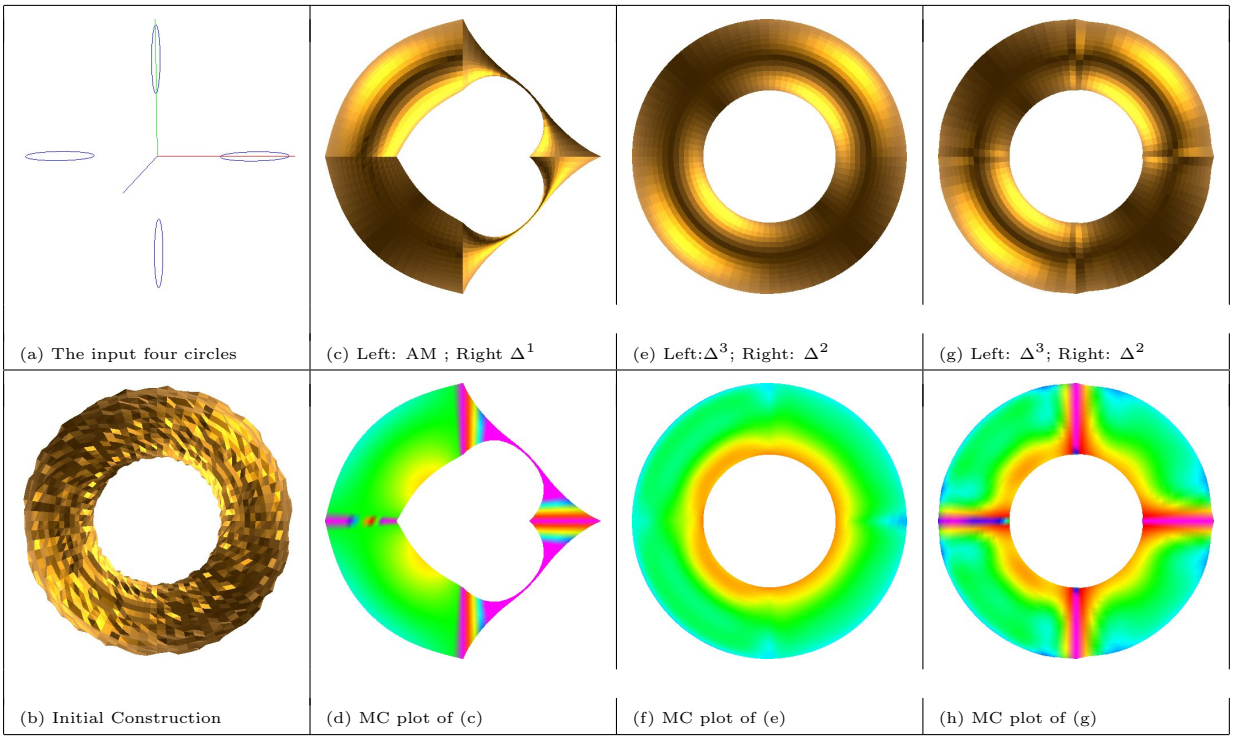


Figure 47: Comparison of different flows. Δ^k represents $2k$ order flow (169) is used. AM denote the averaged mean curvature flow. The time step sizes for the second, fourth and sixth order flows are chosen to be 0.1, 0.0025, and 0.0000625, respectively. Figures (c), (e), (g) are the faired interpolating surface meshes after 6 iterations, where the continuities at the boundary curves are set to 0, 2 and 0, respectively. Figures (d), (f), (h) are the mean curvature (MC) plots of figures (c), (e), (g), respectively.

Fig. 47 shows the combined use of different flows. The aim of this toy example is to illustrate the difference of these flows, especially the continuity on the patch boundaries. Figure (a) shows four circles to be interpolated. Two of the circles are in the xz-plane, the other two are in the yz-plane. (b) shows an initial G^0 surface mesh constructed using [27] with some additional noise added. (c), (e) and (g) are the faired interpolating surfaces after 6 iterations using different combinations of the flows. The time step sizes for the second, fourth and sixth order flows are chosen to be 0.1, 0.0025, and 0.0000625, respectively. Since the higher order flows evolve faster than the lower order flows, we use smaller time step sizes for higher order flows to obtain nearly the same surface evolution speed. Each of the meshes consists of four surface patches. The left two patches are in the regions $R^{-+} := \{(x, y, z) : x \leq 0, y \geq 0\}$ and $R^{--} := \{(x, y, z) : x \leq 0, y \leq 0\}$, respectively, and generated by one type of flow. The right two patches are in the regions $R^{++} := \{(x, y, z) : x \geq 0, y \geq 0\}$ and $R^{+-} := \{(x, y, z) : x \geq 0, y \leq 0\}$, respectively, and generated by a different flow. Figures (d), (f) and (h) are the mean curvature plots of figures (c), (e) and (g), respectively. The mean curvature at each vertex is computed by (171).

The aim of figure (c) is to show the difference between the mean curvature flow and the averaged mean curvature flow, where the left part is generated by the averaged mean curvature flow and the right part is produced by the mean curvature flow. The mean curvature flow shrinks the surface very fast while the averaged mean curvature flow does not. Further evolution using the mean curvature flow will yield a pinch-off of the surface. Therefore, if we model a surface patch using second order flows with G^0 boundary condition, the averaged mean curvature flow is more desirable

than the mean curvature flow.

The patches in R^{-+} and R^{--} of figure (e) are produced by the sixth order flow (169) (with $k = 3$), while the patches in R^{++} and R^{+-} are produced by the fourth order flow (169). As a whole, the surface looks smooth, our curvature plot reveals the smoothness difference at the intersection curves, the sixth order flow gives a smoother result than the fourth order flow.

Figure (g) is produced as (e), but the continuity order at the four circles are set to zero. Hence G^0 continuity is achieved there.

2. Surface Blending

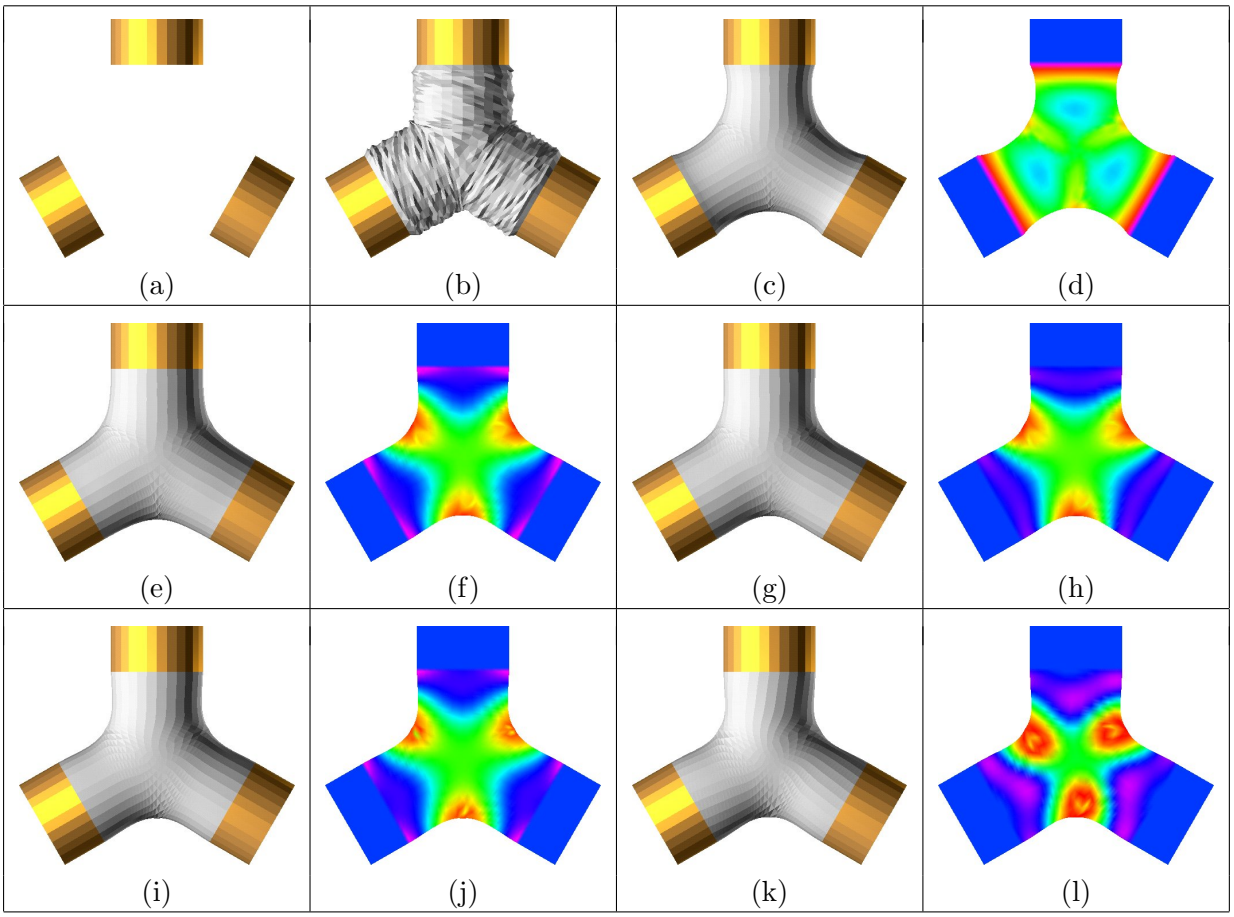


Figure 48: (a) shows three cylinders to be blended. (b) shows the initial construction. (c), (e) and (g) are the faired blending meshes generated using the flow (169) with $k = 1, 2, 3$, respectively. These figures show the results after 32, 32 and 60 iterations with time step sizes 0.01, 0.001, and 0.0001, respectively. (d), (f) and (h) show the mean curvature plots correspondingly. (i) and (k) are the blending meshes generated using the flow (168) with $k = 1, 2$, respectively. These figures show the results after 32 and 60 iterations with time step sizes 0.002 and 0.0002, respectively. Figure (j) and (l) show the mean curvature plots of (i) and (k), respectively

Given a collection surface mesh with boundaries, we construct a fair surface to blend the meshes at the boundaries with specified geometric continuity. Fig 48 shows the case, where three cylinders to be blended are given (figure (a)) with an initial G^0 construction (figure (b)) using [27] with some additional noise added. The blending surfaces (c), (e) and (g) are the faired blending meshes generated using the flow (169) with $k = 1, 2, 3$, respectively. These figures show the results after 32, 32 and 60 iterations with time step sizes 0.01, 0.001, and 0.0001, respectively. Figure (d), (f) and (h) show the mean curvature plots correspondingly. These figures clearly show the difference of smoothness achieved at blending boundaries. The mean curvature flow gives G^0 continuity results.

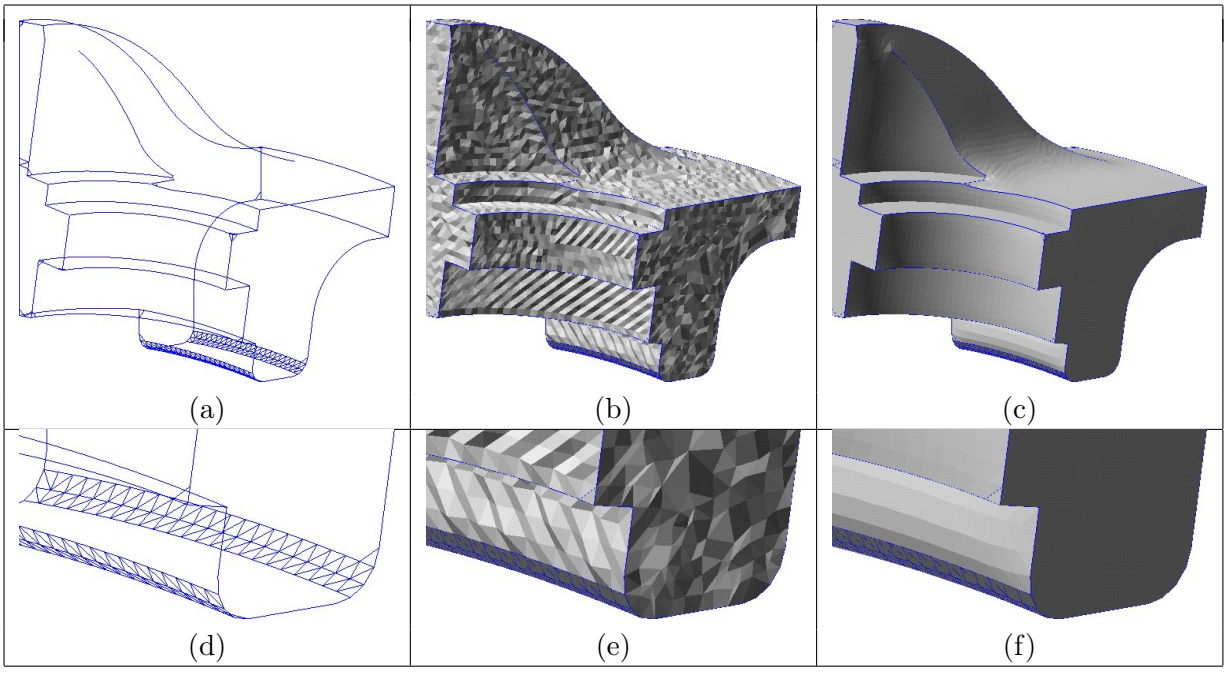


Figure 49: Interpolating curves and patches: (a) shows some input curves with G^0 continuity requirement and some bands of mesh with G^1 continuity requirement. (b) shows an initial construction of the surface mesh. (c) is the faired surfaces, after 12 iterations, generated using the the flow (169) with $k = 2$. The time step size is chosen to be 0.001. (d), (e) and (f) are the zoom in results of (a), (b) and (c), respectively.

The fourth order flow produces smooth surfaces at boundaries. The sixth order flow produces even smoother surfaces as expected.

Fig (i) and (k) are the faired blending meshes generated using the flow (168) with $k = 1, 2$, respectively. These figures show the results after 32 and 60 iterations with time step sizes 0.002 and 0.0002, respectively. Figure (j) and (l) show the mean curvature plots of (i) and (k), respectively. It should be noted that the flows (169) generate little fatter surface than the flows (168).

3. N-sided Hole Filling

Given a surface mesh with a hole, we construct a fair surface to fill the hole with specified geometric continuity on the boundary. Fig 43 shows such an example, where a head mesh with a hole in the nose subregion is given as input (figure (a)). An initial G^0 reconstruction of the nose is shown in (b) using [27] and then evolved with the mean curvature flow. The blending surfaces (figures (c) and (d)) are generated using the flow (169) with $k = 2$ and 3, respectively. It should be observed that the sixth order flow yields a better restoration surface. The head mesh with the hole in the nose subregion is available from http://lsec.cc.ac.cn/˜xuguo/xuguo2.htm.

4. Free-Form Surface Construction

For the free-form surface fiffing problem, we are given some curves, or partial patches, or points as input, and we wish to construct a fair surface mesh to interpolate this multi-dimensional data. Fig. 49 shows the approach of free-form surface construction, where some input curves with G^0 continuity requirement are given to preserve the sharp edges, and also given are some surface bands with a G^1 continuity requirement (see (a)). Figure (b) shows an initial construction of the G^0 surface mesh using the patch filling scheme [238] with added noise. (c) is the faired surfaces, after 12 iterations, generated using the the flow (169) with $k = 2$. The time step size is chosen to be 0.001. Figures (d), (e) and (f) are zoomed in views of (a), (b) and (c), respectively.

Fig. 50 shows the free-form fitting approach from an input triangular mesh, where (a) shows the input surface triangular mesh with a G^1 continuity requirement at the vertices (see (a)). Figure (b)

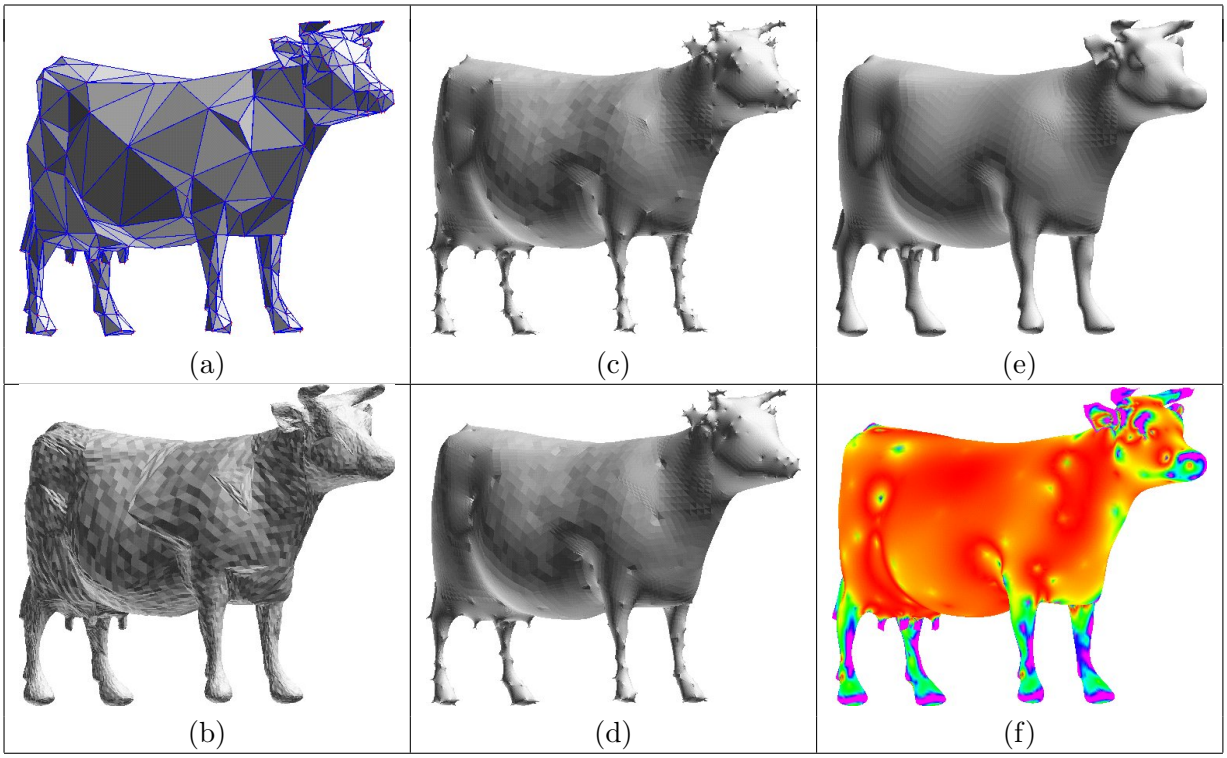


Figure 50: Interpolating points: (a) shows some input points and their triangulation. (b) shows an initial construction of the surface mesh. (c) and (d) are the faired surfaces, after 2 iterations with $\tau^{(n)} = 0.01$, using the mean curvature flow and the averaged mean curvature flow, respectively. (e) is faired surfaces, after 2 iterations with $\tau^{(n)} = 0.001$, using the fourth order flow (169). (f) is the mean curvature plot of (e).

shows an initial construction of the surface mesh, where each input triangle is approximated with 16 sub-triangles. The newly introduced vertices are treated as unknowns and the input vertices are fixed in the fairing process. Figures (c) and (d) are the faired meshes, after 2 iterations with $\tau^{(n)} = 0.01$, generated using the mean curvature flow and the averaged mean curvature flow, respectively. (e) is the faired mesh by fourth order flow, after 2 iterations with $\tau^{(n)} = 0.001$. (f) is the mean curvature plot of (e). The area shrinking of the mean curvature flow makes the input vertices to be interpolated become thorns (see (c)), while the area shrinking and the volume preservation of the averaged mean curvature flow make some of input vertices become thorns and some others become pits (see (d)). However, the fourth order flow does not suffer from this problem (see (e)). The obtained surface interpolates the input points and exhibits G^1 smoothness everywhere as well.

Conclusions We have presented a general scheme for using PDEs to solve several surface modelling problems and with high order boundary continuity conditions. Our scheme has the following features: It produces very fair and desirable solution surfaces. It is simple and easy to implement. Specifically, it solves the free-form blending problem, the N-sided hole filling problem and free-form surface fitting problem in a uniform fashion, and solves the high order boundary continuity problem in an easy and natural way and avoids prior estimation of normals or derivative jets on the boundaries. The implementation results show that our solution works well for a wide range of surface models. Note that the C^1 or higher order continuity interpolatory surface blending solution produced by e.g. [27, 188] for complicated corners, or holes with many boundary curve segments, are usually of very high algebraic degree and thereby prone to be with unsuitable for certain applications. The current solution of starting with G^0 low degree blends, coupled with higher order flow evolution, yields in general a much better alternative for very smooth surface solutions.

Both the geometric flows and quasi geometric flows yield smooth surfaces at the boundaries. However, quasi geometric flows (169) have some attractive features, including ease of implementation, smaller and better behaved coefficient matrices and no requirement of derivatives (normal) estimation.

4 Reconstruction of Point Clouds

4.1 Smooth Reconstruction from Scattered Data

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 [15] 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 [14] have also developed a similar method based on tensor-product Bernstein-Bézier patches.

A different, three-step solution is given by Hoppe et al.[129, 128, 132]. 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 [50, 172], or rather higher order polynomial [196], or a large number of pieces [5] compared to the approach of [15]. The method of [5] 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 [235] 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 [196] that does not split each tetrahedron uses degree 9 polynomials and requires C^4 data at the vertices. In extending the method of [196] 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 [38] 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.

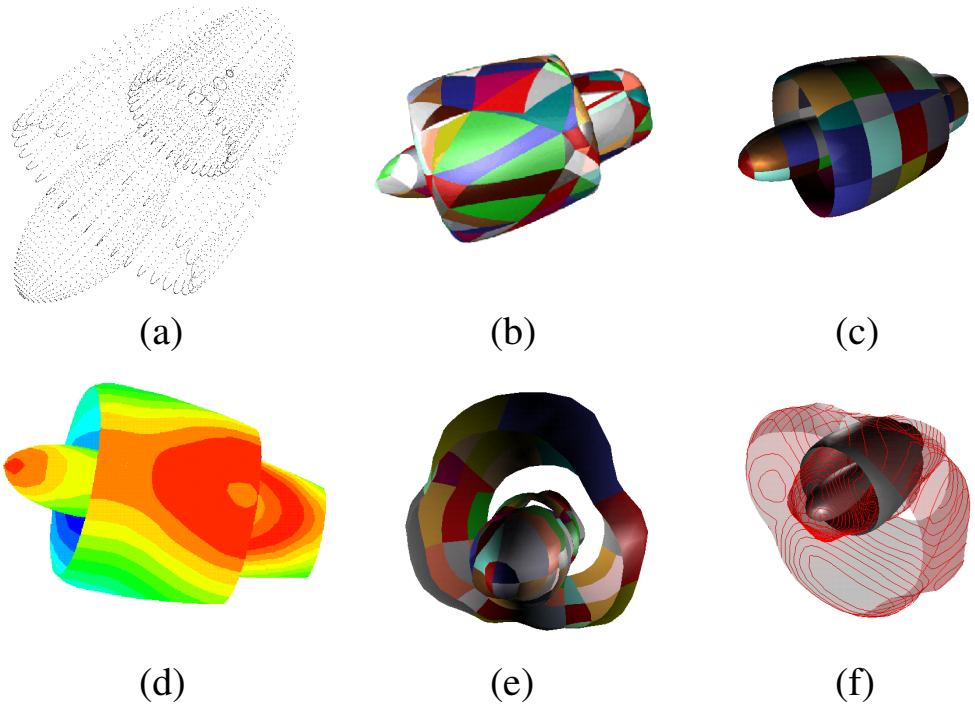


Figure 51: Jet engine model and associated pressure (scalar) field defined on the surface and their respective reconstruction from scattered point data (a) Input point data from the surface of the jet engine cowlings (b) reconstructed model with cubic A-patches defined within a 3D triangulation (c) reconstructed model with bicubic A-patches over a box decomposition (d) isocontours of a pressure field displayed on the jet engine surface (e) reconstructed model of the pressure field defined on the jet engine surface, using bicubic A-patches (f) pressure field with iso-contours displayed surrounding the jet engine.

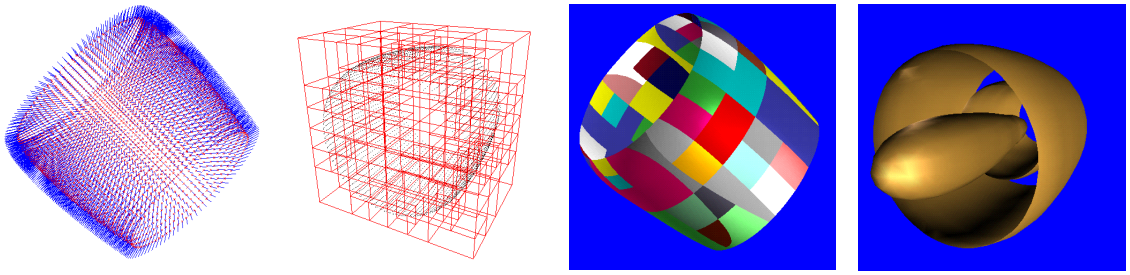
4.2 Volumetric Data Fitting

A-patches can be naturally used in the reconstruction of surfaces and volumetric scalar fields.

Bajaj, Bernardini and Xu [15] reconstruct laser scanned data (point-clouds) using A-patches. Their scheme utilizes an incremental Delaunay 3D triangulation for a more adaptive fit; the dual 3D Voronoi diagram for efficient point location signed distance computations and cubic implicit surface patches. Furthermore, at the same time they also compute a C^1 smooth approximation of the sampled function-on-surface. Bajaj, Bernardini and Xu [14] have also developed a similar method based on tensor-product Bernstein-Bézier patches.

Figure 52 shows several steps of the reconstruction process on one part of the engine. We used the normal propagation method in this example to define an approximate signed distance function. The 3780 data points for the outer cowl are preprocessed to associate local fitting planes and orient the associated normals (Figure 52(a)). The approximation algorithm begins with a given grid (in this case, a uniform subdivision into $5 \times 5 \times 5$ equally-sized boxes) and then adaptively refines it until the error bound conditions are met (the error in this example was set to 0.01 times the max size of the object). The final subdivision is displayed in Figure 52(b). A C^1-smooth piecewise polynomial surface is obtained, as shown in Figure 52(c). The full reconstructed engine is finally shown in Figure 52(d).

In many important cases, a physical phenomenon is measured by sampling the value of a scalar or vector field at points on the surface of some object. For example, one might have sampled the



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

Figure 52: Reconstruction of a jet engine from manifold data: (a) Input points for the outer cowl, with the oriented normals. (b) Octree subdivision generated by the approximation algorithm. (c) Piecewise polynomial approximation. (d) Reconstructed engine.

temperature on the surface of a jet-engine, or the acceleration of a fluid flow on the wing of an airplane. Other cases of interest are electroencephalogram data on the surface of the scalp, or the amount of precipitation on the earth. We will consider the problem of reconstructing both the surface of an object and a (possibly multivariate) field on it from scattered, dense data (what we mean by *dense* will be more clearly stated later). We will assume that both the surface and the field are continuous and have continuous first-order derivatives. Since data measurements are subject to error, we will not try to exactly interpolate the data but rather approximate it within a given tolerance. We will define a way to measure the error-of-fit for both the surface and the data later. The problem may be formally stated as follows:

Definition 4.1. *Given a set of dense scattered points* $P = \{p_i\}_{i=1}^N \subset \mathbb{R}^3$ *on an unknown manifold* M*, and associated data values* $V = \{v_i\}_{i=1}^N$*, construct a smooth surface* $S : s(x, y, z) = 0$*, such that* S *approximates* P *within a given error bound* ϵ_S*, and a smooth function* $F : f(x, y, z)$*, such that* F *approximates the data* V *associated with* P *within a given error bound* ϵ_F*.*

The problem of reconstructing the approximation S to the unknown manifold M has attracted the interest of many authors.

Most of the known methods use parametric or functional surface patches in either local interpolation or global interpolation. A few papers use implicit surface patches. We use a piecewise implicitly defined tensor-product algebraic surface to approximate the unknown surface M.

The problem of interpolating data defined over a *given* manifold in $\mathbb{R}^3$ is commonly referred to as *modeling 3D scattered manifold data* or the *surface-on-surface* problem

Our method is based on constructing an approximation of the signed-distance function $\delta(p, M)$ defined in Section 6.2.2. A similar approach has been used in some of the papers cited above. A novelty of the method proposed is in the *adaptive* approximation of δ with tensor-product Bernstein-Bézier polynomial patches, with the required continuity conditions. Compared to the methods mentioned above, our approach has the advantage of using an octree-like cubic mesh for any scattered data. This makes it easy to use, and adaptively capable of handling irregular data. Moreover, it handles the problems of reconstructing the surface and r scalar fields on it with a uniform approach. Once a piecewise Bernstein-Bézier polynomials representation of the surface and the scalar fields has been constructed, the data can be easily visualized and interacted with. Polygonal shading or ray tracing can be used to display the reconstructed surfaces. Isocontouring, or the normal projection method can be used to display the fields over the surface. The weights (coefficients) of the various patches can be interactively and locally modified.

4.2.1 Outline of the algorithm

If M is a connected and orientable surface in $\mathbb{R}^3$, then it is possible to define (in all $\mathbb{R}^3$) a function $\delta(p)$, called signed-distance, by

$$\delta(p) := sign \cdot dist(p, M)$$

where $dist(p, M)$ denotes the Hausdorff distance from the point p to the surface M and the *sign* is chosen so that $\delta(p)$ is positive when p is on one side of M, and negative when p lies on the other side. Then $\delta(p) = 0$ will recover the surface M.

When only discrete data on a surface is available, an approximate signed-distance can be defined in some appropriate way (see e.g. [167, 130, 15]). Given the signed-distance function, one can approximate it with a piecewise polynomial function $s(x, y, z)$ (in a suitable domain containing P), and then extract the zero-contour of s.

The algorithm proposed consists of the following phases:

1. **Build an approximation of the signed-distance function.** Preprocess the data so that, for a given query point q, the (approximate) value of $\delta(q, M)$ cam be computed. Notice that this requires a topologically consistent reconstruction of the surface orientation at each point. This step can be seen as transforming the problem from a surface-data reconstruction to a volume-data approximation.

2. **Approximate the signed-distance by a piecewise polynomial function.** Build, in an adaptive fashion, a piecewise polynomial, C^1-smooth approximation $s(x, y, z)$ of $\delta(p, M)$. The piecewise polynomial is built by least squares fitting of trivariate polynomials, in each cube of an octree-like subdivision of a domain containing P, to the data points within the cube and to additional samples of the signed-distance function δ defined in phase 1 above. If the error-of-fit in a cube of the subdivision exceeds the given bounds, then the cube is subdivided into eight sub-cubes and the process is repeated in each sub-cube. The reconstructed domain is implicitly defined as $s(x, y, z) = 0$.

3. **Approximate the scalar field defined over M.** Concurrently to the approximation of the signed-distance function, a piecewise polynomial approximation of the scalar field can be computed in a similar fashion by least squares fitting of the scalar field data in each cube in the octree.

In the following sections we will detail the algorithm outlined above.

4.2.2 From surface data to volume data: the signed-distance function

Using the signed-distance function to reconstruct a surface from scattered data points has been considered by several authors.

Moore and Warren [167] use a tetrahedral decomposition of the space, and reconstruct the surface by implicit barycentric Bernstein-Bézier patches. The signed-distance function used is sampled at data points (where it is obviously zero) and at *auxiliary* points, chosen as the vertices of a regular partitioning of each tetrahedron into sub-tetrahedra. If the sampling is dense enough, then the sub-tetrahedra containing data points partition the tetrahedron into two components, so that a sign can be associated with the distance at each vertex of the grid. This dense-sampling assumption can be too restrictive in some practical cases.

Hoppe et al. [130] use a more global approach to correctly orient the approximated manifold. First, for each data point p_i, they compute a best fit plane, and the associated normal $\hat{n}_i$, based on k neighboring points. Then they build the *Riemannian Graph*, $RG(P)$ over P (two points $p_i, p_j \in P$ are connected by an edge in $RG(P)$ iff either p_i is in the k-neighborhood of p_j or p_j is in the k-neighborhood of p_i). Each edge (i, j) is assigned the weight $1 - |\hat{n}_i \cdot \hat{n}_j|$, and a minimum spanning

tree is computed. They then orient the plane associated with the point with the largest z-value so that its normal points toward the positive z-direction, and propagate this orientation to other points traversing the minimum spanning tree. The traversing order implicit in the MST avoids, in many examples illustrated in their paper, an incorrect orientation of parts of the manifold. Their method continues with the construction of a regular subdivision of a parallelepiped containing the data points into cubes. The value of δ is computed at all vertices of the subdivision as the signed distance of the vertex from the oriented plane associated with the closest point in P. An algorithm similar to *marching cubes* is then used to construct a piecewise-linear approximation of the zero contour of δ. In two subsequent steps, described in [131, 127], the constructed mesh is optimized (i.e., the number of triangles is reduced while the distance of the mesh from the data points is kept small) and then a smooth surface is built on it. While this approach gives very convincing results, and allows for smooth objects with sharp features to be correctly reconstructed, the computational time required by the optimization and smoothing steps is significant.

Bajaj et al. [15] propose the use of α-shapes [100, 102] to build a piecewise-linear approximation of the surface being reconstructed. The α-shape is a sub-complex of the Delaunay triangulation of the set of points P. This approach has the advantage of being based on a well-founded mathematical definition of the *shape* of a set of points. The piecewise-linear approximation is then used to compute the value of the signed-distance function at any point. A piecewise polynomial approximation is subsequently built on an adaptive Delaunay 3D triangulation of a domain containing P.

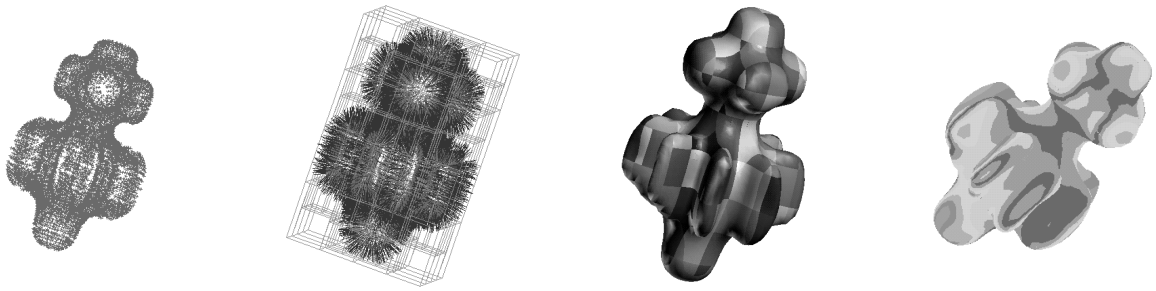
All the three schemes described above have advantages and disadvantages. In all three cases, the method used to define an approximate signed-distance function can be used as the first step of our algorithm. In our current implementation. we are using a *propagation* approach similar to that of [130]:

Algorithm 4.2. *1.* **Local approximation.** *Let P be the given surface data, then for each point $p \in P$, construct a local linear approximation $l_p(x, y, z) = ax + by + cz + d$ by least squares fitting p and other $k(\geq 2)$ nearest points of P. If the points used are collinear the number of points used is increased.*

2. **Orienting normals.** *For each point $p \in P$, construct a normal n_p from the local approximation $l_p(x, y, z) = 0$. That is, $sign \cdot \nabla l_p(p)/\|\nabla l_p(p)\|$. The direction (the sign in the formula) of the normal must be chosen so that all the normals point toward the same side of the surface.*

In the following we assume that the surface data P is ρ-dense, that is any sphere with radius ρ and center in M contains at least one point in P. Our propagation algorithm starts with assigning an orientation to the six points having minimum or maximum x, y or z coordinate (the normal at the point with max z coordinate clearly must point upward, etc). Points whose associated normal has been oriented, but such that neighboring points might not have been oriented are called boundary points and are kept in a list. A point p is extracted from the list and all points contained in a ball centered in p and of radius ρ' (ρ' is a parameter to be chosen a priori, depending on the density ρ of the data set) are oriented (if they had not been oriented before) accordingly to the orientation of p (i.e., in such a way that the the scalar product of the two normals is positive).

3. **signed-distance function evaluation.** *The distance $|\delta(p, M)|$ is computed by first finding a point $q \in P$ that has minimal distance to p. Then $|\delta(p, M)|$ is defined as the local minimal distance from p to $l_q(x, y, z) = 0$ around q. If the query point is outside the sphere $S(q, r)$, with center q and radius r, then use $|\delta(p, M)| = \|p - q\|$. The radius r can be taken to be the maximal distance of the k points that defined l_q from q. The sign of $|\delta(p, M)|$ is taken to be the sign of the inner product of n_q and $p - q$.*



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

Figure 53: Reconstruction of a surface and an associated scalar field from scattered data: (a) Input points. (b) Orientation of normals and octree subdivision. (c) Piecewise polynomial approximation. (d) Reconstructed scalar field.

4.2.3 Piecewise polynomial approximation of signed-distance

In this section, we describe how to use piecewise polynomials to approximate the signed-distance function. Let $D = [\alpha_1, \alpha_2] \times [\beta_1, \beta_2] \times [\gamma_1, \gamma_2]$ be a parallelepiped containing the data set P. The outline of the algorithm is as follows:

Algorithm 4.3. *1* **Initial Partition.** *Construct a partition of D. That is choose a_i, b_j, c_k so that*

$$\begin{array}{ccccccccccc} \alpha_1 & = & a_0 & < & a_1 & < & \dots & < & a_{\ell_1} & = & \alpha_2 \\ \beta_1 & = & b_0 & < & b_1 & < & \dots & < & b_{\ell_2} & = & \beta_2 \\ \gamma_1 & = & c_0 & < & c_1 & < & \dots & < & c_{\ell_3} & = & \gamma_2 \end{array}$$

and partition D by $D = \bigcup D_{ijk}$, with $D_{ijk} = [a_{i-1}, a_i] \times [b_{j-1}, b_j] \times [c_{k-1}, c_k]$ and $i = 1, \dots, l_1, j = 1, \dots, l_2, k = 1, \dots, l_3$.

2 **Local Fitting.** *For each element D_{ijk} that is within the distance ρ from P (if the cube is away from P by a distance bigger than ρ, we regard this cube as containing no surface), construct the (either tri-quadratic or tri-cubic) function $w = s_{ijk}(x, y, z)$ that fits the signed-distance function $\delta(p)$ at $p_s = (x_s, y_s, z_s) \in D_{ijk}$ in the least squares sense. The points p_s used are the points in $P \cap D_{ijk}$ and, additionally, the vertices of a regular grid on the cube D_{ijk} (the number of such points can be chosen depending on the number of data points available). The auxiliary grid points in the least square fit help preventing the function s_{ijk} from having multiple sheets.*

3 **Adaptive Step.** *Compute the algebraic distances of the computed patch from the data points. If the max distance is bigger than the given tolerance ϵ, subdivide D_{ijk} into eight equally-sized sub-cubes and return to Step 2. However, do not subdivide cubes whose side length is less than 2ρ.*

At the end of the iterative, adaptive fitting implemented by the steps outlined above, we have computed local polynomial approximations to the signed-distance function. Obviously, there is no guarantee that adjacent pieces join continuosly. Therefore we need an averaging phase (called free-form blending in [167]) in which values and derivatives at vertices of the octree subdivision are obtained evaluating the polynomials associated with incident cubes and taking their average (possibly weighted to take into account the number of data points used in the fitting and the goodness-of-fit achieved in each cube).

Notice that the adaptive refinement of the subdivision can be represented as an octree data structure. In the following we will refer to the *level* of a cube with the following meaning: cubes at

level 1 are those with the coarser subdivision. Cubes at level $i+1$ have been obtained subdividing a cube at a level i into eight sub-cubes.

Algorithm 4.4. *4* **Extract Values.** *For each vertex of level 1 cubes in the final partition, compute function values and the required derivative values (C^3 data, see Section 4.2.4) by averaging the corresponding values of the neighboring cubes polynomials. Then compute a new polynomial in each level* 1 *cube using the averaged data.*

5 **Recursive C^1 Interpolation.** *Compute a new polynomial at each level* 2 *cube interpolating the C^3 data at its vertices. For vertices lying inside the face of an adjacent level* 1 *cube, first evaluate the polynomial in the adjacent cube to compute the C^3 data. Continue in a similar fashion with level i cubes, for $i > 2$, until all patches have been computed.*

At the end of the top-down construction of interpolants, we have obtained a C^1 smooth, piecewise trivariate polynomial, whose zero-contour approximates the points P.

Step 3 guarantees that the scheme is adaptive. The final partition produced by the algorithm is in general not uniform. Hence two adjacent cubes may have different sizes. Two adjacent cubes that share a part of the common face join in one of the following two fashions:

1. The two faces coincide completely.
2. One contains the other as a proper subset.

If the second case happens, then in Step 5 the interpolant for the larger cube will be computed first, and then this interpolant will be used to compute the C^3 data at the remaining vertices of the smaller cube. This guarantees the global C^1 continuity of the constructed function. We shall detail the construction method for the interpolants in the following sections.

In Step 2, we use algebraic polynomials to fit the data. Since the least squares approximation is not an exact fit, one might question if an approximate fit to the signed-distance could possibly lead to a multi-sheeted surface. For this, a theorem like Theorem 1 in [167] for a tetrahedron can be established for our cubic scheme:

Theorem 4.5. *Let $P_{ijk} = P \cap D_{ijk}$. For a sufficiently small ϵ, if there exists a plane $\pi(x,y,z) = 0$ such that all points of P_{ijk} lie within a distance ϵ from the plane, then the zero contour of the local fitting $s_{ijk}(x,y,z)$ is smooth, single sheeted in D_{ijk} and lies within some distance, depending only on ϵ and the degree of s_{ijk}, from the plane $\pi(x,y,z) = 0$.*

Another way of guaranteeing single-sheetedness is the following. In [186] is has been pointed out that if all the weights increase or decrease monotonically along one of the coordinate directions, then straight lines parallel to that direction intersect the surface patch at most once, i.e. the patch is single-sheeted. We state another characterization in the following:

Lemma 4.6. *If there exists an integer l $(0 < l < m)$ such that*

$$w_{ijk} \leq 0, \quad i = 0,1,...,l-1; \quad j = 0,1,...,n; \; k = 0,1,...,q \tag{188}$$

$$w_{ijk} \geq 0, \quad i = l+1,...,m; \quad j = 0,1,...,n; \; k = 0,1,...,q \tag{189}$$

and there is at least one strict inequality in each set of inequalities, then straight lines parallel to the x-direction intersect the surface patch exactly once. Similar conclusions hold for the y and z-direction intersections.

Proof. The Lemma can be easily proved using the variation-diminishing property of Bernstein-Bézier polynomials. □

Since the averaging in Step 4 will make the final approximation differ from the local approximation of Step 2, one should note that this change may destroy the smooth and single sheeted properties of the local approximation. To avoid this from happening, the averaged values should have a small difference from the local values. If the data is dense and the partition is fine enough, this will be guaranteed.

4.2.4 C^1 Interpolation of C^3 data by $(3,3,3)$- and $(2,2,2)$-polynomials

By C^3 data of a function f at a point p, we mean that we are given values at p for

$$f, \quad \frac{\partial f}{\partial x}, \quad \frac{\partial f}{\partial y}, \quad \frac{\partial f}{\partial z}, \quad \frac{\partial^2 f}{\partial x\,\partial y}, \quad \frac{\partial^2 f}{\partial x\,\partial z}, \quad \frac{\partial^2 f}{\partial y\,\partial z}, \quad \frac{\partial^3 f}{\partial x\,\partial y\,\partial z}. \tag{190}$$

This Section shows a way of constructing tri-cubic and tri-quadratic interpolants over a cube with C^3 data on its vertices so that the composite function is C^1 continuous. The following Lemma tells us how to compute the BB form coefficients around a vertex from the C^3 data there.

Lemma 4.7. *Let $W(x,y,z) = \sum_{i=0}^{m} \sum_{j=0}^{n} \sum_{k=0}^{q} w_{ijk} B_i^m(u) B_j^n(v) B_k^q(w),\ m>0,\ n>0,\ q>0$, be a BB form polynomial on the cube $D = [a_1,a_2]\times[b_1,b_2]\times[c_1,c_2]$. Then W interpolates the C^3 data (190) at the vertex (a_1,b_1,c_1) if and only if*

$$w_{000} = f \tag{191}$$

$$w_{100} = w_{000} + \frac{a_2-a_1}{m}\frac{\partial f}{\partial x} \tag{192}$$

$$w_{010} = w_{000} + \frac{b_2-b_1}{n}\frac{\partial f}{\partial y} \tag{193}$$

$$w_{001} = w_{000} + \frac{c_2-c_1}{q}\frac{\partial f}{\partial z} \tag{194}$$

$$w_{110} = w_{010} + w_{100} - w_{000} + \frac{(a_2-a_1)(b_2-b_1)}{mn}\frac{\partial^2 f}{\partial x \partial y} \tag{195}$$

$$w_{101} = w_{001} + w_{100} - w_{000} + \frac{(a_2-a_1)(c_2-c_1)}{mq}\frac{\partial^2 f}{\partial x \partial z} \tag{196}$$

$$w_{011} = w_{001} + w_{010} - w_{000} + \frac{(b_2-b_1)(c_2-c_1)}{nq}\frac{\partial^2 f}{\partial y \partial z} \tag{197}$$

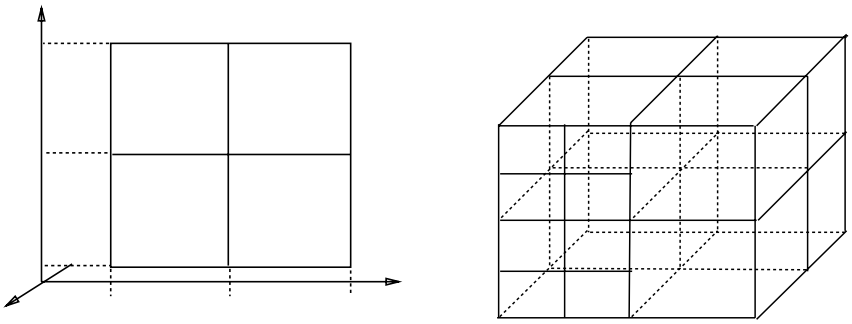
$$\begin{aligned} w_{111} = {} & w_{011} + w_{101} - w_{001} + w_{110} - w_{010} - w_{100} + w_{000} \\ & + \frac{(a_2-a_1)(b_2-b_1)(c_2-c_1)}{mnq}\frac{\partial^3 f}{\partial x \partial y \partial z} \end{aligned} \tag{198}$$

Similar conclusions hold for the other 7 vertices of the cube D.

Lemma 4.8. *Let W_1 and W_2 be polynomials defined on equally sized cubes D_1 and D_2, adjacent along the x-direction (see 84). Then if both W_1 and W_2 interpolate C^3 data at the four common vertices of D_1, D_2, four x-direction collinear conditions are satisfied at each of the common vertices. Similar conclusions hold for y- or z-adjacent cubes.*

Theorem 4.9. *If W_1 and W_2 in Lemma above are tri-cubic, and if both W_1 and W_2 interpolate C^3 data at the common four vertices of two adjacent cubes D_1 and D_2, then W_1 and W_2 are C^1 continuous on the common face of D_1 and D_2.*

The last Theorem says that, if a volume consists of cubes, and if a BB polynomial on each cube is constructed from C^3 data at the vertex by formulae (191)—(198), then the composite function is C^1 continuous. The discussion above also shows that $m=n=q=3$ is the minimal degree for forming C^1 piecewise functions, since there is no degree of freedom left. If a lower degree polynomial is used or some degrees of freedom are required, one has to subdivide each cube into smaller sub-cubes.



(a) (b)

Figure 54: (a) Six collinear conditions. (b) Subdivision of a cube D into eight sub-cubes.

Tri-quadratic interpolation. C^1 interpolation of the C^3 data can also be achieved by a tri-quadratic piecewise polynomial. However, this requires splitting each cube in eight sub-cubes to create additional degrees of freedom in the choice of coefficients.

The first Lemma in the following says that the six C^1 conditions on a plane and around a vertex are not independent.

Lemma 4.10. *Let $R = [a_1, a_3] \times [b_1, b_3]$ be a rectangle in the plane (see Figure 54(a)), and $w_1, \ldots, w_9$ be values on the nine grid points. Then the six collinear conditions*

$$\frac{w_{i+1} - w_i}{a_2 - a_1} = \frac{w_{i+2} - w_{i+1}}{a_3 - a_2}, \qquad i = 1, 4, 7 \tag{199}$$

$$\frac{w_{i+3} - w_i}{b_2 - b_1} = \frac{w_{i+6} - w_{i+3}}{b_3 - b_2}, \qquad i = 1, 2, 3 \tag{200}$$

are satisfied if any five of them are satisfied.

The next Lemma guarantees that by subdivision, the tri-quadratic interpolants over the eight sub-cubes exist uniquely and the composite function is C^1 continuous.

Lemma 4.11. *Let $D = [a_1, a_2] \times [b_1, b_2] \times [c_1, c_2]$ be a given cube. At each of its eight vertices $P_{i_1 i_2 i_3}$, we are given C^3 data. If we subdivide D into eight sub-cubes $D_{i_1 i_2 i_3}$ (see Figure 54(b)), then there exists uniquely one piecewise function W on D such that*

a. $W_{i_1 i_2 i_3} = W|_{D_{i_1 i_2 i_3}}$ is a (2,2,2)-polynomial, that interpolates the set of C^3 data at $P_{i_1 i_2 i_3}$.

b. W is C^1 continuous on D.

c. If W' is defined in the same way over an adjacent cube D', then W and W' are C^1 continuous on the common face of D and D'.

4.3 Hierarchical Multiresolution Reconstruction of Shell Surfaces

Many human manufactured and naturally occurring objects have shell-like structures, that is, the object bodies consist of surfaces with thickness. We call such surfaces *shell surfaces*. The problem of constructing smooth approximations to shell surface objects arises in creating geometric models such as airfoils, tin cans, shell canisters, engineering castings, sea shells, the earth's outer crust, and the human skin, to name just a few.

In engineering, shell structures are often analyzed by finite element methods (see [54, 59, 76, 165, 221]). In these analyses, the shell is often assumed to be uniform in thickness for simplicity, hence the output is often a triangulation that represents the mid-surface of the shell. More accurate finite element analysis of shells uses volume elements, such as hexahedral (see [153]) or pentahedral (see [221]).

In these cases, the output may be a matched triangulation pair. Reference [40] also presents schemes for obtaining matched triangulation pairs for varied scattered and dense surface data inputs. Our aim is to reconstruct smooth shells from matched triangulation pairs. However, we do not assume that the shell is uniform in thickness; instead, we assume we are given two triangulations that represent the boundaries of the shell. These triangulations can be obtained by offsetting the mid-surface triangulation in the normal direction with varying thickness. In the model (such as airfoil, arched roof and dam etc.) construction, we must often respect the geometric data and therefore cannot assume the shell is uniform in thickness. Hence, our problem may be described as follows.

Problem Description. *As input we are given a matched triangulation pair* $\mathcal{T} = \{\mathcal{T}^{(0)}, \mathcal{T}^{(1)}\}$ *with attached normals at each vertex, which presents a linearization of the inner and outer boundary surfaces of a shell domain; also, we are given an error control tolerance* $\epsilon > 0$. *The goal is to reconstruct hierarchical multiresolution smooth shell surfaces whose bounding surfaces provide approximations of* $\mathcal{T}^{(0)}$ *and* $\mathcal{T}^{(1)}$*, respectively, with errors no larger than* ϵ.

The hierarchical scheme is comprised of multiresolutions in two directions. The terminology *Hh-multiresolution* we use is borrowed from *hp*-finite element analysis (see [221]), where their "h" elements denote the mesh size and their "p" elements denote the degree of the shape functions on each mesh element. Here the H-direction multiresolution is a level-of-detail (LOD) representation of the pair of (irregular) triangular meshes, and the h-direction multiresolution is the regular subdivision of each of the triangle pairs. In the geometric modeling problem, using high degree polynomials in general leads to surfaces containing pronounced waves and also increases the computational costs. Hence, we use triangular cubic spline functions on the regularly partitioned triangles.

We extend the reconstruction method of [40] to achieve (a) hierarchical Hh-multiresolution, (b) ϵ-bounded approximations, and (c) the ability to capture sharp curve creases while being C^1 smooth everywhere else. To achieve adaptive multiresolution representations in the H-direction, a hierarchical presentation of the prism scaffold is constructed. For each extracted scaffold from the hierarchy, a sequence of functions (h-direction) is constructed, using triangular splines on regularly subdivided triangular prisms, such that the input data is approximated to within the allowable error ϵ. To get an adaptive reconstruction, combinations of different levels in both the H- and h-directions are allowed.

In Section 4.3.1 we introduce some notation used for describing our algorithm. We then outline in Section 4.3.2 the complete algorithm steps for solving the proposed problem. These steps are detailed in the sections that follow. Two heavy tasks in this algorithm, that are tackled in Sections 4.3.3 and 4.3.5, are the geometric construction of the hierarchical scaffold and the C^1 function construction over the scaffold. The hierarchical construction of the prism scaffold is the same in nature as that of the hierarchical construction of a unique triangulation, but with some adjustments to fit our shell triangulation problem. The C^1 construction in Section 4.3.5 is basically a local interpolation approach, that utilizes mainly the tools of one-dimensional Hermite interpolation, one-dimensional B-splines, two-dimensional triangular B-splines, and transfinite interpolation.

4.3.1 Notation for Shell Surfaces

We assume $\mathcal{T}^{(0)}$ and $\mathcal{T}^{(1)}$ are orientable. For each vertex pair $V_i = \{V_i^{(0)}, V_i^{(1)}\}$ with attached normal pair $\{N_i^{(0)}, N_i^{(1)}\}$, we assume

$$[V_i^{(1)} - V_i^{(0)}]^T N_i^{(s)} > 0, \quad s = 0, 1.$$

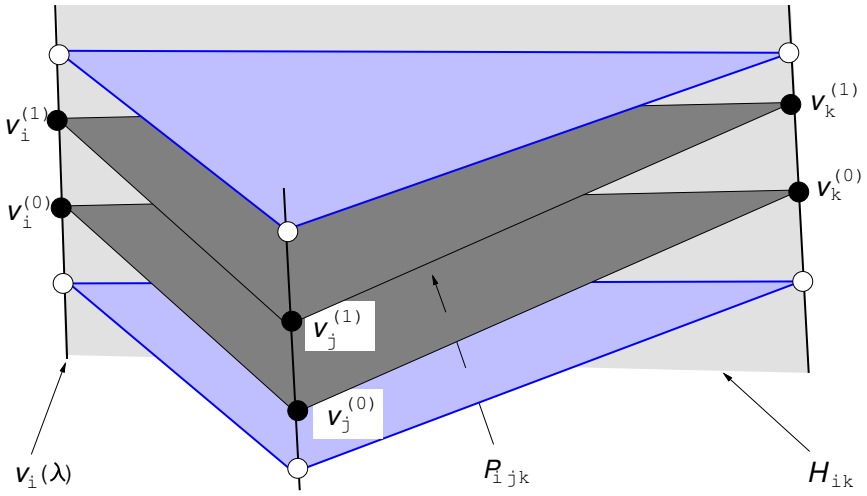


Figure 55: The volume prism cell P_{ijk}, the face $H_{ik}(t,\lambda)$ and the edge $v_i(\lambda)$ defined by a triangle pair $[V_iV_jV_k]$.

This ensures that points in the outer layer are roughly in the same direction from the corresponding points in the inner layer as the normals. With this convention the normals to both the inner and outer surfaces are outward-pointing normals, in contrast to the more common convention where the normals to the inner surface are inward-pointing. For each triangle pair $[V_iV_jV_k]$, we further assume

$$[V_i^{(0)}V_j^{(0)}V_k^{(0)}] \cap [V_i^{(1)}V_j^{(1)}V_k^{(1)}] = \emptyset.$$

Our trivariate function F for constructing the shell is piecewise defined on a collection of prisms. Let $[V_iV_jV_k]$ be a triangle pair. Then the prism, denoted by P_{ijk}, for $[V_iV_jV_k]$ is a volume in $\mathbb{R}^3$ enclosed by the surfaces H_{ij}, H_{jk}, and H_{ki} (see Figure 55), where H_{lm} is a ruled surface defined by V_l and V_m as follows:

$$H_{lm} = \{p : p = b_1 v_l(\lambda) + b_2 v_m(\lambda), \qquad b_1 + b_2 = 1, \qquad \lambda \in \mathbb{R}\}$$

with $v_i(\lambda) = V_i^{(0)} + \lambda N_i, N_i = V_i^{(1)} - V_i^{(0)}$. We will wish to describe points within these prisms in terms of the triangle vertex pairs, as a type of "shell barycentric coordinate." To this end we can explicitly represent the prism P_{ijk} as the volume given by

$$P_{ijk}(I) = \{p : p = b_1 v_i(\lambda) + b_2 v_j(\lambda) + b_3 v_k(\lambda), \qquad b_1 + b_2 + b_3 = 1, b_l \geq 0, \lambda \in I\},$$

where I is a specified interval. This interval contains $[0,1]$ and is usually larger, so that each prism P_{ijk} contains the triangle pair $[V_i^{(0)}V_j^{(0)}V_k^{(0)}]$ and usually extends past its faces, as illustrated in Figure 55. We call (b_1, b_2, b_3, λ) the P_{ijk}–coordinate of $p = p_{ijk}(b_1, b_2, b_3, \lambda) = b_1 v_i(\lambda) + b_2 v_j(\lambda) + b_3 v_k(\lambda)$. For each $\lambda \in I$, $T_{ijk}(\lambda) := \{p : p = b_1 v_i(\lambda) + b_2 v_j(\lambda) + b_3 v_k(\lambda), b_1 + b_2 + b_3 = 1,\ b_l \geq 0\}$ defines a triangle. To ensure that this triangle is non-degenerate, λ is confined to lie in a certain interval I_{ijk}. This interval is computed as follows.

Let $p_{ijk}^{(l)}(\lambda) = \det[n_l, v_j(\lambda) - v_i(\lambda), v_k(\lambda) - v_i(\lambda)]$, $l = i, j, k$. Assume

$$p_{ijk}^{(l)}(\lambda) > 0, \qquad \forall \lambda \in [0,1], \qquad l = i, j, k\ . \tag{201}$$

Consider the real numbers $\lambda_1, \cdots, \lambda_s$ ($s \leq 6$) that solve one of these three equations of degree 2: $p_{ijk}^{(l)}(\lambda) = 0, l = i, j, k$, and define $a = \max(-\infty, \{\lambda_l : \lambda_l < 0\})$, $b = \min(+\infty, \{\lambda_l : \lambda_l > 1\})$, and

$I_{ijk} = (a, b)$. Then I_{ijk} is the largest interval containing $[0, 1]$ such that $P_{ijk}(I_{ijk})$ is non-degenerate. To show this fact, note that a triangle $T_{ijk}(\lambda)$ is non-degenerate if and only if

$$n_l^T [v_j(\lambda) - v_i(\lambda)] \times [v_k(\lambda) - v_i(\lambda)] = p_{ijk}^{(l)}(\lambda) > 0, \tag{202}$$

$l = i, j, k$, where $\times$ denotes the cross product of two vectors. The assumption (201) implies that $[0, 1] \subset I$. Since $p_{ijk}^{(l)}(0) > 0$ and $p_{ijk}^{(l)}(1) > 0$, for $l = i, j, k$, then $p_{ijk}^{(l)}(\lambda) > 0$ for $\lambda \in (a, b)$ and $l = i, j, k$. Since $p_{ijk}^{(l)}(a) = 0$ for $l = i$ or $l = j$ or $l = k$ if $a > -\infty$, a is the infimum of the interval of λ that contains $[0, 1]$ and makes (202) hold. Similarly, b is the supremum of such an interval. Therefore I_{ijk} is the largest interval such that $P_{ijk}(I_{ijk})$ is non-degenerate.

We call the union of all $P_{ijk}(I_{ijk})$ a prism scaffold. For the input triangulation pair, the corresponding scaffold, denoted as $\mathcal{S}_0$, will be the finest level in our hierarchical representation of the scaffold. Note that the triangulation (we always mean the matched triangulation pair) and the scaffold correspond closely. The vertex V_i, edge $[V_iV_j]$ and triangle $[V_iV_jV_k]$ of the triangulation correspond to the edge $v_i(\lambda)$, face H_{ij} and prism P_{ijk} of the scaffold, respectively. Hence, any operation conducted on the triangulation implies the same on the scaffold. For instance, removing a vertex pair from the triangulation and then re-triangulating implies removing an edge from the scaffold and then "re-meshing" the prism scaffold. These operations are performed in building the hierarchical representation of the scaffold in Section 4.3.3.

Given a P_{ijk}-coordinate for a point, it is straightforward to compute its coordinates in the xyz system. However, the inverse is not trivial, since the transforms between them are nonlinear. For a given $p \in \mathbb{R}^3$, we determine $(b_1, b_2, b_3, \lambda)^T$ such that

$$p = b_1 v_i(\lambda) + b_2 v_j(\lambda) + b_3 v_k(\lambda), \qquad b_1 + b_2 + b_3 = 1 \, . \tag{203}$$

It follows from (203) that we have

$$p - v_k(\lambda) = [v_i(\lambda) - v_k(\lambda), v_j(\lambda) - v_k(\lambda)] \, [b_1, \ b_2]^T \, .$$

Therefore,

$$\det[p - v_k(\lambda), v_i(\lambda) - v_k(\lambda), v_j(\lambda) - v_k(\lambda)] = 0 \, . \tag{204}$$

The left-hand side of (204) is a polynomial of degree 3 in λ. Upon solving this equation for λ, we choose the root such that the solution (b_1, b_2, b_3) of (203) satisfies $b_i \geq 0$, $\sum b_i = 1$.

Whenever it is necessary to address the functions that are defined on the level t scaffold (H-direction), the notation $F^{(t)}$ is used. The notation F_σ will be used to address the level σ function in the h-direction.

4.3.2 Algorithm Outline

The hierarchical construction algorithm of the shell structures is comprised of two main phases: the hierarchical construction of the scaffold and of the function over the scaffold. This section gives the algorithm pipeline, with the details of the algorithm provided in the sections that follow.

Step 1. *Construct a C^1 function on the scaffold $\mathcal{S}_0$.*

The finest level scaffold $\mathcal{S}_0$ is built on the input matched triangulation pair. On this scaffold, a C^1 function $F^{(0)}$ is constructed (see Section 4.3.6). This function is regarded as exact when constructing other functions at other resolutions.

Step 2. *Hierarchical representation of scaffold.*

This step constructs a directed acyclic graph (DAG) for the levels of detail of the scaffold. This DAG is built based on the algorithm in [89, 92], with changes to the vertex removal criterion and hole re-triangulation method (see Section 4.3.3). Having such a DAG, we are able to travel from a fine level to a coarse one or vice versa, and extract a required scaffold satisfying a given control error by combining different levels.

Step 3. *Adaptive scaffold extraction.*

For the given control parameters, extract a required scaffold from the DAG that satisfies the given condition (see Section 4.3.4).

Step 4. *Face data construction.*

For each face of the prisms in any level, a C^2 function and C^1 gradient on the face are constructed. All these data form a list. In the DAG structure, each prism should have three pointers that point to the corresponding face data (see Section 4.3.8). Having this data, we are able to construct C^1 functions on any extracted scaffold.

Step 5. *Construct trivariate splines in each prism.*

For the given control error ϵ and the selected scaffold, construct a sequence of C^1 trivariate splines F_σ, $\sigma = 1, 2, \cdots, \Sigma$, so that

$$S_\alpha^{(\sigma)} = \{p : F_\sigma(p) = \alpha,\ \alpha \in [-1, 1]\}, \qquad \sigma = 1, 2, \cdots, \Sigma,$$

are smooth surfaces and $S_{-1}^{(\Sigma)}$ and $S_1^{(\Sigma)}$ are ϵ error-bounded approximations of the inner and outer boundary surfaces of the input shell, respectively. In the process of this construction certain **curve creases** are tagged and captured. This step is described in detail in Sections 4.3.9.

Step 6. *Evaluate and display the shell surface.*

4.3.3 Hierarchical Representation of Prism Scaffold

The hierarchical representation of the scaffold is a sequence $\mathcal{S}_0$, $\mathcal{S}_1$, $\cdots$, $\mathcal{S}_k$ of scaffolds, from the finest level to the coarsest. To construct the hierarchical representation of the scaffold, we perform recursively a vertex removal procedure to form a sequence $\mathcal{T}_0$, $\mathcal{T}_1$, $\cdots$, $\mathcal{T}_k$ of matched triangulation pairs, where $\mathcal{T}_0 = \mathcal{T}$. The policy of the vertex removal is adopted from [89, 92]. That is, if one vertex is selected to be removed at level t, then its neighbor vertices at the same level may not be removed. Hence any two vertices in the set of vertices that are going to be removed are disconnected (see Figure 56). The next level of triangulation is obtained by re-triangulating holes that are left when the vertices are removed.

The hierarchy is stored as a directed acyclic graph (DAG), whose nodes correspond to the prisms of $\mathcal{S}_0$ up to $\mathcal{S}_k$. The leaf nodes correspond to the prisms of $\mathcal{S}_0$. Between the levels t and $t+1$, there is an intermediate level that corresponds to the removed vertices of level t. There is an arc from the star-shaped polygon that is formed when a vertex is removed, to every triangle in level t around the vertex, and to every triangle in level $t+1$ formed by the re-triangulation of the star-shaped polygon. A polygon at the intermediate level between levels t and $t+1$ is called the *parent polygon* of those prisms at level t that linked to it, and it is also called the *child polygon* of those prisms at level $t+1$ that are linked to it. Unchanged triangles between two levels are linked directly by arcs. These descriptions are illustrated by Figure 56.

Some data must be stored along with the DAG. First is the vertex pair list *VertexList*, which is fixed and does not change during the construction of the DAG. Another list is *FaceList*, that of the faces of the prism, which is incremental. The initial list is that of the faces of $\mathcal{S}_0$. Each entry of *FaceList* contains the information of the C^2 function and the C^1 gradient on that face (see Section 4.3.8). When new prisms are produced, the new faces are added to this list. In the DAG, three pointers that point to the three faces of each prism must be stored. Having this information allows us to construct later C^1 functions within each prism cell.

To achieve our goal of building the hierarchical representation of the scaffold, there are two points that need to be addressed. One is the vertex removal criterion, and the other is the re-triangulation.

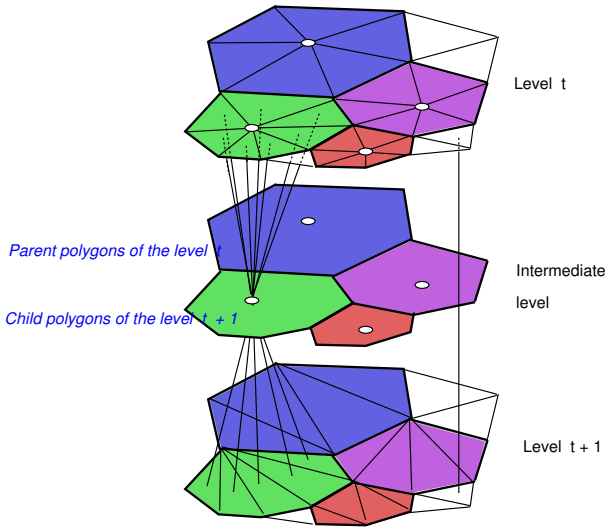


Figure 56: The vertices with circles at the top level are the ones that are removed at level t. The star-shaped polygons, that are shown in the middle of the figure and obtained by removing the selected vertices, are re-triangulated as shown at the bottom. Newly formed prisms at level $t+1$ are linked to those at level t by arcs through the intermediate level. The unchanged prisms are linked directly.

4.3.4 Adaptive Extraction of Shell Surface Support

It is obvious that simply taking a certain level of the scaffold from the hierarchy does not have the adaptive nature. Therefore, it is necessary to combine different level scaffolds to form an adaptive one. The extraction algorithm in [89, 92] can be altered to serve our purpose. From the construction of the DAG we know that each prism in any level has a grade that measures the normal variation. We shall use this grade to control the scaffold extraction for a given control value $g \in [0, \pi/2)$ of the normal variation. To describe the extraction algorithm precisely, we introduce some more notation. Let P be a prism of level t. Then we denote by $G_t(P)$ the collection of all prisms at level t that are in the same child polygon as P, and by $G_t^c(P)$ the collection of all prisms at level $t-1$ that are linked to child polygons of P. Let $Sub_t(P)$ be the collection of all prisms in the levels $t, t-1, \cdots, 0$, that are linked directly or indirectly through intermediate nodes to the prisms in $G_t^c(P)$. That is, $Sub_t(P)$ consists of the prisms in the sub-DAG starting with $G_t(P)$. Then the algorithm for extracting the scaffold can be described by the following C language style pseudo-code:

```
Q_k = S_k;                /* put all the prisms in S_k to Q_k*/
for (t = k;  t > 0;  t --) {
    Q_{t-1} = NULL;
    while (Q_t ≠ NULL ) {
      P = Q_t[0];
      if (G_t(P) ⊄ Q_t) {
          accept all the prisms in G_t(P);
      }   else {
          if (Grade(p) ≤ g for all p ∈ Sub_t(P) ) {
            accept all the prisms in G_t(P)
          } else {
            append to the end of Q_{t-1} all the prisms in G_t^c(P)
          }
```

}
remove from Q_t all the prisms that in $G_t(P)$;
}
}
if ($Q_0 \neq$ NULL) {
accept all the prisms in Q_0;
} .

4.3.5 Construction of C^1 Trivariate Functions on Hierarchy

The C^1 functions on the hierarchy are constructed in three steps: (a) A C^1 function $F^{(0)}$ on $\mathcal{S}_0$ is constructed first (Section 4.3.6). This function serves us as an exact reference while constructing the functions at other levels. (b) C^1 data are computed for each face of each prism in each level (Section 4.3.8). (c) C^1 functions are constructed for each prism of any extracted scaffold that interpolates the vertices of the scaffold and fit $F^{(0)}$ by splines (Section 4.3.9).

4.3.6 Function over the Finest Level $\mathcal{S}_0$

The function $F^{(0)}$, whose level surfaces $F^{(0)}(x, y, z) = -1$ and $F^{(0)}(x, y, z) = 1$ will approximate the inner and outer surfaces, is constructed in two steps. First, function values and gradients (C^1 data) are defined on each of the faces of all the prisms, and second, the function is defined within the prisms, using the C^1 data on the prism faces.

Now we define C^1 data on the faces. Let $H_{lm}(t, \lambda)$ be a face of the prism P_{ijk} where $(l, m) \in \{(i, j), (j, k), (k, i)\}$. Then the function value on this face is defined by cubic Hermite interpolation on the line segment $[v_l(\lambda)\ v_m(\lambda)] = \{p \in \mathbb{R}^3 : p = H_{lm}(t, \lambda), t \in [0, 1]\}$ by interpolating the directional derivatives $D^s_{[v_m(\lambda)-v_l(\lambda)]^s} F(v_l(\lambda))$ and $D^s_{[v_m(\lambda)-v_l(\lambda)]^s} F(v_m(\lambda))$ for $s = 0, 1$. Hence, $F(H_{lm}(t, \lambda))$ can be written as

$$\begin{aligned} F(H_{lm}(t,\lambda)) &= F(v_l(\lambda))H_0^3(t) + F(v_m(\lambda))H_2^3(t) \\ &\quad + [v_m(\lambda) - v_l(\lambda)]^T \nabla F(v_l(\lambda))H_1^3(t) \\ &\quad + [v_m(\lambda) - v_l(\lambda)]^T \nabla F(v_m(\lambda))H_3^3(t)\ , \end{aligned} \tag{205}$$

where $H_0^3(t) = 1 - 3t^2 + 2t^3$, $H_1^3(t) = t - 2t^2 + t^3$, $H_2^3(t) = 3t^2 - 2t^3$, $H_3^3(t) = -t^2 + t^3$ are Hermite interpolation base functions, and

$$F(v_i(\lambda)) = 2\lambda - 1, \qquad \nabla F(v_i(\lambda)) = (1 - \lambda)N_i^{(0)} + \lambda N_i^{(1)}. \tag{206}$$

Here we have normalized the normals $N_i^{(0)}$ and $N_i^{(1)}$ such that $N_i^T N_i^{(0)} = N_i^T N_i^{(1)} = 2$ (recall that $N_i = V_i^{(1)} - V_i^{(0)}$), in order to have $D_{N_i} F = N_i^T \nabla F$ on the edge $v_i(\lambda)$. Let

$$d_1(\lambda) = v_m(\lambda) - v_l(\lambda), \tag{207}$$

$$d_2(t) = (1 - t)N_l + tN_m, \tag{208}$$

$$d_3(t, \lambda) = d_1 \times d_2. \tag{209}$$

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

$$\begin{cases} d_1^T \nabla F(H_{lm}(t,\lambda)) = \dfrac{\partial F(H_{lm}(t,\lambda))}{\partial t}, \\ d_2^T \nabla F(H_{lm}(t,\lambda)) = \dfrac{\partial F(H_{lm}(t,\lambda))}{\partial \lambda}, \\ d_3^T \nabla F(H_{lm}(t,\lambda)) = d_3^T \nabla \breve{F}_{lm}(t,\lambda), \end{cases} \tag{210}$$

where

$$\nabla \breve{F}_{lm}(t,\lambda) = (1-t)\nabla F(v_l(\lambda)) + t\nabla F(v_m(\lambda)) \ . \tag{211}$$

From (210) we have

$$\nabla F(H_{lm}(t,\lambda))^T = [P,Q,R][d_1,d_2,d_3]^{-1} \tag{212}$$

where $[d_1,d_2,d_3]^{-1} = [d_1\|d_2\|^2 - d_2(d_1^T d_2),\ d_2\|d_1\|^2 - d_1(d_1^T d_2),\ d_3]^T/\|d_3\|^2$, and P, Q and R are the right-hand sides of (210).

Next we define C^1 functions within prisms. Let $[V_1V_2V_3]$ be a typical triangle pair. The C^1 function F in the prism P_{123} is defined by the side-vertex scheme defined by Theorem 3.1 in [170]:

$$F(p_{123}(b_1,b_2,b_3,\lambda)) = \sum_{i=1}^{3} w_i D_i(b_1,b_2,b_3,\lambda), \tag{213}$$

where $w_i = \prod_{j\neq i} b_j^2 / \sum_{k=1}^{3}\prod_{j\neq k} b_j^2$, and D_i is defined by Hermite interpolation from the data on the prism faces (see [40] for details). Explicitly,

$$\begin{aligned} D_i(b_1,b_2,b_3,\lambda) &= F(p_i(b_1,b_2,b_3,\lambda))H_0^3(b_i) \\ &\quad + d_i(b_1,b_2,b_3,\lambda)^T\nabla F(p_i(b_1,b_2,b_3,\lambda))H_1^3(b_i) \\ &\quad + F(v_i(\lambda))H_2^3(b_i) + d_i(b_1,b_2,b_3,\lambda)^T\nabla F(v_i(\lambda))H_3^3(b_i), \end{aligned}$$

where

$$\begin{aligned} p_i(b_1,b_2,b_3,\lambda) &= \frac{b_i}{1-b_i}v_j(\lambda) + \frac{b_k}{1-b_k}v_k(\lambda), \\ d_i(b_1,b_2,b_3,\lambda) &= -\frac{b_j}{1-b_i}e_k(\lambda) - \frac{b_k}{1-b_i}e_j(\lambda), \end{aligned}$$

and $(i,j,k) \in \{(1,2,3),(2,3,1),(3,1,2)\}$, $e_k(\lambda) = v_j(\lambda) - v_i(\lambda)$, $e_j(\lambda) = v_k(\lambda) - v_i(\lambda)$.

4.3.7 Minimal Prism with ϵ Offset

As mentioned above, the shell $\{p \in \mathbb{R}^3 : F^{(0)}(p) \in [-1,1]\}$ constructed in this section is considered to be exact, and the other shells constructed later will approximate this shell to within the error ϵ. Therefore, this shell with its ϵ offset is required to be contained in all of the other scaffolds. This requirement will be one of the conditions in building the hierarchical representation of the scaffold. Since testing whether a triangular shell with ϵ offset is contained in another scaffold is time-consuming, we determine a minimal prism that contains the triangular shell with ϵ offset. In building the hierarchical representation, the shell containment requirement will be replaced by the minimal prisms containment requirement. Let $[V_iV_jV_k]$ be a triangle pair. Let $p^{(0)}$ be the point in $\mathbb{R}^3$ with P_{ijk}-coordinate $(b_1^{(0)},b_2^{(0)},b_3^{(0)},\lambda^{(0)})$ and let $p^{(1)}$ be the intersection point of the surface $F^{(0)} = 1$ and the line $b_1^{(0)}v_i(\lambda) + b_2^{(0)}v_j(\lambda) + b_3^{(0)}v_k(\lambda)$, where they intersect at $\lambda = \lambda^{(1)}$. Then

$$\|p^{(0)} - p^{(1)}\| = |\lambda^{(0)} - \lambda^{(1)}|\|b_1^{(0)}N_i + b_2^{(0)}N_j + b_3^{(0)}N_k\| \ .$$

Then we require $|\lambda^{(0)} - \lambda^{(1)}| \geq \epsilon/\sqrt{M}$, where $M := M(N_i,N_j,N_k)$ is the minimal value of the degree two Bézier polynomial $\|b_1N_i + b_2N_j + b_3N_k\|^2$ on the triangle $\{b_1+b_2+b_3 = 1, b_i \geq 0\}$. Let $I_{ijk}^{min} = [a,b]$ be the minimal interval such that $P_{ijk}(I_{ijk}^{min})$ contains the triangular shell. Then we define the minimal prism as $P_{ijk}(I_{ijk}^{\epsilon})$ with $I_{ijk}^{\epsilon} = [a - \epsilon/\sqrt{M}, b + \epsilon/\sqrt{M}]$. The interval $[a,b]$ can be computed by numerical methods (see Section 4.3.1).

4.3.8 Computation of Face Data

The function F in each prism is defined by transfinite interpolation of the data on the face of the prism (see Section 4.3.6 or 4.3.9). To ensure that F is C^1 in the prism, the function and the gradient on the face need to be C^2 and C^1, respectively. Now we define the C^2 function $F(H_{lm})$ and C^1 gradient $\nabla F(H_{lm})$ on every face H_{lm} of each prism in every level. For the finest level, these functions have been defined by (205) and (210). Now we consider the functions on other levels. Though the face data on level $t+1$ could be incrementally computed from the data of level t, we compute data on level $t+1$ from level zero to avoid error accumulation. The DAG constructed enables us to trace back to $\mathcal{S}_0$ to locate the required data from level zero. Let

$$F(H_{lm}(t,\lambda)) = G_{lm}(t,\lambda) + \phi^{\sigma}_{lm}(t) + \psi^{\sigma}_{lm}(t)\lambda, \tag{214}$$

where $G_{lm}(t,\lambda)$ takes the same form as $F(H_{lm})$ in (205), and

$$\phi^{\sigma}_{lm}(t) = \sum_{i=2}^{2^\sigma-2} \phi_i N^{\sigma}_{i3}(t), \qquad \psi^{\sigma}_{lm}(t) = \sum_{i=2}^{2^\sigma-2} \psi_i N^{\sigma}_{i3}(t)\ ,$$

where $\{N^{\sigma}_{i3}(t)\}_{i=-1}^{2^\sigma+1}$ are C^2 cubic B-spline basis functions defined on the uniform knots $t_i = i/2^\sigma,\ i = 0, 1, \cdots,\ 2^\sigma$. Here we shift N^{σ}_{i3} so that t_i is the center of the support $\mathrm{supp}N^{\sigma}_{i3} = ((i-2)/2^\sigma, (i+2)/2^\sigma)$. Note that the function values and the first order derivatives of ϕ^{σ}_{lm} and ψ^{σ}_{lm} are zero at the ends of the interval $[0,1]$.

Since G_{lm} depends on vertex information only and it is easy to construct, we do not store the data of G_{lm}, but only ϕ_i and ψ_i. These parameters are determined by approximating the two intersection curves of the finest level surfaces $F^{(0)} = \pm 1$ with the face H_{lm}, in the least square sense:

$$\int_0^1 \left[F(H_{lm}(t,\lambda_s(t))) + (-1)^s\right]^2\ dt = min, \qquad s = 0, 1, \tag{215}$$

where $\lambda_s(t)$, for fixed t, is defined by the intersection point of the line $(1-t)v_l(\lambda) + tv_m(\lambda)$ with the surface $F^{(0)} + (-1)^s = 0$. The required pieces of the intersection are obtained from the DAG. The minimization in (215) leads to a system of linear equations

$$\sum_{i=2}^{2^\sigma-2} \int_0^1 (\phi_i + \psi_i\lambda_s(t)) N^{\sigma}_{i3}(t) N^{\sigma}_{j3}(t)\ dt = c_j$$

with $c_j = -\int_0^1 \left[G_{lm}(t,\lambda_s(t)) + (-1)^s\right] N^{\sigma}_{j3}(t)\ dt$ and $j = 2, \cdots, 2^\sigma - 2,\ \ s = 0, 1$. The integrations in the system are computed by Gauss-Legendre quadrature rule on each of the sub-intervals $[i/2^\sigma, (i+1)/2^\sigma]$ and then summed up. This order $2(2^\sigma - 2)$ equation can be solved by solving two order $2^\sigma - 2$ linear systems. The intersection point $\lambda_s(t)$ is computed by Newton iteration, and the integer σ is chosen on trial bases. Starting from $\sigma = 1$, we solve the equation and compute the least square error. If the error is larger than the given ϵ, then increase σ by one, until the error is within the tolerance.

Next, we define the gradient $\nabla F(H_{lm}(t,\lambda))$ by the conditions (210), but $\breve{F}_{lm}(t,\lambda)$ is modified by adding a spline function:

$$\nabla\breve{F}_{lm}(t,\lambda) = (1-t)\nabla F(v_l(\lambda)) + t\nabla F(v_m(\lambda)) + \breve{\phi}^{\sigma}_{lm}(t) + \breve{\psi}^{\sigma}_{lm}(t)\lambda \tag{216}$$

with $\breve{\phi}^{\sigma}_{lm}(t) = \sum_{i=2}^{2^\sigma-2} \breve{\phi}_i N^{\sigma}_{i3}(t),\ \breve{\psi}^{\sigma}_{lm}(t) = \sum_{i=2}^{2^\sigma-2} \breve{\psi}_i N^{\sigma}_{i3}(t),$ where $\breve{\phi}_i, \breve{\psi}_i \in \mathbb{R}^3$ are determined by

$$\int_0^1 \|\nabla\breve{F}_{lm}(t,\lambda_s(t)) - \nabla F^{(0)}(H_{lm}(t,\lambda_s(t)))\|^2 dt = min \tag{217}$$

for $s = 0, 1$, and $\lambda_s(t)$ is defined as before. (217) can be solved together with (215) since they share the same coefficient matrix.

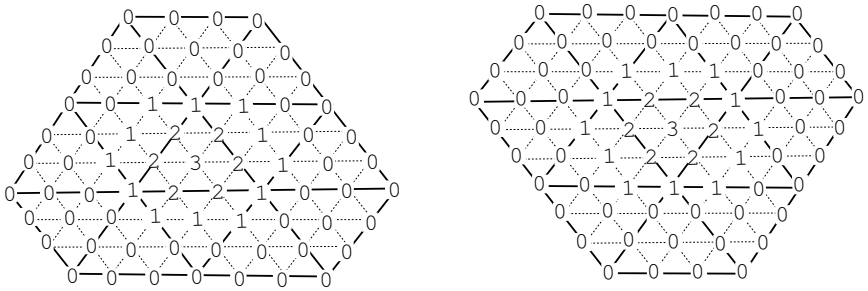


Figure 57: Bézier coefficients for two C^1 cubic spline basis functions. Each is defined on the union of 13 sub-triangles, which forms the support of the function.

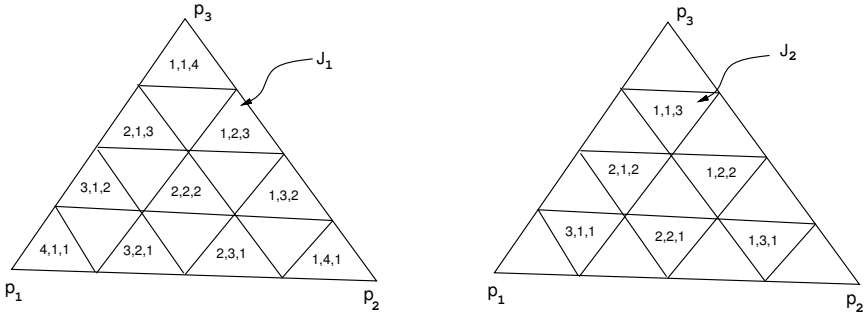


Figure 58: For the regular partition of a triangle with resolution 2^σ, the index set J^σ of the sub-triangles is divided into J_1^σ and J_2^σ. This figure shows them for $\sigma = 2$.

4.3.9 Construction of C^1 Spline Approximations

In this section, we construct a piecewise C^1 function $F = F_\sigma$ ($\sigma \geq 0$ fixed) over the collection of the volumes, such that it (Hermite) interpolates the C^1 data and fits $F^{(0)}$. To achieve ϵ approximation and multiresolution representation in the h-direction, spline functions defined on triangles are utilized in the construction of F. On a triangular domain with a regular partition, C^1 cubic splines defined in BB form were given by Sabin, 1976 (see [200]). Figure 57 gives the BB–form coefficients of a typical base function defined on 13 sub-triangles. Note that these splines in general are not linearly independent (see Bőhm, Farin and Kahmann [58]). However, the collection we use is indeed linearly independent. For a regular partition of a triangle T, we shall associate a base function to each sub-triangle of the partition. To give proper indices for these bases, we label the sub-triangles as T_{ijk} for $(i,j,k) \in J^\sigma = J_1^\sigma \cup J_2^\sigma$, where J_1^σ and J_2^σ are defined as follows:

$$\begin{aligned} J_1^\sigma &= \{(i,j,k): \quad i,j,k \in \{1,2,3,...,2^\sigma\}; \qquad i+j+k = 2^\sigma + 2\}, \\ J_2^\sigma &= \{(i,j,k): \quad i,j,k \in \{1,2,3,...,2^\sigma - 1\}; \qquad i+j+k = 2^\sigma + 1\}, \end{aligned}$$

where 2^σ is the resolution of the partition. Figure 58 gives J_1 and J_2 for $\sigma = 2$. Now we denote the base function defined by Figure 57 with center triangle T_{ijk} as N_{ijk}^σ.

4.3.10 F on Prisms

Let $[V_1V_2V_3]$ be a typical triangle pair. Define

$$F_\sigma(p_{123}(b_1,b_2,b_3,\lambda)) = \sum_{i=1}^{3} w_i D_i(b_1,b_2,b_3,\lambda) + T_\sigma(b_1,b_2,b_3,\lambda), \tag{218}$$

where the first term of left-hand side is in the same form as (213), and the second term is a spline function:

$$T_\sigma(b_1, b_2, b_3, \lambda) = \sum_{(i,j,k)\in J_3^\sigma} (a_{ijk} + w_{ijk}\lambda) N_{ijk}^\sigma(b_1, b_2, b_3)$$

with $J_3^\sigma = \{(i, j, k) \in J^\sigma : i > 1, j > 1, k > 1\}$. This is called a *correction term*, which is used to fit the finest level shell surface in the least square sense:

$$\iint_\Delta [F_\sigma(b_1, b_2, b_3, \lambda_s(b_1, b_2, b_3)) - (-1)^s]^2 dS = min \tag{219}$$

for $s = 0, 1$, where $\lambda_s(b_1, b_2, b_3)$ for each (b_1, b_2, b_3) is defined by the intersection point of the line $b_1v_1(\lambda) + b_2v_2(\lambda) + b_3v_3(\lambda)$ with the surface $F^{(0)} + (-1)^s = 0$. The required pieces of the intersection are obtained from the DAG. The domain Δ in the integration is the unit triangle defined by $\{(b_1, b_2, b_3) : b_1 + b_2 + b_3 = 1, b_i \geq 0\}$. The minimization in (219) leads to a system of linear equations.

4.3.11 Hierarchical Representation of Correction Term

In the construction of $F = F_\sigma$, we have associated it with an integer σ. This integer indicates the level of the hierarchical multiresolution representation of F in the h-direction. However, the construction and expression of F in Section 4.3.9.1 is not incremental, as the construction of $F_{\sigma+1}$ does not utilize the information of F_σ. In this subsection, we revise some parts of the construction in Section 4.3.9.1, so that F is progressively constructed. Now we want to have the following form expression:

$$T_\sigma = T_{\sigma-1} + \sum_{(i,j,k)\in J_3^\sigma \setminus 2*J_3^{\sigma-1}} (a_{ijk}^\sigma + w_{ijk}^\sigma\lambda) N_{ijk}^\sigma,$$

where $T_1 = 0$. Let

$$W^\tau = \text{span}\{N_{ijk}^\tau : \ i \in J_3^\tau \setminus 2 * J_3^{\tau-1}\}, \quad S^\tau = W^2 \oplus W^3 \oplus \cdots \oplus W^\tau.$$

Then S^τ is a C^1 cubic spline function space on a triangle partitioned regularly with resolution 2^τ. Once $T_{\sigma-1}$ has been defined, the coefficients a_{ijk}^σ and w_{ijk}^σ are computed by fitting $F^{(0)}$ in the volume. Since the elements in S^τ have zero function value and zero first order partial derivative values on the boundary of the triangle, we can use different σ for different prisms to get an adaptive construction without destroying the continuity of the composite function. For the prism P_{ijk}, let ϵ_{ijk}^σ be the fitting error. Then for any given fitting error tolerance ϵ, we can choose a minimal σ so that $\epsilon_{ijk}^\sigma \leq \epsilon$. This σ is prism dependent.

Basic Result *The composite function F, defined on any extracted scaffold and for any varying and prism-dependent $\sigma \geq 0$, is C^1.*

5 Reconstruction of Cross-Sectional Polygons and Splines

Measurement based cross-sectional data sets arise from medical imaging (Computed Tomography – CT, Magnetic Resonance Imaging – MRI, Laser Surface Imaging – LSI). Synthetic cross-sectional data sets are generated by computer based simulations – meteorological and thermodynamic simulations, finite element analyses, computational fluid dynamics, etc. Examples of our approach are shown in Figures 59 and 60 which were made in our SHASTRA scientific design environment.

Generating contours in image data, reconstructing digitized signals, and designing scalable fonts are only some of the several applications of spline curve fitting techniques. We generalize past fitting schemes with conic splines [190] and even rational parametric splines [83, 176, 211]. We exhibit efficient techniques to deal with cubic algebraic splines (A-splines) achieving fits with small number

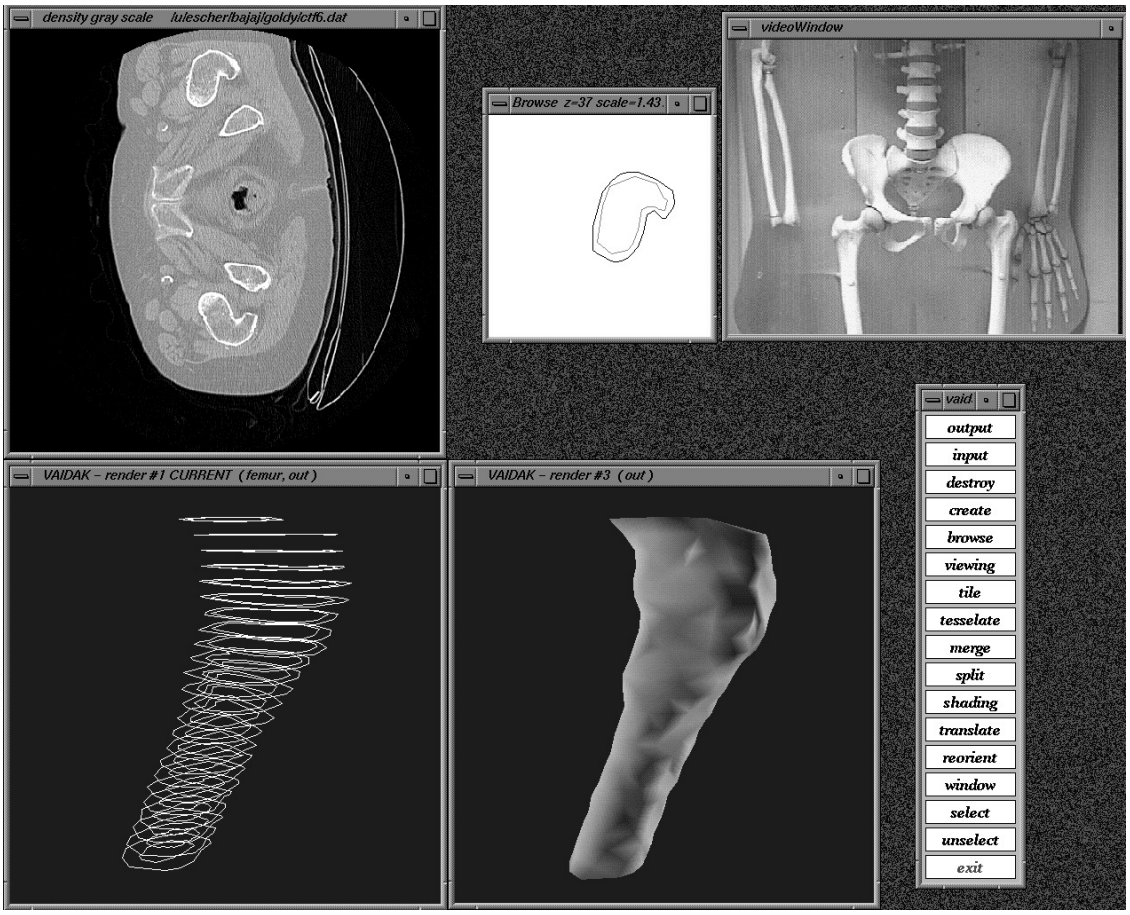


Figure 59: Spline models of a human femur from CT cross-sections.

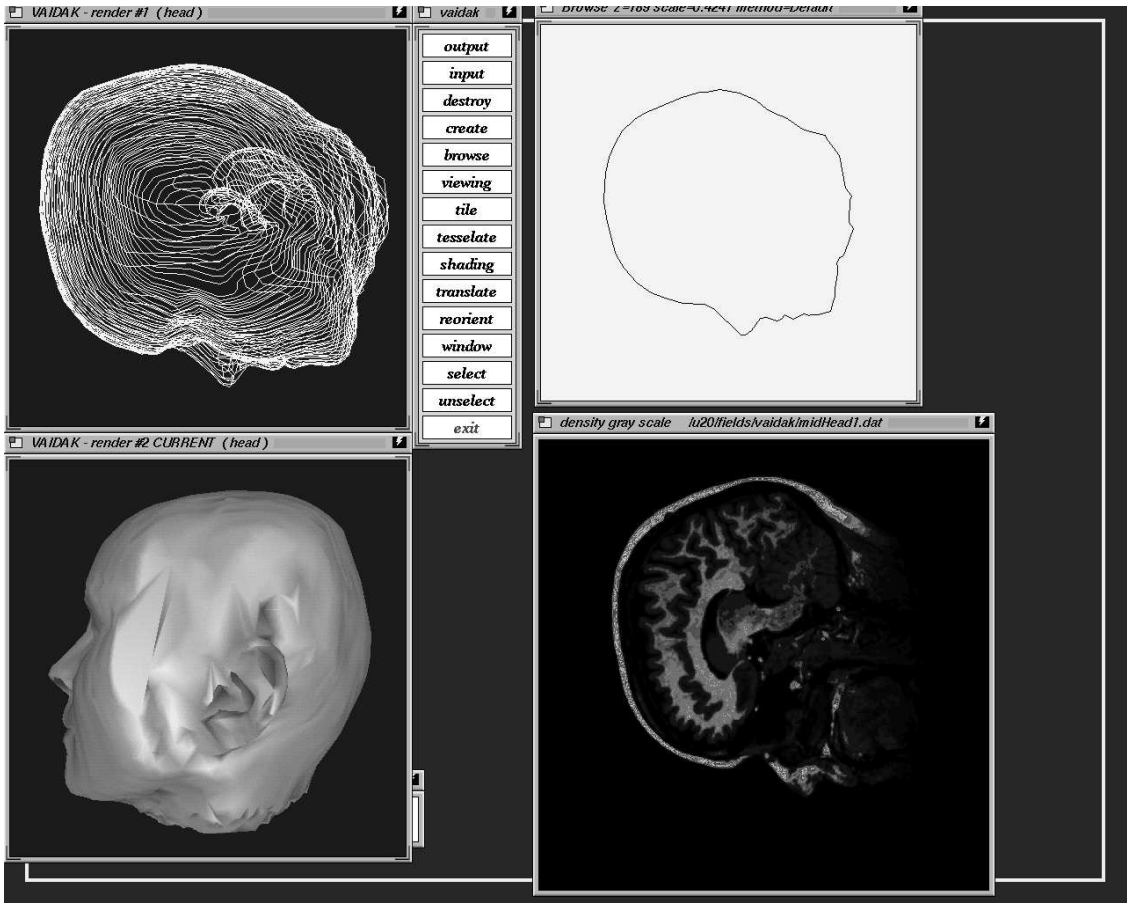


Figure 60: Spline models of a Human Head from MRI cross-sections.

of pieces yet higher order of smoothness/continuity or greater local flexibility for fixed continuity, than prior schemes. The cubic A-splines are continuous chains of cubic implicitly defined algebraic curve segments, $f_i(x, y) = 0$, with $f_i(x, y)$ a bivariate real polynomial, and with achievable local continuity as high as C^3 at the junction points between curve segments.

The primary drawback for the widespread use of splines consisting of implicit algebraic curves is that a single implicitly defined curve may have several real components (ovals) and can possess several real singularities. In [37] we show how to isolate a non-singular and single sheeted segment of implicit algebraic curves and furthermore how to stitch these segments together to form splines.

We focus on the case of cubic A-splines. We provide efficient algorithms for their use in fitting contour image data, ordered digital signal data, as well as randomly sampled scattered data sets. Note that rational parametric cubic splines can only achieve local C^2 continuity [93], compared to the local C^3 continuity of cubic A-splines. The class of rational parametric cubic curves is a strict subset of the class of cubic algebraic curves [225] and also has fewer degrees of freedom (8 versus 9 of the cubic algebraic curve). Note, of course that for fixed continuity (C^k, $k = 0, 1, 2, or 3$), the extra local degrees of freedom which the cubic A-spline segment posseses, allows for greater local flexibility and approximation of the input data.

Related Prior Work:

Since 1960's, considerable work on polynomial spline interpolation and approximation has been done(see [93] for a bibliography). In general, spline interpolation has been viewed as a global fitting problem to scattered data[83, 176, 190, 211]. Local interpolation by polynomials and rational functions is an old technique that traces back to Hermite and Cauchy[63]. However, local interpolation by the zero sets of piecewise polynomials (implicit algebraic curve segments) is relatively new[37, 184, 185, 212]. The papers [184, 185] construct a family of C^1 (actually tangent continuous) and C^2 (actually curvature continuous) cubic algebraic splines. They however use a reduced form of the cubic which guarantees that each segment of the spline is convex and furthermore allows one to achieve C^2 continuity only if the input data is convex. Furthermore, their family of curvature continuous curves [184] can achieve C^2 continuity only if the given data is convex. Their results are a special case of the present paper, as our cubic A-splines are based on the general implicit cubic, and as we show, can always be made to achieve C^3-continuity for arbitrary data, and even C^4-continuity for certain special input data[37].

5.1 Algorithms Using Cubic A-splines

Our data fitting algorithms with C^2 and C^3 cubic A-splines are as follows.

Algorithm 1.

1. *Extract a contour (ordered set of points) from the given input data. See subsection 5.2.*
2. *Compute breakpoints along the contour. These breakpoints points are the junction points for the cubic curves which make up the cubic A-spline. See subsection 5.3.*
3. *Compute derivatives at the junction points using local divided differences along the contour. For C^2 and C^3 continuity one needs upto second and third order derivatives, respectively, at these junction points. See subsection 5.4.*
4. *Construct cubic A-spline fits which interpolate the junction points along with the derivatives, and is least-squares approximate from all the given data between junction points. See subsection 5.5.*

Figure 61: Extracting an iso-contour (left) from a dense MRI slice (right)

5.2 Extracting an Iso-Contour from a Grey-scale Image

For iso-contour extraction from dense image data we use the following algorithm. The dense image data is in the form of a two dimensional array of two byte integers, one array for each planar slice through the object. The value in each cell (pixel) of the array is related to the density of the scanned object at that point in space. Each array may contain any number of iso-contours. To locate the iso-contours :

1. scan for a cell on an initial edge
2. starting at this cell hug the exterior of the cross section working from cell to cell and creating a list of two dimensional points until the beginning is reached or a dead end is found
3. if a dead end is found, backtrack
4. if the path closes and the algorithm does not backtrack to the beginning point then smooth and compress the list of points if necessary.

In our implementation of this algorithm the following heuristic rule was used: if the density value in a cell c is within range and if the density values of all the cells surrounding c are within range, then the cell c is *acceptable*. The point list is smoothed and compressed by growing segments that are within a prescribed constant value of the original polyline.

An example contour extraction is shown in the left part of Figure 61 from the input MRI (Magnetic Resonance Imaging)image slice on the right.

Of course, more sophisticated iso-contour extraction algorithms may also be used, see for e.g. [114]

For arbitrarily scattered data we use the alpha shape generation algorithm of [101] to extract an appropriate contour of the given scattered data points. Examples of this algorithm are shown in Figure 60 for an initially unordered set of point data sampled from a human head profile.

5.3 Computation of Junction Points

The next step is to compute the junction points around the contour. We use a curvature adaptive scheme for the placement of the cubic curve segments that is given in [10]. The points on a unit circle are in one-to-one correspondence with the normal directions, (or alternatively the slopes) of the line segments which make up the polygonal contour. Consider any regular k polygonal subdivision of a circle and number the k discrete normal directions $\mathbf{n}$ of the polygon boundary with integers from 1 to k. See also Figure 63.

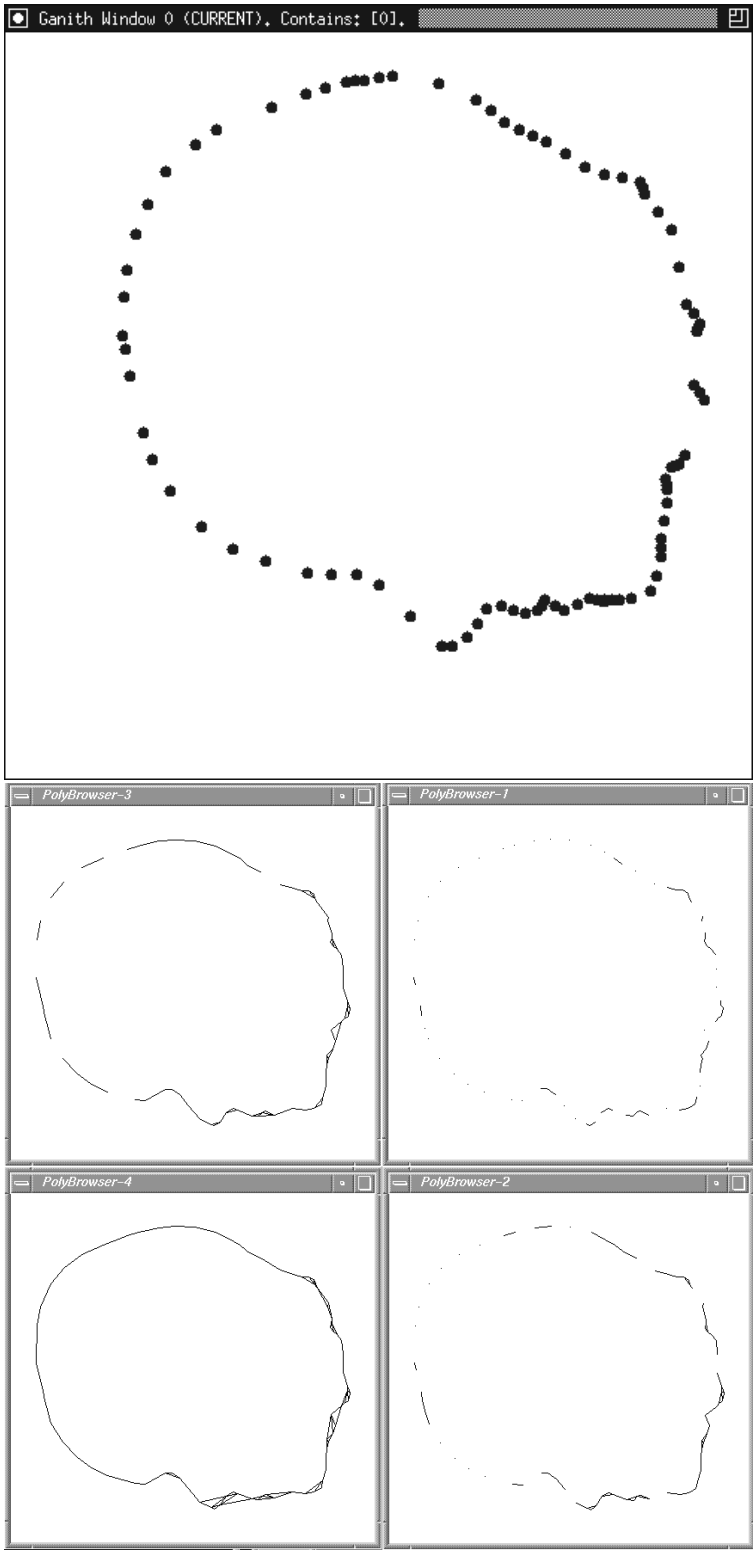


Figure 62: Extracting a contour from scattered data points

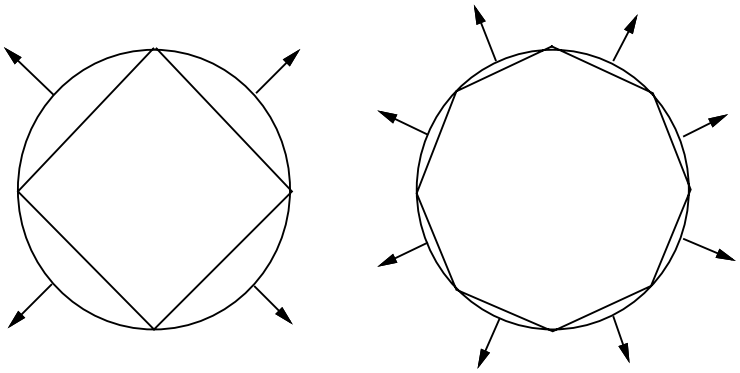


Figure 63: Regular subdivision of the space of normals on a planar contour

Now number each line segment of the contour boundary with the integer i if it has the largest dot product of its normal with the i^{th} normal of the regular polygon. Under this mapping the k discrete normal directions on the circle partitions the polygonal contour on a data slice into groups where the members of a group consist of a connected sequence of line segments having the same assigned number. The endpoints of groups are the contour (junction) points whose two incident line segments have distinct assigned numbers. The line segments of each group are then replaced by a single cubic which C^2 or C^3-interpolates the group endpoints and the locally computed derivatives and simultaneously least-squares approximates the contour line segments that originally formed the group and lies within the junction points. See Figure 64 where junction points are computed for different polygonal subdivisions k of the unit circle.

The C^2 and C^3 interpolation of the pair of endpoints and locally computed derivatives, by cubic A-splines are explained in the next subsections. If the least-squares approximation yields a poor error bound then additional cubics can be used to achieve a better bound. This operation is of course local to the the group and can be achieved by selectively refining the regular polygon edge corresponding to that group, replacing that edge by two or more edges inscribed in the circular arc subtended by that edge. The newly created normal directions are now mapped to the polygonal contour splitting the group into sub-groups. Each sub-group can now be replaced by a cubic, improving the approximation.

5.4 Generating Derivatives at Junction Points

There are various forms of divided-difference methods that extract geometric information around a junction point, from a given list of points [93]. Consider a sequence of points $\cdots, p_{i-2}, p_{i-1}, p_i, p_{i+1}, p_{i+2}, \cdots$ around the junction point p_i and an imaginary power series $C(t)$ from which, we assume, the digitized points near p_i arise, and whose parameter value is $t = 0$ for p_i. Then, the tangent vector of $C(t)$ at $t = 0$ can be approximated by the approximation:

$$C'(0) \approx \frac{\sigma_i}{\mathsf{dist}(p_i, p_{i+1})}(p_{i+1} - p_i) + \frac{1-\sigma_i}{\mathsf{dist}(p_{i-1}, p_i)}(p_i - p_{i-1})$$

where $\sigma_i = \frac{\mathsf{dist}(p_{i-1},p_i)}{\mathsf{dist}(p_i,p_{i+1})+\mathsf{dist}(p_{i-1},p_i)}$ and $\mathsf{dist}(*,*)$ is the distance between two points.

Repeatedly applying this approximation formula, yields compact formulas [28] for higher order divided-differences:

$$\Delta^j p_i = \begin{cases} p_i & \text{if } j = 0 \\ \frac{1}{j}(\frac{\sigma_i}{\mathsf{dist}(p_i,p_{i+1})}(p_{i+1} - p_i) & \\ +\frac{1-\sigma_i}{\mathsf{dist}(p_{i-1},p_i)}(p_i - p_{i-1})) & \text{if } j > 0 \end{cases}$$

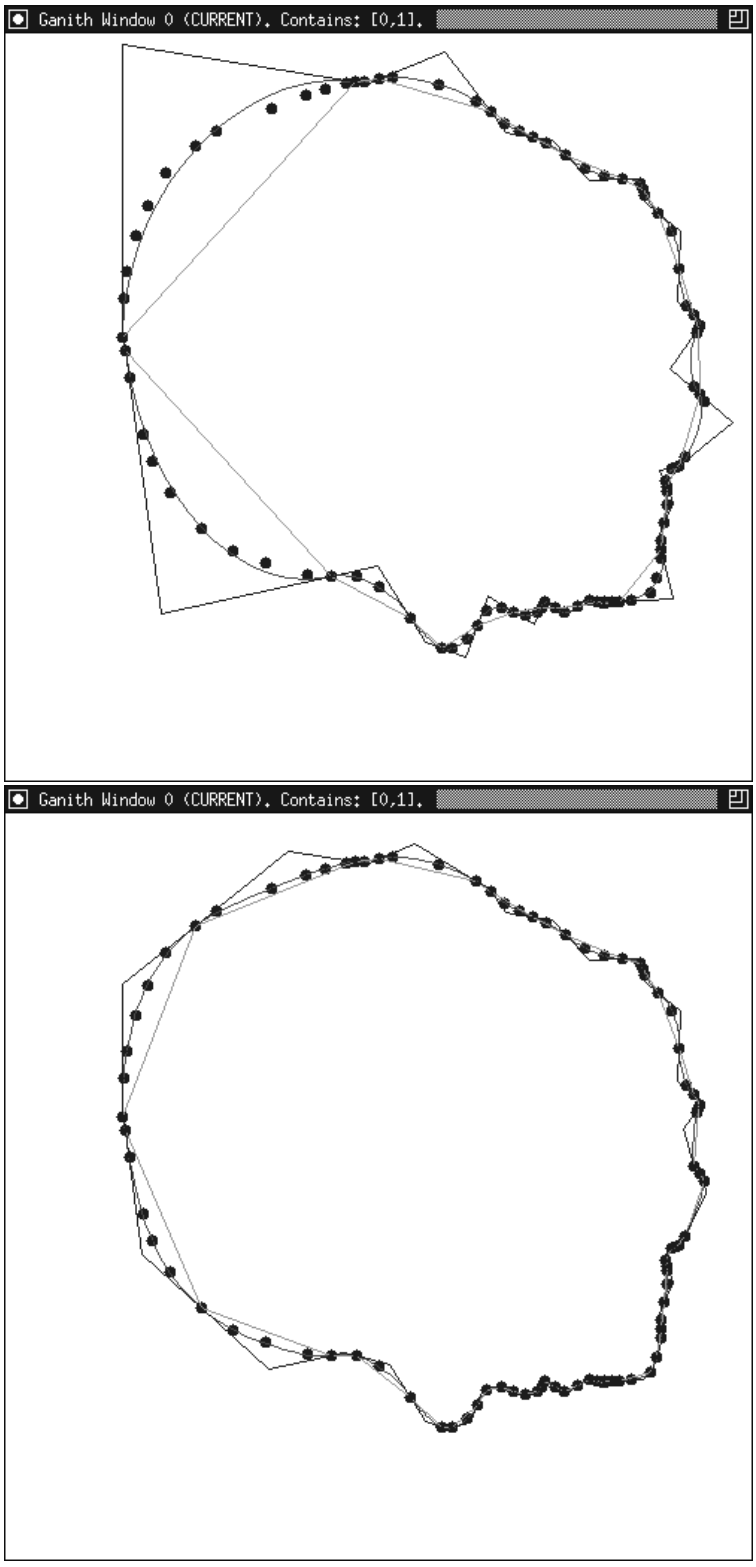


Figure 64: Junction Points and Cubic A-spline Fits

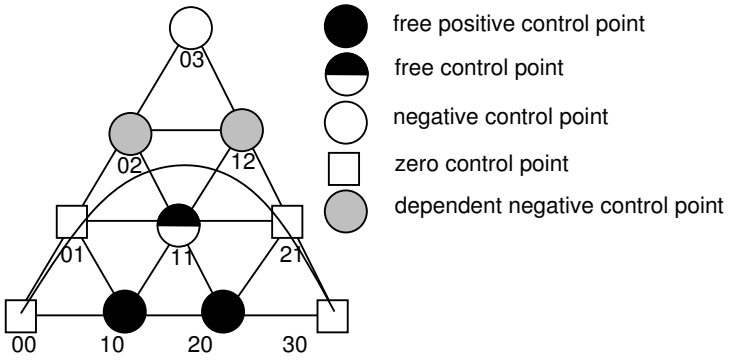


Figure 65: Bernstein Bezier Coefficients of a C^2 Cubic Algebraic Curve

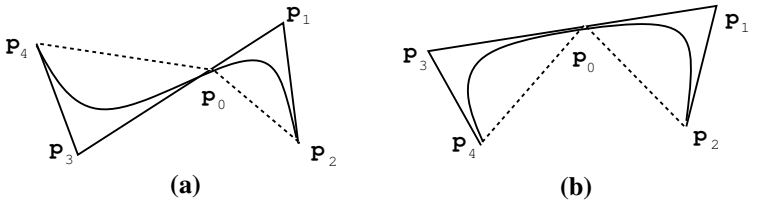


Figure 66: A C^0 polygon and a C^1 polygon

Using this divide-difference operator, a truncated power series is represented as $C_i(t) = \Delta^0 p_i + \Delta^1 p_i t + \Delta^2 p_i t^2 + \cdots + \Delta^k p_i t^k$. The higher order derivatives at the junction points are then approximated by $C^{'}(0)$, $C^{''}(0)$, $C^{'''}(0)$, etc. From these derivatives, we can easily compute the local derivatives $\mathcal{X}^{(k)}_{(p_i,p_{i+1})}$ and $\mathcal{X}^{(k)}_{(p_i,p_{i-1})}$ defined in §2.

5.5 Exact and Least-Squares Fitting with C^2 and C^3 cubic A-splines

Consider a C^1 cubic algebraic curve segment defined over a triangle $p_0p_1p_2$

$$\begin{aligned} F(\alpha_0,\alpha_1) &= -\alpha_1^3 + b_{10}\alpha_0(1-\alpha_0-\alpha_1)^2 + b_{20}\alpha_0^2(1-\alpha_0-\alpha_1) \\ &\quad + b_{02}\alpha_1^2(1-\alpha_0-\alpha_1) + b_{12}\alpha_0\alpha_1^2 + b_{11}\alpha_0\alpha_1(1-\alpha_0-\alpha_1) \end{aligned} \tag{220}$$

with

$$b_{10} > 0, \quad b_{20} > 0, \quad b_{02} \le 0, \quad b_{12} \le 0 \tag{221}$$

By differentiating $F(\alpha_0,\alpha_1) = 0$ about α_1 we have the following formulas for $\alpha^{(k)}_{0i} = \alpha^{(k)}_{0i}(0)$:

$$\alpha^{(1)}_{00} = 0 \qquad \alpha^{(1)}_{02} = -1$$

$$\frac{\alpha^{(2)}_{00}}{2!} = -\frac{b_{02}}{b_{10}}, \qquad \frac{\alpha^{(2)}_{02}}{2!} = \frac{b_{12}}{b_{20}}, \tag{222}$$

$$\frac{\alpha^{(3)}_{00}}{3!} = \frac{b_{10} - b_{10}b_{02} + b_{11}b_{02}}{b_{10}^2}, \qquad \frac{\alpha^{(3)}_{02}}{3!} = \frac{-b_{20} + b_{20}b_{12} - b_{11}b_{12}}{b_{20}^2} \tag{223}$$

From these formulas and the sign requirement (221), we can derive the following algorithm for constructing C^2 continuous A-spline curve(see [37] for detail):

Algorithm 2. *Let $\{\widehat{q_iv_iq_{i+1}}\}_{i=0}^m$ form a C^1 polygonal contour of the junction points (see Figure 66).*

1. *Specify the second derivative values such that* $\mathcal{X}^{(2)}_{(q_i,v_i)} = 0$ *if* q_i *is of a Case(a)-join, or* $\mathcal{X}^{(2)}_{(q_i,v_i)}\Delta(q_i, v_i, q_{i+1}) \geq 0$ *if* q_i *is of a Case(b)-join for* $i = 1, 2, \ldots, m$, *and* $\mathcal{X}^{(2)}_{(q_0,v_0)}\Delta(q_0, v_0, q_{0+1}) \geq 0$, $\mathcal{X}^{(2)}_{(q_{m+1},v_m)}\ \Delta(q_m, v_m, q_{m+1}) \leq 0$.

2. *Compute* b_{02} *and* b_{12} *by (222) for each triangle. Determine the three remaining degrees of freedom* $b_{10} > 0$, $b_{20} > 0$ *and* b_{11} *by least-squares approximation of the given data within the triangle, or via a default choice if there are not enough data points within the triangle.*

For achieving C^3 continuity, we specify the second and third local derivatives at the junction points. These derivatives need to satisfy some of the following conditions in order to have the coefficients of the BB-form have the required signs(see (221)

$$\mathcal{X}^{(3)}_{(p_0,p_1)}\Delta(p_0, p_1, p_2) > 0 \tag{224}$$

$$\left(\frac{\alpha^{(3)}_{00}}{6} - \frac{\alpha^{(2)}_{00}}{2} \right) b_{10} = 1 - b_{11}\frac{\alpha^{(2)}_{00}}{2}, \quad \left(\frac{\alpha^{(3)}_{02}}{6} - \frac{\alpha^{(2)}_{02}}{2} \right) b_{20} = -1 - b_{11}\frac{\alpha^{(2)}_{02}}{2} \tag{225}$$

On the triangle $p_0p_1p_2$ and at point p_0 we have the inequalities.

$$\pm \begin{cases} 1 - \frac{\|p_1-p_0\|^3 b_{11} D_2}{\Delta(p_0,p_1,p_2)} > 0 \\ \frac{\|p_1-p_0\|^4 D_3}{\Delta(p_0,p_1,p_2)} + 2\left(\frac{D_2}{\Delta(p_0,p_1,p_2)} \right)^2 \|p_1 - p_0\|^4 \langle p_1 - p_0, p_2 - p_0\rangle - \frac{\|p_1-p_0\|^3 D_2}{\Delta(p_0,p_1,p_2)} > 0 \end{cases} \tag{226}$$

where $D_k = \frac{\mathcal{X}^{(k)}_{(p_0,p_1)}}{k!}$, and at p_2

$$\pm \begin{cases} -1 - \frac{\|p_1-p_2\|^3 b_{11} D_2}{\Delta(p_0,p_1,p_2)} > 0 \\ \frac{\|p_1-p_2\|^4 D_3}{\Delta(p_0,p_1,p_2)} + 2\left(\frac{D_2}{\Delta(p_0,p_1,p_2)} \right)^2 \|p_1 - p_2\|^4 \langle p_1 - p_2, p_2 - p_0\rangle - \frac{\|p_1-p_2\|^3 D_2}{\Delta(p_0,p_1,p_2)} > 0 \end{cases} \tag{227}$$

where $D_k = \frac{\mathcal{X}^{(k)}_{(p_2,p_1)}}{k!}$. For $\widehat{p_4p_3p_0}$ at p_0, we have

$$\pm \begin{cases} -1 + \frac{\|p_3-p_0\|^3 b_{11} D_2}{\Delta(p_4,p_3,p_0)} > 0 \\ \frac{\|p_3-p_0\|^4 D_3}{\Delta(p_4,p_3,p_0)} + 2\left(\frac{D_2}{\Delta(p_4,p_3,p_0)} \right)^2 \|p_3 - p_0\|^4 \langle p_3 - p_0, p_0 - p_4\rangle + \frac{\|p_3-p_0\|^3 D_2}{\Delta(p_4,p_3,p_0)} > 0 \end{cases} \tag{228}$$

Algorithm 3. *Let* $\{\widehat{q_iv_iq_{i+1}}\}_{i=0}^m$ *form a* C^1 *polygon of the junction points and assume* $\langle v_i - q_i, q_{i+1} - q_i\rangle > 0$, $\langle v_{i-1} - q_i, q_{i-1} - q_i\rangle > 0$ *if* $1 \leq i \leq m$.

1. *At each junction point* q_i $(i = 0, 1, \ldots, m+1)$, *specify the second and third order derivatives as follows(regard* q_i, v_i, q_{i+1} *as* p_0, p_1, p_2 *for* $i \geq 0$ *and* q_{i-1}, v_{i-1}, q_i *as* p_4, p_3, p_0 *for* $i \leq m+1$*):*
   (a) $\mathcal{X}^{(2)}_{(q_i,v_i)} = 0$, $\mathcal{X}^{(3)}_{(q_i,v_i)}$ *satisfy (224) if* q_i *is of a Case(a)-join and* $1 \leq i \leq m$.
   (b) $\mathcal{X}^{(2)}_{(q_i,v_i)}\Delta(q_i, v_i, q_{i+1}) > 0$, $\mathcal{X}^{(2)}_{(q_i,v_i)}$ *and* $\mathcal{X}^{(3)}_{(q_i,v_i)}$ *satisfy both +(226) and −(228) if* q_i *is of a Case(b)-join and* $1 \leq i \leq m$.
   (c) *For* $i = 0$ *and* $i = m+1$, $\mathcal{X}^{(2)}_{(q_0,v_0)}\Delta(q_0, v_0, q_{0+1}) \geq 0$, $\mathcal{X}^{(2)}_{(q_{m+1},v_m)}\Delta(q_m, v_m, q_{m+1}) \leq 0$, *and* $\mathcal{X}^{(3)}_{(q_0,v_0)}$ *and* $\mathcal{X}^{(3)}_{(q_{m+1},v_m)}$ *satisfy +(226) and −(227), respectively.*

2. *For each triangle, compute* b_{10} *and* b_{20} *using (225); compute* b_{02} *and* b_{12} *using (222). The remaining single degree of freedom* $b_{11} \leq 0$ *is chosen by least-squares approximation of the given data points interior to the triangle or via a default choice if there are not enough data points within the triangle.*

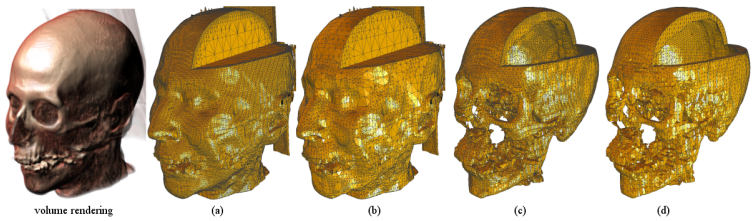


Figure 67: Adaptive tetrahedral meshes extracted from UNC Head (CT, 129×129×129). Isovalues (α_{in}, α_{out}) = (1000, 50) in (a)(b), and (1000, 120) in (c)(d); error tolerance ε_{in} = 0.0001, ε_{out} = (a): 0.0001, (b): 2.856, (c): 2.627, (d): 9.999. *in* and *out* represent inner and outer isosurface respectively. The number of elements and the extraction time are listed in Figure 70.

5.6 Surplus Degrees of Freedom

For the above C^2 and C^3 data fitting algorithms, after satisfying the derivatives at the junction points, there still exists three and one remaining degrees of freedom, respectively. These degrees of freedom can be used to locally control the shape of the curve in each triangle. For example, if b_{10} and b_{20} are given in Algorithm 2, then b_{11} can be chosen so that the curve in the triangle can be as high as the top vertex (when b_{11} tends to ∞) or as low as the bottom edge (when b_{11} tends to $-\infty$). In Algorithm 3, the curve can vary between the two limit curves(one corresponding to $b_{11} = 0$, other corresponding to $b_{11} = -\infty$). As we early mentioned in both Algorithm 2 and Algorithm 3, we currently use these degrees of freedom to least-squares approximate points in the triangle and based on the sign requirements (221). However, there exist the possibility that there are not enough data points within a triangle to determine these coefficients. Default choices of values for the undetermined coefficients are used in this case. One method used to determine these default values of the coefficients is to locally approximate a quadratic or linear curve with the triangle, which tends to avoid sharp changes in the geometry of the spline curve. The linear or quadratic approximation is easily achieved by using degree elevation formulas (see [108]). The second approach is to minimize the energy over the triangle on which the curve is defined. That is,

$$min = \iint_{\Delta} \left(\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right) dx\, dy$$

where $f(x, y) = 0$ is the curve in the triangle Δ in xy-system.

6 Contour-Based Meshing

6.1 Adaptive and Quality 3D Meshing from Imaging Data

We present an algorithm to extract adaptive and quality 3D meshes directly from volumetric imaging data - primarily Computed Tomography (CT) and Magnetic Resonance Imaging (MRI). The extracted tetrahedral and hexahedral meshes are extensively used in finite element simulations. Our comprehensive approach combines bilateral and anisotropic (feature specific) diffusion filtering, with contour spectrum based, isosurface and interval volume selection. Next, a top-down octree subdivision coupled with the dual contouring method is used to rapidly extract adaptive 3D finite element meshes from volumetric imaging data. The main contributions are extending the dual contouring method to crack free interval volume tetrahedralization and hexahedralization with feature sensitive adaptation. Compared to other tetrahedral extraction methods from imaging data, our method generates better quality adaptive 3D meshes without hanging nodes. Our method has the properties of crack prevention and feature sensitivity.

6.1.1 Overview

The development of finite element simulations in medicine, molecular biology, engineering and geosciences has increased the need for high quality finite element meshes. Although there has been tremendous progresses in the area of surface reconstruction and 3D geometric modeling, it still remains a challenging process to generate 3D geometric models directly from imaging data, such as CT, MRI and signed distance function (SDF) data. The image data can be represented as $V = \{F(i,j,k)|i,j,k$ are indices of x,y,z coordinates in a rectilinear grid$\}$. V is the volume containing function values $F(i,j,k)$ at the indices i,j,k.

For accurate and efficient Finite Element Method (FEM) calculations, it is important to have accurate and high quality models, minimize the number of elements and preserve features. We present a comprehensive approach to extract tetrahedral and hexahedral meshes directly from imaging data.

$$S_F(c) = \{(x,y,z) : F(x,y,z) = c\} \quad (229)$$

$$I_F(\alpha_1,\alpha_2) = \{(x,y,z) : \alpha_1 < F(x,y,z) < \alpha_2\} \quad (230)$$

Given volumetric imaging data and two isovalues α_1, α_2, the main steps to extract tetrahedral/hexahedral meshes from the interval volume, I_F, between the two isosurfaces (Equation (1)) are as follows:

1. Volumetric Denoising
2. Contour spectrum based interval volume selection.
3. Adaptive 3D meshing with feature sensitivity.
4. Quality improvement

As a preprocessing step, the bilateral prefiltering coupled with anisotropic diffusion method [36] is applied to volumetric data. Accurate gradient estimation can also be obtained. The Contour Spectrum [31] provides quantitative metrics of a volume to help us select two suitable isovalues for the interval volume.

We extend the idea of dual contouring to interval volume tetrahedralization and hexahedralization from volumetric Hermite data (position and normal information). Dual Contouring [136] analyzes those edges that have endpoints which lie on different sides of the isosurface, called *sign change edge*. Each edge is shared by four (uniform case) or three (adaptive case) cells, and one minimizer is calculated for each of them by minimizing a predefined Quadratic Error Function (QEF) [116].

$$QEF[x] = \sum_i (n_i \cdot (x - p_i))^2 \quad (231)$$

where p_i, n_i represent the position and unit normal vectors of the intersection point respectively. For each sign change edge, a quad or a triangle is constructed by connecting the minimizers. These quads and triangles provide an approximation of the isosurface.

Each sign change edge belongs to a boundary cell. In our tetrahedral mesh extracting process, we give a systematic way to tetrahedralize the volume in the boundary cell. For uniform grids, it is easy to deal with the interior cells. We only need to decompose each cell into five tetrahedra in a certain way. For the adaptive case, it is more complicated. In order to avoid introducing hanging nodes, which are strictly prohibited in finite element meshes, we design an algorithm to tetrahedralize the interior cell depending on the resolution levels of all its neighbors. As a byproduct, the uniform hexahedral mesh extraction algorithm is simpler. We analyze each interior vertex (a grid point inside the interval volume) which is shared by eight cells.

In Dual Contouring, QEF is used for isosurface extraction and sharp features can be preserved. But how to identify features such as sharp edges and facial features (like nose, eyes, mouth and ears)?

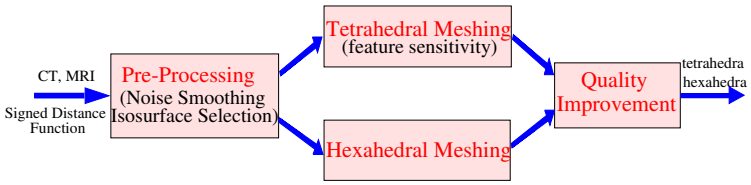


Figure 68: Overview for 3D mesh extraction

We adopt a different error function to identify those features sensitively. The edge contraction method is used to improve the mesh quality.

In the last twenty years, the techniques of CT and MRI have developed rapidly. Computer visualization, and engineering calculation (FEM) require certain kinds of mesh extracted from these scanned volume data.

Multiresolution Isosurface Extraction The predominant algorithm for isosurface extraction from volume data is Marching Cubes (MC) [160], which computes a local triangulation within each cube to approximate the isosurface by using a case table of edge intersections. Furthermore, the asymptotic decider was proposed to avoid ambiguities existing in MC [173] [157]. For efficient isosurface extraction, [44] starts from seed cells and traces the rest of the isosurface components by contour propagation.

When the adjacent cubes have different resolution levels, the cracking problem will happen. To keep the face compatibility, the gravity center of the coarser triangle is inserted, and a fan of triangles are used to approximate the isosurface [228]. A surface wave-front propagation technique [231] is used to generate multiresolution meshes with good aspect ratio. By combining SurfaceNets [118] and the extended Marching Cubes algorithm [141], octree based Dual Contouring [136] can generate adaptive multiresolution isosurfaces with good aspect ratio and preserve sharp features.

Quality and Feature Preserving Isosurface MC can not detect sharp features of the extracted isosurface, and severe alias artifacts appear. The enhanced distance field representation and the extended MC algorithm [141] were introduced to extract feature sensitive isosurfaces from volume data. The grid snapping method reduces the number of elements in an approximated isocontour and also improves the aspect ratio of the elements [166]. [53] studied how to generate triangular meshes with bounded aspect ratios from a planar point set. [164] proposed an algorithm to triangulate a d-dimensional region with a bounded aspect ratio.

Quality Meshing MC is extended to extract tetrahedral meshes between two isosurfaces directly from volume data [115]. A different and systematic algorithm, Marching Tetrahedra (MT), was proposed for interval volume tetrahedralization [174]. A multiresolution 3D meshes [245] can be generated by combining recursive subdivision and edge-bisection methods. Poor quality tetrahedra called slivers are notoriously common in 3D Delaunay triangulations. Sliver exudation [71] is used to eliminate those slivers. A deterministic algorithm [70] was presented for generating a weighted Delaunay mesh with no poor quality tetrahedra including slivers. Shewchuk [216] provides some valuable conclusions on quality measures for FEM.

Our comprehensive 3D meshing method is displayed in Figure 68. We first use the anisotropic diffusion method coupled with bilateral prefiltering to remove noise from imaging data. Depending on the application, suitable isosurfaces are selected for the interval volume by using the contour spectrum and the contour tree. We then begin to extract 3D meshes from the interval volume, and a feature sensitive error function is adopted to reduce the number of elements while preserving features. Finally, the edge contraction method is used to improve the mesh quality.

Since noise influences the accuracy of the extracted meshes, it is important to remove it before the mesh extracting process. We use the anisotropic diffusion method [36] to smooth noise. In

order to obtain more accurate computation of curvature and gradient for the anisotropic diffusion tensor, the bilateral prefiltering combining the domain and range filtering together is chosen instead of Gaussian filtering because it can preserve features such as edges and corners.

Mesh extraction from imaging data requires selecting suitable boundary isosurfaces. We use a user interface called Contour Spectrum [31], to find isosurfaces of interest. The Contour Spectrum computes quantitative properties such as surface area, volume, and gradient integral of contours, and helps to choose suitable isosurfaces by showing the related spectrum in a 2D plane. A contour tree [62] can be used to capture the topological information on each isosurface and help choose isosurfaces with desirable topology.

6.1.2 3D Mesh Extraction

In this section, our goal is to tetrahedralize or hexahedralize the interval volume between two isosurfaces by using an octree-based data structure. First, we discuss triangulation in 2D problems, then we extend it to 3D tetrahedralization. A hexahedral mesh generation algorithm is presented at the end of this section. Here are definitions used in the algorithm description.

Sign Change Edge A sign change edge is an edge whose one vertex lies inside the interval volume (we call it the interior vertex of this sign change edge), while the other vertex lies outside.

Interior Edge in Boundary Cell In a boundary cell, those edges with both vertices lying inside the interval volume are called interior edges.

Interior Cell Different from the boundary cell, all the eight vertices of an interior cell lie inside the interval volume.

Interior Face in Boundary Cell In the boundary cell, those faces with all four vertices lying inside the interval volume are called interior faces.

Hanging Node A hanging node is one that is attached to the corner of one triangle but does not attach to the corners of the adjacent triangles. For example, a T-Vertex.

Uniform Tetrahedral Extraction For isosurface extraction, we only need to analyze boundary cells – those cells that contain sign change edges. There are four neighbor cubes which share the same sign change edge. Dual Contouring generates one minimal vertex for each neighbor cube by minimizing the QEF, then connects them to generate a quad. By marching all sign change edges, the isosurface is obtained. For tetrahedral mesh extraction, cells inside the interval volume should also be set as leaves besides boundary cells.

6.1.3 Uniform 2D Triangulation

Figure 69(1) is a uniform triangulation example of the area interior to the isocontour in two dimensions. There are three different cases which need to be dealt with separately.

1. Sign change edge - find the QEF minimizers of two cells which share the edge. Then the two minimizers and the interior vertex of the edge construct a triangle (blue).

2. Interior edge in boundary cell - find the QEF minimizer of the boundary cell. Then the minimizer and this interior edge construct a triangle (yellow).

3. Interior cell - decompose each interior cell into two triangles (pink).

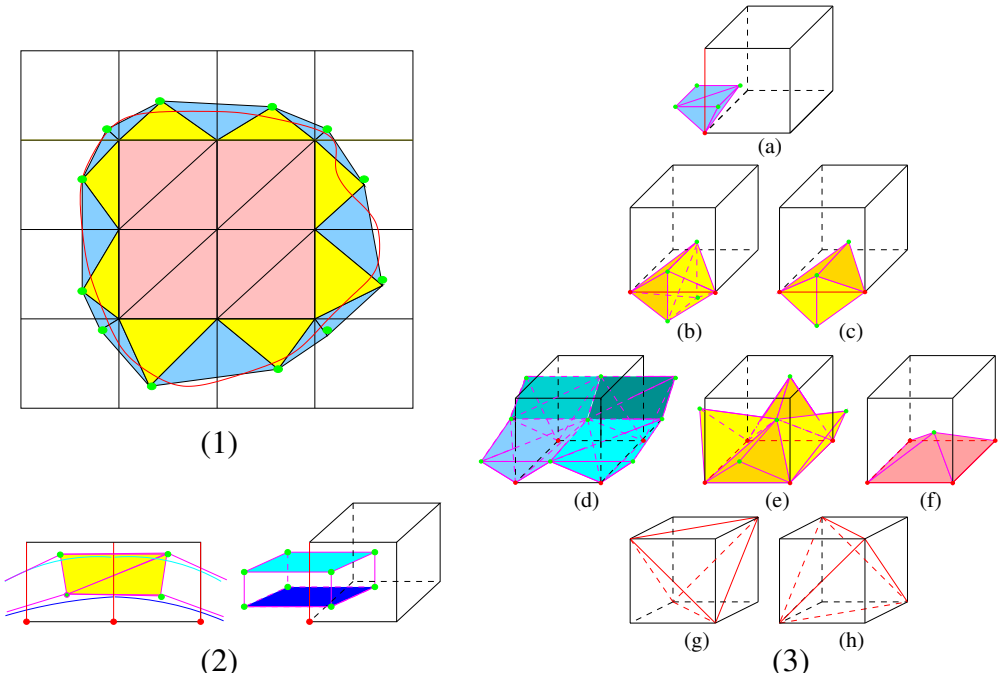


Figure 69: (1) - Uniform Triangulation, the red curve represents the isocontour. (2) - Sign Change Edge Passed Across by Two Isosurfaces, Left (2D) : the cyan and blue curves represent the two isocontours; Right (3D): the cyan and blue quads approximate the two isosurfaces. The red edges are sign change edges. (3) - Case Table of Uniform Tetrahedralization - the red vertex means it lies interior to the interval volume. In (1)(2)(3), green points represent minimizers.

6.1.4 Uniform 3D Tetrahedralization

Compared to 2D triangulation, three dimensional tetrahedral meshing is more complicated.

1. Sign change edge – decompose the quad into two triangles, then each triangle and the interior vertex of this edge construct a tetrahedron. In Figure 69(3a), the red line represents the sign change edge, and two blue tetrahedra are constructed.

2. Interior edge in boundary cell – find the QEF minimizers of the boundary cell and its boundary neighbor cells, then two adjacent minimizers and the interior edge construct a tetrahedron. In Figure 69(3b)(3c), the red cube edge represents the interior edge. (b) gives four minimizers to construct four edges, each of which construct a tetrahedron with the interior edge, so totally four tetrahedra are constructed. While (3c) assumes the cell below this boundary cell is interior to the interval volume, so there is no minimizer for it. Therefore we obtain three minimizers, and only two tetrahedra are constructed.

3. Interior face in boundary cell – find the QEF minimizer of the boundary cell, then the interior face and the minimizer construct a pyramid, which can be decomposed into two tetrahedra (Figure 69(3f)). Figure 69(3d)(3e)(3f) give a sequence how to generate tetrahedra when there is only one interior face in the boundary cell. (3d) analyzes four sign change edges, (3e) deals with four interior edges and (3f) fills the gap.

4. Interior cell – decompose the interior cube into five tetrahedra. There are two different decomposition ways (Figure 69(3g)(3h)). For two adjacent cells, we choose a different decomposition method to avoid the diagonal choosing conflict problem.

If two isosurfaces pass across the same sign change edge, we can split the cell into eight cubes in the octree data structure, then analyze each small cubes separately. In another approach, we need to analyze the sign change edge twice (Figure 69(2)) and fill gaps in the boundary cell. In 2D, two minimizers are obtained for the inner isosurface, and similarly two minimizers are calculated for the outer isosurface. They construct a quad, which can be decomposed into two triangles. For 3D, a hexahedron is built between the two surfaces for the sign change edge. The hexahedron can be split into five tetrahedra. Two different isosurfaces can not intersect with each other since one point can not have two isovalues. However, the two quads approximating the two isosurfaces may intersect because of bad gradient vectors. This can be solved by splitting the cell into eight cubes.

Adaptive Tetrahedral Extraction Uniform tetrahedralization usually gives an over-sampled mesh. Adaptive tetrahedral meshing is a good and effective way to reduce the number of elements while preserving the accuracy requirement.

First, we split the volume data by using the octree data structure to obtain denser cells along the boundary, and coarser cells inside the interval volume. The QEF value is calculated for each octree cell, and a much more efficient octree is built by comparing the QEF value with a given error tolerance ε and using the bottom-up algorithm. Leaves of the octree have different resolution levels. The next step is to analyze each leaf.

Each leaf cell may have neighbors at different levels. An edge in a leaf cell may be divided into several edges in its neighbor cells. Therefore it is important to decide which edge should be analyzed. The Dual Contouring method provides a good rule to follow – we always choose the minimal edges. Minimal edges are those edges of leaf cubes that do not properly contain an edge of a neighboring leaf.

Similar to uniform tetrahedral mesh extraction, we need to analyze the sign change edge, the interior edge and the interior face in the boundary cell, and the interior cell. When we analyze boundary cells, only minimal edges and minimal faces are analyzed. Compared to the uniform case, the only difference is in how to decompose the interior cell into tetrahedra without hanging nodes.

Data Set	Type	Resolution	Number of Tetrahedra (Extraction Time (unit : ms))			
			(a)	(b)	(c)	(d)
UNC Head (Skin)	CT	$129 \times 129 \times 129$	935124 (17406)	545269 (10468)	–	–
UNC Head (Skull)	CT	$129 \times 129 \times 129$	–	–	579834 (10203)	166271 (306
Poly	CT	$257 \times 257 \times 257$	276388 (5640)	63325 (1672)	14204 (672)	–
Knee	SDF	$65 \times 65 \times 65$	70768 (1360)	94586 (1782)	93330 (1750)	72366 (1406

Figure 70: Data Sets and Test Results. The CT data sets are re-sampled to fit into the octree representation (Figure 67, 75).

6.1.5 Adaptive 2D Triangulation

Figure 71(left) gives an example of how to triangulate the interior area of an isocontour. Similarly, we need to analyze the following problems:

1. Sign change edge – if the edge is minimal, deal with it as in the uniform case (blue triangles).
2. Interior edge in boundary cell – if the edge is minimal, analyze it as in the uniform case (yellow triangles).
3. Interior cell – Figure 71 (right) lists all the cases of how to decompose the interior cell into triangles. In order to obtain triangles with good aspect ratio, we restrict the neighboring level difference to be ≤ 2.

Compared to the uniform case, the triangulation of interior cells is more complicated (Figure 71). All neighbors of an interior cell need to be checked because the neighbor cells are used to decide if there are any middle points on the shared edge. Suppose the resolution level of this cell is κ, we group into five cases according to the number of edges whose level is greater than κ. The i^{th} group means there are number i edges whose level is greater than κ, where $i = 0, \ldots, 4$. For each subdivided edge, it may be subdivided more than once, or the neighbor cell may have a higher level than (κ+1). So we need to search all the middle points on this edge. A top-down or a bottom-up algorithm can be used here to find the resolution level of its neighbors, and find out all the middle points on the edge. If all the four edges have already been subdivided, then we can use the recursion method to march each of the four smaller cells with the same algorithm. In this way, hanging nodes are removed effectively.

6.1.6 Adaptive 3D Tetrahedralization

For 3D adaptive tetrahedralization, we use the same algorithm with the uniform case to analyze the boundary cell.

1. Sign change edge – if the edge is minimal, deal with it as in the uniform case.
2. Interior edge in the boundary cell – if the edge is minimal, deal with it as in the uniform case.
3. Interior face in boundary cell – identify all middle points on the four edges, and decompose the face into triangles as in the adaptive 2D case, then calculate the minimizer of this cell, each triangle and this minimizer construct a tetrahedron.
4. Interior cell – decompose each face of the cube into triangles, just as how to deal with the interior cell for the adaptive 2D triangulation (Figure 71), then insert a Steiner point at the cell center. Each triangle and the Steiner point construct a tetrahedron.

By using the above algorithm, we extract tetrahedral meshes from volumetric imaging data successfully. Figure 72 (right) gives one example.

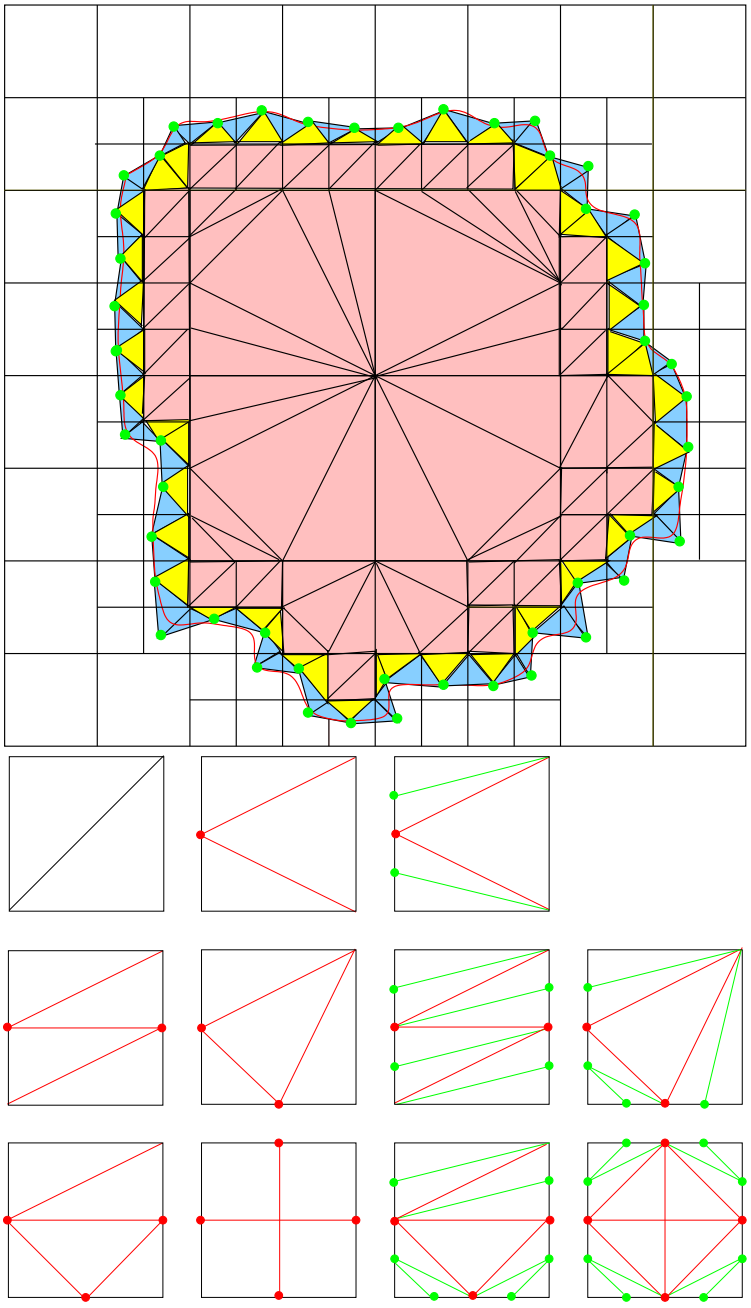


Figure 71: Top: Adaptive Triangulation. The red curve represents the isocontour, green points represent minimizers. Bottom: Case Table for Decomposing the Interior Cell into Triangles. Red points and red lines mean its neighbors have level (κ+1); green points and green lines mean its neighbors have a higher level than (κ+1).

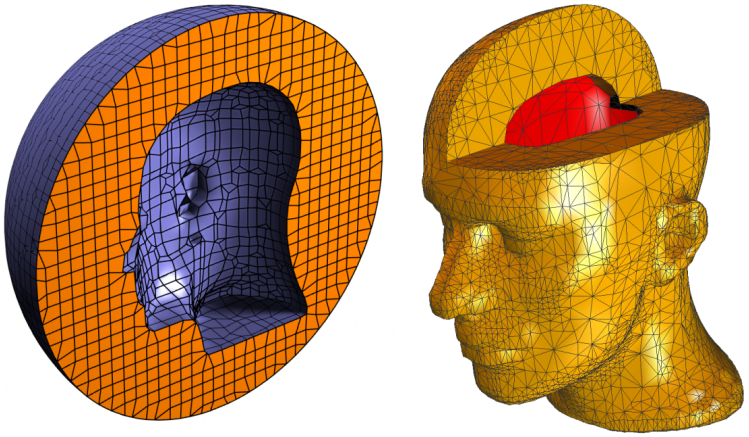


Figure 72: Left - hexahedralization of the volume between the human head and a sphere boundary; Right - an adaptive tetrahedral mesh.

Hexahedral Extraction Finite element calculations sometimes require hexahedral meshes instead of tetrahedral meshes. Each hexahedron has eight points. In the tetrahedralization process we deal with edges shared by at most four cells. This means that we can not get eight minimizers for each edge. But, each vertex is shared by eight cells, and we can calculate a minimizer for each of them. These eight minimizers can then be used to construct a hexahedron. Figure 72 (left) shows an example used to solve electromagnetic scattering problems.

6.1.7 Error Metric

For efficiency and accuracy during calculations, finite element applications require the number of elements to be as small as possible, while preserving necessary features. For a given precision requirement, the uniform mesh is always over-sampled with unnecessary small elements. Adaptive meshes are therefore preferable.

For the adaptive mesh, an error function and an error tolerance ε are required, which set the criteria to identify where we should select higher level (denser mesh) and where lower level (coarser mesh) should be chosen. In order to minimize the number of elements while preserving features, it is important to have a feature sensitive error function.

The Dual Contouring algorithm can preserve sharp features by using the QEF error function. Examples show that it is not sensitive to some features, for example, facial features, like the nose, eyes, mouth and ears of the human head model in Figure 73. Here we choose the Euclidean distance error (EDerror) function to identify features.

For level (i), the eight vertices' function values are given, and a trilinear function is defined in Equation (4), from which the function values of 12 edge middle points, 6 face middle points and 1 center point can be obtained. For level (i+1), the function values of all vertices are given. The error function is defined in Equation (5).

$$\begin{aligned} f^i(x,y,z) &= f_{000}(1-x)(1-y)(1-z) + f_{011}(1-x)yz \\ &+ f_{001}(1-x)(1-y)z + f_{101}x(1-y)z \\ &+ f_{010}(1-x)y(1-z) + f_{110}xy(1-z) \\ &+ f_{100}x(1-y)(1-z) + f_{111}xyz \end{aligned} \tag{232}$$

$$EDerror = \sum \frac{|f^{i+1} - f^i|}{|\nabla f^i|} \tag{233}$$

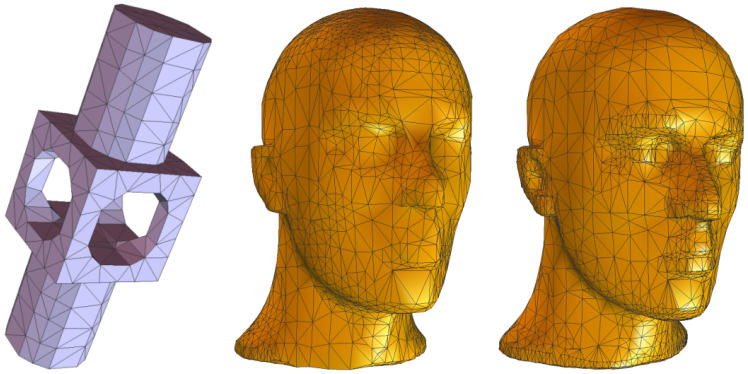


Figure 73: Sharp edge features (left); Facial features: QEF (middle, 2952 triangles) and EDerror (right, 2734 triangles). Better feature adaptation (eyes, nose, mouth and ears) is shown in the right picture.

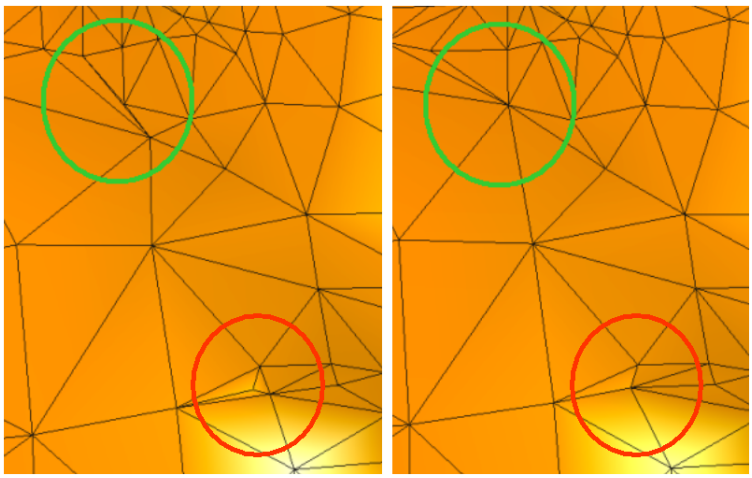


Figure 74: Quality Improvement - Left: no edge contraction, circles mark triangles with bad aspect ratio; Right: poor quality triangles disappear after iterative edge contraction.

The two error functions are compared in Figure 73. It is obvious that the EDerror can also preserve sharp edges, and is more sensitive to the areas where nose, eyes, mouth and ears are located on the human head model. That is because EDerror is the Euclean distance which measures error in a better way [191] than QEF. Furthermore, QEF only measures function value at a minimizer point for each cell, while EDerror compares function value at all vertices and edge/face middle points.

6.1.8 Quality Improvement

The above 3D mesh extraction algorithm can tetrahedralize the interval volume, and the extracted meshes have better quality than meshes from other methods such as MC and MT. However, it can not guarantee that all the elements have good quality. For example, sliver triangles or tetrahedra exist. In order to measure tetrahedra's quality, three quality parameters are borrowed from the ABAQUS document (a FEM software).

- Tetrahedral Quality Measure = volume of tetrahedron / volume of equilateral tetrahedron with same circum-sphere radius (> 0.02)

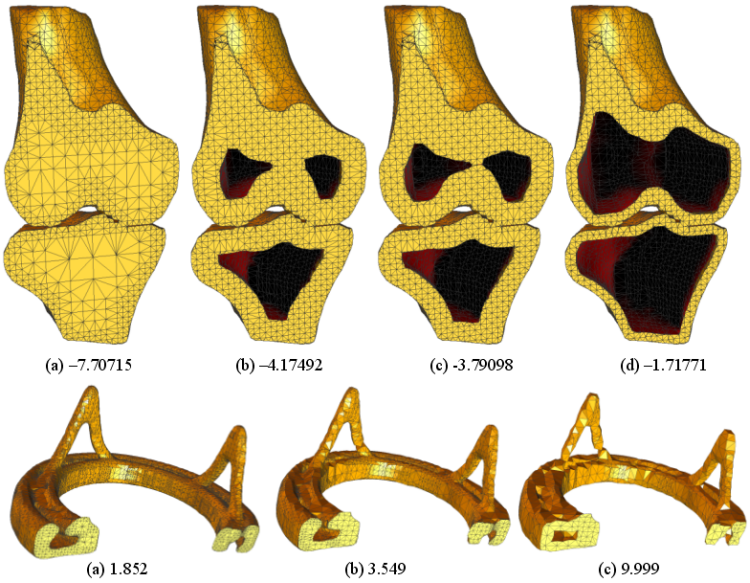


Figure 75: Upper row: Knee (SDF) – error tolerances $\varepsilon_{in} = \varepsilon_{out} = 0.0001$; isovalues α_{out} = -0.02838, α_{in} are listed below each picture. Bottom row: Heart Valve (Poly, CT) – isovalues (α_{in}, α_{out}) = (1000, 75); error tolerances $\varepsilon_{in} = 0.0001$, ε_{out} are listed below each picture.

- Min/Max Angles – with minimum angle $\alpha > 10°$ and maximum angle $\beta < 160°$.
- Right-hand-side principle

In the process of improving the mesh quality, edge contraction is a direct method to eliminate sliver tetrahedra. For each tetrahedron, first calculate the three quality parameters. If the tetrahedron's orientation is Left-hand-side, swap any two vertices' index number. If Tetrahedral Quality Measure ≤ 0.02 or Min Angle $\leq 10°$ or Max Angle $\geq 160°$, contract the shortest edge. Be careful not to merge vertices on surfaces to vertices inside the interval volume. Figure 6.2.6 shows an example.

6.1.9 Results

We developed an interactive program for 3D mesh extraction and rendering from a volume. In the program, the error tolerance and the isovalues can be changed interactively. The results were computed on a PC equipped with a Pentium III 800 MHz processor and 1 GB main memory.

Figure 70, 75 provide information about data sets and test results. As a preprocessing, we calculate min/max values for each octree cell to visit only cells contributing to mesh extraction and to compute QEF values only in those cells at run time. Extraction time in the table includes octree traversal, QEF computation and actual mesh extraction, given isovalues and error tolerance values for inner and outer surfaces as run time parameters. If we fix isovalues, and change error tolerance interactively, the computed QEF is reused and thus the whole extraction process is accelerated.

To extract 3D meshes from the surface data, we computed SDF from the surface and performed mesh extraction (Figure 75 (knee)). The results from CT data are shown in Figure 67 and 75 (heart valve). The number of elements is controlled by changing error tolerance. In Figure 75 (upper row), the sequence of images are generated by changing the isovalue of the inner isosurface. The topology of the inner isosurface can change arbitrarily.

We have presented an algorithm to extract adaptive and high quality 3D meshes directly from volumetric imaging data. By extending the dual contouring method [136], our method can generate

3D meshes with good properties such as no hanging nodes, sharp feature preservation and good aspect ratio. Using an error metric which is normalized by the function gradient, the resolution of the extracted mesh is adapted to the features sensitively. The resulting meshes are useful for efficient and accurate FEM calculations.

6.2 Adaptive and Quality Quadrilateral/Hexahedral Meshing from Volumetric Imaging Data

We describe an algorithm to extract adaptive and quality quadrilateral/hexahedral meshes directly from volumetric imaging data. First, a bottom-up surface topology preserving octree-based algorithm is applied to select a starting octree level. Then the dual contouring method is used to extract a preliminary uniform quad/hex mesh, which is decomposed into finer quads/hexes adaptively without introducing any hanging nodes. The positions of all boundary vertices are recalculated to approximate the boundary surface more accurately. Mesh adaptivity can be controlled by a feature sensitive error function, the regions that users are interested in, or finite element calculation results. Finally, the relaxation based technique is deployed to improve mesh quality. Several demonstration examples are provided from a wide variety of application domains. Some extracted meshes have been extensively used in finite element simulations.

6.2.1 Overview

Unstructured quadrilateral/hexahedral mesh generation attracts many researchers' interest because of its important applications in finite element simulations. However, it still remains a challenging and open problem to generate adaptive and quality quad/hex meshes directly from volumetric imaging data, such as Computed Tomography (CT), Magnetic Resonance Imaging (MRI) and Signed Distance Function (SDF) data.

The volumetric imaging data V is a sequence of sampled functional values on rectilinear grids, and can be written as $V = \{F(i,j,k)|i,j,k$ are indices in x,y,z coordinates in a rectilinear grid$\}$. An isosurface or a level set corresponding to the isovalue α is defined as $S_F(\alpha) = \{(x,y,z)|F(x,y,z) = \alpha\}$, and an interval volume between two isosurfaces $S_F(\alpha_1)$, $S_F(\alpha_2)$ is defined as $I_F(\alpha_1,\alpha_2) = \{(x,y,z)|\alpha_1 \leq F(x,y,z) \leq \alpha_2\}$. We present an approach to extract adaptive and quality quadrilateral meshes for the isosurface $S_F(\alpha)$, and hexahedral meshes for an interval volume $I_F(\alpha_1,\alpha_2)$ with isosurfaces as boundaries. In some finite element simulations, both interior and exterior hexahedral meshes are required, for example, the interior mesh of the volume inside the solvent accessibility surface of the biomolecule mouse acetylcholinesterase (mAChE) [217], and the exterior mesh between the solvent accessibility surface and an outer sphere. Since the most important part in the geometric structure of mAChE is the cavity, we need to generate finer mesh for it (Figure 76). Our approach can also generate adaptive and quality interior and exterior hexahedral meshes.

The main steps to extract adaptive and quality quadrilateral and hexahedral meshes from volumetric data are as follows:

1. The selection of a starting octree level for uniform mesh generation with correct topology.
2. Crack-free and adaptive quad/hex meshing without any hanging nodes.
3. Quality improvement.

In order to generate uniform quadrilateral and hexahedral meshes with correct topology, we select a suitable starting octree level using a bottom-up surface topology preserving octree-based algorithm. An approach provided in [136] is used to check whether a fine isosurface is topologically equivalent to a coarse one or not. Generally correct topology is guaranteed in the uniform mesh.

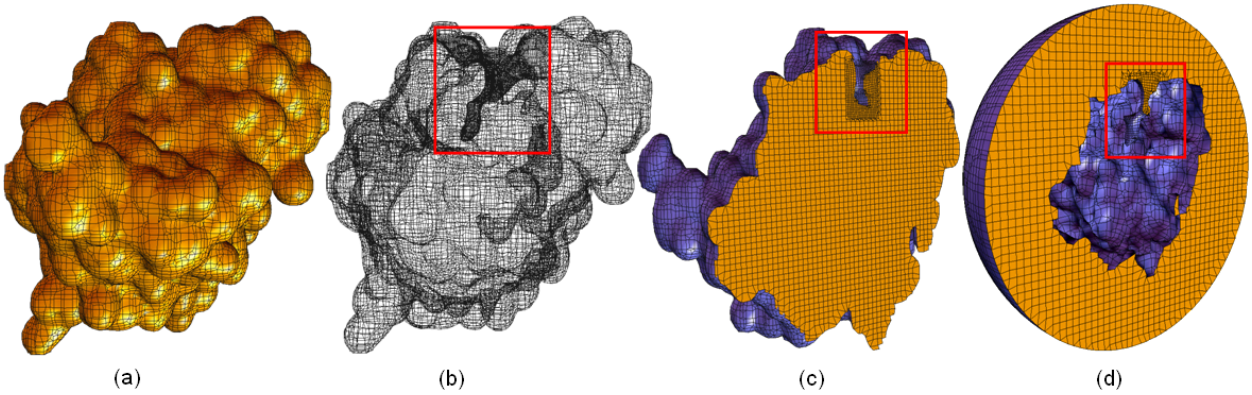


Figure 76: Adaptive quadrilateral and hexahedral meshes of a biomolecule mAChE. (a) - the quadrilateral mesh of the molecular surface; (b) - the wireframe of the adaptive quadrilateral mesh of the molecular surface; (c) - the adaptive hexahedral mesh of the interior volume; (d) - the adaptive hexahedral mesh of the exterior volume between the molecular surface and an outer sphere. Finer meshes are generated in the region of the cavity, while coarser meshes are kept in other areas. The cavity is shown in the red boxes.

The dual contouring method [136] proposes an algorithm to extract a uniform quadrilateral mesh for an isosurface by analyzing each *sign change edge*, whose two ending points lie in different sides of the isosurface. In the octree-based data structure, each sign change edge is shared by four octree leaves, and one minimizer point is obtained for each leaf cell by minimizing a predefined quadratic error function (QEF) [116]. The four minimizer points construct a quad, and the union of all the generated quads provides an approximation to this isosurface.

Starting from a uniform quadrilateral mesh, we use templates to refine each quad adaptively. The position of each vertex is recalculated by moving it toward the isosurface along its normal direction, which is represented by trilinear interpolation functions within octree leaf cells. The dual contouring isosurface extraction method has been extended to uniform hexahedral mesh generation [242] [241]. Predefined three dimensional templates are used to generate adaptive hexahedral meshes.

The mesh adaptivity can be controlled according to various requirements by a feature sensitive error function [242] [241], areas that users are interested in, or results from finite element calculations. Users can also design an error function to control the mesh adaptivity according to their specific requirements.

Generally, the extracted quadrilateral and hexahedral meshes can not be used for finite element calculations directly since some elements have poor quality. We choose corresponding metrics to measure the quality of quadrilateral and hexahedral meshes respectively, then the relaxation based technique is deployed to improve mesh quality. Some of the generated meshes have been used in finite element simulations.

Previous Work As a structured method, quad/hex mapped meshing [79] generates the most desirable meshes if opposite edges/faces of the domain to be meshed have equal numbers of divisions or the same surface mesh. However, it is always difficult to decompose an arbitrary geometric configuration into mapped meshable regions. In the CUBIT project [202] at Sandia National Labs, a lot of research has been done to automatically recognize features and decompose geometry into mapped meshable areas or volumes.

As reviewed in [183] [224], there are indirect and direct methods for unstructured quad/hex mesh generation. The indirect method is to generate triangular/tetrahedral meshes first, then

convert them into quads/hexes. The direct method is to generate quads/hexes directly without first going through triangular/tetrahedral meshing.

Unstructured Quad Mesh Generation: The indirect method is to convert triangles into quads by dividing a triangle into three quads, or combining adjacent pairs of triangles to quads [149].

There are three main categories for unstructured direct quad mesh generation, quad meshing by decomposition, advancing front quad meshing and isosurface extraction. The decomposition technique is to divide the domain into simpler regions which can be resolved by templates [8] [222]. The second category is to utilize a moving front method for direct placement of nodes and elements. Starting with an initial placement nodes on the boundary, Zhu et al. [246] formed individual elements by projecting edges towards the interior. As a part of CUBIT [202], the paving algorithm places elements starting from the boundary and works in [56]. Different from the decomposition and the advancing front techniques, the dual contouring method [136] extracts uniform quadrilateral meshes from volumetric imaging data to approximate isosurfaces which can be an arbitrary geometry.

Unstructured Hex Mesh Generation: Eppstein [104] started from a tetrahedral mesh to decompose each tetrahedron into four hexahedra. Although this method avoids many difficulties, it rapidly increases the number of elements and tends to introduce bad shape elements.

There are five distinct methods for unstructured direct all-hex mesh generation: grid-based, medial surface, plastering, whisker weaving and isosurface extraction. The grid-based approach generates a fitted 3D grid of hex elements on the interior of the volume, and hex elements are added at the boundaries to fill gaps [206] [208] [209]. The grid-based method is robust, but tends to generate poor quality elements at the boundaries. Medial surface methods are to decompose the volume to map meshable regions, and fill the volume with hex elements using templates [192] [193]. Plastering places elements on boundaries first and advances towards the center of the volume [60] [55]. Whisker weaving first constructs the spatial twist continuum (STC) or dual of the hex mesh, then the hex elements can be fitted into the volume using the STC as a guide [223]. Medial surface methods, plastering and whisker weaving have successfully generated hex meshes for some geometry, but have not been proven to be robust and reliable for an arbitrary geometric domain. Zhang et al. [242] [241] extended the dual contouring isosurface extraction method [136] to uniform hexahedral mesh generation. This method is robust and reliable for an arbitrary geometry, but adaptive meshes are preferable and mesh quality needs to be improved.

Quality Improvement: As the simplest and most straight forward method, Laplacian smoothing relocates the vertex position at the average of the nodes connecting to it [109]. There are a variety of other smoothing techniques based on a weighted average of the surrounding nodes and elements. The averaging method may invert or degrade the local quality, but it is simple to implement and in wide use. Instead of relocating vertices based on a heuristic algorithm, people utilized an optimization technique to improve the mesh quality. The optimization algorithm measures the quality of the surrounding elements to a node and attempts to optimize it. The algorithm is similar to a minimax technique used to solve circuit design problems [65]. The optimization-based smoothing yields better results but it is more expensive than Laplacian smoothing. Therefore, some people [61] [112] [113] recommended a combined Laplacian/optimization-based approach.

Staten et al. [219] [138] proposed algorithms to improve node valence for quadrilateral meshes. One special case of cleanup in hexahedral meshes for the whisker weaving algorithm is presented in [163]. Schneiders [207] proposed algorithms and a series of templates for quad/hex element decomposition. A subdivision algorithm was proposed for the refinement of hexahedral meshes [35].

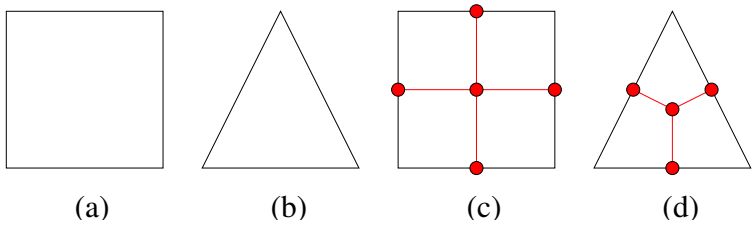


Figure 77: The templates to decompose a quad or a triangle into quads. Red points are newly inserted at the middle of edges or the element center. (a) - a quad before splitting; (b) - a triangle before splitting; (c) - a quad is split into four quads; (d) - a triangle is split into three quads.

6.2.2 Starting Octree Level Selection

There are three main steps in our adaptive and quality quadrilateral and hexahedral mesh extraction from volumetric data. First, we need to choose a suitable starting octree level to generate the uniform mesh with correct topology. Then pre-defined templates are used to refine the uniform mesh adaptively. The positions of all boundary vertices are recalculated, and the mesh adaptivity can be controlled by an error function designed in multiple ways. Finally, the relaxation based technique is used to improve mesh quality.

The bottom-up surface topology preserving octree-based algorithm is used to select a starting octree level. Suppose the volume data has the dimension of $(2^n + 1)^3$, so the deepest octree level is n. For an isosurface, we first compare the surface topology at Level n and Level $(n - 1)$. If the surface topology is equivalent, then we continue comparing the surface topology at Level $(n - 1)$ and Level $(n - 2)$ until we find the surface topology at two neighboring levels, e.g. Level i and Level $(i - 1)$ $(i = n, \ldots, 1)$, is different from each other. Then we will select i as the starting octree level.

We assign a sign to each grid point in the volumetric data. If the function value at a grid point is greater than the isovalue, then the sign is 1, otherwise it is 0. An approach is described in [136] to check whether a fine isocontour is topologically equivalent to a coarse one or not. The fine and coarse isocontour is topologically equivalent with each other if and only if the sign of the middle vertex of a coarse edge/face/cube is the same as the sign of at least one vertex of the edge/face/cube which contains the middle vertex. Generally we guarantee the correct topology for the boundary surfaces by choosing a suitable starting octree level, and correct topology will be preserved in the process of adaptive mesh refinement.

6.2.3 Quad Isosurface Extraction

Finite element calculations sometimes require quadrilateral meshes instead of triangular meshes. It is more challenging to generate quadrilateral meshes since not every polygon can be decomposed into quads directly. The uniform quadrilateral mesh extraction algorithm is simpler [136], but adaptive meshes are more preferable than uniform ones. There are two main problems in adaptive quadrilateral mesh extraction.

1. How to decompose a quad into finer quads.

2. How to calculate the positions of vertices.

Mesh Decomposition **Indirect Method:** In the dual contouring isosurface extraction method [136], an error function is defined to control where we should generate fine meshes, and where we should keep coarse ones. In the adaptive octree data structure, either a sign change edge is shared

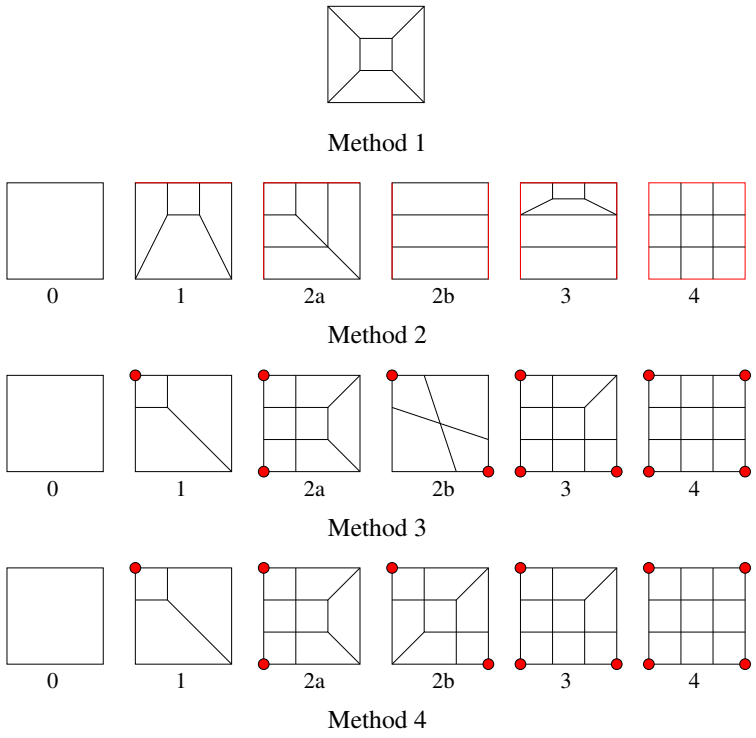


Figure 78: Three different methods to define templates for adaptive quadrilateral isosurface extraction. In Method 1, the quad needs to be refined; In Method 2 and 3, octree leaf cells generating red minimizer points need to be refined.

by three cells resulting in a triangle, or it is shared by four cells and a quad is generated. Therefore, the isosurface is represented by a union of quads and triangles. In order to obtain a all-quad mesh, the indirect method splits each quad into four quads and each triangle into three quads by inserting points at the middle of edges and at the center of the element as shown in Figure 77. The idea of the indirect method is simple and easy to implement, but the number of elements increases ($2 \sim 3$) times all over the original mesh.

Direct Method: At the selected starting octree level, the dual contouring isosurface extraction method [136] generates uniform quadrilateral meshes by analyzing each sign change edge which is shared by four leaf cells. Adaptive quadrilateral meshes can be obtained from the uniform mesh by using some templates. There are multiple ways to define templates for adaptive quadrilateral mesh construction, therefore criteria needs to be set to evaluate them in order to generate meshes with good quality. Here we define some requirements for templates:

1. All resulting elements are quads.
2. No hanging nodes exist.
3. The resulting mesh approximates the object surface accurately.
4. The resulting elements have good aspect ratio.
5. The resulting mesh introduces small number of new elements and vertices.

Figure 78 shows three methods to define templates for adaptive quadrilateral mesh generation starting from a uniform mesh with correct topology. In the uniform case, each sign change edge is shared by four cells and four minimizer points are obtained to construct a quad. In Method 1,

if the maximum error function value (for example, the feature sensitive error function defined in [242] [241]) of the four cells is greater than a threshold ϵ, then the four octree cells containing the sign change edge should be subdivided, and the quad generated from this edge should be refined. This method does not consider its neighboring information, each quad is refined independently. If a quad needs to be refined, then the resulting mesh has 5 elements and 4 newly inserted vertices. In Method 2 [209] and 3, various decomposition methods are chosen according to the cell which generates a quad node and also needs to be refined. Method 2 and 3 are only different in Case (2b), Method 2 generates less elements and extra vertices, but the quad quality is worse than Method 3.

We can use the above five template requirements to compare the three methods in Figure 78. It is obvious that all the three methods only generate quad elements, and no hanging nodes are introduced. Compared with Method 1, Methods 2 and 3 insert extra nodes on the quad edges as well as inside the quad, so they can approximate the surface more accurately. Comparing the worst aspect ratio of the resulting quad elements in Method 2 and 3, we can see that Method 3 generates quads with better quality. The number of elements and the number of newly inserted vertices for each template are listed in Figure 79. Method 3 is preferable by balancing the five criteria.

Method	Number of	0	1	2a	2b	3	4
2	elements	1	3	7	4	8	9
	vertices	0	3	8	4	10	12
3	elements	1	3	7	7	8	9
	vertices	0	3	8	8	10	12

Figure 79: The number of elements and the number of newly inserted vertices for templates in Methods 2 and 3 shown in Figure 78.

Vertex Position Calculation In the process of mesh refinement, new vertices are inserted according to the pre-defined templates. The next step is to update the positions of existing vertices and calculate the positions of newly inserted vertices.

In Figure 80, we assume that the leaf cell can be divided into four subcells in the finest resolution level, therefore the real isosurface (the red curve) is represented by a union of three trilinear interpolation functions within the subcells. For each existing minimizer point, first we find the octree leaf cell containing it in the current resolution level, then move it toward the isosurface within this leaf cell along its normal direction. The intersection point is more accurate to represent this boundary vertex than the minimizer point. If the calculated intersection point lies outside this cell unfortunately because of bad normal vectors, we will still keep the old position and normal vectors for it.

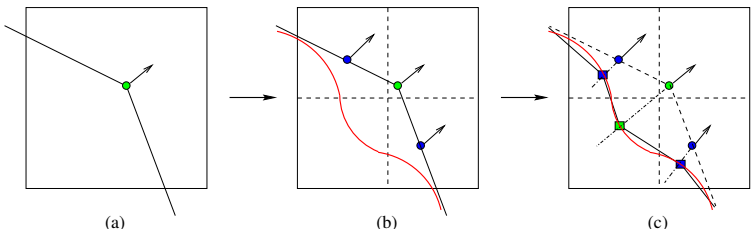


Figure 80: The calculation of vertex positions. The red curve is the real isocontour. The green circle point represents an existing minimizer point of this leaf cell, and blue circle points are two newly inserted vertices. The arrows are their normal vectors, and the green and blue box points are the resulting vertices.

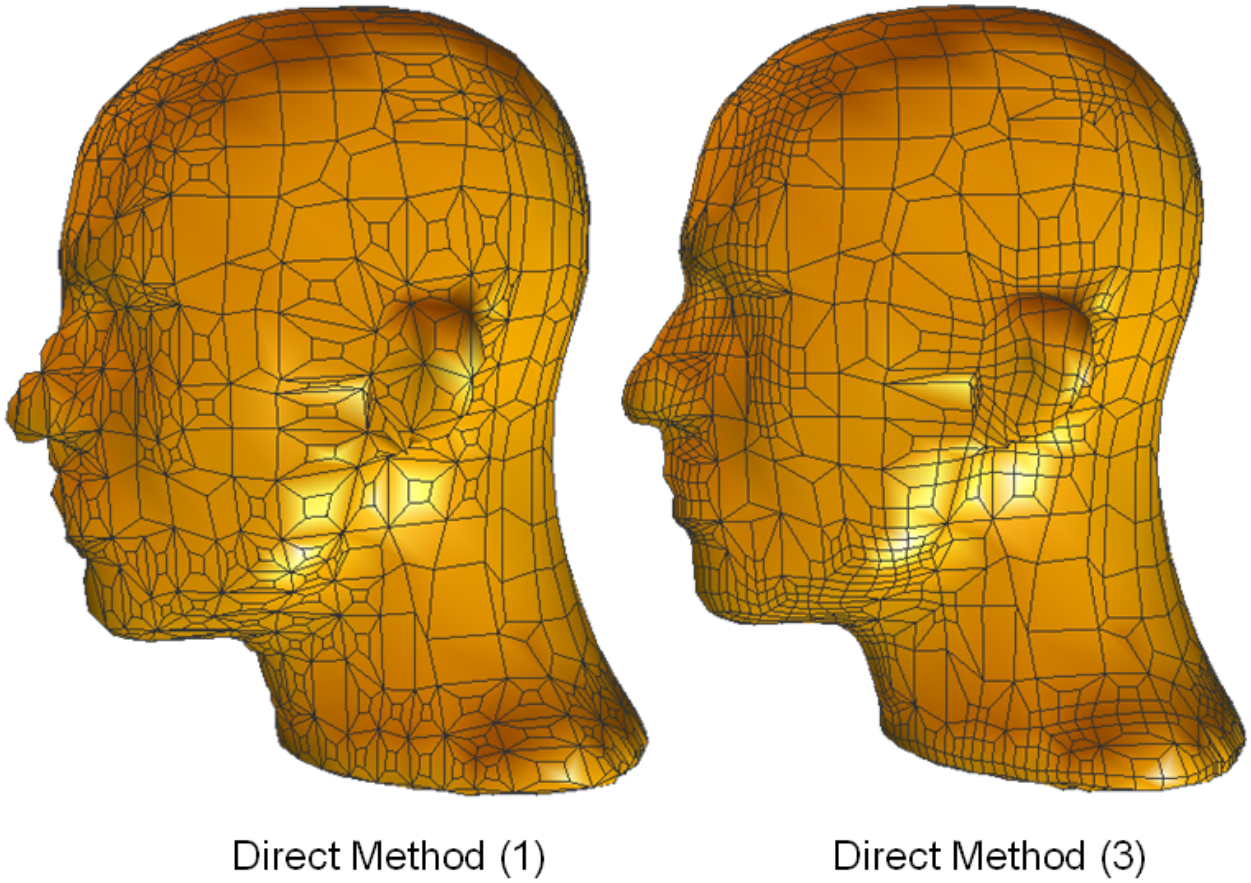


Figure 81: Adaptive quad meshes generated from two direct methods. A feature sensitive error function [242] [241] is chosen for mesh adaptivity, the isovalue $\alpha = 0$, the error tolerance $\varepsilon = 0.4$. Method 1 generates a bad nose, and Method 3 generates a better result.

For those newly inserted vertices, we first calculate their position and normal vectors by the linear interpolation of the four vertices of the original quad. Then we will move them toward the isosurfaces in the same way as we update the positions of existing vertices.

Figure 81 shows adaptive quadrilateral meshes of the human head generated from two direct methods, Method 1 and Method 3 shown in Figure 78. It is obvious that the original uniform mesh is refined adaptively, and the new vertex positions are closer to the isosurface. Method 1 generates a bad nose, and Method 3 approximates the isosurface more accurately than Method 1 because it introduces extra vertices on the refined edges of each original quad. The mesh adaptivity is controlled by a feature sensitive error function [242] [241], which is sensitive to facial features such as the nose, the eyes, the mouth and the ears.

6.2.4 Hexahedral Mesh Extraction

The dual contouring method [136] has been extended to uniform hexahedral mesh generation by analyzing each interior vertex (a grid point inside the interval volume) shared by eight different cells, which are either boundary cells or interior cells [242] [241]. One minimizer is calculated for each boundary cell, and the cell center is set as the minimizer for each interior cell. Those eight minimizers construct a hexahedron. In this section, we will focus on adaptive hexahedral mesh generation.

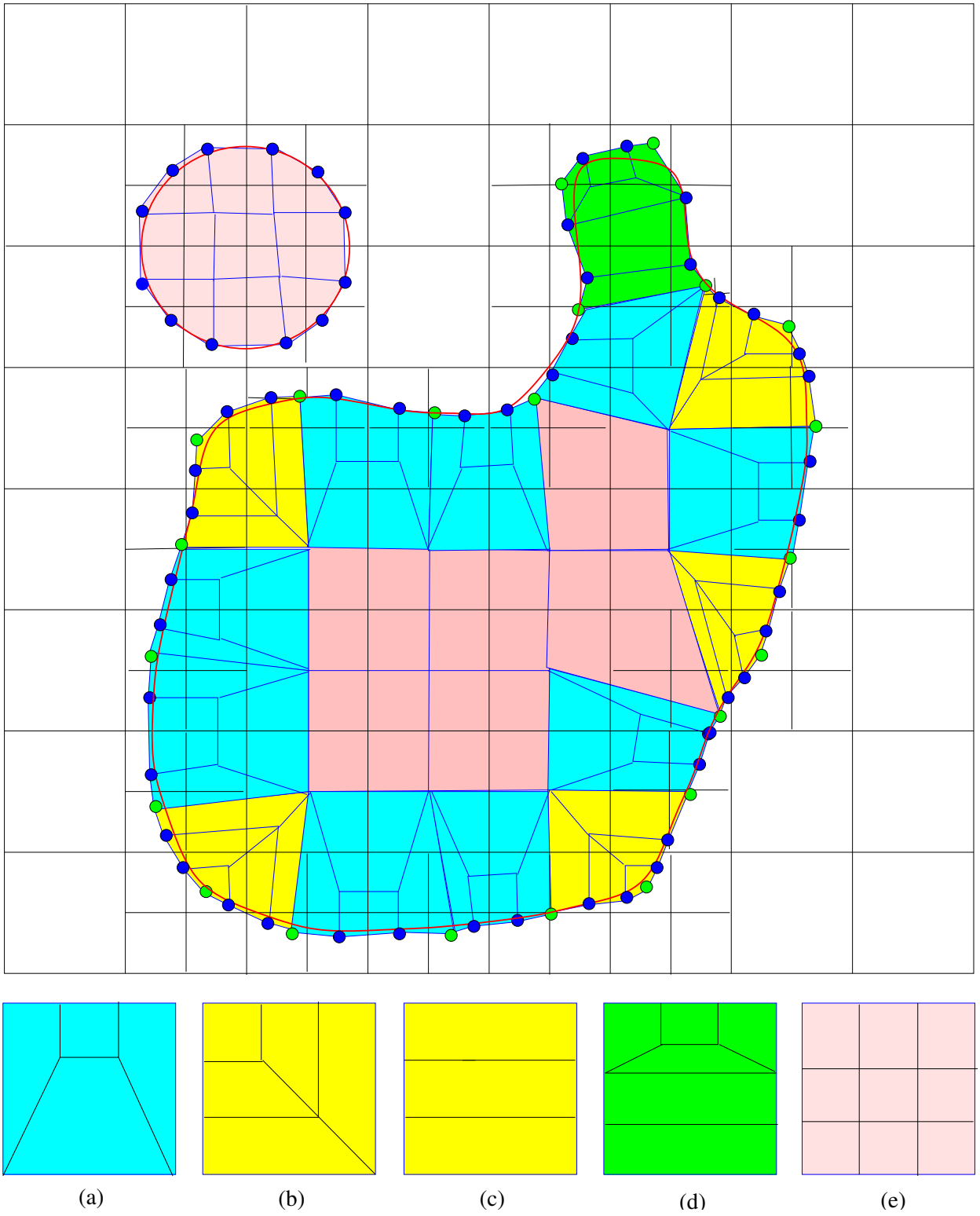


Figure 82: Top row - an example of adaptive quad mesh generation in 2D. Each green point represents a minimizer point of a cell to be refined, and the red curve represents the real isocontour. Bottom row - the decomposition templates of Method 3 shown in Figure 78.

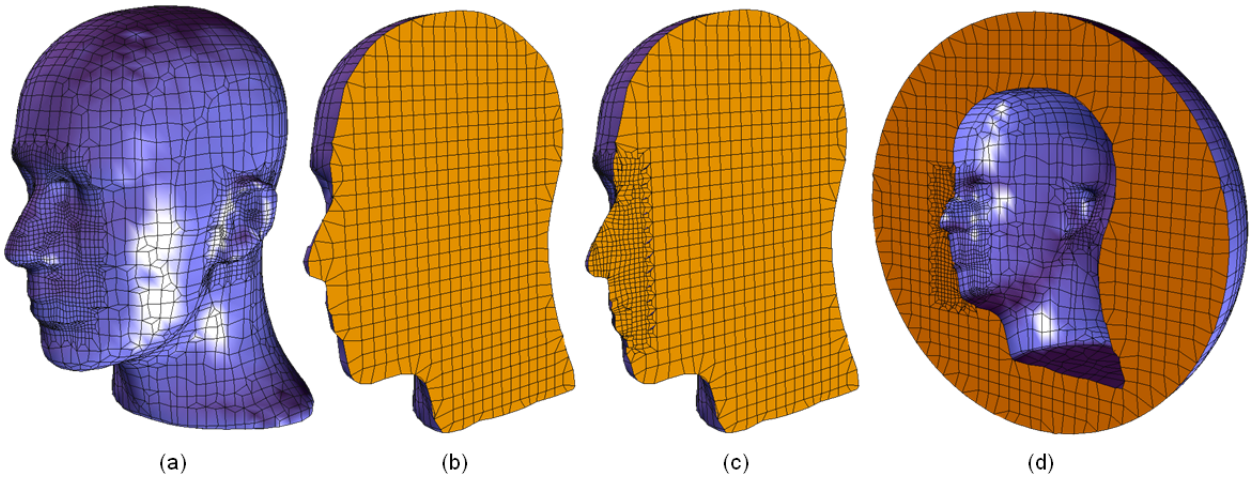


Figure 83: Adaptive hexahedral mesh decomposition (Method 1). Left - a 2D example; Middle - a small hexahedron is inserted; Right - the top face of the original hexahedron needs to be refined.

2D Mesh Decomposition In 2D, the uniform quadrilateral mesh can be constructed by analyzing each interior grid point, which is shared by four cells. One minimizer point is calculated for each cell, therefore four minimizer points are obtained and they construct a quad. All the templates defined in Figure 78 can be used here for adaptive 2D mesh generation. Figure 82 shows an example of adaptive quadrilateral mesh extraction using Method 3. When we analyze each cell to calculate the minimizer point, we compare the feature sensitive error function of this cell with a threshold ϵ. If the error function value of a cell is greater than ϵ, then this cell needs to be subdivided. An interior grid point is shared by four cells, therefore there are a total of $2^4 = 16$ configurations. Due to the symmetry, there are five basic templates for the quad refinement. A uniform quadrilateral mesh can be refined adaptively by using those templates.

3D Mesh Decomposition **Indirect Method:** Adaptive and quality tetrahedral meshes have been generated from volumetric imaging data [242] [241], therefore we can obtain hexahedral meshes by decomposing each tetrahedron into four hexahedra.

Direct Method: Not all the direct methods for adaptive 2D mesh generation shown in Figure 78 can be extended to 3D. There are two main methods for adaptive hexahedral mesh generation, one is extended from the first 2D direct method and the other one is derived from part of the third 2D direct method.

Extended from the first 2D direct method in Figure 78, Method 1 refines each hexahedron independently as shown in Figure 83. It first splits each hexahedron into seven ones by inserting one small hex in its center, and each face of the original hex is contained in a hex independently. If one face needs to be refined, then the hex containing it will be refined as shown in the right picture of Figure 83. If there are i $(i = 1, \ldots, 6)$ faces that need to be refined for a hexahedron, then the resulting mesh has $(6 - i + 1 + 6i = 5i + 7)$ elements and $8(i + 1)$ newly inserted vertices.

Method 2 is derived from part of the third 2D direct method shown in Figure 78. In the process of refinement, this method considers whether the error function value of each cell is greater than a threshold ε or not. One hexahedron has a total of eight vertices, so there are $(2^8 = 256)$ configurations. Due to the symmetry, there are 22 necessary templates [226], but not all the templates can be decomposed into hexahedra. Figure 84 shows five templates for adaptive hexahedral decomposition and the detailed view [209], which are much more complicated than the templates of 2D quadrilateral decomposition. Figure 85 lists the number of elements and the number of newly inserted vertices for each template.

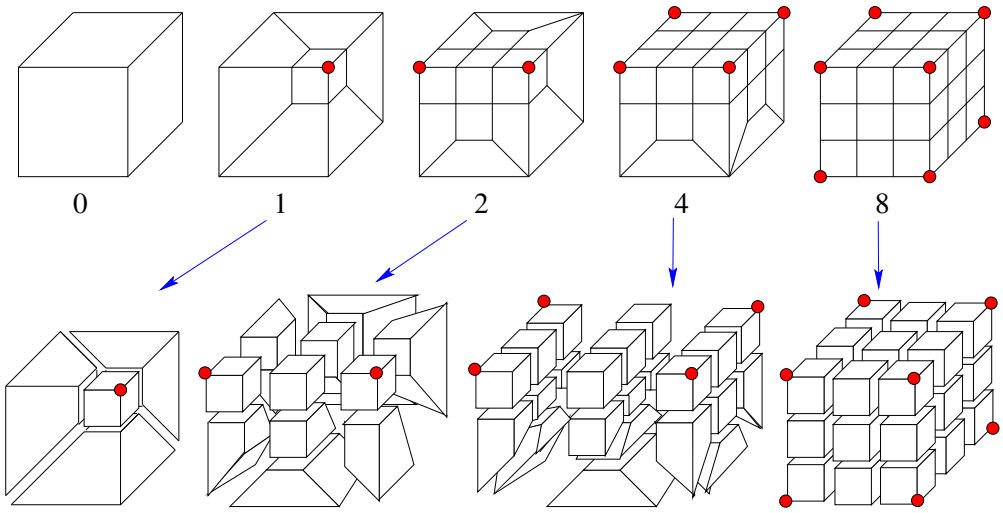


Figure 84: Templates of adaptive hexahedral mesh decomposition (Method 2) according to the cells to be refined from which red minimizer points are generated. The bottom row shows the detailed decomposition format.

Method	Number of	0	1	2	4	8
2	elements	1	4	11	22	27
	vertices	0	7	19	39	56

Figure 85: The element number and the newly inserted vertex number of Method 2 within refined hexahedra shown in Figure 84.

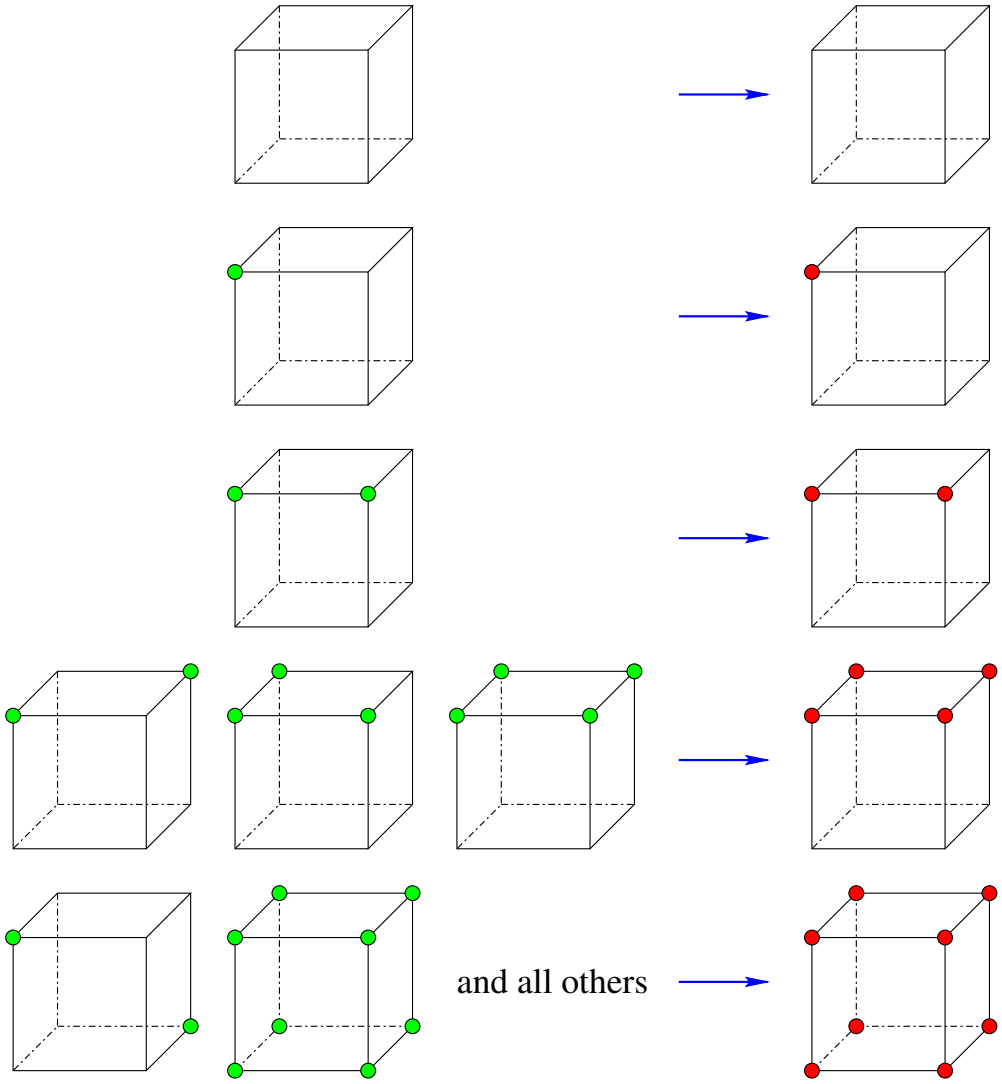


Figure 86: The Look-Up table for converting an arbitrary configuration to one of the five templates in Figure 84. Each green node represents the cell from which the minimizer point is generated needs to be refined. The sign of the cell generating a red node is 1, otherwise the sign is 0.

We set a sign for each leaf cell at the uniform starting octree level indicating if this cell needs to be refined or not. For each leaf cell, the feature sensitive error function is calculated and compared with a threshold ϵ. If the function value is greater than ϵ, then the sign is set to be 1, otherwise it is 0. For each hexahedron extracted from the uniform level, we check if it belongs to one of the templates shown in Figure 84. If not, we need to convert it by looking up the table shown in Figure 86. We keep updating the sign for each leaf cell until no sign changes, at this time all the generated hexahedra in the uniform level are in the format of the five templates shown in Figure 84, then we can construct an adaptive hexahedral mesh using the corresponding templates.

Each hexahedron is constructed by eight minimizer points, which are calculated from leaf cells in the uniform octree level. The error function of the cell generating a minimizer point is either greater than the threshold ϵ or $\leq \epsilon$, therefore there are a total of $2^8 = 256$ configurations for a hexahedron. Figure 86 shows the Look-Up table for converting an arbitrary configuration to the five templates shown in Figure 84. The green node means the error function of the cell generating this minimizer point is greater than the threshold ϵ. The red node means the sign of the cell

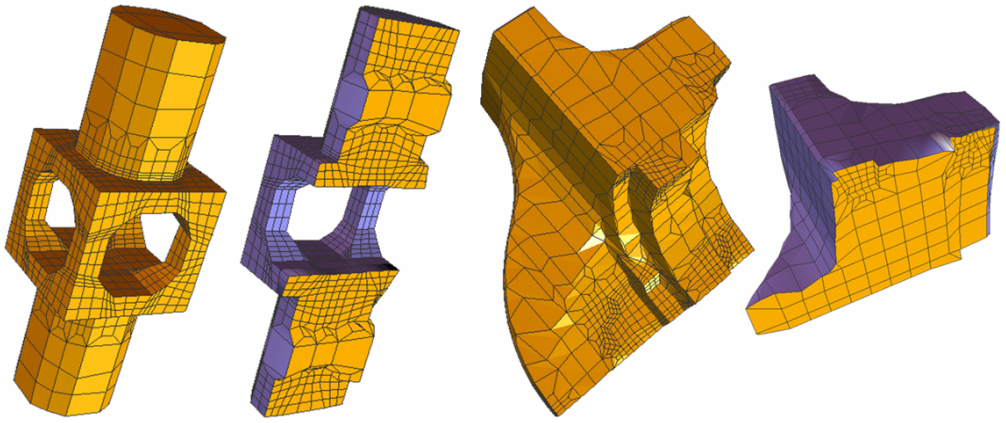


Figure 87: Sharp features are preserved. From left to right: an adaptive quad mesh of a mechanical part; an adaptive hex mesh of a mechanical part; an adaptive quad mesh of a fandisk, an adaptive hex mesh of a fandisk.

generating this node is set to be 1, otherwise the sign is 0.

In the process of adaptive hexahedral mesh generation, we need to insert extra vertices and detect if they lie on the boundary or not. If a vertex lies on a boundary edge or a boundary face, then it is a boundary vertex. Otherwise it lies interior to the interval volume. There is a special case that we need to be careful, a vertex lying on an edge whose two ending points are on the boundary, or lying on a face whose four points are on the boundary, may not be on the boundary. For those extra vertices lying inside the interval volume, we choose the linear interpolation of the eight vertices of the original hexahedron. For those existing and newly inserted vertices lying on the boundary isosurface, we first compute their positions from the linear interpolation, then move them toward the isosurface to obtain their new positions as we do for the adaptive quadrilateral isosurface extraction.

Figure 88 compares adaptive hexahedral meshes of the human head generated from Method 1 and Method 2. It is obvious that Method 2 constructs a better nose than Method 1 because it introduces extra vertices on edges of refined hexes resulting in a more accurate approximation, and Method 2 tends to generate meshes with better quality than Method 1. The extracted surface mesh from Method 2 is a little different from the result of the third method shown in Figure 81, since only templates 0, 1, 2a and 4 of the third method in Figure 78 are adopted, while templates 2b and 3 are not used. Since we still use QEF for computing minimizing vertices, we can also preserve sharp edges and corners (Figure 87).

6.2.5 Mesh Adaptivity

In order to generate accurate meshes with the minimal number of elements and vertices, it is important to choose a good error metric to decide where we should generate a finer mesh and where a coarser mesh should be kept. There are three main ways to control the mesh adaptivity. Users can also design an error function based on their specific requirements.

- The feature sensitive error function.
- Areas that users are interested in.
- Finite element calculation results.

The feature sensitive error function [242] [241] is defined as the difference of trilinear interpolation functions between coarse and fine octree levels normalized by the gradient magnitude.

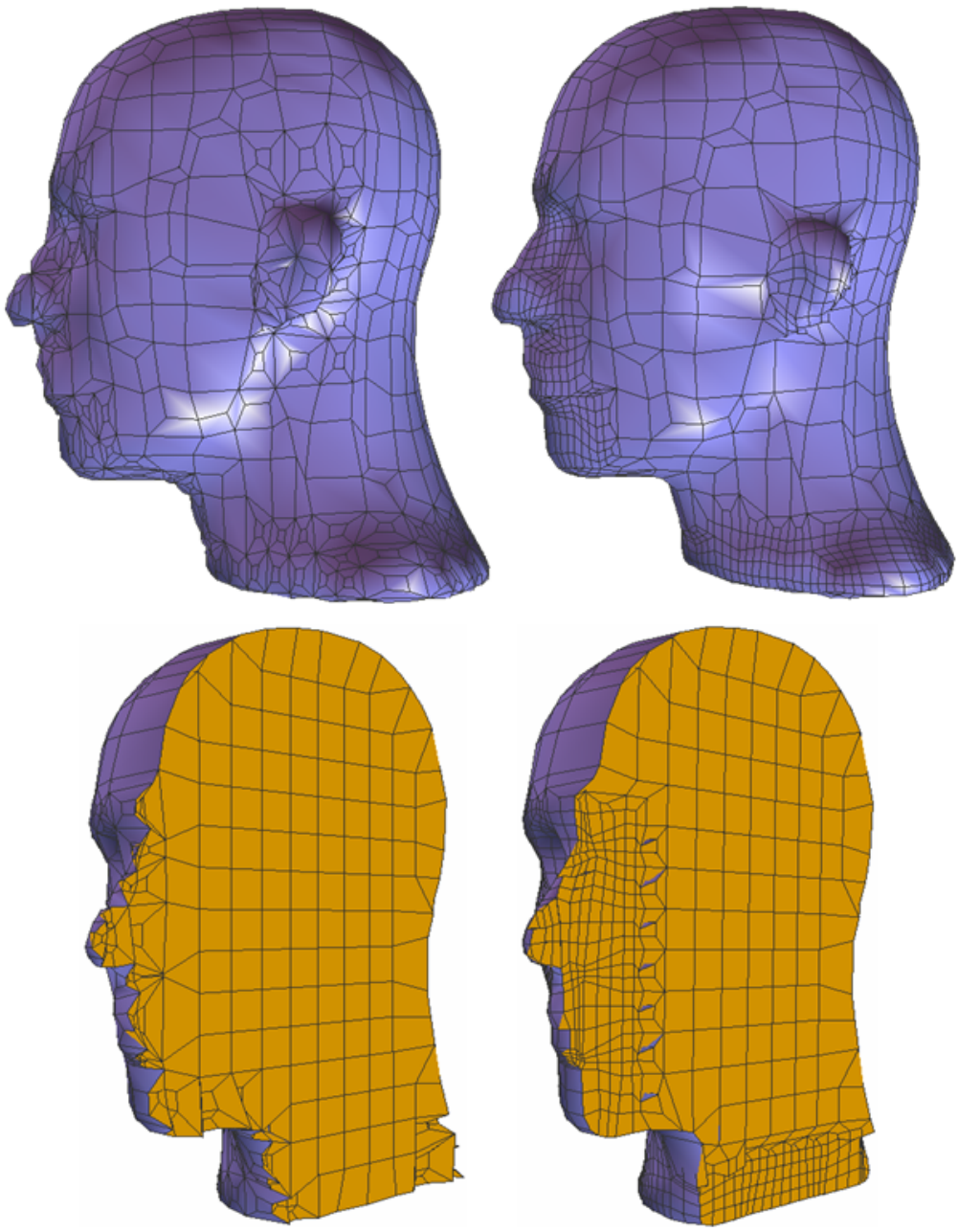


Figure 88: Adaptive hexahedral meshes from Method 1 (left) and Method 2 (right) for the human head. Top row shows the boundary isosurfaces, it is obvious that Method 1 generates bad nose as Figure 81. Bottom row shows cross sections, the right part of elements are removed.

Type	DataSet	MeshSize (Vertex♯, Elem♯)	Scaled Jacobian (best,aver.,worst)	Condition Number (best,aver.,worst)	Oddy Metric (best,aver.,worst)	Inverted Elem♯
quad	Bubble^b	(208, 206)	(1.0, 0.92, 0.36)	(1.0, 1.12, 2.77)	(0.0, 0.61, 13.37)	0
	Bubble^a	-	(1.0, 0.94, 0.62)	(1.0, 1.07, 1.60)	(0.0, 0.34, 3.13)	0
	Head^b	(714, 712)	(1.0, 0.92, 0.06)	(1.0, 1.13, 17.41)	(0.0, 0.98, 604.24)	0
	Head^a	-	(1.0, 0.92, 0.37)	(1.0, 1.10, 2.73)	(0.0, 0.48, 12.93)	0
	mAChE^b	(19998, 20013)	(1.0, 0.90, 0.04)	(1.0, 1.17, 27.63)	(0.0, 1.29, 1524.67)	0
	mAChE^a	-	(1.0, 0.90, 0.16)	(1.0, 1.15, 6.26)	(0.0, 0.87, 76.28)	0
hex	Head^b	(1210, 812)	(1.0, 0.85, 1.9e-3)	(1.0, 2.62, 519.74)	(0.0, 12.88, 6.95e3)	1
	Head^a	-	(1.0, 0.85, 0.02)	(1.0, 1.98, 46.34)	(0.0, 5.03, 638.83)	0
	mAChE^b	(81233, 70966)	(1.0, 0.94, 5.2e-5)	(1.0, 2.07, 1.92e4)	(0.0, 18.35, 1.58e6)	5
	mAChE^a	-	(1.0, 0.94, 0.01)	(1.0, 1.40, 74.73)	(0.0, 2.37, 1379.81)	0

Figure 89: The comparison of the three quality criteria (the scaled Jacobian, the condition number and Oddy metric) before/after the quality improvement for quadrilateral meshes of bubble, head and mAChE. DATA^b – before quality improvement; DATA^a – after quality improvement.

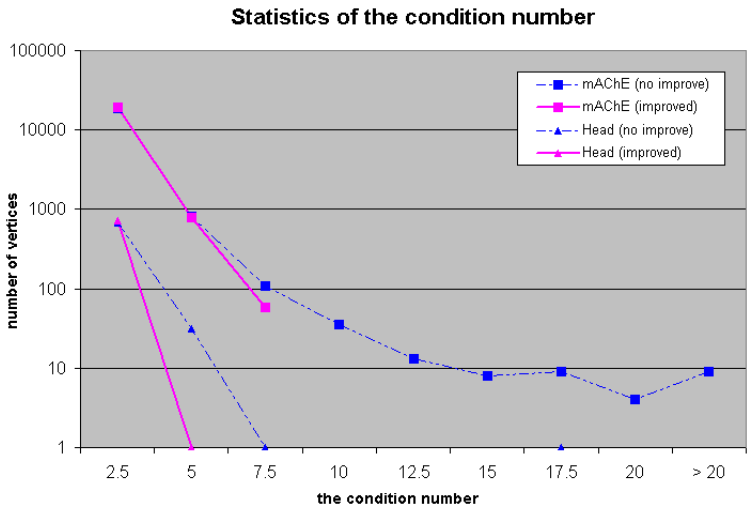


Figure 90: The histogram of the condition number for quadrilateral meshes of mAChE and the human head.

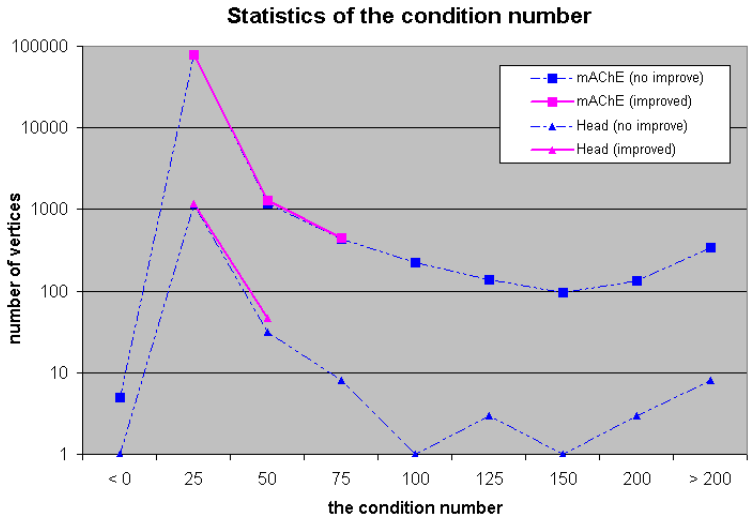


Figure 91: The histogram of the condition number for hexahedral meshes of mAChE and the human head.

DataSet	Type	Dimension	Number of Elements (Extraction Time (unit : ms))			
			(a)	(b)	(c)	(d)
Bubble	SDF	65^3	206 (172)	1478 (329)	1854 (344)	–
Head	SDF	65^3	1942 (594)	812 (375)	4049 (750)	17905 (3267)
Knee	SDF	65^3	4058 (735)	1386 (453)	7111 (797)	36207 (1516)
Skull	CT	129^3	–	–	20416 (9893)	10827 (9205)
Skin	CT	129^3	20999 (9955)	61244 (14565)	–	–
mAChE	Given	257^3	20013 (6080)	–	70966 (11690)	38939 (7955)

Figure 92: Data Sets and Test Results. The CT data sets are re-sampled to fit into the octree representation. Rendering results for each case are shown in Figure 96, 93, 94, 95 and 76. Skull and Skin are extracted from the UNC Head model.

It is sensitive to areas of large geometric feature since it directly measures the surface difference between coarse and fine levels, for example, the facial features (nose, eyes, mouth and ears) in the head model as shown in Figure 88 and 93.

Sometimes, people are interested in some special areas based on their physical or biological applications. For example, there is a cavity in the structure of the biomolecule called mouse acetylcholinesterase (mAChE) [217]. A finer mesh is required around the cavity area while a coarse mesh needs to be kept in other regions. In this situation, the error function should be defined by regions. Figure 76 shows the adaptive quadrilateral and hexahedral meshes for the biomolecule mAChE, and it is obvious that the mesh adaptivity is controlled by regions.

In finite element simulations, we first need to construct meshes to represent the analyzed geometric domain, then solve ordinary/partial differential equations over it using the finite element method. For accurate and efficient finite element analysis, adaptive meshes are preferable. The mesh adaptivity can be controlled directly by finite element solutions to balance the error of finite element solutions over each element. Figure 96 shows quad meshes of a bubble model. The mesh adaptivity is controlled by its deformation obtained from the finite element analysis.

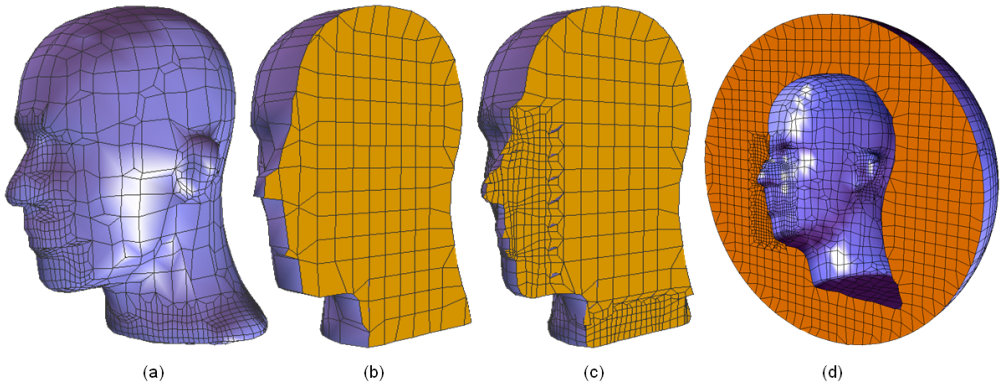


Figure 93: Quadrilateral and hexahedral meshes of the human head. (a) - an adaptive quadrilateral mesh; (b) - the uniform hexahedral mesh at a chosen starting level; (c) - an adaptive interior hexahedral mesh controlled by the feature sensitive error function; (d) - an adaptive exterior hexahedral mesh controlled by the feature sensitive error function.

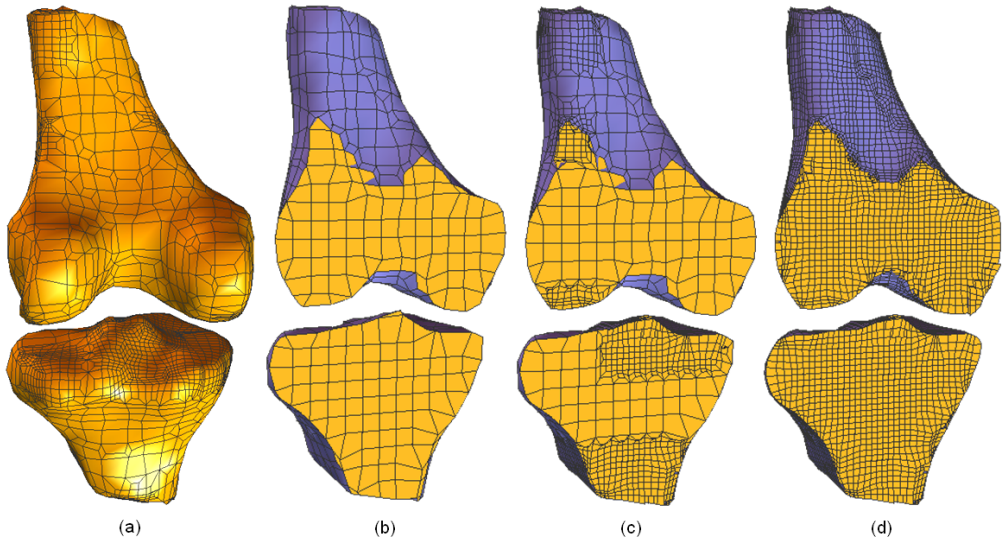


Figure 94: Quadrilateral and hexahedral meshes of the knee. (a) - an adaptive quadrilateral mesh; (b) - the uniform hexahedral mesh at a chosen starting level; (c) - an adaptive hex mesh controlled by the feature sensitive error function; (d) - all the hexahedral elements in (b) are refined.

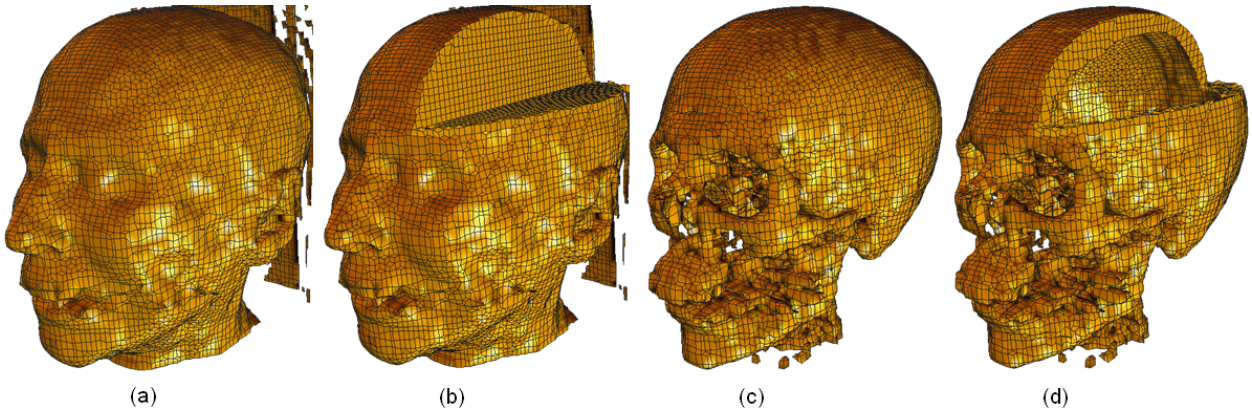


Figure 95: Quadrilateral and hexahedral meshes are extracted from a CT-scanned volumetric data (UNC head). (a) - the quadrilateral mesh of the skin; (b) - the hexahedral mesh of the volume inside the skin; (c) - the quadrilateral mesh of the skull isosurface; (d) - the hexahedral mesh of the skull.

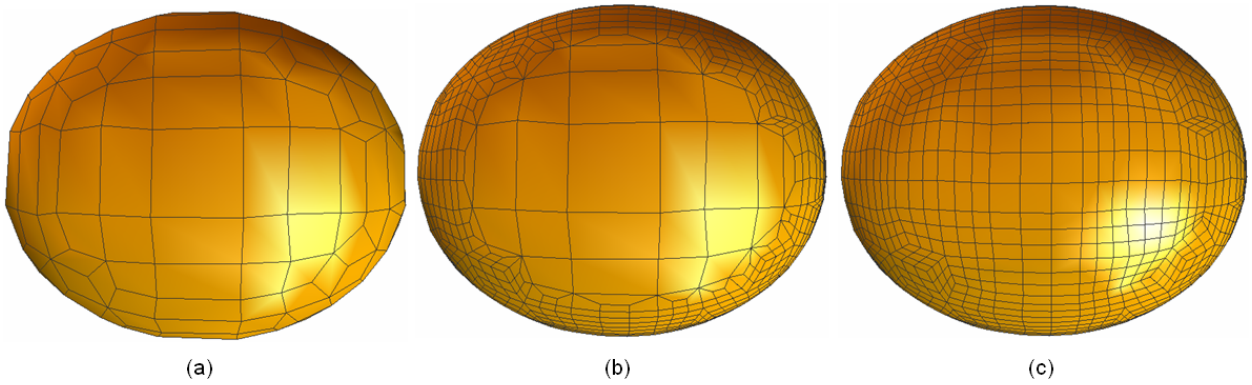


Figure 96: Quadrilateral meshes of a bubble model. (a) - the uniform mesh at a chosen starting level; (b) - an adaptive mesh controlled by finite element solutions (deformation); (c) - a mesh generated by refining all the elements in (a).

6.2.6 Quality Improvement

Quality improvement is a necessary step for finite element mesh generation. First we need to choose corresponding quality metrics to measure the quality of quadrilateral and hexahedral meshes. Here we select the scaled Jacobian, the condition number of the Jacobian matrix and Oddy metric [177] as our metrics [139][140][142].

Assume $x \in \Re^3$ is the position vector of a vertex in a quad or a hex, and $x_i \in \Re^3$ for $i = 1, \ldots, m$ are its neighboring vertices, where $m = 2$ for a quad and $m = 3$ for a hex. Edge vectors are defined as $e_i = x_i - x$ with $i = 1, \ldots, m$, and the Jacobian matrix is $J = [e_1, ..., e_m]$. The determinant of the Jacobian matrix is called *Jacobian*, or *scaled Jacobian* if edge vectors are normalized. An element is said to be *inverted* if one of its Jacobians ≤ 0. We use the *Frobenius norm* as a matrix norm, $|J| = (tr(J^T J)^{1/2})$. The condition number of the Jacobian matrix is defined as $\kappa(J) = |J||J^{-1}|$, where $|J^{-1}| = \frac{|J|}{det(J)}$. Therefore, the three quality metrics for a vertex x in a quad or a hex are defined as follows:

$$Jacobian(x) = det(J) \tag{234}$$

$$\kappa(x) = \frac{1}{m}|J^{-1}||J| \tag{235}$$

$$Oddy(x) = \frac{(|J^T J|^2 - \frac{1}{m}|J|^4)}{det(J)^{\frac{4}{m}}} \tag{236}$$

where $m = 2$ for quadrilateral meshes and $m = 3$ for hexahedral meshes.

In the process of mesh quality improvement, our goal is to remove inverted elements and improve the worst condition number of the Jacobian matrix. First the averaging method is used to remove inverted elements. We calculate the scaled Jacobian for a vertex in each element, and relocate this vertex by the average of all its neighbors if the Jacobian is negative. Then we calculate the condition number of the Jacobian matrix for a vertex in each quad or hex, and find the vertex with the maximum value. We compute the new position for this vertex using the conjugated gradient method with the condition number (Equation 235) as objective.

If the relocated vertex is an interior node, then we replace the location of this vertex with the calculated new position. If this vertex lies on the boundary, then we calculate its new position and move it toward the isosurface along its normal direction. We keep reducing the maximum condition number for quad or hex meshes until we arrive a given threshold. In this way, we can improve the worst condition number of the Jacobian matrix, as well as improving the other two metrics, the scaled Jacobian and Oddy metric. However, it is possible to produce an invalid mesh containing inverted elements. We choose a 'smart' smoothing method [112], which relocates the point only if the mesh quality is improved.

Figure 89 shows the improvement of the worst values of the scaled Jacobian, the condition number and Oddy metric. The histograms of the condition number (Figure 90 and 91) show the overall quality of quad and hex meshes for the human head model and a biomolecule mAChE. By Comparing the three quality metrics before and after quality improvement, we can see that the worst parameters are improved significantly.

6.2.7 Results and Applications

We have developed an interactive program for adaptive and quality quadrilateral/hexahedral mesh extraction and rendering from volumetric imaging data, and plugged it into our LBIE-Mesh software (Level Set Boundary and Interior-Exterior Mesher), which can generate adaptive and quality 2D (triangular/quadrilateral) and 3D (tetrahedral/hexahedral) meshes from volume data. The algorithm of tetrahedral mesh generation is described in [242] [241]. In this software, error tolerances and isovalues can be changed interactively. Our results were computed on a PC equipped with a Pentium III 800MHz processor and 1GB main memory.

Our algorithm has been used to generate quadrilateral and hexahedral meshes for some signed distance function data such as the bubble (Figure 96), the human head (Figure 93) and the knee model (Figure 94). We also extracted meshes for the skin and the skull from a CT scanned data (the UNChead, Figure 95), and tested the algorithm on biomolecular data (mAChE, Figure 76). Figure 92 shows the information for each dataset and results. The results consist of the number of elements, the extraction time and images with respect to different isovalues and error tolerances. Extraction time includes octree traversal, QEF computation and mesh extraction.

Figure 96 shows the extracted quadrilateral meshes for a bubble, which has been used in the simulation of drop deformation using the finite element method. First, we generate a uniform quad mesh for the original state of the bubble. Then we get finite element solutions such as the deformation from finite element analysis, and use the error of the deformation over each element to control the mesh adaptivity. Finally we can provide an adaptive and quality quad mesh to limit the maximum error of finite element solutions within a threshold.

Some physically-based simulations need both interior and exterior hexahedral meshes. For example, when people are analyzing the electromagnetic scattering over the human head, hex meshes of the volume interior to the head surface and hex meshes exterior to the head surface but inside an outer sphere are needed at the same time. Figure 93 shows the extracted interior and exterior meshes for a head model. The facial features such as nose, eyes, mouth and ears are kept, and fine meshes are generated in those regions. Figure 1 shows another example of interior and exterior hexahedral meshes, the biomolecule mAChE. The mesh adaptivity is controlled by regions, fine meshes are generated in the area of cavity.

Conclusions We have presented an algorithm to extract adaptive and quality quadrilateral and hexahedral meshes directly from volumetric imaging data. First, a bottom-up surface topology preserving octree-based algorithm is used to select a starting octree level, at which we extract uniform meshes with correct topology using the dual contouring isosurface extraction method [136] [242] [241]. Then we extended it to adaptive quadrilateral and hexahedral mesh generation using some predefined templates without introducing any hanging nodes. The position of each boundary vertex is recalculated to approximate the isosurface more accurately. The mesh adaptivity can be controlled in three ways, the feature sensitive error function [242] [241], the areas that users are interested in and finite element solutions. Users can also design their own error function to control the mesh adaptivity according to their specific requirements. Finally, three various quality metrics are selected to measure the mesh quality, and the relaxation based technique is used to improve it. The resulting meshes are extensively used for efficient and accurate finite element calculations. Some of them have been used successfully.

6.3 Efficient Delaunay Mesh Generation From Sampled Scalar Functions

Many modern research areas face the challenge of meshing level sets of sampled scalar functions. While many algorithms focus on ensuring geometric qualities of the output mesh, recent attention has been paid to building topologically accurate Delaunay conforming meshes of any level set from such volumetric data.

We present an algorithm which constructs a surface mesh homeomorphic to the true level set of the sampled scalar function. The presented algorithm also produces a tetrahedral volumetric mesh of good quality, both interior and exterior to the level set. The meshing scheme presented substantially improves over the existing algorithms in terms of efficiency. Finally, we show that when the unknown sampled scalar function, for which the level set is to be meshed, is approximated by a specific class of interpolant, the algorithm can be simplified by taking into account the nature of the interpolation scheme so as to circumvent some of the critical computations which tend to produce numerical instability.

6.3.1 Problem and Motivation

A wide variety of science and engineering applications rely on accurate level set triangulation. This is especially true for multiscale models in biology, such as macromolecular structures extracted from reconstructed single particle cryo-EM (Electron Microscopy), cell-processes and cell-organelles extracted from TEM (Tomographic Electron Microscopy), and even trabecular bone models extracted from SR-CT (Synchrotron Radiation Micro-Computed Tomography) imaging. Computational analysis of these models for estimation of nano, micro, or mesoscopic structural properties depends on the mesh representation of the contour components respecting their topological features.

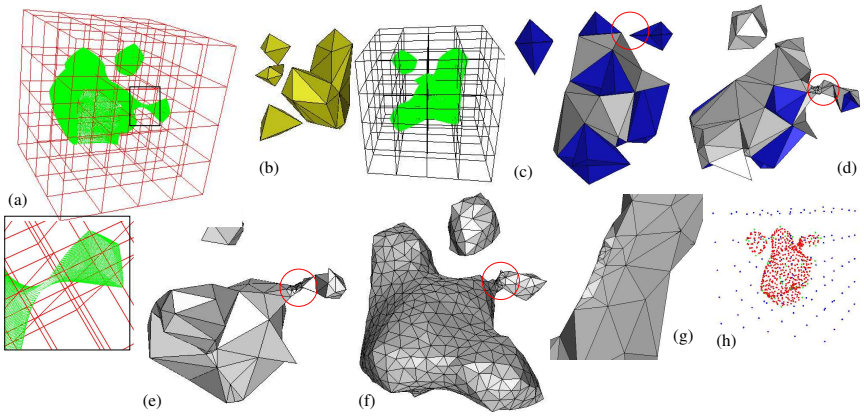


Figure 97: Various stages of our algorithm. (a) A rectilinear grid with sample values of an unknown function at the grid points. Within cells, the function is approximated with a trilinear interpolant. For the purpose of visualization only, we collect a set of points (green) on the surface and display them. A narrow region of the surface is magnified below. (b) Another view of the data (right) and the same view of the mesh generated by Marching Cubes [159]. Note that the mesh is disconnected in the thin region. (c) The mesh generated by the restricted Delaunay triangulation of only edge and grid points. Blue facets have a grid point as at least one of their vertices. This point set is still not sufficient to produce a Delaunay-conforming mesh. (d) The mesh generated by addition of sample points of Σ. The topology is now recovered (Property I). (e,f) Geometrical refinement for progressively smaller value of ϵ. (g) Even in the magnified portion of the thin region, the triangles approximate the geometry nicely (Property II). (h) All the points involved in construction of the mesh including grid points (blue), edge points (green), and new points added by the algorithm (red). Observe that in order to recover the topology and reduce the geometric error in the approximation, many surface sample points are added to the point set.

Our goal is to find an algorithm to solve the following problem. The input to the algorithm is a rectilinear sampling of a bounded domain of an unknown scalar function F. The rectilinear grid need not be uniform; it may be adaptive as in the case of an octtree. The user then specifies a local interpolant to generate a level set approximation for any isovalue v; we use Σ to denote this level set of the interpolating function. Since the function F is unknown, we must assume that the local interpolant produces a good approximation of the function F within each cell of the grid. Our algorithm is general enough to use any local interpolant, however, in our experience, a trilinear interpolant is the most natural choice.

Our goal is construct a mesh in an efficient manner such that the following properties hold:

I **Topological Guarantee:** M is homeomorphic to Σ.

II **Geometrical Guarantee:** The Hausdorff distance from M to Σ is within a user-specified

bound ϵ.

III **Delaunay Conformity:** M is a subcomplex of the Delaunay triangulation of the vertex set of M.

IV **Adaptivity:** The user can decimate part of the volumetric data and still preserve properties I, II, and III.

Once a surface M is generated that is Delaunay conforming, it is possible to improve the mesh quality by applying any Delaunay refinement algorithm. We detail such an algorithm in Section 6.3.4. Figure 97 visually illustrates a toy data set (a), the failure of a typical isocontouring method, in this case Marching Cubes, in reconstructing the level set (b), generation of the correct topology by our algorithm (c-d), geometric refinement (e-g), and the final Delaunay-conforming surface sample (h).

At this point, we emphasize the novelty of our approach. Although there exist algorithms which can be applied to solve the problem as stated, such algorithms are devised in a more general setting and thus do not exploit the natural structure of the volumetric data. Our approach has a number of unique advantages over its predecessors. As part of the algorithm, we collect some of the grid points around Σ and use them to build a Delaunay conforming mesh efficiently. Once the surface mesh is created and forced to conform to the Delaunay triangulation of the point set, these grid points can either be removed or used to construct an interior or exterior tetrahedral volume mesh.

Additionally, as a result of noise in the input data or a poor choice of the isovalue v, there may exist topological anomalies in the surface Σ. In mesh processing literature, such anomalies have been referred to as "topological noise" [232, 23]. The term "noise" indicates undesirable features of small geometric size that prevent the mesh from being used for further processing. Methods have been developed to remove such artifacts provided that the point sample of Σ is sufficiently dense near the anomalies; our method guarantees this density. Therefore, the output mesh M can be applied without any further refinement to any point processing algorithm that uses prior Delaunay-based reconstruction of geometry to detect and selectively remove these topological features.

Finally, algorithms involving volumetric data and meshes often become computationally demanding due to the large size of data sets. The adaptivity of the algorithm (property IV) provides one way to ease calculations by down-sampling less important regions of the volumetric data.

6.3.2 Prior Work

Mesh generation techniques have received significant attention in the past two decades. Two works in particular, one by Chew [74] and one by Edelsbrunner and Shah [103], have spawned important and relevant results in the field. We will address the prominent successors of each of these works and compare the relative advantage of our approach.

Chew provided one of the first meshing algorithms for curved surfaces with provably good geometry in [74], although the algorithm as stated in the paper could not guarantee topological correctness. Boissonnat and Oudot showed how Chew's algorithm can be applied to produce a dense sample of Σ and subsequently mesh it. Oudot, Rineau and Yvinec recently improved that method by including sliver exudation, that is, the process of removing tetrahedra with very small dihedral angles from a mesh [182]. Alliez, Cohen-Steiner, Yvinec and Desbrun have also adapted the method to produce nicely graded meshes, i.e. meshes where the tetrahedra vary in size gradually based on their distance to the surface [6].

Separately, Edelsbrunner and Shah established a criterion called the *closed ball property* for ensuring that a mesh is Delaunay conforming [103]. We explain the closed ball property in Section 6.3.3. Recent work by Cheng, Dey, Ramos, and Ray uses this property to provide a method

for constructing Delaunay conforming meshes that avoids the need to estimate local feature size [69]. This is a significant development as approximating local feature size is computationally expensive and not always numerically robust. Cheng, Dey and Ramos have extended this strategy for piecewise smooth complexes [72]. Very recently, Dey and Levine [94] have given a two phase algorithm to mesh isosurfaces from imaging data.

All of these approaches to mesh generation are useful, however, they all rely on an oracle to know whether an arbitrary ray intersects the surface Σ. The implementation of such an oracle becomes computationally prohibitive when applied to piecewise interpolated surfaces. For example, a common data set size is 100^3 vertices, meaning there exist 100^3 functions in the piecewise decomposition. Thus, a single ray may pass through over a hundred separate function domains, making intersection calculations expensive. As we detail in Section 6.3.4, a major advantage of the algorithm presented here is that we take advantage of the original rectilinear scaffolding from the data to substantially reduce the computational overhead required.

Our work also improves upon existing methods for isosurface construction. Many well-known techniques exist for isosurfacing including marching cubes [159], active snakes, dual contouring [243], and higher order interpolation [43]. A variety of approaches have also been developed to provide hierarchical isosurface extraction and interior meshing [234, 125, 133]. Shewchuk and Labelle recently provided a straightforward isosurfacing and stuffing algorithm with good quality tetrahedra [144]. Relatively few works, however, take into account the effect of a trilinear interpolant within each grid cell [169, 73, 158]. Attali and Lachaud gave an algorithm for construction of Delaunay conforming isosurfaces [7]. Their method, however, relies on a specific rule established by Lauchaud in [145] that does not accommodate interpolation within grid cells. Further, none of these techniques generalize easily to data that is sampled adaptively or to arbitrary interpolants.

6.3.3 Background

The Voronoi and Delaunay diagrams of a point set P, denoted $\operatorname{Vor} P$ and $\operatorname{Del} P$ respectively, play an important role in the computations involved. Due to page limitations, we do not go into the detail of their construction and refer the reader to any standard computational geometry textbook, e.g. [90].

Given a point set P chosen from the same space in which Σ is embedded, the *restricted Delaunay triangulation of* P, denoted $\operatorname{Del} P|_\Sigma$, is defined to be the set of Delaunay objects of $\operatorname{Del} P$ whose dual Voronoi objects have non-zero intersection with Σ. If P is chosen in a way that respects the structure and local feature size of Σ, $\operatorname{Del} P|_\Sigma$ will be a mesh with the desired properties.

Edelsbrunner and Shah gave a sufficient criterion called the *closed ball property* [103], sometimes referred to as the *topological ball property*, for $\operatorname{Del} P|_\Sigma$ to be homeomorphic to Σ. A Voronoi object V of dimension k satisfies the closed ball property if $V \cap \Sigma = \emptyset$ or $V \cap \Sigma$ is homeomorphic to a closed ball of dimension $k-1$. Accordingly, a point set P is said to satisfy the closed ball property if every Voronoi object of $\operatorname{Vor} P$ satisfies the closed ball property. Using the notion of *transversality* as defined in [121], their criterion can now be stated precisely.

Theorem 6.1. *[103] If Σ intersects each Voronoi object of $\operatorname{Vor} P$ transversally and $\operatorname{Vor} P$ satisfies the closed ball property, then $\operatorname{Del} P|_\Sigma$ is homeomorphic to Σ.*

We will show in Section 6.3.4 that our algorithm produces a mesh satisfying the closed ball property. This ensures that the mesh is Delaunay conforming (property III from Section 6.3.1) and, by the theorem, that the mesh is homeomorphic to Σ (property I).

6.3.4 Algorithm

In this section, we describe the algorithm, analyze its efficiency and present some simplifying results for a specific choice of local interpolant. Figure 98 shows an overview of the process in two

dimensions.

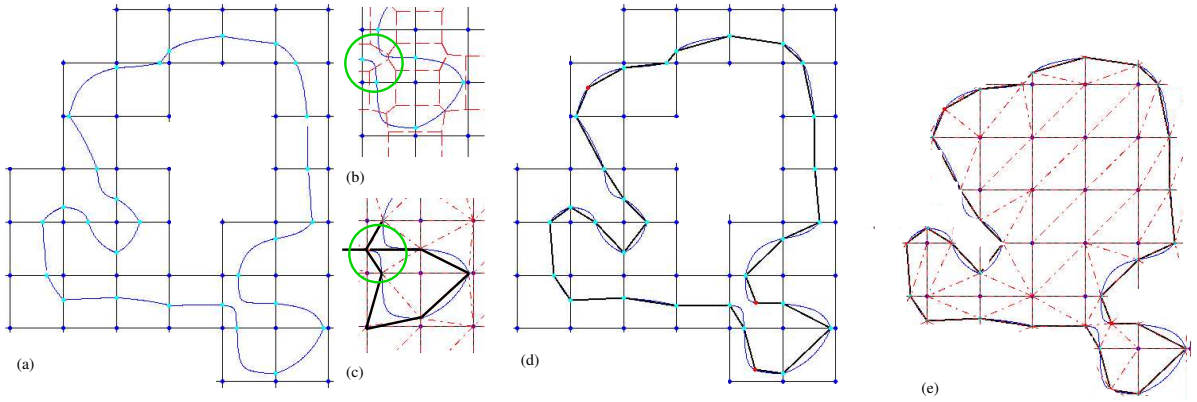


Figure 98: A 2D example using bilinear interpolation of Σ demonstrating the importance of the closed ball property. (a) Grid points (dark blue) and edge points (light blue) relevant to our algorithm are shown. Note that in the 3D case we form a layer of grid points twice as thick. (b) A portion of the Voronoi diagram is shown in red and a location where the closed ball property is violated is circled. (c) Since the closed ball property is violated, the restricted Delaunay diagram (black) has incorrect topology in the circled region. (d) Red points are inserted where the closed ball property is violated and the restricted Delaunay graph is formed (black). (e) By including those grid points interior to Σ, we efficiently produce an interior mesh that does not alter the Delaunay conforming surface mesh.

Algorithm Description Our algorithm is motivated primarily by the work of Cheng et al. in [69] who build a Delaunay conforming approximation of the level set of any general implicit function. Since our problem is focused on locally interpolated functions, we take advantage of the natural scaffolding of the input grid to substantially improve the computational efficiency of the algorithm. Moreover, we also show that once the Delaunay conforming surface mesh is extracted, it is quite straightforward to build a tetrahedral volumetric mesh of good quality. We call the algorithm for extracting a surface mesh DelSurfMesh and the extension to build a Delaunay tetrahedral interior or exterior mesh DelVolMesh.

In [69], the authors start with a small sample of points lying on the surface to be meshed. They keep refining the mesh until its vertex set satisfies the closed ball property, thereby providing a Delaunay conforming mesh homeomorphic to Σ. To ensure that a point set P satisfies the closed ball property, it is necessary to check the intersection of Σ with each Voronoi edge, facet, and cell of $\mathrm{Vor}\, P$. While checking intersections is a computationally expensive task for a general implicit function, it is even more burdensome for the case of a piecewise locally interpolated function as given in our problem. For example, to determine if a certain Voronoi edge intersects Σ more than once, it is necessary to search all voxels containing any subset of Σ which are stabbed by the Voronoi edge, as Voronoi edges may be incident upon many voxels. Matters become worse for Voronoi facets or cells which may touch large regions of the domain.

It is in regards to this difficulty that our approach becomes significant. We exploit the "gridded" nature of the input data set to put $O(1)$ bounds on our intersection calculations. To start, we construct an initial sampling by computing all the points where a grid edge intersects Σ. These points serve as the initial sampling of Σ. However, if we compute the Voronoi diagram of these

E points alone, the Voronoi cells can intersect an arbitrary number of voxels, meaning we will still have trouble verifying and enforcing the closed ball property. To circumvent this problem, we compute a protective layer of grid points near Σ which we denote G. The selection of G traps Voronoi cells of E points into a few voxels. As the algorithm progresses, it adds more points to the existing samples and the nature of the insertion process ensures that the Voronoi cells of these new points are also trapped in a constant number of voxels. We derive bounds on the size of Voronoi cells of the initial samples and the new points for uniform and non-uniform rectilinear gridding (Octtree) in Section 6.3.4.

We now give the pseudocode of the algorithm DelSurfMesh and describe the specifics of the steps subsequently (Figure 99).

DelSurfMesh(Σ)
1 Compute the point set E sampling Σ.
2 Compute the protective layer G of grid points.
3 Compute the Voronoi and Delaunay diagrams of $E \cup G$.
4 Insert new sample points (N) repeatedly until $\mathrm{Vor}\,(E \cup G \cup N)$ satisfies closed ball property.
5 Output the Restricted Delaunay triangulation $\mathrm{Del}\,(E \cup G \cup N)|_{\Sigma}$.

Figure 99: Pseudo-code of the DelSurfMesh algorithm.

We have already described how we choose the initial set of points on (E) and near (G) the surface Σ. The next task is to ensure that the closed ball property holds for the set of points. For a general interpolant within every grid cell, we employ the method given in [69]. For completeness, we briefly describe how the closed ball property can be violated and what measures are to be taken. This process is thus divided into three sub-steps CBP_VE for Voronoi edges, CBP_VF for Voronoi facets, and CBP_VC for Voronoi cells. As we show in Section 6.3.4, one can simplify this process considerably further if the typical trilinear interpolant is used, thereby improving the robustness and efficiency of the algorithm.

- CBP_VE: A Voronoi edge VE violates the closed ball property if it intersects Σ in more than one point. If this occurs, the intersection point which is farthest from the Delaunay triangle dual to VE is inserted into the triangulation.
- CBP_VF: A Voronoi facet VF violates the closed ball property if it intersects Σ in more than one component or if the intersection includes a closed loop in the interior of VF. In either case, the intersection point farthest from the Delaunay edge dual to VF is inserted into the triangulation.
- CBP_VC: A Voronoi cell VC violates the closed ball property if it intersects Σ in more than one component, if the intersection includes an isolated component of Σ inside VC, or if the intersection includes a surface of positive genus with one or more disks removed.

Note, the above properties are to be checked in the order given. Once a violation is detected, a new point is inserted into the existing Delaunay triangulation, the triangulation is updated, and the processes must begin again. Figure 100 shows different situations that can arise in this context.

Mesh Refinement Although the topology of M is now correct, it may be possible that the geometry of M is not approximated sufficiently for an application purpose. Hence, we allow a user input ϵ and refine M as follows. For each Voronoi edge VE that intersects Σ, we compute the unique point $p \in VE \cap \Sigma$ and the circumcenter c of the dual Delaunay face to VE. If the distance

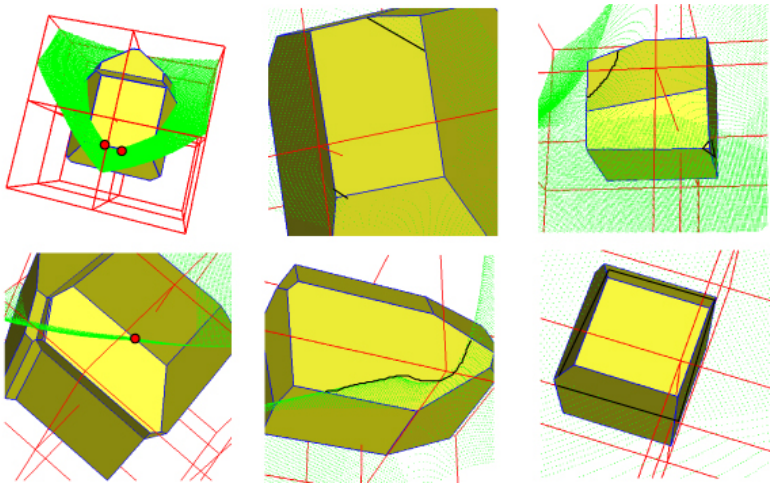


Figure 100: Top row shows three samples scenarios where the closed ball property is violated for a Voronoi Edge (left), Voronoi Facet (middle) and Voronoi Cell (right). The bottom row shows the cases where the closed ball property is satisfied for Voronoi objects of the corresponding dimension (in the top row).

between p and c is more than ϵ, we add p to the vertex set and regenerate the restricted Delaunay mesh. Every Voronoi edge dual to a restricted Delaunay facet is a normal approximation to Σ locally meaning the distance between p and c is an upper bound for the Hausdorff distance from Σ to the restricted Delaunay mesh. Hence, this process will yield a mesh satisfying property II.

In our current implementation, we maintain the 3D Vor/Del diagram of the point set throughout the process. Very recently, it was shown by Dey and Levine that this is not necessary [94]; one can recover the geometry while manipulating only the 2D mesh data structure of the surface, as long as there are enough points sampling a topologically correct approximation of Σ.

Tetrahedral Meshing of Interior/Exterior At this stage, we have an accurate mesh approximation of the level set both topologically and geometrically. Since the mesh is already embedded in a Delaunay mesh that includes some grid points, we already have a tetrahedral mesh of both the interior and exterior of Σ. In order to improve the quality of the mesh elements, we use the algorithm DelVolMesh defined as follows.

Without loss of generality, we describe how DelVolMesh is used to generate a tetrahedral mesh of the interior of Σ. The input to DelVolMesh is $\mathrm{Del}\,(G \cup E \cup N)$, the volumetric mesh generated from DelSurfMesh. Note that $\mathrm{Del}\,(G \cup E \cup N)$ has the output of DelSurf Mesh, $\mathrm{Del}\,(G \cup E \cup N)|_{\Sigma}$, as a subcomplex. We form a set G' of all grid vertices of the original rectilinear scaffolding which have function values less than the isovalue and do not belong to G. These points are distributed through the interior of Σ evenly (or evenly relative to an adaptive gridding) and we add them to the Delaunay mesh of the volume.

Here, our protective layer of grid points is crucially important. As we add the points of G', some triangles of the Delaunay mesh will necessarily change but the Delaunay triangles among surface points (E and N vertices) will be unaffected. This is a direct consequence of the fact that a point of G' is, by construction of G, closer to points of G than to points of $E \cup N$. Therefore, $\mathrm{Del}\,(E \cup N \cup G \cup G')$ will still have M as the restricted Delaunay diagram. By throwing out the points of G exterior to Σ, we are left with a tetrahedral mesh of the volume with good quality tetrahedra and a Delaunay conforming surface mesh. The 2D analogue of DelVolMesh is shown in Figure 98 (d) and (e).

Efficiency Since our algorithm uses the natural structure of the rectilinear input data to construct Voronoi and Delaunay diagrams, we are able to provide two important results that reduce the computational burden. We state and discuss the significance of each one.

Theorem 6.2. *There is an $O(1)$ bound on the number of voxels that a Voronoi cell of an E or N point may intersect. In the case of uniform rectilinear gridding with voxels that are cubes, this bound is four voxels.*

Proof. We prove the following lemma for the simplest case in 2D which will be the crux of the argument for the proof of the more general cases.

Lemma 6.3. *For data given on an equally spaced 2D grid, the Voronoi cell of an E point is bounded within two pixels.*

Proof. Let $e \in E$ lie on the edge ε be between grid points $g_1, g_2 \in G$ and let $VC(e)$ be the (2D) Voronoi cell defined by e. There exist two lines perpendicular to ε, one passing through the midpoint of g_1 and e and one passing through the midpoint of g_2 and e. By definition, $VC(e)$ is necessarily contained between these two lines which bounds $VC(e)$ to a single column of pixels.

Now consider one of of the two pixels containing both g_1 and g_2. Denote its other vertices as g_3 and g_4. Let s denote the side length of a pixel. Then both $||e - g_3|| > s/2$ and $||e - g_4|| < s/2$. For any point f on the edge between g_3 and g_4, observe that $||f - g_3|| \leq s/2$ or $||f - g_4|| \leq s/2$. Therefore, each point on the edge between g_3 and g_4 is closer to one of those grid points than it is to e, and hence cannot belong to $VC(e)$. This bounds $VC(e)$ to the two pixels containing ε, proving the lemma. A picture version of this proof appears in Figure 101a. □ □

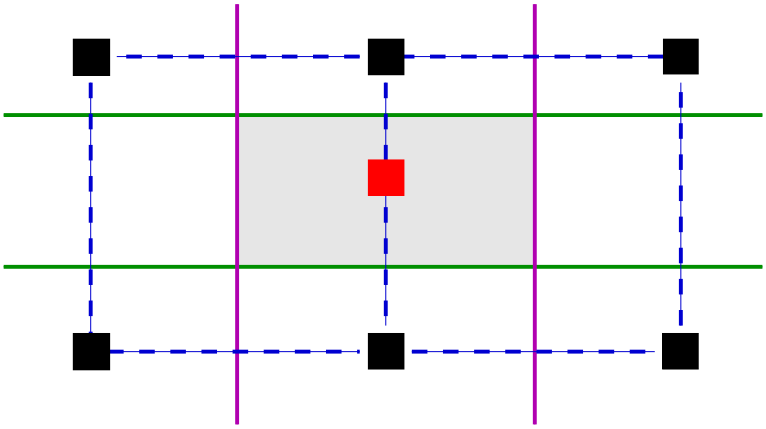


Figure 101: (a) A picture proof of Lemma 6.3. The Voronoi cell of the edge point e must lie in the shaded region bounded by the solid lines (green). By symmetry, this restricts the Voronoi cell to a two pixel range. (b) A picture proof of Theorem 6.2 in the general 2D case. Here, each pixel has dimensions s_1 by s_2 with $\lceil s_1/s_2 \rceil = 3$. By symmetry, the Voronoi cell of e is restricted to a six pixel range. (c) A Voronoi cell of an edge point in 3D, visibly contained within four voxels.

Continuing with the 2D case and the same notation, suppose that each pixel is a rectangle with side lengths s_1 and s_2. Suppose ε has length s_1. As in the proof of the lemma, we can immediately restrict $VC(e)$ to a column of pixels. Consider only the range of $2\lceil s_1/s_2 \rceil$ pixels closest to ε. This range contains the two squares of side length s_1 with ε as one edge. Therefore, we may repeat the analysis in the proof of the lemma using the corners of this range instead of the corners of the pixel. Hence, $VC(e)$ is bounded within $2\lceil s_1/s_2 \rceil$ pixels. Figure 101b shows a simple example.

Now consider the 3D case where each voxel has dimensions s_1, s_2, and s_3 and again assume $e \in E$ lies on the edge ε with side length s_1. There are two planes perpendicular to ε which bounds $VC(e)$ to a flat grid of voxels. In either of the two rectilinear dimensions of this grid, we apply the same analysis as above to conclude that $VC(e)$ intersects at most $2(\lceil s_1/s_2 \rceil + \lceil s_1/s_3 \rceil)$ voxels. Note that this constant reduces to 4 when $s_1 = s_2 = s_3$.

The proof for points of type N is similar. Finally, we note that as we add more E and N points to the diagram, the Voronoi cells can only get smaller. Therefore, the presence of other E or N points in the diagram is irrelevant to the bound. □ □

Theorem 6.2 has significant implications for solving the ray intersection problem discussed in Section 6.3.2. A priori, Voronoi cells may touch an arbitrary number of voxels which cause an explosion in computational time when checking if the closed ball property is satisfied. With Theorem 6.2, the computational time can be bounded in advance.

Notably, the actual bound depends on the gridding scheme of the input, not the choice of interpolant. Accordingly, the width of the protective layer of grid points collected during the algorithm depends on the gridding scheme as well. For the case of adaptive gridding in an octtree construction, the user must require the "level difference" between two adjacent cells to be no more than some fixed number k. Given k, a loose bound on the number of voxels a Voronoi cell of an E or N point may intersect is $4k^2$, as an edge is incident on at most k^2 cells in each of the other two orthogonal direction. This bound may be improved to 4 for E points and $3k + 1$ for N points by considering limiting cases and using the convexity of Voronoi cells.

We have inserted the G points in order to decrease computations required to check the closed ball property, however we have increased the complexity of the Voronoi and Delaunay triangulations themselves. By Theorem 6.4 below, the new edges and facets formed by the addition of these G points do not add any significant burden to the closed ball property confirmation process. Further, these facets and edges are used in the algorithm DelVolMesh described in Section 6.3.4.

Theorem 6.4. *Let VF be a Voronoi facet formed between two grid points and VE a Voronoi edge formed among three grid points at any point in the algorithm. Then $VF \cap \Sigma = \emptyset$ and $VE \cap \Sigma = \emptyset$.*

Proof. It suffices to prove the theorem for E points, since the addition of N points only makes existing Voronoi elements smaller. First we treat the 2D case. Consider a pixel with grid points g_1, g_2, g_3, g_4 and two edge points p and q. (If the pixel has more (four) or fewer (zero) edge points, the theorem is vacuous.) The edge points may occur on adjacent or opposite edges of the pixel, as shown in Figures 102 a and b, respectively. We consider the adjacent case first and suppose that the edges on which p and q lie intersect at g_4. Let R denote the region bounded by the rectangle formed with p and q as opposite corners (the shaded yellow regions of Figure 102). Note that $\Sigma \subset R$ since we have chosen a bilinear interpolant. Hence, it suffices to show that the Voronoi edges formed between g_1, g_2 and between g_1, g_3 do not intersect R. If x is a point on the Voronoi edge between g_1, g_2 then by definition $||x - g_1|| = ||x - g_2||$ and $||x - z|| \geq ||x - g_2||$ for all $z \in G \cup E - \{g_1, g_2\}$. However, if $x \in R$ then $||x - q|| < ||x - g_2||$, based on the labelling of Figure 102a. Hence, the Voronoi edge between g_1, g_2 cannot intersect R. Similarly, the Voronoi edge between g_1, g_3 cannot intersect R, proving this case. The case where p and q lie on opposite edges is similar.

The proof for the 3D case is analagous to the 2D case. It suffices to prove the claim for a closed Voronoi facet between two grid points g_1, g_2 as this includes the relevant Voronoi edges. Consider just one of the voxels into which this facet extends. In each of the rectinear directions away from the g_1, g_2 edge, there exists a closest point of E. Taking these two E points and the g_1, g_2 edge, we uniquely define a box. Using the definitions of the Voronoi diagram construction, we can similarly show that Σ lies outside of the box and the Vornoi facet lies inside it, thereby proving the theorem. □ □

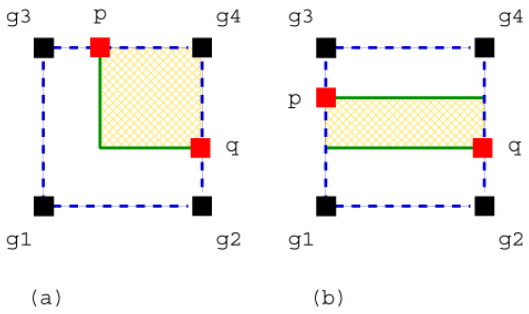


Figure 102: Proof of Theorem 6.4 in the 2D case. We show that Σ lies entirely inside the shaded yellow region, while the Voronoi edges formed between two grid points necessarily lie outside of it.

Simplifying Results Thus far, we have addressed how the inclusion of the grid points G reduces computational overhead. We now examine how particular choices of the interpolant can further ease the calculations. First we consider the process CBP_VF. If $VF \cap \Sigma \neq \emptyset$, then the intersection is a compact 1-manifold by the transversality assumption. All compact 1-manifolds are homeomorphic to a finite collection of line segments and circles and we distinguish between the case of no circles and at least one circle. If no circles exist, the closed ball property is violated if and only if more than two edges of the facet intersect Σ which is easy to check.

If at least one circle occurs in the intersection of Σ and a Voronoi facet, calculations can become more subtle. In the case of trilinear interpolation, these types of intersections do arise in two fashions. First, there exist configurations of relative function values that produce a tunnel topology inside a single voxel, for example, Case 13.5.2 as defined in [158]. In some cases, a Voronoi facet will pass through the entire tunnel, causing an interior loop in its intersection with Σ. In this case, we can detect the topology of the cell by the function values and add the "shoulder points" as described in [158]. The shoulder points are positioned so that the Voronoi diagram breaks these interior loops on Voronoi facets, thereby easing calculations. Alternatively, it may occur that the interior loop of a Voronoi facet passes through multiple voxels, which still provides some difficulties. We conjecture that this phenomenon happens only if the interpolated function is non smooth at a voxel edge or vertex and are working to simplify this problem.

Now we turn to CBP_VC. For this process, use of the trilinear interpolant provides much stronger results to simplify calculations. First, if Σ intersects a Voronoi cell, the intersection manifold must have a boundary component by the following Theorem.

Theorem 6.5. *If each facet of a Voronoi cell VC has empty intersection with Σ and Σ is defined by a trilinear interpolant, then $VC \cap \Sigma = \emptyset$. That is, there cannot be a component of Σ entirely inside a single Voronoi cell.*

Proof. The trilinear interpolation method precludes the existence of isolated surface components within a single voxel. Therefore, every component of Σ has a point of G in its interior and hence is sampled by at least six points of E, corresponding to the six rectilinear directions. Thus, each component of Σ intersects at least six Voronoi cells. A similar proof holds if the data is not given on a rectilinear grid. □ □

Therefore, the process of checking the closed ball property for a Voronoi cell is reduced to checking if the intersection along the facets of a cell is a single closed curve (or empty). A priori, the possibility remains that the intersection surface is a manifold of positive genus with a disc removed. Detection of such cases has been discussed in detail in [69]. However, if our algorithm is applied in the common case of trilinear interpolation with voxels that are cubes, these difficult cases can also be excluded, leading to a much simpler algorithm.

Theorem 6.6. *Let VC be a Voronoi cell whose facets and edges satisfy the closed ball property and let Σ be defined by a trilinear interpolant on a uniform grid with voxels that are cubes. Then the closed ball property is satisfied for VC if and only if $VC \cap \Sigma$ has a single boundary component.*

Proof. It suffices to exclude the minimum case where $VC \cap \Sigma$ is homeomorphic to a torus with a disc removed. To generate such a surface by trilinear interpolation on a grid of cubes requires more than four voxels since the intersection of Σ with a voxel face cannot contain closed loops. By Theorem 6.2, the surface therefore intersects more than one Voronoi cell. □ □

6.3.5 Implementation and Results

We have used CGAL [64] to build and maintain the Voronoi and Delaunay diagram of the point set. We also needed to compute the intersection of a ray with the surface Σ inside certain voxels; for this task, we have used another publicly available library called SYNAPS [195].

As described in the problem statement in Section 6.3.1, only the parameter ϵ is needed to run the algorithm on a given data set. This input dictates the desired upper bound on the Hausdorff distance between the surface approximation M and the interpolated surface Σ. Given this parameter, we reduce the amount of computation needed by snapping some of the points of E in the following manner. If there exist points $g \in G$ and e_0, e_1, $e_2 \in E$ such that the e_i are on grid edges incident to g and $d(g, e_i) < \epsilon$ for $i = 0, 1, 2$, we snap the three e_i points onto the common point g. Since all e_i points are within ϵ distance of the grid point and Σ passes through each e_i, the closet point of Σ to g cannot be more than ϵ away, which is within the user specified limit of tolerance.

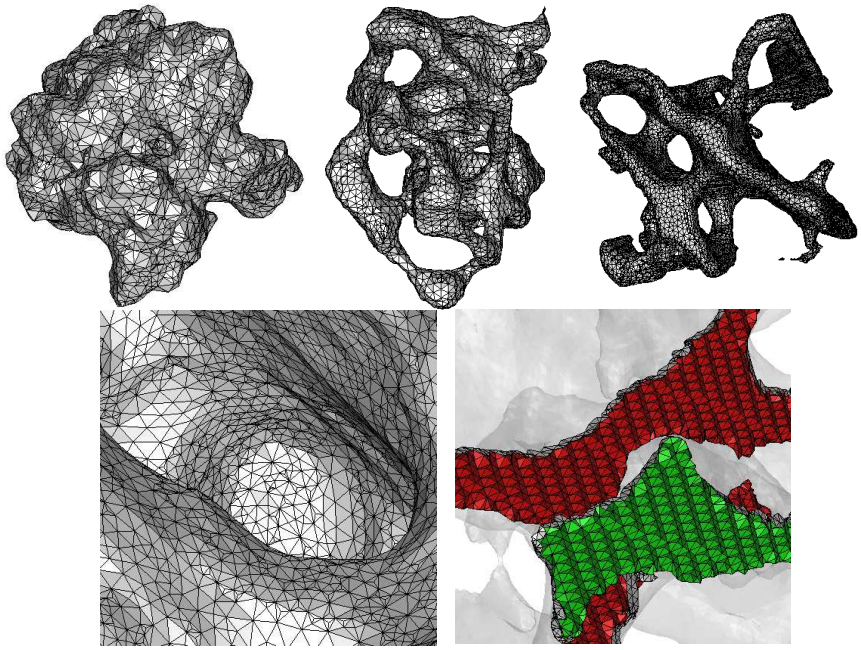


Figure 103: Performance of the meshing algorithm. The top row shows the geometry, surface mesh, a closeup on the surface mesh and a cut-away of the volumetric tetrahedral mesh for 1CID. The second and the third rows show the same for 1MAG and BONE, respectively.

Although we have reduced the complexity of the ray-surface intersection problem as describe in Section 6.3.4, we still have to compute it for a constant number of voxels for every Voronoi edge. This was accomplished by parameterizing the Voronoi edge segment between two Voronoi vertices p_0 and p_1 with a real number t. Inside every candidate voxel, we then detect the intersection points of the segment with the interpolating polynomial function using the Algebraic solver available in

the library SYNAPS. If an intersection point lies inside the corresponding voxel, we consider it as a valid intersection.

The performance of the algorithm on biological entities at various scales is shown in Figure 103.

The 1CID data set was obtained via blurring the coordinates of the atoms available from the Protein Data Bank (PDB) [52]. The resulting 3D scalar volume represents the electron density of the protein molecule T Cell Surface Glycoprotein CD4 . The goal in this case is to extract a well-sampled Delaunay conforming surface mesh so that one can analyze the secondary structure using the unstable manifolds of the index 1 and index 2 saddle points of a suitable distance function [24].

1MAG, shown in the second row of Figure 103, is the PDB entry of the ion channel Gramicidin A, obtained from soil bacteria Bacillus brevis. The molecular surface has many tunnels as shown in the left most sub-figure in the second row. However, only the tunnel in the middle is topologically significant as it is important to preserve the property of the ion channel. Once the isosurface has been extracted, it is therefore necessary to remove all the other tunnels and preserve the main tunnel. The algorithm for such removal of topological features has been described in [23] which relies on a Delaunay conforming isosurface in order to detect and remove unwanted topological features.

Finally, BONE data is shown in the third row of Figure 103. This data set is provided by our collaborator from the University of Rome. The goal here is to mesh the internal bone structure for stress-strain analysis.

The size of the 1CID volume data set is 128^3. It took 1 second to build the initial triangulation, 45 seconds to enforce the closed ball property, and 35 seconds to recover the geometry for $\epsilon = 0.00001$. The size of the 1MAG dataset is 128^3. It took 1.5 seconds to build the initial triangulation, 20 seconds to enforce the closed ball property, and 28 seconds to recover the geometry for $\epsilon = 0.00001$. The size of the BONE dataset is 64^3. It took 5 seconds to build the initial triangulation, 65 seconds to enforce the closed ball property, and 89 seconds to recover the geometry for $\epsilon = 0.0001$. Note, the time taken to mesh does not depend on the size of the volume data set since we were able to enforce that the ray-surface intersection calculation is of constant complexity. However, as the level set passes through more voxels, the number of points $(E \cup G)$ increases and that increases the time complexity of the algorithm.

We have presented an improvement over traditional level set meshing approaches. In particular, we claimed that the isosurfaces extracted via well-known approaches such as marching cubes or dual contouring are not suitable as they not only fail to capture the topology in a provable manner but also do not produce sufficient samples for the extracted approximation to be embedded in the Delaunay triangulation of the sampled set of points. On the other hand, there are elegant approaches for meshing general implicit surfaces, yet these algorithms suffer from the computational overhead of computing the intersection of a ray and the level set. Typically, the level set of a sampled scalar function is approximated via a number of piecewise interpolating functions and therefore finding the intersection of a ray with the level set requires a large number of checks. In light of this, we have presented an algorithm which not only overcomes the sampling issue of traditional isosurfacing techniques but also adopts a well-known provable algorithm [69] for generating a good sample of the isosurface efficiently. We have also shown that for the commonly used trilinear interpolant, many of the difficult cases that arise in implementation of the algorithm can be avoided, thereby improving numerical robustness.

Scope As noted earlier, the scope of this algorithm is immense. First, we have not assumed any particular interpolation scheme in order to prove that we need to check only a few voxels when detecting ray-surface intersection. Therefore, it is possible to extend the algorithm to any higher order interpolant, for example cubic A-patches [19]. Second, adaptive sampling of the scalar function poses no problem to this algorithm as explained previously. Finally, since we exploit

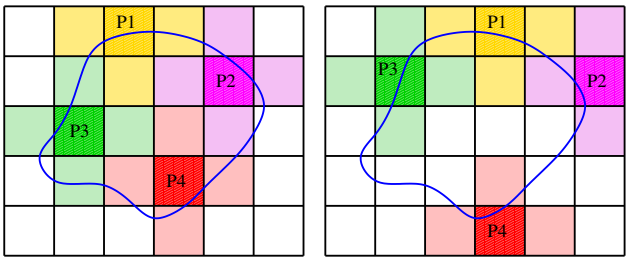


Figure 104: Data may be partitioned to run the algorithm in parallel. In this 2D example, the voxels indicated by the P_i are to be processed. One can send data from each P_i voxel to a separate processor,along with the collar of protective voxels around it indicated by the same color shading. Two iterations are shown.

the local nature of the interpolated surface Σ, the algorithm can easily be parallelized which is especially important for the large data sets often used in the multiscale biological models we have discussed. Figure 104 indicates how a data set might be partitioned to speed up computational time the analogous 2D case. The different colored pixels marked with P_i can be processed concurrently on separate processors as the Voronoi cells of the E and N points in those pixels are guaranteed to be inside the protective collar of surrounding pixels. We note that not all pixels can be processed concurrently as the protective collars must be non-overlapping. However, by iterating the process, the most difficult computations can be done in parallel.

Limitations One limitation of our algorithm is that it tends to generate more samples due to the layer of G points that we use to restrict the search space. Therefore it is necessary to apply adaptive sampling of the scalar function such that bigger voxels are automatically created in regions of less detail and smaller voxels in regions of high detail. By the results of Section 6.3.4, the efficiency of the algorithm remains the same.

Future Work The extracted mesh typically has a "gridding" artifact because of the influence of the grid. Further, sometimes the uniformity of the grid causes it to be over-sampled. Therefore, ideally, one would decimate the grid so that the G vertices lie close to the medial axis after the isovalue is known. We plan to develop a scheme for such an optimal placement of the grid vertices. Also, the efficiency of the core computation requires a scaffolding structure. Hence, once the Delaunay conforming surface mesh is extracted, it is not clear how the surface can be decimated (if needed) by throwing away some of the grid vertices, while still efficiently checking that the resulting (decimated) point sample satisfies the closed ball property.

7 Modeling and Visualizing Functions on Surfaces

7.1 Modeling Scattered Surface Data On Curved Surfaces

We consider the following problem: Given a set of four dimensional points $P = \{(x_i, y_i, z_i, F_i)\}_{i=1}^{M}$ with (x_i, y_i, z_i) on a given smooth surface D, called *domain surface*, construct an interpolation function F, known as *on-surface*, such that $F(x_i, y_i, z_i) = F_i, \quad i = 1, \cdots, M$, and visualize the on-surface, where the domain surface could be a set of expressions and no part of the surface is plane.

The problem of constructing interpolants on physical objects arises in some application areas such as characterizing the rain fall on the earth, the pressure on the wing of an airplane and the temperature on a human body. The problem was first proposed as an open question by Barnhill

[47] in 1985. After that, a considerable number methods have been developed for dealing with it (for surveys see [48], [111]). Most of them interpolate the scattered data over planar or spherical domain surfaces. In [49] and [110], the domain surface are generalized to convex surface and topological genus zero surface, respectively. Recently, Pottmann [189] presented a method which does not possess similar restrictions on the domain surface but requires it to be of C^2. In [50] this restriction was left and the on-surface is constructed by transfinite interpolation. It seems that, the currently know approaches possess restrictions either on domain surfaces or on-surfaces. The domain surfaces are usually assumed to be spherical, convex or genus zero. The on-surface are not always polynomial [50], [172] or rather higher order polynomial [196] or more pieces of surface patch [5] compare with the present approach. The aim of this is to design an polynomial interpolation scheme over a collection of tetrahedra without restrictions mentioned above.

Related to the interpolation over tetrahedra, a well known method is the Clough-Tocher scheme [5] that split a tetrahedron into 4 subtetrahedra and a degree 5 polynomial is built from C^2 data over each subtetrahedron. The function in global is C^1. Another Clough-Tocher scheme[236] that need only C^1 data for constructing C^1 function split the tetrahedron into 12 and a cubic is used over each subtetrahedra. A C^1 scheme [196] that does not split the tetrahedron uses degree 9 polynomial and C^4 data. Going to C^2 scheme, a known result is to use degree 17 polynomial and C^8 data. Comparing with these approaches, our construction has no splitting and lower degrees. The reason for achieving this is that a simplicial hull is a neighborhood of a surface. The solution to our proposed problem involves the following steps:

a. Construct a triangular approximation T of the domain surface D.

b. Generate C^1 or C^2 data at the vertices of the triangulation T.

c. Bulid a simplicial hull $\sum$ over the triangulation T.

d. Modeling the on-surface over $\sum$ by interpolating the C^1 or C^2 data locally.

e. Visualizing the on-surface.

We will not address the first two steps. A algorithm for the construction of the triangulation of the given surface is proposed in [75]. However, we require our triangulation to satisfy certain conditions which will be discussed in Section 3. The C^1 or C^2 data at the vertices can be generated by a technique described in [181]. We shall detail step **c**, **d** and **e** in §3, §4, and §5 respectively, following the notation and preliminary section.

7.1.1 Notation and Preliminary Details

Bernstein-Bezier (BB) Form: Let $p_1,\ p_2,\ p_3,\ p_4 \in \mathbb{R}^3$ be affine independent. Then the tetrahedron with vertices $p_1,\ p_2,\ p_3$, and p_4 is the convex hull defined by $[p_1p_2...p_4] = \{p \in \mathbb{R}^3 : p = \sum_{i=1}^4 \alpha_i p_i, \alpha_i \geq 0, \sum_{i=1}^j \alpha_i = 1\}$. For any $p = \sum_{i=1}^4 \alpha_i p_i \in [p_1p_2...p_4]$, $\alpha = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)^T$ is the barycentric coordinate of p. Any polynomial $f(p)$ of degree n can be expressed as Bernstein-Bezier(BB) form over $[p_1p_2...p_4]$ as

$$f(p) = \sum_{|\lambda|=n} b_\lambda\ B_\lambda^n(\alpha), \quad \lambda \in \mathcal{Z}_+^4 \tag{237}$$

where $B_\lambda^n(\alpha) = \frac{n!}{\lambda_1!\lambda_2!\lambda_3!\lambda_4!}\ \alpha_1^{\lambda_1}\alpha_2^{\lambda_2}\alpha_3^{\lambda_3}\alpha_4^{\lambda_4}$ is Bernstein polynomial, $|\lambda| = \sum_{i=1}^4 \lambda_i$ with $\lambda = (\lambda_1, \lambda_2, \lambda_3, \lambda_4)^T = \sum_{i=1}^4 \lambda_i e_i$, $\alpha = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)^T$ 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.

Lemma 2.1. *Let $f(p) = F(\alpha) = \sum_{|\lambda|=n}\ b_\lambda B_\lambda^n(\alpha)$ with α is the barycentric coordinates of p. Then*

for any given $p^{(1)}$ and $p^{(2)}$, with $\alpha^{(1)}$ and $\alpha^{(2)}$ be their barycentric coordinates, we have

$$\nabla f(p)^T(p^{(1)}-p^{(2)}) = n\sum_{|\lambda|=n-1} b^1_\lambda(\alpha^{(1)}-\alpha^{(2)})B^{n-1}_\lambda(\alpha)$$

$$(p^{(1)}-p^{(2)})^T\nabla^2 f(p)(p^{(1)}-p^{(2)}) = n(n-1)\sum_{|\lambda|=n-2} b^2_\lambda(\alpha^{(1)}-\alpha^{(2)})B^{n-2}_\lambda(\alpha)$$

where $\nabla f(p) = [\frac{\partial f(p)}{\partial x}\ \ \frac{\partial f(p)}{\partial y}\ \ \frac{\partial f(p)}{\partial z}]^T$, $\nabla^2 f(p) = [\nabla\frac{\partial f(p)}{\partial x},\ \nabla\frac{\partial f(p)}{\partial y}\ \ \nabla\frac{\partial f(p)}{\partial z}]$ *and* $b^r_\lambda(\alpha^{(1)}-\alpha^{(2)}) = \sum_{|j|=r} b_{\lambda+j}\ B^r_j(\alpha^{(1)}-\alpha^{(2)})$

See [108] for the two dimensional case of the above lemma. From this lemma we have

Corollary 2.2. *Let* $f(p) = \sum_{|\lambda|=n}\ b_\lambda B^n_\lambda(\alpha)$ *be defined on the tetrahedron* $[p_1p_2p_3p_4]$, *then*

$$b_{(n-1)e_i+e_j} = b_{ne_i} + \frac{1}{n}(p_j-p_i)^T\nabla f(p_i), \quad j=1,2,3,4; \quad j\neq i \tag{238}$$

$$\begin{aligned} b_{(n-2)e_i+e_j+e_k} &= -b_{ne_i} + b_{(n-1)e_i+e_j} + b_{(n-1)e_i+e_k} \\ &+ \tfrac{1}{n(n-1)}(p_j-p_i)^T\nabla^2 f(p_i)(p_k-p_i), \quad j\neq i, k\neq i \end{aligned} \tag{239}$$

The corollary tell us that the weights around a vertex can be computed from the given C^2 data.

Lemma 2.3 ([108]). *Let* $f(p) = \sum_{|\lambda|=n}\ a_\lambda B^n_\lambda(\alpha)$ *and* $g(p) = \sum_{|\lambda|=n}\ b_\lambda B^n_\lambda(\alpha)$ *be two polynomials defined on two tetrahedra* $[p_1p_2p_3p_4]$ *and* $[p'_1p_2p_3p_4]$, *respectively. Then*
(i) f *and* g *are* C^0 *continuous at the common face* $[p_2p_3p_4]$ *if and only if*

$$a_\lambda = b_\lambda, \quad for\ \ any\ \ \lambda = 0\lambda_2\lambda_3\lambda_4, \quad |\lambda| = n \tag{240}$$

(ii) f *and* g *are* C^1 *continuous at the common face* $[p_2p_3p_4]$ *if and only if (240) holds and*

$$b_{1\lambda_2\lambda_3\lambda_4} = \beta_1 a_{1\lambda_2\lambda_3\lambda_4} + \beta_2 a_{0\lambda_2\lambda_3\lambda_4+0100} + \beta_3 a_{0\lambda_2\lambda_3\lambda_4+0010} + \beta_4 a_{0\lambda_2\lambda_3\lambda_4+0001} \tag{241}$$

(iii) f *and* g *are* C^2 *continuous at the common face* $[p_2p_3p_4]$ *if and only if (240)-(241) holds and*

$$\begin{aligned} b_{2\lambda_2\lambda_3\lambda_4} &= \beta_1^2 a_{2\lambda_2\lambda_3\lambda_4} + 2\beta_1\beta_2 a_{0\lambda_2\lambda_3\lambda_4+1100} + 2\beta_1\beta_3 a_{0\lambda_2\lambda_3\lambda_4+1010} + 2\beta_1\beta_4 a_{0\lambda_2\lambda_3\lambda_4+1001} \\ &+ \beta_2^2 a_{0\lambda_2\lambda_3\lambda_4+0200} + 2\beta_2\beta_3 a_{0\lambda_2\lambda_3\lambda_4+0110} + 2\beta_2\beta_4 a_{0\lambda_2\lambda_3\lambda_4+0101} \\ &+ \beta_3^2 a_{0\lambda_2\lambda_3\lambda_4+0020} + 2\beta_3\beta_4 a_{0\lambda_2\lambda_3\lambda_4+0011} + \beta_4^2 a_{0\lambda_2\lambda_3\lambda_4+0002} \end{aligned} \tag{242}$$

where $\beta = (\beta_1,\beta_2,\beta_3,\beta_4)^T$ *are defined by the relation* $p'_1 = \beta_1p_1+\beta_2p_2+\beta_3p_3+\beta_4p_4, \quad |\beta| = 1$.

In Lemma 2.3, if we divide (241) and (242) by β_4^2, then the C^1 and C^2 conditions become

$$a_{0\lambda_2\lambda_3\lambda_4+0001} = \mu_1 a_{1\lambda_2\lambda_3\lambda_4} + \mu_2 b_{1\lambda_2\lambda_3\lambda_4} + \mu_3 a_{1\lambda_2\lambda_3\lambda_4+0100} + \mu_4 a_{1\lambda_2\lambda_3\lambda_4+0010} \tag{243}$$

$$\begin{aligned} &\mu_1(\mu_1 a_{2\lambda_2\lambda_3\lambda_4} + \mu_3 a_{0\lambda_2\lambda_3\lambda_4+1100} + \mu_4 a_{0\lambda_2\lambda_3\lambda_4+1010} - a_{0\lambda_2\lambda_3\lambda_4+1001}) \\ &= \mu_2(\mu_2 b_{2\lambda_2\lambda_3\lambda_4} + \mu_3 b_{0\lambda_2\lambda_3\lambda_4+1100} + \mu_4 b_{0\lambda_2\lambda_3\lambda_4+1010} - b_{0\lambda_2\lambda_3\lambda_4+1001}) \end{aligned} \tag{244}$$

respectively, where $\mu_1 = -\frac{\beta_1}{\beta_4}, \mu_2 = \frac{1}{\beta_4}, \mu_3 = -\frac{\beta_2}{\beta_4}, \mu_4 = -\frac{\beta_3}{\beta_4}$, that is $p_4 = \mu_1p_1+\mu_2p'_1+\mu_3p_2+\mu_4p_3$.

It is not difficult to show from Corollary 2.2 the following

Lemma 2.4. *Let* $f(p)$ *and* $g(p)$ *be defined as Lemma 2.3. If the coefficients of* f *and* g *around the vertices are determined by (238)–(239), then the* C^1 *and* C^2 *conditions (241)–(242) related only to these coefficients are satisfied.*

Degree Elevation. The polynomial $f(p)$ defined in (237) can be written as one of degree $n+1$ (see e.g. [108]). $f(p) = \sum_{|\lambda|=n+1}\ (Eb)_\lambda\ B^{n+1}_\lambda(\alpha), \quad \lambda\in\mathcal{Z}^4_+$ where $(Eb)_\lambda = \frac{1}{n+1}\sum_{i=1}^4 \lambda_i b_{\lambda-e_i}$. We shall use these formulas in approximating a lower degree polynomial.

7.1.2 Simplicial Hull

Let $[p_ip_j]$ 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 $[p_ip_j]$ is *positive convex*. If both the numbers are zero then we say it is *zero convex*. A *Negative 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*. Let $[p_ip_jp_k]$ 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 $[p_ip_jp_k]$ 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 $[p_ip_jp_k]$ are labeled as *non-convex*.

A *simplicial hull* of T, denoted by $\sum$, is a collection of non-degenerated tetrahedra which satisfies:

(1). Each tetrahedron in $\sum$ has either a single edge of T(then it will be called *edge tetrahedron*) or a single face of T(then it will be called *face tetrahedron*). (2). For each face of T there is only one face tetrahedron in $\sum$ if the face is convex, or there are only two face tetrahedra in $\sum$ if the face is non-convex. (3). Two face tetrahedra that share a common edge do not intersect at other part. This condition is referred to as *non-self-intersection* of the the tetrahedra. (4). For each edge there is only one pair common face sharing edge tetrahedra in $\sum$ on one side of T if the edge is convex or only two pair common face sharing edge tetrahedra in $\sum$ each pair on the different side if the edge is non-convex such that the pair blend the two adjacent face tetrahedra in the same side, in case they exist. (5). For each vertex, the tangent plane defined by the normal is contained in the neighboring tetrahedra in the neighborhood of the vertex. This condition is called *tangent plane containment.*

Therefore, a simplicial hull of T is a neighborhood of T that contains the tangent planes at the vertices and no self-intersection. It should be noted that, for the given T there may exist infinity many simplicial hulls or there may no simplicial hull exists. In the following, we describe a scheme for building a simplicial hull.

1. Build 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 passing through the center of the inscribed circle. Then choose points p_4 and/or q_4 off each side of the plane $\langle p_1p_2p_3\rangle$ to be the farthermost intersection points between L and the tangent planes at 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. If F is positive convex, then p_4 is chosen on the side opposite to the direction of the normals, and a single face tetrahedron $[p_1p_2p_3p_4]$ is formed. If F is negative convex, then q_4 is chosen on the same side as the normals and again the single face tetrahedron $[p_1p_2p_3q_4]$ is formed.

2. Build 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 pair 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 $p''_1 = \frac{(1-t)}{2}(p_2 + p_3) + \frac{t}{2}(p'_4 + p_4), \ \ t \geq 1$, so that p''_1 is above $[p_2, p_3]$. Similarly, $q''_1 \in (q_4q'_4)$ or is below (q_4, q'_4), say $q''_1 = \frac{(1-t)}{2}(p_2 + p_3) + \frac{t}{2}(q'_4 + q_4), \ \ t \geq 1$, so that q''_1 is below $[p_2, p_3]$. If the edge $[p_2p_3]$ is positive (or negative) convex, only the first (or second) pair above are needed.

There is one pitfall in step 2 of the construction: If the point p_4 and p'_4(or q_4 and q'_4) have to be chosen far away from the face centers due to the tangent containment condition, the face tetrahedra may intersect each other. However, if the triangulation is locally even, then we can get rid of the possible trouble. A triangulation of a smooth surface is said to be *locally even* if for every face, say $[p_0p_1p_2]$, each angle defined by the normal at the vertex and the normal of the face $[p_0p_1p_2]$ is less than $\tan^{-1}(\frac{2s\tan(\frac{1}{2}\min\{\alpha_0,\alpha_1,\alpha_2\})}{\| \ \|p_j-p_i\|(p_k-p_i)+\|p_k-p_i\|(p_j-p_i) \ \|})$, where s is the area of the face $[p_0p_1p_2]$, α_i is

the dihedral angle of the edge of the face $[p_0p_1p_2]$ and $0 \le i, j, k \le 2$ are distinct. This condition guarantees that the face tetrahedron constructed has height(the distance between the top vertex p_4 or q_4 to the face) at most $r\tan(\frac{1}{2}\min\{\alpha_0, \alpha_1, \alpha_2\})$, where r is the radius of the inscribed circle. Hence the dihedral angles at the bottom edges of the tetrahedron are less than $\frac{1}{2}\min\{\alpha_0, \alpha_1, \alpha_2\}$. Therefore, there is no intersection between two face tetrahedra. This fact is summarized as

Theorem 3.1. *If the triangulation T is locally even, then we can build a simplicial hull without self-intersection.*

It should be noted that if a triangulation of the domain surface is not locally even, we can always modify it to obtain the required triangulation by inserting points.

In the following we shall assume the triangulation is locally even and a simplicial hull is built. We further assume that each edge of the triangulation is convex and any two adjacent faces are not coplanar. Again, this can be achieved by inserting point in the triangulation.

7.1.3 C^1/C^2 Interpolation by Cubic/Quintic

Suppose we have established a simplicial hull $\sum$ for the given triangulation T. Now we construct a C^1/C^2 function f over $\sum$ such that f has the given C^1/C^2 data at each vertex. Let $V_1 = [p_1p_2p_3p_4]$, $V_2 = [p_1'p_2p_3p_4']$, $W_1 = [p_1''p_2p_3p_4]$, $W_2 = [p_1''p_2p_3p_4']$, $V_1' = [p_1p_2p_3q_4]$, $V_2' = [p_1'p_2p_3q_4']$ and the cubic/quintic polynomials f_i over V_i, g_i over W_i and f_i' over V_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

$$\begin{array}{lcll} p_1'' & = & \beta_1^1 p_1 \ + \beta_2^1 p_2 + \beta_3^1 p_3 + \beta_4^1 p_4, & \beta_1^1 + \beta_2^1 + \beta_3^1 + \beta_4^1 = 1 \\ p_1'' & = & \beta_1^2 p_1' + \beta_2^2 p_2 + \beta_3^2 p_3 + \beta_4^2 p_4', & \beta_1^2 + \beta_2^2 + \beta_3^2 + \beta_4^2 = 1 \\ p_1'' & = & \mu_1 p_4 + \mu_2 p_4' + \mu_3 p_2 + \mu_4 p_3, & \mu_1 + \mu_2 + \mu_3 + \mu_4 = 1 \end{array} \tag{245}$$

C^1 **Cubic Scheme** (1). The number 0 weights are given by the function values at the vertices. (2). The number 1 weights are determined by formula (238) from C^1 data. (3). The number 2 weights, that is $a_{1110}^{(i)}$, are free. (4). The number 3 weights are determined by C^1 conditions (241) and (243). (5). The number 4 weights are free. (6). The number 5 weights are determined by C^1 conditions (241). (7). The number 6 weights are free. (8). The number 7 weights are determined by C^1 conditions (243).

The remaining weights with index $\lambda_1\lambda_2\lambda_3\lambda_4$ are determined by C^1 condition (241) for $\lambda_4 \le 1$ and freely chosen for $\lambda_4 > 1$.

C^2 **Quintic Scheme** (1). The number 0 weights are given by the function values at the vertices. For examples, $a_{5e_i}^{(1)} = f(p_i)$, $i = 1, 2, 3$. (2). The number 1 weights are determined by formula (238). (3). The number 2 weights are determined by formula (239). (4). The number 3 weights, that is $a_{1220}^{(i)}, a_{2210}^{(i)}$ and $a_{2120}^{(i)}$, are free. (5). The number 4 weights are determined by C^1 conditions (241), that is $b_{1220}^{(i)} = \beta_1^{(i)}a_{1220}^{(i)} + \beta_2^{(i)}a_{0320}^{(i)} + \beta_3^{(i)}a_{0230}^{(i)} + \beta_4^{(i)}a_{0221}^{(i)}$, $i = 1, 2$, $b_{1220}^{(i)} = \mu_1 a_{0221}^{(1)} + \mu_2 a_{0221}^{(2)} + \mu_3 a_{0320}^{(i)} + \mu_4 a_{0230}^{(i)}$ It follows from these equations that

$$\mu_1 a_{0221}^{(1)} + \mu_2 a_{0221}^{(2)} - \beta_4^{(i)} a_{0221}^{(i)} = \beta_1^{(i)} a_{1220}^{(i)} + (\beta_2^{(i)} - \mu_3) a_{0320}^{(i)} + (\beta_3^{(i)} - \mu_4) a_{0230}^{(i)} \tag{246}$$

for $i = 1, 2$. The coefficient matrix A of (246) for the unknowns $a_{0221}^{(i)}$ is $A = \begin{bmatrix} \mu_1 - \beta_4^{(1)} & \mu_2 \\ \mu_1 & \mu_2 - \beta_4^{(2)} \end{bmatrix}$ This matrix is nonsingular if and only if p_1, p_1', p_2 and p_3 are not coplanar(see the Appendix of [17]). Hence (246) has unique solution under our assumptions. (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 to the vertex p_2. Suppose there are k

triangles(hence k edges) around p_2 and there are r times convexity change of the edges, then by C^1 and C^2 conditions, we have $6k+r$ equations That is, crossing each face, we have two equations, and cross each triangle, we have one equation. The number of related unknowns is $6k+r$ also. That is, k number 5 weights and $5k$ number 6 weights and one more unknown is related when one time convexity change of the edges occur. Now we investigate these equations. It follows from (241) and (242) that

$$b_{1211}^{(i)} = \beta_1^{(i)} a_{1211}^{(i)} + \beta_2^{(i)} a_{0311}^{(i)} + \beta_3^{(i)} a_{0221}^{(i)} + \beta_4^{(i)} a_{0212}^{(i)} \tag{247}$$

$$\begin{aligned} b_{2210}^{(i)} &= \beta_1^{(i)}\beta_1^{(i)} a_{2210}^{(i)} + 2\beta_1^{(i)}\beta_2^{(i)} a_{1310}^{(i)} + 2\beta_1^{(i)}\beta_3^{(i)} a_{1220}^{(i)} + 2\beta_1^{(i)}\beta_4^{(i)} a_{1211}^{(i)} \\ &+ \beta_2^{(i)}\beta_2^{(i)} a_{0410}^{(i)} + 2\beta_2^{(i)}\beta_3^{(i)} a_{0320}^{(i)} + 2\beta_2^{(i)}\beta_4^{(i)} a_{0311}^{(i)} \\ &+ \beta_3^{(i)}\beta_3^{(i)} a_{0230}^{(i)} + 2\beta_3^{(i)}\beta_4^{(i)} a_{0221}^{(i)} + \beta_4^{(i)}\beta_4^{(i)} a_{0212}^{(i)} \end{aligned} \tag{248}$$

for $i = 1, 2$. (248) can be written briefly as

$$b_{2210}^{(i)} = 2\beta_1^{(i)}\beta_4^{(i)} a_{1211}^{(i)} + \beta_4^{(i)}\beta_4^{(i)} a_{0212}^{(i)} + \gamma \tag{249}$$

where γ is the known terms in (248). Since (see (243) and (244))

$$b_{2210}^{(1)} = \mu_1 b_{1211}^{(1)} + \mu_2 b_{1211}^{(2)} + \cdots \tag{250}$$

$$\mu_1^2 b_{0212}^{(1)} - \mu_1 b_{1211}^{(1)} = \mu_2^2 b_{0212}^{(2)} - \mu_2 b_{1211}^{(2)} + \cdots \tag{251}$$

where $\cdots$ are known terms, then by substituting (247) into (250) and (251) and then eliminating $b_{2210}^{(i)}$ from (249) and (250) we get three equations related to four unknowns which could be written as:

$$\begin{aligned} &\begin{bmatrix} \beta_4^{(1)} - \mu_1 & -\mu_2 \\ -\mu_1 & \beta_4^{(2)} - \mu_2 \end{bmatrix} \begin{bmatrix} \beta_4^{(1)} & 0 \\ 0 & \beta_4^{(2)} \end{bmatrix} \begin{bmatrix} a_{0212}^{(1)} \\ a_{0212}^{(2)} \end{bmatrix} \\ &= -\begin{bmatrix} 2\beta_4^{(1)} - \mu_1 & -\mu_2 \\ -\mu_1 & 2\beta_4^{(2)} - \mu_2 \end{bmatrix} \begin{bmatrix} \beta_1^{(1)} & 0 \\ 0 & \beta_1^{(2)} \end{bmatrix} \begin{bmatrix} a_{1211}^{(1)} \\ a_{1211}^{(2)} \end{bmatrix} + \cdots \end{aligned} \tag{252}$$

$$\begin{bmatrix} -\mu_1(\beta_4^{(1)} - \mu_1) & \mu_2(\beta_4^{(2)} - \mu_2) \end{bmatrix} \begin{bmatrix} a_{0212}^{(1)} \\ a_{0212}^{(2)} \end{bmatrix} - \begin{bmatrix} \mu_1\beta_1^{(1)}, -\mu_2\beta_1^{(2)} \end{bmatrix} \begin{bmatrix} a_{1211}^{(1)} \\ a_{1211}^{(2)} \end{bmatrix} = \cdots \tag{253}$$

Since the coefficient matrix of (252) is nonsingular, by solving $[a_{0212}^{(1)}\ a_{0212}^{(2)}]^T$ from (252) and then substituting it into (253), we get one equation relating to the unknowns $a_{1211}^{(1)}$, $a_{1211}^{(2)}$. Let the equation be in the form

$$\phi a_{1211}^{(1)} + \psi a_{1211}^{(2)} = \omega \tag{254}$$

Therefore, these unknowns form a closed chain around the vertex p_2. But one should note that the chain will go to other side of the triangles if the edges change their convexity(from positive convex to negative convex or from negative convex to positive convex). For example, if the edge $[p_2p_3]$ is positive convex, while the edge $[p_1p_2]$ is negative convex, then the chain is changed by add one more equation related to the two unknowns $a_{1211}^{(1)}$ and $c_{1211}^{(1)}$ from C^1 condition $c_{1211}^{(1)} = \alpha_1 a_{2210}^{(1)} + \alpha_2 a_{1310}^{(1)} + \alpha_3 a_{1220}^{(1)} + \alpha_4 a_{1211}^{(1)}$, where $q_4 = \alpha_1 p_1 + \alpha_2 p_2 + \alpha_3 p_3 + \alpha_4 p_4$ Again, this equation is in the same form as (254). The coefficient matrix of all these equations related to the vertex p_2 is in the form of

$$A = \begin{bmatrix} \phi_1 & \psi_1 & & \\ & \phi_2 & \psi_2 & \\ & & \ddots & \\ \psi_{k+r} & & & \phi_{k+r} \end{bmatrix}$$

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. In this specified case, $a^{(1)}_{1202}$ and $a^{(1)}_{2102}$ do not involve in any equation, since there is no neighbor tetrahedron. These two weights are defined by C^2 condition of crossing the face $[p_1p_2p_3]$.

(7). The number 7 weights are similarly determined as that of number 6. (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 $6k$. That is for $i=1,2$

$$b^{(i)}_{1202}=\beta^{(i)}_1a^{(i)}_{1202}+\beta^{(i)}_2a^{(i)}_{0302}+\beta^{(i)}_3a^{(i)}_{0212}+\beta^{(i)}_4a^{(i)}_{0203} \tag{255}$$

$$\begin{aligned} b^{(i)}_{2201} &= \beta^{(i)}_1\beta^{(i)}_1a^{(i)}_{2201}+2\beta^{(i)}_1\beta^{(i)}_2a^{(i)}_{1301}+2\beta^{(i)}_1\beta^{(i)}_3a^{(i)}_{1211}+2\beta^{(i)}_1\beta^{(i)}_4a^{(i)}_{1202}\\ &+ \beta^{(i)}_2\beta^{(i)}_2a^{(i)}_{0401}+2\beta^{(i)}_2\beta^{(i)}_3a^{(i)}_{0311}+2\beta^{(i)}_2\beta^{(i)}_4a^{(i)}_{0302}\\ &+ \beta^{(i)}_3\beta^{(i)}_3a^{(i)}_{0221}+2\beta^{(i)}_3\beta^{(i)}_4a^{(i)}_{0212}+\beta^{(i)}_4\beta^{(i)}_4a^{(i)}_{0203} \end{aligned} \tag{256}$$

and

$$b^{(1)}_{3200}=\mu_1b^{(1)}_{2201}+\mu_2b^{(2)}_{2201}+\cdots \tag{257}$$

$$\mu_1^2b^{(1)}_{1202}-\mu_1b^{(1)}_{2201}=\mu_2^2b^{(2)}_{1202}-\mu_2b^{(2)}_{2201}+\cdots \tag{258}$$

Substitute (255) and (256) into (258), 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 that in the same form as (254). Then a system like (254) need to solve. However, if the surrounding tetrahedra at the same side at p_2 are not closed, the matrix A is in the form of $A=\begin{bmatrix}\phi_1&\psi_1&&\\&\ddots&\ddots&\\&&\phi_k&\psi_k\end{bmatrix}$ which can be changed to $A=\begin{bmatrix}A_1&0\\0&A_2\end{bmatrix}$ if one unknown, say the l-th is chosen to be a free parameter. Hence the system of equations is break into two sub-systems. Each of them can be solved easily.

(10). For the number 10 weights, we have six equations parallel to the equations (256)–(258) with all the index changed by the rule

$$\text{The index of the number 10 weright} = \text{The index of the number 9 weright} - e_2 + e_3 \tag{259}$$

and seven independent weights. By chosing one of them, say $b^{(i)}_{3110}$, to be a free parameter, the system can be solved. (11). The number 11 weights are determined in the same way as the that of number 9. (12). The number 12 and 13 weights are free, the number 14 are determined by C^1 and C^2 conditions. That is $b^{(i)}_{1103}$ are defined by (241). $b^{(i)}_{2102}$ are defined by (242). For $b^{(i)}_{3101}$, we have by (243) and (244) $\mu_1b^{(1)}_{3101}+\mu_2b^{(2)}_{3101}=b^{(1)}_{4100}+\gamma$, $-\mu_1b^{(1)}_{3101}+\mu_2b^{(2)}_{3101}=\mu_2^2b^{(2)}_{2102}-\mu_1^2b^{(2)}_{2102}+\delta$, where γ and δ are known terms. Hence $b^{(1)}_{3101}=\frac{b^{(1)}_{4100}-\mu_2^2b^{(2)}_{2102}+\mu_1^2b^{(2)}_{2102}+\gamma-\delta}{2\mu_1}$, $b^{(2)}_{3101}=\frac{b^{(1)}_{4100}+\mu_2^2b^{(2)}_{2102}-\mu_1^2b^{(2)}_{2102}+\gamma+\delta}{2\mu_1}$

(13). The number 15 weights are similar to that of number 14, the index is changed by the rule (259).

(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 (241) and (242) of crossing the face for $\lambda_4\le 2$ and freely chosen for $\lambda_4>2$

In summary, the construction step 1–14 is according to the C^1 and C^2 conditions across the common tetrahedra faces that are over or below the original triangulation. The 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 function is global C^2 continuous.

The Use of Free Weights In both of the C^1 and C^2 schemes described above, there are some free weights under determined. These parameters can be used to control the shape of the on-surface to get the required smoothness and desirable shaped surface. We now propose three approaches or their combinations to use them. The first is to control interactively the shape by adjusting the free weights. The second is to interpolate the on-surface data in the tetrahedron considered. However, if the free weights considered have to be determined before others, then the first or the third approach should be used for them. The third approach is to approximate a lower degree polynomial in the least square sense. For example, by the degree elevation formula, the number 2 weights of the C^1 scheme can be determined by

$$a^{(i)}_{1110} = \frac{1}{4}(a^{(i)}_{1200} + a^{(i)}_{2100} + a^{(i)}_{2010} + a^{(i)}_{1020} + a^{(i)}_{0210} + a^{(i)}_{0120}) - \frac{1}{6}(a^{(i)}_{3000} + a^{(i)}_{0300} + a^{(i)}_{0030})$$

In the way, the number 4 weights of the C^1 scheme are

$$a^{(i)}_{0003} = \frac{1}{3}[2(q^{(i)}_{0101} + q^{(i)}_{1001} + q^{(i)}_{0011}) - (a^{(i)}_{0300} + a^{(i)}_{3000} + a^{(i)}_{0030})]$$

$$a^{(i)}_{0102} = \frac{1}{3}(2q^{(i)}_{0101} + a^{(i)}_{0003}), \quad a^{(i)}_{1002} = \frac{1}{3}(2q^{(i)}_{1001} + a^{(i)}_{0003}), \quad a^{(i)}_{0012} = \frac{1}{3}(2q^{(i)}_{0011} + a^{(i)}_{0003})$$

where

$$q^{(i)}_{0101} = \frac{3}{4}(a^{(i)}_{1101} - a^{(i)}_{1011} + a^{(i)}_{0111} + a^{(i)}_{0201}) - \frac{1}{4}(q^{(i)}_{1100} - q^{(i)}_{1010} + q^{(i)}_{0110} + a^{(i)}_{0300})$$

$$q^{(i)}_{1001} = \frac{3}{4}(a^{(i)}_{1101} + a^{(i)}_{1011} - a^{(i)}_{0111} + a^{(i)}_{2001}) - \frac{1}{4}(q^{(i)}_{1100} + q^{(i)}_{1010} - q^{(i)}_{0110} + a^{(i)}_{3000})$$

$$q^{(i)}_{0011} = \frac{3}{4}(-a^{(i)}_{1101} + a^{(i)}_{1011} + a^{(i)}_{0111} + a^{(i)}_{0021}) - \frac{1}{4}(-q^{(i)}_{1100} + q^{(i)}_{1010} + q^{(i)}_{0110} + a^{(i)}_{0030})$$

$$q^{(i)}_{1100} = \frac{1}{4}(3a^{(i)}_{1200} + 3a^{(i)}_{2100} - a^{(i)}_{0300} - a^{(i)}_{3000})$$

$$q^{(i)}_{1010} = \frac{1}{4}(3a^{(i)}_{2010} + 3a^{(i)}_{1020} - a^{(i)}_{0030} - a^{(i)}_{3000})$$

$$q^{(i)}_{0110} = \frac{1}{4}(3a^{(i)}_{0210} + 3a^{(i)}_{0120} - a^{(i)}_{0300} - a^{(i)}_{0030})$$

7.1.4 Visualization and Examples

We can visualize the graph of the interpolation function on the domain surface D either by drawing the contours on the surface D or by drawing the graph of the interpolation function F.

Plots the contours We display the contours on the domain surface by showing different colors in the region between two contours. In our approach, we achieve this by first generating triangle approximation of the domain surface, and then generating the corresponding four dimensional triangles on F, and finally intersecting these triangles with iso-value to get the line segments of the contours. Let w be a given iso-value, $[p_1p_2p_3]$ be a triangle on D. WLG, we may assume $F(p_1) \le F(p_2) \le F(p_3)$. Then if $w < F(p_1)$ or $w > F(p_3)$, the triangle does not intersect the iso-value. If $w \in [F(p_1), F(p_3)]$, say $w \in [F(p_1), F(p_2)]$, let $t_1 = \frac{w-F(p_1)}{F(p_2)-F(p_1)}$, $\quad t_2 = \frac{w-F(p_1)}{F(p_3)-F(p_1)}$, $q_1 = t_1p_1 + (1-t_1)p_2$, $\quad q_2 = t_1p_1 + (1-t_2)p_3$, then $[q_1q_2]$ is one segment of the contour $F(p) = w$. The collection of all of these line segments form a piecewise approximations to the contours. By increasing the resolution of the triangulation of the domain surface, we can get better approximation of the contours. Figure 5.1 and 5.2 show this feature of an engine.

Plot the on-surface Since the contours may not clearly indicate the geometric shape of the on-surface, one often plot the on-surface in one way or another. One approach is to use a radial projection from the center of the domain. But if the domain surface is not convex or has no-zero genus. This method have difficulties in plotting the surface. Another more natural way is to use the normal projection, that is, project the point p on the domain surface D to a distance proportional to $F(p)$ in the normal direction of D at p:

$$G(p) = p + L\frac{\nabla f(p)(F(p) - F_{min})}{\|\nabla f(p)\|(F_{max} - F_{min})}$$

where L is a positive scaler, F_{min} and F_{max} are minimum and maximum values of F on D. However if L is not given properly, then the surface G may self-intersect in case of the domain surface is not convex. To avoid this happen, we can take $L < \gamma = \min_{p\in D}\{R_{min}(p)\}$, where $R_{min}(p)$ is the minimal principal curvature radius at p of surface D. Since D is piecewise C^1/C^2 continuous surface and it is C^∞ within each tetrahedron, hence $\gamma > 0$. Therefore the main task is to compute γ. In general, it is not easy to compute the exact value of γ, but an approximate γ can serve our purpose as well. When we produce the triangulation of each surface patch on D, we can compute $R_{min}(p)$ for vertices p. And then take $\gamma = \min_{p\in D}\{R_{min}(p) : p \text{ is a vertex}\}$. By implicit function theorem, we could derive the formula for principal curvature radius. For an implicit surface $f(x_1, x_2, x_3) = 0$, define $f_i = \frac{\partial f}{\partial x_i}$, $f_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$. Then the principal curvature radius are the absolute values of the roots of the equation $ax^2 + bx + c = 0$, where $a = \sum_{i=1}^3 f_i^2(f_{i+1,i+1}f_{i+2,i+2} - f_{i+1,i+2}^2) + 2\sum_{i\neq j,k\neq i,k\neq j} f_i f_j(f_{ik}f_{jk} - f_{ij}f_{kk})$, $b = \|\nabla f\|(\sum_{i=1}^3 f_i^2(f_{i+1,i+1} + f_{i+2,i+2}) - 2\sum_{i\neq j} f_i f_j f_{ij})$, $c = \|\nabla f\|^4$, where the indices should mod by 3.

Another approach to determine L is to take

$$L < \delta = \min_{p\in T_d}\left\{\min_q\left\{\frac{\|q-p\|^2\|\nabla f(p)\|}{2(q-p)^T\nabla f(p)} : [p,q] \in T_d \text{ and } (q-p)^T\nabla f(p) > 0\right\}\right\}$$

where T_d is final triangulation of D for display the surface D and F, $p \in T_d$ means p is a vertex and $[p, q] \in T_d$ means $[p, q]$ is an edge. It should be noted that if D is convex(i.e., $(q-p)^T\nabla f(p) \leq 0$) or planar(i.e., $(q-p)^T\nabla f(p) = 0$), δ is arbitrary.

7.2 Algebraic Spline Molecular Surfaces

The computation of electrostatic solvation energy (also known as polarization energy) for biomolecules plays an important role in the molecular dynamics simulation [137], the analysis of stability in protein structure prediction [218], and the protein-ligand binding energy calculation [143]. The explicit model of the solvent provides the most rigorous solvation energy calculation [175]. However, due to the large amount of solvent molecules, most of the computation time is spent on the trajectories of the solvent molecules, which severely increases the computation cost of this method [198]. An alternative method is to represent the solvent implicitly as a dielectric continuum [204], then the electrostatic potential is known by solving the Poisson-Boltzmann (PB) equations [45][161]. A more efficient method is to approximate the PB electrostatic solvation energy by the generalized Born (GB) model [220][51][151], which computes the electrostatic solvation energy ΔG_{elec} as

$$G_{\text{pol}} = -\frac{\tau}{2}\sum_{i,j}\frac{q_i q_j}{[r_{ij}^2 + R_i R_j \exp(-\frac{r_{ij}^2}{F R_i R_j})]^{\frac{1}{2}}}, \tag{260}$$

where $\tau = \frac{1}{\varepsilon_p} - \frac{1}{\varepsilon_w}$, ε_p is the solute (low) dielectric constant, ε_w is the solvent (high) dielectric constant, q_i is the atomic charge of atom i, r_{ij} is the distance between atom i and j, F is an empirical factor (could be 4 [220] or 8 [151]), and R_i is the effective Born radius of atom i. The effective

Born radius reflects how deep an atom is buried in the molecule and consequently determines the importance to the polarization. The formulation of the effective Born radii is derived in [117]:

$$R_i^{-1} = \frac{1}{4\pi} \int_\Gamma \frac{(\mathbf{r} - \mathbf{x}_i) \cdot \mathbf{n}(\mathbf{r})}{|\mathbf{r} - \mathbf{x}_i|^4} \, dS, \tag{261}$$

where Γ is the molecular surface of the solute, $\mathbf{x}_i$ is the center of atom i, and $\mathbf{n}(\mathbf{r})$ is the unit normal of the surface at $\mathbf{r}$. The details of the derivation of (261) and a fast evaluation algorithm based on the fast Fourier transform (FFT) for (261) is discussed in [34]. Since the numerical integrations are done on the molecular surface Γ, an accurate and analytic representation of Γ is needed.

Figure 105: Three molecular surfaces are shown for two atoms in two dimension. The boundary of the union of balls (pink) with the van der Waals radii is the VWS. The SAS (purple) is the union of augmented van der Waals spheres with each radius enlarged by the radius of a solvent probe (light blue). The SES (the blue curve) is boundary of all possible solvent probes that do not intersect with the interior of the VWS.

Three well-known molecular surfaces are shown in Figure 105 in 2D. The van der Waals surface (VWS) is the union of a set of spheres with atomic van der Waals radii. The solvent accessible surface (SAS) is the union of augmented van der Waals spheres with each radius enlarged by the solvent probe radius (normally taken as 1.4 $\mathbf{r}A$) [148]. The solvent excluded surface (SES, also called molecular surface or Connolly surface) is the boundary of the union of all possible solvent probes that do not intersect with the interior of the VWS [78][197]. As described in [78], the SES consists of the convex spherical patches which are parts of the VWS as well, the toroidal pathces and the concave spherical patches, which are generated by the probes rolling along the intersections of neighboring atoms. The VWS causes an overestimation of the electrostatic solvation energy, while the SAS leads to an underestimation [151]. The SES is the most accurate when it is applied in the energetic calculation and therefore it is most often used to model the molecular surface. However the SES still has one significant drawback: it contains cusps when the rolling probe self-intersects, which may cause singularity in the Born radii and the force calculations.

In the energetic computation, knowing the patch complexes of the molecular surface is not enough. For convenience, an analytical representation of the molecular surface is needed and the singularity should also be avoided. One way to generate such a model is to define an analytical volumetric density function, for example, the summation of Gaussian functions [120], Fermi-Dirac switching function [152], or piecewise polynomials [151], and approximate the SES by an iso-contour of the density function. Techniques of fast extracting an iso-contour of smooth kernel functions are developed in [16][33]. However the error of the generated isosurface could be large and result in inaccurate energy computation. A NURBS representation for the SES is presented in [30]. Although it provides a parametric approximation to the SES, it does not solve the singularity problem. Edelsbrunner [99] defines another paradigm of a smooth surface referred to as *skin* which is based on the Voronoi, Delaunay, and Alpha complexes of a finite set of weighed points. The *skin* model has good geometric properties such as it is free of singularity and it can be decomposed into a collection of quadratic patches. Triangulation schemes based on the *skin* model are provided in [67][68]. However when applied to the energetic computation, the *skin* triangulation which in fact is a linear approximation to the SES has to be very dense to gain accuracy, which causes oversampling on the surface and hence makes the computation very slow. Therefore it still remains a challenge to generate a model for the molecular surface which is accurate, smooth, and computable.

The main contribution of this work is to provide a method to model the SES as piecewise algebraic spline patches with certain continuity at the boundary of the patches. Each patch has dual

implicit and parametric representations. Hence high order implicit surfaces can be parameterized onto a planer domain and therefore higher order quadrature rules of 2D such as the Gaussian quadrature rules can be easily applied to the energetic computation. Moreover, because higher order spline patches are used to approximate the SES, fewer number of triangles are needed to obtain the same accuracy in the energetic computation as the linear model. The algebraic spline patches are generated based on the prism scaffold built surrounding the original triangular mesh of the SES and are defined implicitly by simple BB spline functions. Previous work on constructing piecewise spline patches within a simplical hull over a triangular mesh includes generating quadric patches [86], cubic patches [123][88], and nonsingular and single sheeted cubic patches [20] in a tetrahedra scaffold. We also show that the so generated algebraic spline patches are error bounded and free of singularity under certain conditions.

7.2.1 Algebraic spline model

Algorithm Sketch There are four main steps in our ASMS construction algorithm: (1) construct an initial triangular mesh of the SES; (2) build a prism scaffold surrounding the triangulation; (3) define a piecewise polynomial with certain continuity; (4) extract the 0-contour of the piecewise polynomial. We are going the explain each step in detail in the following and discuss how to make use the parametrization of the ASMS in the numerical integration.

Initial triangulation of the MS So far a lot of work has been done on the triangulation of the SES or its approximation [68][4][147][244][32]. The ASMS generation could be applied to any of these triangulations. In our current research we use the triangulation generated by a program in the software TexMol [32][21] as the initial. Features of the molecular surface are well preserved in this triangulation. We then decimate the mesh [42] to obtain a coarser one.

Implicit/parametric patches generation Given the triangulation mesh T, let $[\mathbf{v}_i\mathbf{v}_j\mathbf{v}_k]$ be one of the triangles where $\mathbf{v}_i,\ \mathbf{v}_j,\ \mathbf{v}_k$ are the vertices of the triangle. Suppose the unit normals of the surface at the vertices are also known, denoted as $\mathbf{n}_l,\ (l = 1, j, k)$. Let $\mathbf{v}_l(\lambda) = \mathbf{v}_l + \lambda\mathbf{n}_l$. First we define a prism (Figure 106) $D_{ijk} := \{\mathbf{p} :\ \mathbf{p} = b_1\mathbf{v}_i(\lambda) + b_2\mathbf{v}_j(\lambda) + b_3\mathbf{v}_k(\lambda),\ \lambda \in I_{ijk}\}$, where (b_1, b_2, b_3) are the barycentric coordinates of points in $[\mathbf{v}_i\mathbf{v}_j\mathbf{v}_k]$, and I_{ijk} is a maximal open interval containing 0 and for any $\lambda \in I_{ijk}$, $\mathbf{v}_i(\lambda)$, $\mathbf{v}_j(\lambda)$, $\mathbf{v}_k(\lambda)$ are not collinear and $\mathbf{n}_i$, $\mathbf{n}_j$, $\mathbf{n}_k$ point to the same side of the plane $P_{ijk}(\lambda) := \{\mathbf{p} :\ \mathbf{p} = b_1\mathbf{v}_i(\lambda) + b_2\mathbf{v}_j(\lambda) + b_3\mathbf{v}_k(\lambda)\}$.

Figure 106: A prism D_{ijk} constructed based on the triangle $[\mathbf{v}_i\mathbf{v}_j\mathbf{v}_k]$.

Next we define a function in the Benstein-Bezier (BB) basis over the prism D_{ijk}:

$$F(b_1, b_2, b_3, \lambda) = \sum_{i+j+k=n} b_{ijk}(\lambda) B^n_{ijk}(b_1, b_2, b_3), \tag{262}$$

where $B^n_{ijk}(b_1, b_2, b_3)$ is the Bezier basis

$$B^n_{ijk}(b_1, b_2, b_3) = \frac{n!}{i!j!k!} b_1^i b_2^j b_3^k.$$

We approximate the molecular surface by the zero contour of F, denoted as S. In order to make S smooth, the degree of the Bezier basis n should be no less than 3. For simplicity, here we consider the case of $n = 3$. The control coefficients $b_{ijk}(\lambda)$ should be properly defined such that S

Figure 107: The control coefficients of the cubic Bezier basis of function F.

is continuous. In Figure 107 we show the relationship of the control coefficients and the points of the triangle when $n = 3$. Next we are going to discuss these coefficients are defined.

Since S passes through the vertices $\mathbf{v}_i,\ \mathbf{v}_j,\ \mathbf{v}_k$, we define

$$b_{300} = b_{030} = b_{003} = \lambda. \tag{263}$$

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

$$b_{210} = \lambda + \frac{1}{3}\mathbf{n}_i \cdot (\mathbf{v}_j(\lambda) - \mathbf{v}_i(\lambda)). \tag{264}$$

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

To obtain the C^1 continuity at the midpoints of the edges of T, we define b_{111} by using the side-vertex scheme [171]:

$$b_{111} = w_1 b_{111}^{(1)} + w_2 b_{111}^{(2)} + w_3 b_{111}^{(3)}, \tag{265}$$

where

$$w_i = \frac{b_j^2 b_k^2}{b_2^2 b_3^2 + b_1^2 b_3^2 + b_1^2 b_2^2}, \qquad i = 1, 2, 3,\ i \neq j \neq k.$$

Next we are going to define $b_{111}^{(1)}$, $b_{111}^{(2)}$ and $b_{111}^{(3)}$ such that the C^1 continuity is obtained at the midpoint of the edge $\mathbf{v}_j\mathbf{v}_k$, $\mathbf{v}_i\mathbf{v}_k$ and $\mathbf{v}_i\mathbf{v}_j$. Consider the edge $\mathbf{v}_i\mathbf{v}_j$ for instant. Recall that any point $\mathbf{p} = (x, y, z)$ in D_{ijk} can be represented by

$$(x, y, z)^T = b_1\mathbf{v}_i(\lambda) + b_2\mathbf{v}_j(\lambda) + b_3\mathbf{v}_k(\lambda). \tag{266}$$

Therefore differentiating both sides of (266) with respect to x, y and z, respectively, yields

$$I_3 = \begin{pmatrix} \frac{\partial b_1}{\partial x} & \frac{\partial b_2}{\partial x} & \frac{\partial \lambda}{\partial x} \\ \frac{\partial b_1}{\partial y} & \frac{\partial b_2}{\partial y} & \frac{\partial \lambda}{\partial y} \\ \frac{\partial b_1}{\partial z} & \frac{\partial b_2}{\partial z} & \frac{\partial \lambda}{\partial z} \end{pmatrix} \begin{pmatrix} (\mathbf{v}_i(\lambda) - \mathbf{v}_k(\lambda))^T \\ (\mathbf{v}_j(\lambda) - \mathbf{v}_k(\lambda))^T \\ (b_1\mathbf{n}_i + b_2\mathbf{n}_j + b_3\mathbf{n}_k)^T \end{pmatrix}, \tag{267}$$

where I_3 is a 3×3 unit matrix. Denote

$$T := \begin{pmatrix} (\mathbf{v}_i(\lambda) - \mathbf{v}_k(\lambda))^T \\ (\mathbf{v}_j(\lambda) - \mathbf{v}_k(\lambda))^T \\ (b_1\mathbf{n}_i + b_2\mathbf{n}_j + b_3\mathbf{n}_k)^T \end{pmatrix}, \tag{268}$$

and let $A = \mathbf{v}_i(\lambda) - \mathbf{v}_k(\lambda)$, $B = \mathbf{v}_j(\lambda) - \mathbf{v}_k(\lambda)$ and $C = b_1\mathbf{n}_i + b_2\mathbf{n}_j + b_3\mathbf{n}_k$, then $T = (A\ B\ C)^T$. From (267) we have

$$\begin{pmatrix} \frac{\partial b_1}{\partial x} & \frac{\partial b_2}{\partial x} & \frac{\partial \lambda}{\partial x} \\ \frac{\partial b_1}{\partial y} & \frac{\partial b_2}{\partial y} & \frac{\partial \lambda}{\partial y} \\ \frac{\partial b_1}{\partial z} & \frac{\partial b_2}{\partial z} & \frac{\partial \lambda}{\partial z} \end{pmatrix} = T^{-1} = \frac{1}{\det(T)}\left(B \times C,\ C \times A,\ A \times B\right). \tag{269}$$

According to (262), at the midpoint of $\mathbf{v}_i\mathbf{v}_j$, $(b_1, b_2, b_3) = (\frac{1}{2}, \frac{1}{2}, 0)$, we have

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

By (265), at $(b_1, b_2, b_3) = (\frac{1}{2}, \frac{1}{2}, 0)$ we have $b_{111} = b_{111}^{(3)}$. Therefore the gradient at $(\frac{1}{2}, \frac{1}{2}, 0)$ is

$$\begin{aligned}\nabla F &= T^{-1}(\frac{\partial F}{\partial b_1}, \frac{\partial F}{\partial b_2}, \frac{\partial F}{\partial \lambda})^T \\ &= \frac{\mathbf{n}_i + \mathbf{n}_j}{4} + \frac{1}{2\det(T)}[3(b_{210} - b_{111}^{(3)})B \times C + \ 3(b_{120} - b_{111}^{(3)})C \times A + A \times B] \end{aligned} \tag{270}$$

Define vectors

$$\begin{aligned}&\mathbf{d}_1(\lambda) = \mathbf{v}_j(\lambda) - \mathbf{v}_i(\lambda) = B - A, \\ &\mathbf{d}_2(b_1, b_2, b_3) = b_1\mathbf{n}_i + b_2\mathbf{n}_j + b_3\mathbf{n}_k = C, \\ &\mathbf{d}_3(b_1, b_2, b_3, \lambda) = \mathbf{d}_1 \times \mathbf{d}_2 = B \times C + C \times A. \end{aligned} \tag{271}$$

Let

$$\mathbf{c} = C(\frac{1}{2}, \frac{1}{2}, 0), \tag{272}$$

$$\mathbf{d}_3(\lambda) = \mathbf{d}_3(\frac{1}{2}, \frac{1}{2}, 0, \lambda) = B \times \mathbf{c} + \mathbf{c} \times A. \tag{273}$$

Let $\nabla F = \nabla F(\frac{1}{2}, \frac{1}{2}, 0)$. In order to have C^1 continuity at $(\frac{1}{2}, \frac{1}{2}, 0)$, we should have $\nabla F \cdot \mathbf{d}_3(\lambda) = 0$. Therefore, by (270) and (273), we have

$$b_{111}^{(3)} = \frac{\mathbf{d}_3(\lambda)^T(3b_{210}B \times \mathbf{c} + 3b_{120}\mathbf{c} \times A + A \times B)}{3||\mathbf{d}_3(\lambda)||^2}. \tag{274}$$

Similarly, we may define $b_{111}^{(1)}$ and $b_{111}^{(2)}$.

Now the function $F(b_1, b_2, b_3, \lambda)$ is well defined. The next step is to extract the zero level set S. Given the barycentric coordinates (b_1, b_2, b_3) of a point in the triangle $[\mathbf{v}_i\mathbf{v}_j\mathbf{v}_k]$, we find the corresponding λ by solving the equation $F(b_1, b_2, b_3, \lambda) = 0$ for λ and this could be done by the Newton's method. Then we may get the corresponding point on S as

$$(x,\ y,\ z)^T = b_1\mathbf{v}_i(\lambda) + b_2\mathbf{v}_j(\lambda) + b_3\mathbf{v}_k(\lambda). \tag{275}$$

Smoothness

Theorem 7.1. *The ASMS S is C^1 at the vertices of T and the midpoints of the edges of T.*

Theorem 7.2. *S is C^1 everywhere if every edge $\mathbf{v}_i\mathbf{v}_j$ of T satisfies $\mathbf{n}_i \cdot (\mathbf{v}_i - \mathbf{v}_j) = \mathbf{n}_j \cdot (\mathbf{v}_j - \mathbf{v}_i)$.*

Theorem 7.3. *S is C^1 everywhere if the unit normals at the vertices of T are the same.*

Proofs of the theorems are shown in the Appendix.

Parametrization and quadrature In this section, we would like to show how the ASMS is applied to the computation of (261). Since we use the ASMS to represent the molecular surface, now $\Gamma = S$. Let $f = \frac{(\mathbf{r}-\mathbf{x}_i)\cdot\mathbf{n}(\mathbf{r})}{|\mathbf{r}-\mathbf{x}_i|^4}$. We decompose the entire surface S into patches $\{S_j\}$ with S_j being the AMSM generated over triangle j, then we have

$$\int_S f(\mathbf{x})\ dS = \sum_j \int_{S_j} f(\mathbf{x})\ dS. \tag{276}$$

For any point $\mathbf{x} = (x, y, z)$ on S_j, by the inverse map of (275), one can uniquely map $\mathbf{x}$ to a point in triangle j and get its baricentric coordinates (b_1, b_2, b_3) with $b_3 = 1 - b_1 - b_2$. Therefore, x, y, z can be represented in terms of (b_1, b_2):

$$x = x(b_1, b_2,), \qquad y = y(b_1, b_2), \qquad z = z(b_1, b_2)$$

Replacing (x, y, z) with (b_1, b_1, b_3) in (276) and letting

$$g(b_1, b_2) = f(x(b_1, b_2), y(b_1, b_2), z(b_1, b_2)),$$

we get

$$\int_{S_j} f(\mathbf{x})\ dS = \int_{\sigma_j} g(b_1, b_2)\sqrt{EG - F^2}\ db_1 db_2, \tag{277}$$

where

$$\begin{aligned} E &= (\frac{\partial x}{\partial b_1})^2 + (\frac{\partial y}{\partial b_1})^2 + (\frac{\partial z}{\partial b_1})^2, \\ F &= \frac{\partial x}{\partial b_1}\frac{\partial x}{\partial b_2} + \frac{\partial y}{\partial b_1}\frac{\partial y}{\partial b_2} + \frac{\partial z}{\partial b_1}\frac{\partial z}{\partial b_2}, \\ G &= (\frac{\partial x}{\partial b_2})^2 + (\frac{\partial y}{\partial b_2})^2 + (\frac{\partial z}{\partial b_2})^2. \end{aligned}$$

We then apply the Gaussian quadrature to (277):

$$\int_{\sigma_i} g(b_1, b_2)\sqrt{EG - F^2}\ db_1 db_2 \approx \sum_{k=1}^{n} W_k g(b_1^k, b_2^k)\sqrt{EG - F^2}|_{b_1^k, b_2^k}, \tag{278}$$

where (b_1^k, b_2^k, b_3^k) and W_k are the Gaussian integration nodes and weights on the triangles.

7.2.2 Error of the ASMS model

In order to show the error of S to the true surface S_0, we do a test on some typical surfaces (Table 3) $S_0 := \{(x, y, z) : z = f(x, y), (x, y) \in [0, 1]^2\}$ which are considered as the true surfaces. We generate a triangulation mesh over the true surface with the maximum edge length h being 0.1. Based on the mesh, we construct the ASMS model S. The error of S to S_0 is defined as max $\frac{||\mathbf{p}-\mathbf{q}||}{||\mathbf{q}||}$, where $\mathbf{p} \in S$, $\mathbf{q} \in S_0$, and $\mathbf{p}$ and $\mathbf{q}$ have the same (b_1, b_2, b_3) volume coordinates but different λ coordinates. We sample $(\mathbf{p},\ \mathbf{q})$ on the surfaces and compute the maximum relative error. For the point pair $\mathbf{p}(b_1, b_2, b_3, \lambda_p)$ and $\mathbf{q}(b_1, b_2, b_3, \lambda_q)$ defined above, we prove that their Euclidean distance is bounded by the difference of their λ coordinates.

Lemma 7.4. *The error of the approximation point* $\mathbf{p}$ *to the true point* $\mathbf{q}$ *is bounded by* $|\lambda_p - \lambda_q|$.

Proof.

$$\begin{aligned} ||\mathbf{p} - \mathbf{q}|| &\le b_1||\mathbf{v}_i(\lambda_p) - \mathbf{v}_i(\lambda_q)|| + b_2||\mathbf{v}_j(\lambda_p) - \mathbf{v}_j(\lambda_q)|| + b_3||\mathbf{v}_k(\lambda_p) - \mathbf{v}_k(\lambda_q)|| \\ &\le |\lambda_p - \lambda_q|(b_1||\mathbf{n}_i|| + b_2||\mathbf{n}_j|| + b_3||\mathbf{n}_k||) \\ &= |\lambda_p - \lambda_q| \end{aligned}$$

□

Table 3: Relative error and Convergence

Function $(x, y) \in [0, 1]^2$	$\max\{\frac{\|\mathbf{p}-\mathbf{q}\|}{\|\mathbf{q}\|}\}$	C
$z = 0$	0	0
$z = x^2 + y^2$	2.450030e-05	1.010636e-2
$z = x^3 + y^3$	1.063699e-04	2.610113e-2
$z = e^{-\frac{1}{4}[(x-0.5)^2+(y-0.5)^2]}$	5.286856e-07	6.288604e-5
$z = 1.25 + \frac{\cos(5.4y)}{6+6(3x-1)^2}$	2.555683e-04	4.58608e-2
$z = \tanh(9y - 9x)$	1.196519e-02	1.896754e-1
$z = \sqrt{1 - x^2 - y^2}$	8.614969e-05	1.744051e-1 (h^4)
$z = [(2 - \sqrt{1 - y^2})^2 - x^2]^{1/2}$	1.418242e-05	1.748754e-02

Table 4: Error of ASMS to the SES

1GCQ		1ML0		1KKL	
No. of Δs	ε_{max}	No. of Δs	ε_{max}	No. of Δs	ε_{max}
16,312	0.266069	18,400	0.233949	19,968	0.260418
32,624	0.142149	36,864	0.142380	39,544	0.134689
65,456	0.082550	73,736	0.083895	79,096	0.085855

To study the rate of converges of S to S_0, we gradually refine the initial mesh. Since the error is bounded by $|\lambda_p - \lambda_q|$, we compute the ratio of the maximum difference of λ_p and λ_q to h, h^2, h^3, and so forth. As h decreases, we check if the ratio converges or not, which allows us to know the highest rate of convergence of S to S_0. For most of the test functions in Table 3, we observe that S converges to S_0 as fast as $O(h^3)$. We also observe that for the case $z = \sqrt{1 - x^2 - y^2}$, the rate of convergence reaches $O(h^4)$. We show the limit of the ratio $\frac{|\lambda-\lambda'|}{h^3}$ as $h \downarrow 0$, denoted as C, in Table 3. Hence we draw the following claim:

Claim: Let h be the maximum side length of triangulation mesh T, $\mathbf{p}$ be the point on the ASMS, $\mathbf{q}$ be the corresponding point on the true surface, then $\mathbf{p}$ converges to $\mathbf{q}$ at the rate of $O(h^3)$. i.e. There exists a constant C such that $||\mathbf{p} - \mathbf{q}|| \leq Ch^3$.

4600 Triangles 9216 Triangles 18434 Triangles

Figure 108: The top row is the triangulation of the SES of protein 1ML0 with different number of triangles. The bottom row is the ASMS generated from the above corresponding triangulation.

We generated the ASMS for the real proteins based on different size of meshes (Figure 108) and show the error of the ASMS to the SES of three proteins: 1GCQ (843 atoms), 1ML0 (1051 atoms), and 1KKL (1276 atoms) in Table 4. Here the SES is modeled as a level set of the summation of fast decaying Gaussian functions. The ASMS is generated from the triangulation of the SES at different resolution. The number of triangles of the initial meshes are listed in Table 4. The error ε_{max} is defined as the one-way Hausdorff distance from the ASMS to the SES: $\varepsilon_{max} = \max_{\mathbf{p}\in\text{ASMS}} \min_{\mathbf{q}\in\text{SES}} ||\mathbf{p} - \mathbf{q}||$. As we see in the table, the errors are small and decrease rapidly as the initial triangulation becomes dense.

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

Figure 109: Molecular models of a protein(1HIA). (a) is The atomic model. (b) is the initial dense mesh of the SES (27480 triangles). (c) is the decimated mesh of the SES model (7770 triangles). (d) is the ASMS (7770 patches) generated from (c).

(a) (b) (c)
(d) (e) (f)

Figure 110: The top row are the models of 1CGI and the bottom row are the models of 1PPE. (a) and (d) are the atomic structures of the proteins. (b) and (e) are the decimated triangular meshes of the proteins with 8712 triangles and 6004 triangles, respectively. (c) and (f) are the ASMS models generated from (b) and (e), respectively.

7.2.3 Application to the biomolecular energetic computation

We apply the ASMS model to the GB electrostatic solvation energy computations of the example proteins 1HIA (693 atoms), 1CGI (852 atoms), and 1PPE (436 atoms). The ASMS models S for the proteins are generated based on the initial mesh with different number of triangles (Table 5). We show the ASMS of the example molecules generated from the decimated triangulations in Figure 109 and Figure 110. As a comparison, we compute the polarization energy $G_{pol}$for both the ASMS and the piecewise linear (PL) surfaces and show the energy results and the timing in Table 5. For all the computations, a 4-point Gaussian quadrature rule over a triangle [97] is used for the numerical integration in (278) when computing the Born radii. The running time contains the time cost of computing the integration nodes over the surfaces, computing the Born radii, and evaluating G_{pol}. If we consider the energy computed from the dense mesh as accurate, as we see from the table, the G_{pol} computed from the coarse PL model has a large error, however for the coarse ASMS model, it is very close to the dense mesh result but with less time. On the other hand, to get a energy result of the same accuracy, fewer number of triangles are needed for the ASMS model than the PL model. For example, for the protein 1CGI, the G_{pol} computed from the ASMS with 3674 triangles is -1394.227 kcal/mol. However to get a similar result, 8712 triangles are needed for the piecewise linear model. Therefore the ASMS model is much more efficient in the energetic computation than trivial piecewise linear models.

Table 5: electrostatic solvation energy and timing

Protein ID	No. of Triangles	G_{pol} (kcal/mol)		Timing (s)	
		PL	AS	PL	AS
1CGI	29108	-1371.741894	-1343.1496	39.64	40.31
	8712	-1399.194841	-1346.2230	12.94	12.64
	3674	-1678.444735	-1394.2270	7.40	6.11
1HIA	27480	-1361.226603	-1340.6384	30.23	31.18
	7770	-1389.017538	-1347.8067	9.43	9.93
	3510	-1571.890827	-1388.4665	5.21	5.21
1PPE	24244	-835.563905	-825.3252	17.27	18.26
	6004	-852.713039	-828.2158	5.09	5.39
	2748	-933.956234	-845.5085	2.74	3.27

We have introduced a method to generate a model for the molecular surface. Like the other molecular surface models, this ASMS model is smooth and close to the SES as long as the initial triangulation is based on the SES. In addition, it has dual implicit and parametric representations. The implicit representation enables us to flexibly vary the surface by selecting different level sets, while the parametric representation allows us easily apply the ASMS to the numerical computations, such as the numerical integrations involved in the finite element method or the boundary element method. Moreover, unlike the other piecewise linear models, the ASMS surface is of higher degree, therefore, to get the same accuracy, fewer number of triangles (roughly one-third of the PL model) are needed for the ASMS when it is applied to the numerical integrations. For many large system problems, for example the atomistic molecular dynamics simulations, efficient computation is the most concerning issue, hence he ASMS is very suitable to be used in this kind of problems. We should mention that, while not detailed, the algorithm of Section 7.2.1 can, by repeated evocation, yield a hierarchical multiresolution spline model of the molecular surface. In the future research we could extend this algebraic patch model to the electrostatic solvation forces calculation which is crucial in the molecular dynamics simulations. Fast and accurate numerical integration is also one of the main tasks of the force calculation and is more challenging because the integration domain contains not only the surface but also a skin layer over each atom.

References

[1] S. Abhyankar and C. Bajaj. Automatic rational parameterization of curves and surfaces iv: Algebraic space curves. *ACM Transactions of Graphics*, 8(4):324–333, 1989.

[2] S. Abhyankar and C. Bajaj. Computations with algebraic curves. In N. . Lecture Notes in Computer Science, editor, *Proc. of the Intl. Symposium on Symbolic and Algebraic Computation*, pages 279–284. Springer-Verlag, 1989.

[3] M. Abramowitz and I. Stegun. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. Dover publications, 1964.

[4] N. Akkiraju and H. Edelsbrunner. Triangulating the surface of a molecule. *Discrete Applied Mathematics*, 71:5–22, 1996.

[5] P. Alfeld. A trivariate Clough-Tocher scheme for tetrahedral data. *Computer Aided Geometric Design*, 1:169–181, 1984.

[6] P. Alliez, D. Cohen-Steiner, M. Yvinec, and M. Desbrun. Variational tetrahedral meshing. In *SIGGRAPH 2005*, pages 617–625, 2005.

[7] D. Attali and J.-O. Lachaud. Delaunay conforming iso-surface, skeleton extraction and noise removal. *Comp. Geom.: Theory and Appl.*, 19:175–189, 2001.

[8] P. Baehmann, S. Wittchen, M. Shephard, K. Grice, and M. Yerry. Robust geometrically based, automatic two-dimensional mesh generation. *Int. J. Numer. Meth. Engng*, 24:1043–1078, 1987.

[9] C. Bajaj. Geometric modeling with algebraic surfaces. In D. Handscomb, editor, *The Mathematics of Surfaces III*, pages 3 –48. Oxford University Press, 1990.

[10] C. Bajaj. Electronic Skeletons: Modeling Skeletal Structures with Piecewise Algebraic Surfaces. In *Curves and Surfaces in Computer Vision and Graphics II*, pages 230–237, Boston, MA, 1991.

[11] C. Bajaj. Surface fitting with implicit algebraic surface patches. In H. Hagen, editor, *Topics in Surface Modeling*, pages 23 – 52. SIAM Publications, 1992.

[12] 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.

[13] C. Bajaj. Implicit surface patches. In J. Bloomenthal, editor, *Introduction to Implicit Surfaces*, pages 98 – 125. Morgan Kaufman Publishers, 1997.

[14] 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, Computer Sciences Department, Purdue University, 1994.

[15] 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. ACM SIGGRAPH, Addison Wesley, August 6-11 1995.

[16] C. Bajaj, J. Castrillon-Candas, V. Siddavanahalli, and Z. Xu. Compressed representations of macromolecular structures and properties. *Structure*, 13:463–471, 2005.

[17] C. Bajaj, J. Chen, and G. Xu. Interactive modeling with a-patches. Computer science technical report, csd-tr-93-002, Purdue University, 1993.

[18] C. Bajaj, J. Chen, and G. Xu. Modeling with Cubic A-Patches. *ACM Transactions on Graphics*, 14(2):103–133, 1995.

[19] C. Bajaj, J. Chen, and G. Xu. Modeling with cubic A-patches. *ACM Transactions on Graphics*, 14(2):103–133, 1995.

[20] C. Bajaj, J. Chen, and G. Xu. Modeling with cubic A-patches. *ACM Transactions on Graphics*, 14:103–133, 1995.

[21] C. Bajaj, P. Djeu, V. Siddavanahalli, and A. Thane. Texmol: Interactive visual exploration of large flexible multi-component molecular complexes. *Proc. of the Annual IEEE Visualization Conference*, pages 243–250, 2004.

[22] C. Bajaj and S. Evans. Splines and geometric modeling. *Handbook of discrete and computational geometry*, pages 833–850, 1997.

[23] C. Bajaj, A. Gillette, and S. Goswami. Topology based selection and curation of level sets. In *TopoInVis 2007*, Accepted.

[24] C. Bajaj and S. Goswami. Automatic fold and structural motif elucidation from 3d em maps of macromolecules. In *ICVGIP 2006*, pages 264–275, 2006.

[25] C. Bajaj, C. Hoffmann, R. Lynch, and J. Hopcroft. Tracing surface intersections. *Computer aided geometric design*, 5(4):285–307, 1988.

[26] C. Bajaj and I. Ihm. Algebraic surface design with Hermite interpolation. *ACM Transactions on Graphics*, 11(1):61–91, 1992.

[27] C. Bajaj and I. Ihm. Algebraic surface design with Hermite interpolation. *ACM Transactions on Graphics*, 11(1):61–91, 1992.

[28] C. Bajaj and I. Ihm. Low degree approximation of surfaces of revolved objects. In *Proceedings of Graphics Interface'93*, pages 33–41, Toronto, Canada, 1993.

[29] C. Bajaj, I. Ihm, and J. Warren. Higher-order interpolation and least-squares approximation using implicit algebraic surfaces. *ACM Transactions on Graphics (TOG)*, 12(4):347, 1993.

[30] C. Bajaj, H. Lee, R. Merkert, and V. Pascucci. Nurbs based b-rep models from macromolecules and their properties. *In Proceedings of Fourth Symposium on Solid Modeling and Applications*, pages 217–228, 1997.

[31] C. Bajaj, V. Pascucci, and D. Schikore. The contour spectrum. In *Proceeding of IEEE Visualization 1997*, pages 167–174, 1997.

[32] C. Bajaj and V. Siddavanahalli. An adaptive grid based method for computing molecular surfaces and properties. ICES Technical Report TR-06-57, 2006.

[33] C. Bajaj and V. Siddavanahalli. Fast error-bounded surfaces and derivatives computation for volumetric particle data. ICES Technical Report TR-06-06, 2006.

[34] C. Bajaj, V. Siddavanahalli, and W. Zhao. Fast algorithms for molecular interface triangulations and solvation energy computations. *ICES Report*, 07-06, 2007.

[35] C. Bajaj, J. Warren, and G. Xu. A subdivision scheme for hexahedral meshes. *The Visual Computer*, 18(5-6):343–356, 2002.

[36] C. Bajaj, Q. Wu, and G. Xu. Level set based volume anisotropic diffusion. In *ICES Technical Report 301, The Univ. of Texas at Austin, www.ticam.utexas.edu/ccv/papers/*, 2003.

[37] C. Bajaj and G. Xu. *A-Splines: Local Interpolation and Approximation using C^k-Continuous Piecewise Real Algebraic Curves*. Computer Science Technical Report, CAPO-92-44, Purdue University, 1992.

[38] C. Bajaj and G. Xu. Modeling Scattered Function Data on Curved Surface. In J. Chen, N. Thalmann, Z. Tang, and D. Thalmann, editor, *Fundamentals of Computer Graphics*, pages 19 – 29, Beijing, China, 1994.

[39] C. Bajaj and G. Xu. Piecewise rational approximations of real algebraic curves. *J. Comput. Math*, 15(1):55–71, 1997.

[40] C. Bajaj and G. Xu. Smooth multiresolution reconstruction of free-form fat surfaces. TICAM Report 99-08, Texas Institute for Computational and Applied Mathematics, The University of Texas at Austin, March 1999.

[41] C. Bajaj and G. Xu. Anisotropic diffusion of surfaces and functions on surfaces. *ACM Transactions on Graphics (TOG)*, 22(1):4–32, 2003.

[42] C. Bajaj, G. Xu, R. Holt, and A. Netravali. Hierarchical multiresolution reconstruction of shell surfaces. *CAGD*, 19:89–112, 2002.

[43] C. Bajaj, G. Xu, and Q. Zhang. Smooth surface constructions via a higher-order level-set method. In *Proc. of CAD/Graphics 2007*, Accepted.

[44] C. L. Bajaj, V. Pascucci, and D. Schikore. Fast isocontouring for improved interactivity. In *Proceedings of the IEEE Symposium on Volume Visualization, ACM Press*, pages 39–46, 1996.

[45] N. Baker, M. Holst, and F. Wang. Adaptive multilevel finite element solution of the Poisson-Boltzmann equation II. Refinement at solvent-accessible surfaces in biomolecular systems. *J. Comput Chem.*, 21:1343–1352, 2000.

[46] E. Bansch, P. Morin, and R. Nochetto. Finite element methods for surface diffusion. *Free Boundary Problems: Theory and Applications*, page 53, 2004.

[47] R. Barnhill. Surfaces in computer aided geometric design : A Survey with New Results. *Computer Aided Geometric Design*, 2:1–17, 1985.

[48] R. Barnhill and T. Foley. Methods for constructing surfaces on surfaces. In G.Farin, editor, *Geometric Modeling: Methods and their Applications*, pages 1–15. Springer, Berlin, 1991.

[49] R. Barnhill, B. Piper, and K. Rescorla. Interpolation to arbitrary data on a surface. In G.Farin, editor, *Geometric Modeling*, pages 281–289. SIAM, Philadelphia, 1987.

[50] 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.

[51] D. Bashford and D. A. Case. Generalized Born models of macromolecular solvation effects. *Annu. Rev. Phys. Chem.*, 51:129–152, 2000.

[52] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, I. Shindyalov, and P. Bourne. The Protein Data Bank. *Nucleic Acids Research*, pages 235–242, 2000.

[53] M. W. Bern, D. Eppstein, and J. R. Gilbert. Provably good mesh generation. In *IEEE Symposium on Foundations of Computer Science*, pages 231–241, 1990.

[54] M. Bernadou and J. M. Boisserie. *The Finite Element Method in Thin Shell Theory: Application to Arch Dam Simulations*. Birkhäuser, 1982.

[55] T. Blacker and R. Myers. Seams and wedges in plastering: a 3d hexahedral mesh generation algorithm. *Engineering With Computers*, 2:83–93, 1993.

[56] T. Blacker and M. Stephenson. Paving: A new approach to automated quadrilateral mesh generation. *Int. J. Numer. Meth. Engng*, 32:811–847, 1991.

[57] J. Bloomenthal. Polygonization of implicit surfaces. *Computer Aided Geometric Design*, 5(4):341–355, 1988.

[58] W. Böhm, G. Farin, and J. Kahmann. A survey of curve and surface methods in CAGD. *Computer Aided Geometric Design*, 1:1–60, 1984.

[59] M. L. Bucalem and K. J. Bathe. Finite element analysis of shell structures. *Archives Comput. Methods Engrg.*, 4:3–61, 1997.

[60] S. Canann. Plastering and optismoothing: new approaches to automated, 3d hexahedral mesh generation and mesh smoothing. *Ph.D. Dissertation, Brigham Young University, Provo, UT*, 1991.

[61] S. Canann, J. Tristano, and M. Staten. An approach to combined laplacian and optimization-based smoothing for triangular, quadrilateral and quad-dominant meshes. In *7th International Meshing Roundtable*, pages 479–494, 1998.

[62] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. *Computational Geometry : Theory and Applications*, 24(2):75–94, 2003.

[63] M. Cauchy. *Cours d'analyse de l'Ecole Royale Polytechnique*. premiére, L'Imprimerie Royale, Paris, 1821.

[64] CGAL Consortium. CGAL: Computational Geometry Algorithms Library.

[65] C. Charalambous and A. Conn. An efficient method to solve the minimax problem directly. *SIAM Journal of Numerical Analysis*, 15(1):162–187, 1978.

[66] I. Chavel. *Riemannian geometry: a modern introduction*. Cambridge Univ Pr, 2006.

[67] H. Cheng and X. Shi. Guaranteed quality triangulation of molecular skin surfaces. *IEEE Visualization*, pages 481–488, 2004.

[68] H. Cheng and X. Shi. Quality mesh generation for molecular skin surfaces using restricted union of balls. *IEEE Visualization*, pages 51–58, 2005.

[69] S.-W. Cheng, T. Dey, E. Ramos, and T. Ray. Sampling and meshing a surface with guaranteed topology and geometry. *SCG '04: Proc. of the 20th Annual Symposium on Computational Geometry*, pages 280–289, 2004.

[70] S.-W. Cheng and T. K. Dey. Quality meshing with weighted delaunay refinement. In *Proc. 13th ACM-SIAM Sympos. Discrete Algorithms (SODA 2002)*, pages 137–146, 2002.

[71] S.-W. Cheng, T. K. Dey, H. Edelsbrunner, M. A. Facello, and S. Teng. Sliver exudation. *Journal of ACM*, 47:883–904, 2000.

[72] S.-W. Cheng, T. K. Dey, and E. A. Ramos. Delaunay refinement for piecewise smooth complexes. In *SODA*, pages 1096–1105, 2007.

[73] E. Chernyaev. Marching cubes 33: Construction of topologically correct isosurfaces. *Technical Report. CERN CN/95-17*, 1995.

[74] L. Chew. Guaranteed-quality mesh generation for curved surfaces. In *Proc. SoCG '93*, pages 274–280, 1993.

[75] L. P. Chew. Guaranteed-Quality Mesh Generation for Curved Surface. In *9th Annual Computational Geometry*, pages 274–280, CA,USA, May 1993.

[76] F. Cirak, M. Ortiz, and P. Schroder. Subdivision surfaces: A new paradigm for thin-shell finite-element analysis. Technical report, http://www.multires.caltech.edu/pubs/, 1999.

[77] U. Clarenz, U. Diewald, and M. Rumpf. Anisotropic Geometric Diffusion in Surface Processing. In *Proceedings of Viz2000, IEEE Visualization*, pages 397–505, Salt Lake City, Utah, 2000.

[78] M. L. Connolly. Analytical molecular surface calculation. *J. Appl. Cryst*, 16:548–558, 1983.

[79] W. A. Cook and W. R. Oakes. Mapping methods for generating three-dimensional meshes. *Computers in Mechanical Engineering*, pages 67–72, 1982.

[80] S. Coons. Surfaces for computer-aided design of space forms, 1967.

[81] M. Cox. The numerical evaluation of B-splines. *J. Inst. Math. Appl*, 10(134-149):47, 1972.

[82] H. Curry and I. Schoenberg. On spline distributions and their limits: the P olya distribution functions, Abstract 380t. *Bull. Amer. Math. Soc*, 53:1114, 1947.

[83] H. Curry and I. Schoenberg. On pólya frequency functions iv: the fundamental spline functions and their limits. *J. d'Analyse Math.*, 17:71–107, 1966.

[84] W. Dahmen. Subdivision algorithm converge quadratically. *J. of Comput. Appl. Math.*, 16:145–158, 1986.

[85] 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, 1989.

[86] 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, 1989.

[87] W. Dahmen and T.-M. Thamm-Schaar. Cubicoids: modeling and visualization. *Computer Aided Geometric Design*, 10:93–108, 1993.

[88] W. Dahmen and T.-M. Thamm-Schaar. Cubicoids: modeling and visualization. *Computer Aided Geometric Design*, 10:89–108, 1993.

[89] M. de Berg and K. T. G. Dobrindt. On levels of detail in terrains. *Graphical Models and Image Processing*, 60:1–12, 1998.

[90] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 1997.

[91] C. De Boor. On calculating with B-splines* 1. *Journal of Approximation Theory*, 6(1):50–62, 1972.

[92] L. De Floriani, P. Magillo, and E. Puppo. Data structures for simplicial multi-complexes. In R. H. Güting, D. Papadias, and F. Lochovsky, editors, *Proc. 6th International Symposium on Spatial Databases*, Hong Kong, July 1999.

[93] C. deBoor. *A Practical Guide to Splines*. Springer-Verlag, 1978.

[94] T. Dey and J. Levine. Delaunay meshing of isosurfaces. In *Proc. Shape Modeling International [to appear]*, 2007.

[95] M. Do Carmo. *Riemannian geometry*. Birkhauser, 1992.

[96] M. Do Carmo and M. Do Carmo. *Differential geometry of curves and surfaces*. Prentice-Hall Englewood Cliffs, NJ, 1976.

[97] D. Dunavant. High degree efficient symmetrical Gaussian quadrature rules for the triangle. *International Journal of Numerical Methods in Engineering*, 21:1129–1148, 1985.

[98] G. Dziuk. An algorithm for evolutionary surfaces. *Numerische Mathematik*, 58(1):603–611, 1990.

[99] H. Edelsbrunner. Deformable smooth surface design. *Discrete Computational Geometry*, 21:87–115, 1999.

[100] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Trans. Inform. Theory*, pages 551–559, 1981.

[101] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel. On the Shape of a Set of Points in the Plane. *IEEE Trans. on Information Theory*, 29:4:551–559, 1983.

[102] H. Edelsbrunner and E. M
"ucke. Three-dimensional alpha shapes. In *Proceedings of the 1992 workshop on Volume visualization*, page 82. ACM, 1992.

[103] H. Edelsbrunner and N. Shah. Triangulating topological spaces. *Intl. Journal of Comput. Geom. and Appl.*, 7:365–378, 1997.

[104] D. Eppstein. Linear complexity hexahedral mesh generation. In *Symposium on Computational Geometry*, pages 58–67, 1996.

[105] C. Epstein and M. Gage. The curve shortening flow. *Wave Motion: Theory, Modeling, and Computation*, 1987.

[106] J. Escher, U. Mayer, and G. Simonett. The surface diffusion flow for immersed hypersurfaces. *SIAM Journal on Mathematical Analysis*, 29(6):1419–1433, 1998.

[107] J. Escher and G. Simonett. The volume preserving mean curvature flow near spheres. *Proceedings of the American Mathematical Society*, 126(9):2789–2796, 1998.

[108] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide, Second Edition.* Academic Press Inc., 1990.

[109] D. Field. Laplacian smoothing and delaunay triangulations. *Communications in Applied Numerical Methods*, 4:709–712, 1988.

[110] T. Foley, D. Lane, G. Nielson, R. Franke, and H. Hagen. Interpolation of scattered data on closed surfaces. *Computer Aided Geometric Design*, 7:303–312, 1990.

[111] R. Franke. Recent advances in the approximation of surfaces from scattered data. In C.K.Chui, L.L.Schumarker, and F.I.Utreras, editors, *Multivariate Approximation*, pages 275–335. Academic Press, New York, 1987.

[112] L. Freitag. On combining laplacian and optimization-based mesh smoothing techniqes. *AMD-Vol. 220 Trends in Unstructured Mesh Generation*, pages 37–43, 1997.

[113] L. Freitag and C. Ollivier-Gooch. Tetrahedral mesh improvement using swapping and smoothing. *Int. J. Numer. Meth. Engng*, 40:3979–4002, 1997.

[114] D. Fritsch. A medial description of greyscale image structure by gradient-limited diffusion. In Visualization in Biomedical Computing, pages 105–117, 1992.

[115] I. Fujishiro, Y. Maeda, H. Sato, and Y. Takeshima. Volumetric data exploration using interval volume. *IEEE Transactions on Visualization and Computer Graphics*, 2(2):144–155, 1996.

[116] M. Garland and P. Heckbert. Simplifying surfaces with color and texture using quadric error metrics. In *IEEE Visualization*, pages 263–270, 1998.

[117] A. Ghosh, C. S. Rapp, and R. A. Friesner. Generalized Born model based on a surface integral formulation. *J. Phys. Chem. B*, 102:10983–10990, 1998.

[118] S. F. Gibson. Using distance maps for accurate surface representation in sampled volumes. In *Proceedings of the 1998 IEEE symposium on Volume visualization*, pages 23–30, 1998.

[119] G. Golub and C. Van Loan. *Matrix Computation.* The Johns Hopkins Univ. Press, Baltimore, MD, 1983.

[120] J. A. Grant and B. T. Pickup. A gaussian description of molecular shape. *J. Phys. Chem.*, 99:3503–3510, 1995.

[121] V. Guillemin and A. Pollack. *Differential Topology.* Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1974.

[122] B. Guo. *Modeling Arbitrary Smooth Objects with Algebraic Surfaces.* PhD thesis, Computer Science, Cornell University, 1991.

[123] B. Guo. *Modeling arbitrary smooth objects with algebraic surfaces.* PhD thesis, Cornell University, 1991.

[124] B. Guo. Surface generation using implicit cubics. In N. Patrikalakis, editor, *Scientific Visualization of Physical Phenomena*, pages 485–530. Springer-Verlag,Tokyo, 1991.

[125] I. Guskov, A. Khodakovsky, P. Schroder, and W. Sweldens. Hybrid meshes: multiresolution using regular and irregular refinement. In *SCG '02: Proc. of the 18th Annual Symposium on Computational Geometry*, pages 264–272, 2002.

[126] P. Hanrahan. Ray tracing algebraic surfaces. *ACM SIGGRAPH Computer Graphics*, 17(3):83–90, 1983.

[127] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface reconstruction. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 295–302. ACM, 1994.

[128] 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.

[129] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics*, 26(2):71–78, 1992.

[130] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *COMPUTER GRAPHICS-NEW YORK-ASSOCIATION FOR COMPUTING MACHINERY-*, 26:71–71, 1992.

[131] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, page 26. ACM, 1993.

[132] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimzation. *Computer Graphics*, 27:19–26, Aug 1-6 1993.

[133] K. Hormann, U. Labsik, M. Meister, and G. Greiner. Hierarchical extraction of iso-surfaces with semi-regular meshes. In *SMA '02: Proc. of 7th ACM Symposium on Solid Modeling and Applications*, pages 53–58, 2002.

[134] G. Huisken. The volume preserving mean curvature flow. *Journal f
"ur die reine und angewandte Mathematik (Crelles Journal)*, 1987(382):35–48, 1987.

[135] I. Ihm and B. Naylor. Piecewise Linear Approximations of Digitized Space Curves with Applications. In N. Patrikalakis, editor, *Scientific Visualization of Physical Phenomena*, pages 545–569. Springer-Verlag, Tokyo, 1991.

[136] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. In *Proceedings of SIGGRAPH*, pages 339–346, 2002.

[137] M. Karplus and J. A. McCammon. Molecular dynamics simulations of biomolecules. *Nature Structural Biology*, 9:646–652, 2002.

[138] P. Kinney. Cleanup: Improving quadrilateral finite element meshes. In *6th International Meshing Roundtable*, pages 437–447, 1997.

[139] P. Knupp. Achieving finite element mesh quality via optimization of the jacobian matrix norm and associated quantities. part i - a framework for surface mesh optimization. *Int. J. Numer. Meth. Engng*, 48:401–420, 2000.

[140] P. Knupp. Achieving finite element mesh quality via optimization of the jacobian matrix norm and associated quantities. part ii - a framework for volume mesh optimization and the condition number of the jacobian matrix. *Int. J. Numer. Meth. Engng*, 48:1165–1185, 2000.

[141] L. Kobbelt, M. Botsch, U. Schwanecke, and H. Seidel. Feature sensitive surface extraction from volume data. In *SIGGRAPH 2001*, pages 57–66, 2001.

[142] C. Kober and M. Matthias. Hexahedral mesh generation for the simulation of the human mandible. In *9th International Meshing Roundtable*, pages 423–434, 2000.

[143] B. Kuhn and P. A. Kollman. A ligand that is predicted to bind better to avidin than biotin: insights from computational fluorine scanning. *J. Am. Chem. Soc*, 122:3909–3916, 2000.

[144] F. Labelle and J. Shewchuk. Isosurface stuffing: Fast tetrahedral meshes with good dihedral angles. In *SIGGRAPH (to appear)*, 2007.

[145] J.-O. Lachaud. Topologically defined iso-surfaces. In *DCGA '96: Proc. 6th Intl. Workshop on Discr. Geom. for Comp. Imagery*, pages 245–256, 1996.

[146] J. Lang. Adaptive multilevel solution of nonlinear parabolic PDE systems: theory, algorithm, and applications. *Lecture Notes in Computational Science and Engineering*, 16, 2000.

[147] P. Laug and H. Borouchaki. Molecular surface modeling and meshing. *Engineering with Computers*, 18:199–210, 2002.

[148] B. Lee and F. M. Richards. The interpretation of protein structure: estimation of static accessiblilty. *J. Mol. Biol.*, 55:379–400, 1971.

[149] C. Lee and S. Lo. A new scheme for the generation of a graded quadrilateral mesh. *Computers and Structures*, 52(5):847–857, 1994.

[150] E. Lee. The rational Bezier representation for conics, in geometric modeling: Algorithms and new trends, 1987.

[151] M. S. Lee, M. Feig, F. R. S. Jr., and C. Brooks, III. New analytic approximation to the standard molecular volume definition and its application to generalized Born calculations. *J. Comput Chem.*, 24:1348–1356, 2003.

[152] M. S. Lee, F. R. Salsbury, and C. Brooks, III. Novel generalized Born methods. *J Chemical Physics*, 116(24):10606–10614, 2002.

[153] W. K. Liu, Y. Guo, S. Tang, and T. Belytschko. A multiple-quadrature eight-node hexahedral finite element for large deformation elastoplastic analysis. *Computer Methods in Applied Mechanics and Engineering*, 154:69–132, 1998.

[154] S. Lodha. *Surface Approximation with Low Degree Patches with Multiple Representations.* PhD thesis, Computer Science, Rice University, 1992.

[155] C. Loop and T. DeRose. A multisided generalization of Bézier surfaces. *ACM Transactions on Graphics (TOG)*, 8(3):234, 1989.

[156] C. Loop and T. DeRose. Generalized B-spline surfaces of arbitrary topology. *ACM SIGGRAPH Computer Graphics*, 24(4):356, 1990.

[157] A. Lopes and K. Brodlie. Improving the robustness and accuracy of the marching cubes algorithm for isosurfacing. In *IEEE Transactions on Visualization and Computer Graphics*, pages 19–26, 2003.

[158] A. Lopes and K. Brodlie. Improving the robustness and accuracy of the marching cubes algorithm for isosurfacing. In *IEEE Transactions on Visualization and Computer Graphics*, volume 9, pages 16–29, 2003.

[159] W. Lorensen and H. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM SIGGRAPH '87*, pages 163–169, 1987.

[160] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of SIGGRAPH 1987*, pages 163–169, 1987.

[161] J. D. Madura, J. M. Briggs, R. C. Wade, M. E. Davis, B. A. Luty, A. Ilin, J. Antosiewicz, M. K. Gilson, B. Bagheri, L. R. Scott, and J. A. McCammon. Electrostatics and diffusion of molecules in solution: simulations with the university of houston brownian dynamics program. *Computer Physics Communications*, 91:57–95, 1995.

[162] M. Meyer, M. Desbrun, P. Schr
"oder, and A. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. *Visualization and mathematics*, 3(7):34–57, 2002.

[163] S. Mitchell and T. Tautges. Pillowing doublets: Refining a mesh to ensure that faces share at most one edge. In *4th International Meshing Roundtable*, pages 231–240, 1995.

[164] S. A. Mitchell and S. A. Vavasis. Quality mesh generation in higher dimensions. *SIAM Journal on Computing*, 29(4):1334–1370, 2000.

[165] H. Mollmann. *Introduction to the Theory of Thin Shells*. Chichester, New York, 1981.

[166] D. Moore. Compact isocontours from sampled data. *Graphics Gems III*, pages 23–28, 1992.

[167] D. Moore and J. Warren. Approximation of dense scattered data using algebraic surfaces. In *Proceedings of the Twenty-Fourth Annual Hawaii International Conference on System Sciences, 1991*, pages 681–690, 1991.

[168] W. Mullins. Theory of thermal grooving. *Journal of Applied Physics*, 28:333, 1957.

[169] B. K. Natarajan. On generating topologically consistent isosurfaces from uniform samples. *The Visual Computer*, 11:52–62, 1994.

[170] G. Nielson. The side-vertex method for interpolation in triangles. *Journal of Approximation Theory*, 25:318–336, 1979.

[171] G. Nielson. The side-vertex method for interpolation in triangles. *J. Apprrox. Theory*, 25:318–336, 1979.

[172] G. Nielson, T. Foley, B. Hamann, and D. Lane. Visualizing and Modeling Scattered Multivariate Data. *IEEE Computer Graphics And Applications*, 11:47–55, 1991.

[173] G. M. Nielson and B. Hamann. The asymptotic decider: Resolving the ambiguity in marching cubes. In *Proceedings of Visualization '91*, pages 83–90, 1991.

[174] G. M. Nielson and J. Sung. Interval volume tetrahedrization. In *IEEE Visualization '97*, pages 221–228, 1997.

[175] M. Nina, D. Beglov, and B. Roux. Atomic radii for continuum electrostatics calculations based on molecular dynamics free energy simulations. *J. Phys. Chem. B*, 101:5239–5248, 1997.

[176] G. Nürnberger, L. Schumaker, M. Sommer, and H. Strauß. Generalized tchebycheffian splines. *SIAM J. Math. Anal.*, 15:790–804, 1984.

[177] A. Oddy, J. Goldak, M. McDill, and M. Bibby. A distortion metric for isoparametric finite elements. *Transactions of CSME, No. 38-CSME-32, Accession No. 2161*, 1988.

[178] Y. Ohtake, A. Belyaev, and I. Bogaevski. Polyhedral surface smoothing with simultaneous mesh regularization. In *Geometric Modeling and Processing 2000. Theory and Applications. Proceedings*, pages 229–237, 2000.

[179] Y. Ohtake, A. Belyaev, and I. Bogaevski. Mesh regularization and adaptive smoothing. *Computer-Aided Design*, 33(11):789–800, 2001.

[180] B. O'Neill. *Elementary differential geometry*. Academic Pr, 1997.

[181] K. Opitz and H. Pottmann. Computing Shortest Paths on Polyhedra: Applications in Geometric Modeling and Scientfic Visualization. *International Journal of Computational Geometry and Applications*, 1995.

[182] S. Oudot, L. Rineau, and M. Yvinec. Meshing volumes bounded by smooth surfaces. In *Proc. 14th Intl. Meshing Roundtable*, pages 203–219, 2005.

[183] S. Owen. A survey of unstructured mesh generation technology. In *7th International Meshing Roundtable*, pages 26–28, 1998.

[184] M. Paluszny and R. R. Patterson. A family of curvature continuous cubic algebraic splines. In Curves and Surfaces in Computer Vision and Graphics III, SPIE, pages 48–57, 1992.

[185] M. Paluszny and R. R. Patterson. A family of tangent continuous algebraic splines. *ACM Transaction on Graphics*, 12,3:209–232, 1993.

[186] N. Patrikalakis and G. Kriezis. Representation of piecewise continuous algebraic surfaces in terms of B-splines. *The visual computer*, 5(6):360–370, 1989.

[187] J. Peters. Evaluation and approximate evaluation of the multivariate Bernstein-Bézier form on a regularly partitioned simplex. *ACM Transactions on Mathematical Software*, 20(4):460–480, 1994.

[188] J. Peters and M. Wittman. Smooth blending of basic surfaces using trivariate box spline. *IMA 96, The Mathematics of Surfaces*, 1996.

[189] H. Pottmann. Interpolation on surfaces using minimum norm networks. *Computer Aided Geometric Design*, 9:51–67, 1992.

[190] V. Pratt. Techniques for conic splines. *sig85*, 19(3):151–159, 1985.

[191] V. Pratt. Direct least-squares fitting of algebraic surfaces. In *ACM SIGGRAPH 1987*, pages 145–152, 1987.

[192] M. Price and C. Armstrong. Hexahedral mesh generation by medial surface subdivision: Part i. *Int. J. Numer. Meth. Engng*, 38(19):3335–3359, 1995.

[193] M. Price and C. Armstrong. Hexahedral mesh generation by medial surface subdivision: Part ii. *Int. J. Numer. Meth. Engng*, 40:111–136, 1997.

[194] L. Ramshaw. *Blossoming: A connect-the-dots approach to splines*. Digital Systems Research Center, 1987.

[195] G. D. Reis, B. Mourrain, R. Rouillier, and P. Trebuchet. An environment for symbolic and numeric computation. In *Proc. Internat. Conf. on Mathematical Software*, pages 239–249, 2002.

[196] K. Rescorla. C^1 Trivariate Polynomial Interpolation. *Computer Aided Geometric Design*, 4:237–244, 1987.

[197] F. M. Richards. Areas, volumes, packing, and protein structure. *Annu. Rev. Biophys. Bioeng.*, 6:151–176, 1977.

[198] B. Roux and T. Simonson. Implicit solvent models. *Biophysical Chemistry*, 78:1–20, 1999.

[199] Y. Saad. *Iterative methods for sparse linear systems*. Society for Industrial Mathematics, 2003.

[200] M. Sabin. *The use of piecewise form of numerical representation of shape*. PhD thesis, Hungarian Academy of Science, Budapest, 1976.

[201] G. Salmon. A treatise on conic sections, 1879.

[202] Sandia. Cubit mesh generation tookkit. web site: http://sass1693.sandia.gov/cubit.

[203] G. Sapiro. *Geometric Partial Differential Equations and Image Analysis*. Cambridge, University Press, 2001.

[204] M. Schaefer and M. Karplus. A comprehensive analytical treatment of continuum electrostatics. *J. Phys. Chem.*, 100:1578–1599, 1996.

[205] R. Schneider and L. Kobbelt. Geometric fairing of irregular meshes for free-form surface design. *Computer aided geometric design*, 18(4):359–379, 2001.

[206] R. Schneiders. A grid-based algorithm for the generation of hexahedral element meshes. *Engineering With Computers*, 12:168–177, 1996.

[207] R. Schneiders. Refining quadrilateral and hexahedral element meshes. In *5th International Conference on Grid Generation in Computational Field Simulations*, pages 679–688, 1996.

[208] R. Schneiders. An algorithm for the generation of hexahedral element meshes based on an octree technique. In *6th International Meshing Roundtable*, pages 195–196, 1997.

[209] R. Schneiders, R. Schindler, and F. Weiler. Octree-based generation of hexahedral element meshes. In *5th International Meshing Roundtable*, pages 205–216, 1996.

[210] I. Schoenberg. Contributions to the problem of approximation of equidistant data by analytic functions. *IJ Schoenberg: selected papers*, page 58, 1988.

[211] I. Schoenberg and A. Whitney. On pólya frequency functions iii. the positivity of translation determinants with applications to the interpolation problem by spline curves. *Trans. Amer. Math. Soc.*, 74:246–259, 1953.

[212] T. Sederberg. Planar piecewise algebraic curves. *Computer Aided Geometric Design*, 1(3):241–255, 1984.

[213] T. Sederberg. Piecewise algebraic surface patches. *Computer Aided Geometric Design*, 2:53–59, 1985.

[214] T. Sederberg, D. Anderson, and R. Goldman. Implicitization, inversion, and intersection of planar rational cubic curves. *Computer Vision, Graphics and Image Processing*, 31:89–102, 1985.

[215] T. Sederberg and A. Zundel. Scan line display of algebraic surfaces. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, page 156. ACM, 1989.

[216] J. R. Shewchuk. What is a good linear element? interpolation, conditioning, and quality measures. In *Proceedings of International Meshing Roundtable*, pages 115–126, 2002.

[217] Y. Song, Y. Zhang, T. Shen, C. L. Bajaj, J. A. McCammon, and N. A. Baker. Finite element solution of the steady-state smoluchowski equation for rate constant calculations. *Biophysical Journal*, 86(4):1–13, 2004.

[218] J. Srinivasan, T. E. Cheatham, P. Cieplak, P. A. Kollman, and D. A. Case. Continuum solvent studies of the stability of dna, rna, and phosphoramidate-dna helices. *J. Am. Chem. Soc*, 120:9401–9409, 1998.

[219] M. Staten and S. Canann. Post refinement element shape improvement for quadrilateral meshes. *AMD-Trends in Unstructured Mesh Generation*, 220:9–16, 1997.

[220] W. C. Still, A. Tempczyk, R. C. Hawley, and T. Hendrickson. Semianalytical treatment of solvation for molecular mechanics and dynamics. *J. Am. Chem. Soc*, 112:6127–6129, 1990.

[221] B. Szabo and I. Babuska. *Finite Element Analysis*. Wiley, New York, 1991.

[222] J. Talbert and A. Parkinson. Development of an automatic, two dimensional finite element mesh generator using quadrilateral elements and bezier curve boundary definitions. *Int. J. Numer. Meth. Engng*, 29:1551–1567, 1991.

[223] T. Tautges, T. Blacker, and S. Mitchell. The whisker-weaving algorithm: a connectivity based method for constructing all-hexahedral finite element meshes. *Int. J. Numer. Meth. Engng*, 39:3327–3349, 1996.

[224] S.-H. Teng and C. W. Wong. Unstructured mesh generation: Theory, practice, and perspectives. *International Journal of Computational Geometry and Applications*, 10(3):227–266, 2000.

[225] R. Walker. *Algebraic Curves*. Springer Verlag, New York, 1978.

[226] F. Weiler, R. Schindler, and R. schneiders. Automatic goemtry-adaptive generation of quadrilateral and hexahedral element meshes for the fem. *Numerical Grid Generation in Computational Field Simulations*, pages 689–697, 1996.

[227] R. Westermann, C. Johnson, and T. Ertl. A level-set method for flow visualization. In *Proceedings of the conference on Visualization'00*, page 154. IEEE Computer Society Press, 2000.

[228] R. Westermann, L. Kobbelt, and T. Ertl. Real-time exploration of regular volume data by adaptive reconstruction of isosurfaces. *The Visual Computer*, 15(2):100–111, 1999.

[229] B. White. Evolution of curves and surfaces by mean curvature. In *Proc. of the Inernational Congress of Mathematicians*, volume 1, pages 525–538, 2002.

[230] T. Willmore. *Riemannian geometry*. Oxford University Press, USA, 1996.

[231] Z. Wood, M. Desbrun, P. Schroder, and D. Breen. Semi-regular mesh extraction from volumes. In *Visualization 2000 Conference Proceedings*, pages 275–282, 2000.

[232] Z. Wood, H. Hoppe, M. Desbrun, and P. Schroder. Removing excess topology from isosurfaces. *ACM Transactions on Graphics*, 23(2):190–208, April 2004.

[233] Z. Wood, P. Schr
"oder, D. Breen, and M. Desbrun. Semi-regular mesh extraction from volumes. In *Proceedings of the conference on Visualization'00*, pages 275–282. IEEE Computer Society Press, 2000.

[234] Z. Wood, P. Schroder, D. Breen, and M. Desbrun. Semi-regular mesh extraction from volumes. In *VIS '00: Proc. of the Conference on Visualization 2000*, pages 275–282. IEEE Computer Society Press, 2000.

[235] A. Worsey and G. Farin. An n-dimensional Clough-Tocher element. *Constructive Approximation*, 3(2):99–110, 1987.

[236] A. Worsey and B. Piper. A trivariate Powell-Sabin interpolant. *Computer Aided Geometric Design*, 5:177–186, 1988.

[237] G. Xu. Convergence of discrete Laplace-Beltrami operators over surfaces*. *Computers & Mathematics with Applications*, 48(3-4):347–360, 2004.

[238] G. Xu, C. Bajaj, and H. Huang. *C^1 Modeling with A-patches from Rational Trivariate Functions. Computer Aided Geometric Design*, 18, 2001.

[239] G. Xu, Q. Pan, and C. Bajaj. Discrete surface modelling using partial differential equations. *Computer Aided Geometric Design*, 23(2):125–145, 2006.

[240] G. Xu and Y. Shi. Progressive computation and numerical tables of generalized Gaussian quadrature formulas. *CHINESE JOURNAL OF NUMERICAL MATHEMATICS AND APPLICATIONS*, 28(2):10, 2006.

[241] Y. Zhang, C. Bajaj, and B.-S. Sohn. 3d finite element meshing from imaging data. *Submitted to the special issue of Computer Methods in Applied Mechanics and Engineering (CMAME) on Unstructured Mesh Generation, www.ices.utexas.edu/∼jessica/meshing*, 2003.

[242] Y. Zhang, C. Bajaj, and B.-S. Sohn. Adaptive and quality 3d meshing from imaging data. In *ACM Symposium on Solid Modeling and Applications*, pages 286–291, 2003.

[243] Y. Zhang, C. Bajaj, and B.-S. Sohn. Adaptive and quality 3d meshing from imaging data. In *Proc. of 8th ACM Symposium on Solid Modeling and Applications*, pages 286–291, June 2003.

[244] Y. Zhang, G. Xu, and C. Bajaj. Quality meshing of implicit solvation models of biomolecular structures. *Cagd*, 23:510–530, 2006.

[245] Y. Zhou, B. Chen, and A. Kaufman. Multiresolution tetrahedral framework for visualizing regular volume data. In *IEEE Visualization*, pages 135–142, 1997.

[246] J. Zhu, O. Zienkiewicz, E. Hinton, and J. Wu. A new approach to the development of automatic quadrilateral mesh generation. *Int. J. Numer. Meth. Engng*, 32:849–866, 1991.