Chapter 3: Piecewise Polynomial Curves and Surfaces (Finite Elements)

Chandrajit Bajaj and Andrew Gillette

October 25, 2010

Contents

1 Piecewise Polynomials

1.1 Barycentric and Bernstein-Bézier Bases

Barycentric coordinates are a natural method for describing a function defined on a triangle. We begin with the simplest possible case of describing a line passing through a triangle. Let T be a triangle with vertices (x_1, y_1), (x_2, y_2), (x_3, y_3). To represent a line C implicitly, we could find real coefficients $c_{ij} \in \mathbb{R}$ of a function

$$g(x, y) = \sum_{i+j \leq 1} c_{ij} x^i y^j = c_{00} + c_{10} x + c_{01} y$$

such that $C = \{g = 0\}$. Since the function g is defined in terms of the global coordinates x and y, it is not immediately obvious given the coefficients c_{ij} whether C passes through T at all. Thus, we transform the (x, y) coordinates to **real barycentric coordinates** $(\lambda_1, \lambda_2, \lambda_3)$ via

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix}$$

Under the mapping, (x_1, y_1) becomes $(1, 0, 0)$, (x_2, y_2) becomes $(0, 1, 0)$, and (x_3, y_3) becomes $(0, 0, 1)$. We can now seek real coefficients $\gamma_i \in \mathbb{R}$ of a function

$$g(\lambda_1, \lambda_2, \lambda_3) = \sum_{i=1}^{3} \gamma_i \lambda_i$$

such that $C = \{g = 0\}$. The coefficients are very easily described: γ_i is the value of g at (x_i, y_i). Accordingly, if at least one γ_i is positive and at least one is negative, C will pass through T, intersecting at the edges between vertices with opposite signs. Thus, barycentric coordinates give us an easy way to define and control the shape of lines through a triangle.

To describe more complicated curves through T, we use a generalization of barycentric coordinates. Fix a degree $n \geq 1$ and compute the trinomial expansion of

$$(\lambda_1 + \lambda_2 + \lambda_3)^d = 1.$$

This will yield $\binom{d+2}{2}$ terms of the form $\lambda_1^i \lambda_2^j \lambda_3^k$ with $i + j + k = n$. These functions, called the **Bernstein polynomials**, form a basis for degree n polynomials in $\mathbb{R}^2$ and can be used analogously to the linear case. In standard coordinates, we could find real coefficients $c_{ij} \in \mathbb{R}$

$$g(x, y) = \sum_{i+j \leq n} c_{ij} x^i y^j$$

such that $C = \{g = 0\}$. This problem is much more difficult than the linear case, making the barycentric coordinate change essential. We seek instead the real **Bernstein-Bézier coefficients** $\gamma_{ijk} \in \mathbb{R}$ of the function

$$g(\lambda_1, \lambda_2, \lambda_3) = \sum_{i+j+k=n} \gamma_{ijk} \frac{d!}{i!j!k!} \lambda_1^i \lambda_2^j \lambda_3^k \tag{1}$$

such that $C = \{g = 0\}$. As with barycentric coordinates, the coefficient at a vertex of the triangle is exactly the value of g at the vertex, for example

$$\gamma_{300} = g(1, 0, 0) = g(x_1, y_1).$$

The remaining coefficients control the properties of g (and hence its level sets) within T. The coefficients are associated to the **domain points** on a regular subdivision of the triangle. We show examples of such subdivisions for $n = 2$ and $n = 3$ in Figure 1.

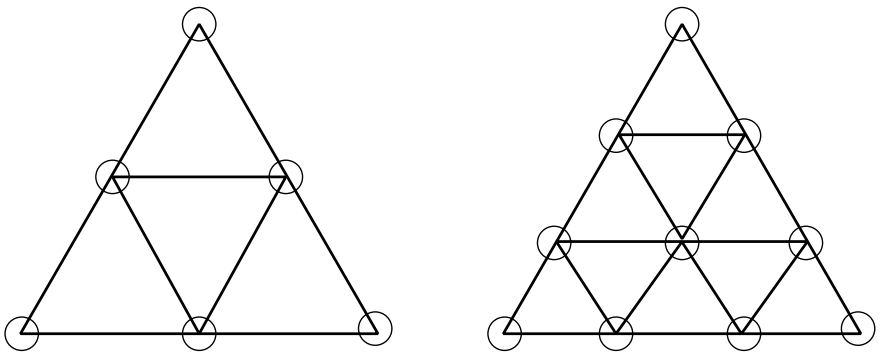

Figure 1: Domain points associated to Bernstein-Bézier coefficients for $n = 2$ (left) and $n = 3$ (right).

The bases are defined for restricted subdomains of the defining space as opposed to the power basis which is defined for all points of the space. The example formulations given below are defined for values of each of the variables x, y and z in the unit interval [0,1].

Bernstein-Bézier Basis (BB)

Univariate:

$$P(x) = \sum_{j=0}^{m} w_j B_j^m(x)$$

where

$$B_i^m(x) = \binom{m}{i} x^i (1-x)^{m-i}$$

Bivariate: (1) Tensor:

$$P(x,y) = \sum_{i=0}^{m}\sum_{j=0}^{n} w_{ij} B_i^n(x) B_j^n(y)$$

(2) Barycentric:

$$P(x,y) = \sum_{i=0}^{m}\sum_{j=0}^{m-i} w_{ij} B_{ij}^n(x,y)$$

where

$$B_{ij}^m(x,y) = \binom{m}{ij} x^i y^j (1-x-y)^{m-i-j}$$

Trivariate: (1) Tensor:

$$P(x,y,z) = \sum_{i=0}^{m}\sum_{j=0}^{n}\sum_{k=0}^{p} w_{ijk} B_i^n(x) B_j^n(y) B_k^n(z)$$

(2) Mixed:

$$P(x,y,z) = \sum_{i=0}^{m}\sum_{j=0}^{m-i}\sum_{k=0}^{p} \mathbf{b}_{ijk} B_{ij}^n(x,y) B_k^n(z)$$

(3) Barycentric:

$$P(x,y,z) = \sum_{i=0}^{m}\sum_{j=0}^{m-i}\sum_{k=0}^{m-i-j} w_{ijk} B_{ijk}^m(x,y,z)$$

where

$$B_{ijk}^m(x,y,z) = \binom{m}{ijk} x^i y^j z^k (1-x-y-z)^{m-i-j-k}$$

Multivariate (n-variate):

$$p(x_1,\ldots,x_d) = \sum_{i_1=0}^{n_1} \cdots \sum_{i_d=0}^{n_d} b_{i_1 i_2 \ldots i_d}\ B_{i_1}^{n_1}(t_1) B_{i_2}^{n_2}(t_2) \ldots B_{i_d}^{n_d}(t_d) \tag{2}$$

where

$$(x_1,\ldots,x_d)^T \in [a_1,b_1] \times [a_2,b_2] \times \ldots \times [a_d,b_d]$$

$$t_i = \tfrac{x_i - a_i}{b_i - a_i}, \qquad i = 1,2,\ldots,d$$

$$B_i^n(t) = \tfrac{n!}{i!(n-i)!}\ t^i (1-t)^{n-i}$$

Bernstein Form on a Simplex

Let $\mathbf{p}_0,\ldots,\mathbf{p}_d \in \mathbb{R}^d$ be affine independent. That is the matrix $\begin{bmatrix} \mathbf{p}_0 & \mathbf{p}_1 & \cdots & \mathbf{p}_d \\ 1 & 1 & \ldots & 1 \end{bmatrix}$ is non-singular. Let

$$\begin{bmatrix} x_1 \\ \vdots \\ x_d \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_0 & \mathbf{p}_1 & \cdots & \mathbf{p}_d \\ 1 & 1 & \dots & 1 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_d \end{bmatrix} \tag{3}$$

Then the Bernstein form on the simplex $[\mathbf{p}_0, \dots, \mathbf{p}_d]$ is

$$p(x_1, \dots, x_d) = \sum_{i_0+i_1+\dots+i_d=n} b_{i_0 i_1 \dots i_d} \; B^n_{i_0 i_1 \dots i_d}(\alpha_0, \dots, \alpha_d) \tag{4}$$

where

$$(x_1, \dots, x_d)^T \in [\mathbf{p}_0, \dots, \mathbf{p}_d],$$

$$B^n_{i_0 i_1 \dots i_d}(\alpha_0, \dots, \alpha_d) = \frac{n!}{i_0! i_1! \dots i_d!} \alpha_0^{i_0} \alpha_1^{i_1} \dots \alpha_d^{i_d}$$

Since

$$\sum_{j=0}^{d} i_j = n, \qquad \sum_{j=0}^{d} \alpha_j = 1$$

(1.6) can also be written as

$$p(x_1, \dots, x_d) = \sum_{i_1+\dots+i_d \le n} b_{i_1 \dots i_d} \tilde{B}^n_{i_1 \dots i_d}(\alpha_1, \dots, \alpha_d) \tag{5}$$

where

$$\tilde{B}^n_{i_1 \dots i_d}(\alpha_1, \dots, \alpha_d) = \frac{n!}{i_1! i_2! \dots i_d! (n - i_1 - \dots - i_d)!} \alpha_1^{i_1} \alpha_2^{i_2} \dots \alpha_d^{i_d} (1 - \alpha_1 - \dots - \alpha_d)^{n - i_1 - \dots - i_d}$$

and $b_{i_1 \dots i_d} = b_{i_0 \dots i_d}$.

Mixed Multivariate Bernstein Form

Let $d = d_1 + d_2$, $\mathbf{p}_0, \dots, \mathbf{p}_{d_1} \in \mathbb{R}^{d_1}$ be affine independent. Then the mixed Bernstein form is

$$p(x_1, \dots, x_d) = \sum_{i_1+\dots+i_{d_1} \le m} \; \sum_{j_1=0}^{n_1} \dots \sum_{j_{d_2}=0}^{n_{d_2}} b_{i_1 \dots i_{d_1} j_1 \dots j_{d_2}} \; \tilde{B}^m_{i_1 \dots i_{d_1}}(\alpha_1, \dots, \alpha_{d_1}) \; B^{n_2}_{j_1}(t_1) \dots B^{n_{d_2}}_{j_{d_2}}(t_{d_2}) \tag{6}$$

where

$$(x_1, \dots x_d)^T \in [\mathbf{p}_0, \dots, \mathbf{p}_{d_1}] \times [a_1, b_1] \times \dots \times [a_{d_2}, b_{d_2}]$$

$$\begin{bmatrix} x_1 \\ \vdots \\ x_{d_1} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_0 & \mathbf{p}_1 & \cdots & \mathbf{p}_{d_1} \\ 1 & 1 & \dots & 1 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{d_1} \end{bmatrix}$$

and

$$t_i = \frac{x_{d_1+i} - a_i}{b_i - a_i} \qquad i = 1, 2, \dots, d_2$$

If $d_1 = 0$, then p is the Bernstein form on hypercube. If $d_2 = 0$, then p is the Bernstein form on simplex.

1.2 B-Spline Basis

The B-spline basis over the unit interval [0,1] is easily generated by a fractional linear recurrence as given below for the univariate case. The bivariate and trivariate forms can also be similarly generated from this in either tensor product or barycentric form, as given for the BB form above.
Univariate:

$$\mathbf{P}_n = \sum_{l=0}^{m} \mathbf{p}_l N_l^n(x)$$

where

$$N_l^1(x) = \begin{cases} 1 \text{ for } x_l \leq x_{l+1} \\ 0 \text{ otherwise.} \end{cases}$$

and knot sequence $0 = u_0 \leq u_1 < \ldots < u_{m+1} = 1$

$$N_l^n(x) = \frac{x - x_{l-1}}{x_{l+n-1} - x_{l-1}} N_l^{n-1}(x) + \frac{x_{l+n} - x}{x_{l+n} - x_l} N_{l+1}^{n-1}(x)$$

Both the parametric and the implicit representation of algebraic curve segments and algebraic surface patches can be represented in either of the above BB or B-spline bases. Note that the canonical representation of a parametric plane curve segment and surface patch in x, y, z space are given as follows.
Curve:

$$\begin{cases} x = P_1(t), \\ y = P_2(t), \\ w = P_3(t). \end{cases}$$

Surface:

$$\begin{cases} x = P_1(s,t), \\ y = P_2(s,t), \\ z = P_3(s,t), \\ w = P_4(s,t). \end{cases}$$

where the P_i are polynomials in any of the above appropriate bases and the variables/parameters s, and t range over the unit interval [0,1].

An implicit curve segment and surface patch can be defined in x, y, z space by
Curve:

$$z = P(x,y) \wedge z = 0$$

Surface:

$$w = P(x,y,z) \wedge w = 0$$

where the P is a polynomial in any of the above appropriate basis and the variables x, y, z range over the unit interval [0,1].

The work of characterizing the BB form of polynomials within a tetrahedron such that the zero contour of the polynomial is a single sheeted surface within the tetrahedron, has been attempted in the past. In [90], Sederberg showed that if the coefficients of the BB form of the trivariate polynomial on the lines that parallel one edge, say L, of the tetrahedron, all increase (or decrease) monotonically in the same direction, then any line parallel to L will intersect the zero contour algebraic surface patch at most once. In [59], Guo treats the same problem by enforcing monotonicity conditions on a cubic polynomial along the direction from one vertex to a point of the opposite face of the vertex. From this he derives a condition $a_{\lambda - e_1 + e_4} - a_\lambda \geq 0$ for all $\lambda = (\lambda_1, \lambda_2, \lambda_3, \lambda_4)^T$ with $\lambda_1 \geq 1$, where a_λ are the coefficients of the cubic in BB form and e_i is the i-th unit vector. This

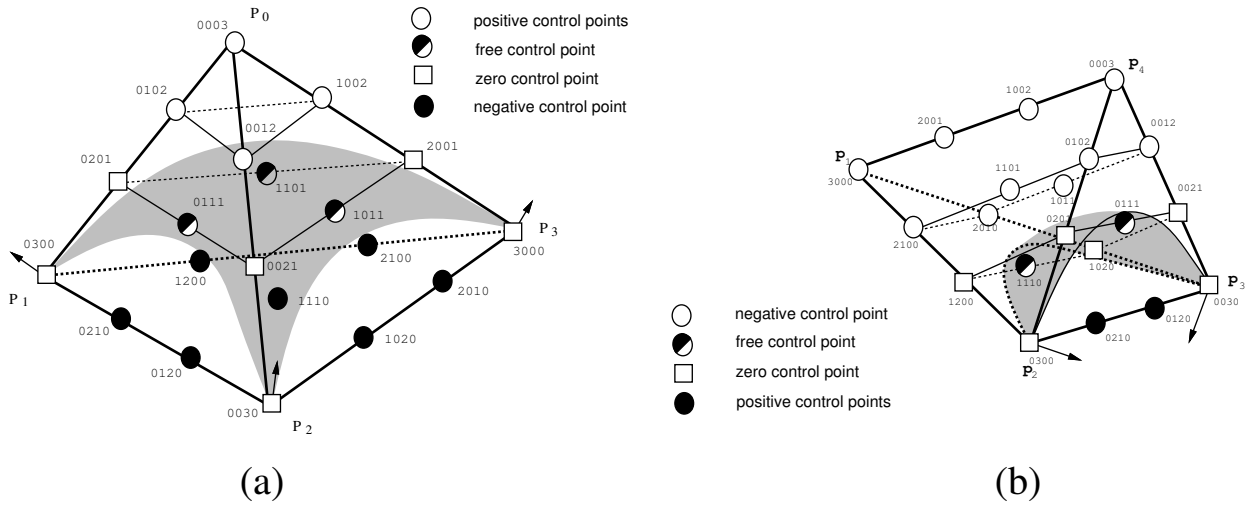

Figure 2: (a) A three sided patch tangent at P_1, P_2, and P_4. (b) A degenerate four sided patch tangent to face $[p_1p_2p_4]$ at p_2 and $[p_1p_3p_4]$ at p_3

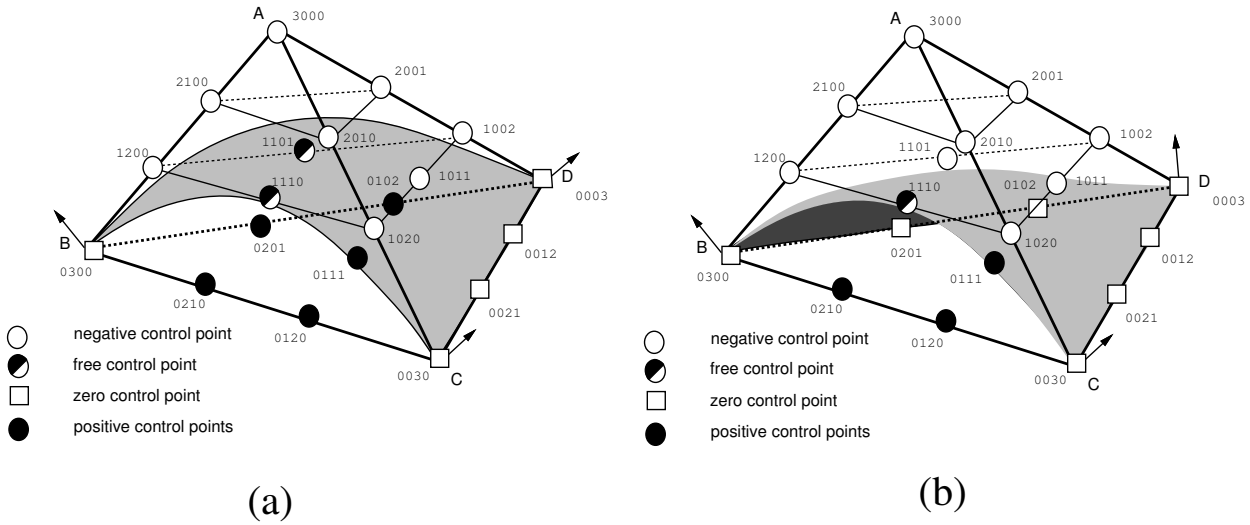

Figure 3: (a) A three sided patch interpolating the edge CD (b) A three sided patch interpolating edges BD and CD

condition is difficult to satisfy in general, and even if this condition is satisfied, one still cannot avoid singularities on the zero contour.

In[14, 15] sufficient conditions of a smooth, single sheeted zero contour generalizes Sederberg's condition and provides with an efficient way of generating nice implicit surface patches in BB-form (called A-patches for algebraic patches). See Figures 2 and 3.

1.3 Trimmed Freeform Patches

Definition 1.1. *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* smoothly contain $\boldsymbol{p}$ *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)**

1.4 Implicit Algebraic Surface Patches

An implicit algebraic surface patch can be defined in x, y, z space by :

$$w = P(x, y, z) \wedge w = 0$$

where the P is a polynomial in any of the above appropriate basis and the variables x, y, z range over the unit interval [0,1]. Alternatively, the surface patch can be defined by a closed cycle of trimming curves which may be defined with rational parametric equations or implicitly or both. In section 3 the surfaces patches are defined implicitly with a closed triple (triangle) of rational trimming curves.

As in the *Rational parametric representation*, we have many elements over which we can define the patches. In general, the simplest polyhedron we consider are

- *Cube (Tensor domain)* The parameters x, y, z are defined over the interval [0,1]. ($x \in [0, 1], y \in [0, 1], z \in [0, 1]$) This yield a tensor product Bernstein-Bézier coordinate system for trivariate polynomials.
- *Tetrahedron* The parameters x, y, z are defined with the condition: $0 \leq x + y + z \leq 1$. This yields a barycentric coordinate system for trivariate polynomials.
- *Triangular prism* The parameters x, y, z are now defined as follows: z is defined over the interval [0,1]. ($z \in [0, 1]$) and x, y range over $0 \leq x + y \leq 1$
- *Square pyramid* The parameters x and y satisfy $x \in [0, 1]$ and $y \in [0, 1]$, while z satisfies the condition $0 \leq x + y + z \leq 1$

2 Piecewise Representation of Curves

2.1 Implicit Curves, Parametric Curves

2.1.1 Approximatory A-splines in $\mathbb{R}^2$

Let $\mathbf{p}_1$, v_1 and $\mathbf{p}_2$ be three affine independent points in the xy-plane (see Figure 2.1.1). Then we consider the two line segments $[\mathbf{p}_1 v_1]$ and $[v_1 \mathbf{p}_2]$ as a segment of a polygon, denoted by $\widehat{\mathbf{p}_1 v_1 \mathbf{p}_2}$. We shall consider v_1 as a controller and $\mathbf{p}_1$ and $\mathbf{p}_2$ as interpolation points. An arbitrary polygon chain(or polygon for brevity) consists of a sequence of consecutive polygon segments denoted by $\{\widehat{\mathbf{p}_i v_i \mathbf{p}_{i+1}}\}_{i=0}^{m}$. A polygon $\{\widehat{\mathbf{p}_i v_i \mathbf{p}_{i+1}}\}_{i=0}^{m}$ is said to be of type G^1 (see Figure 2.1.1) if

$$(v_i - \mathbf{p}_{i+1}) = \alpha_i (v_{i+1} - \mathbf{p}_{i+1}), \quad \alpha_i < 0, \quad \text{for } i = 0, \cdots, m.$$

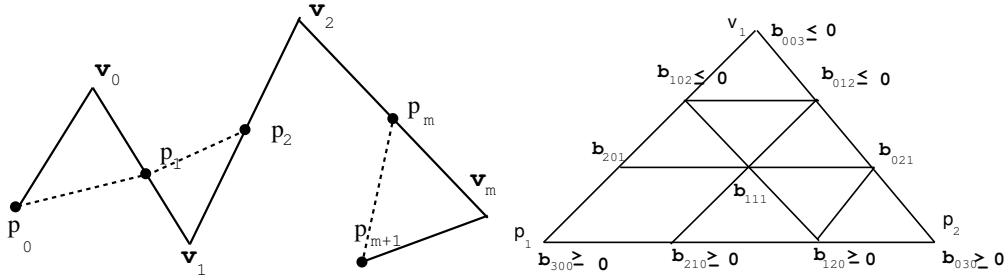

Figure 4: Left: The scaffolding used to create an A-spline. The implicitly defined curve will be contained within each triangle $\mathbf{p}_i, \mathbf{p}_{i+1}, v_i$ and meet with G^1 continuity at each $\mathbf{p}_i$. Note that this is an interpolatory scheme for the $\mathbf{p}_i$ and an approximatory scheme for the v_i. A constraint of this method is that each triple $v_i, \mathbf{p}_{i+1}, v_{i+1}$ must be collinear. Right: Bézier coefficients for a cubic interpolant using this method.

If $\mathbf{p}_0 = \mathbf{p}_{m+1}$, then the polygon is closed. Note that a G^1 polygon can be trivially constructed from an arbitrary polygon by inserting one vertex per edge of the polygon.

2.1.2 Single Sheeted, Singularity-Free A-Splines

Let $F(\alpha_1, \alpha_2, \alpha_3)$ be defined as (1) on the triangle $[\mathbf{p}_1\mathbf{p}_2 v_1]$. Since there is constant multiplier to the equation $F(\alpha_1, \alpha_2, \alpha_3) = 0$, we may assume $b_{00n} = -1$ if $b_{00n} \neq 0$.

Theorem 2.1. *For the given polynomial $F(\alpha_1, \alpha_2, \alpha_3)$ defined as (1), if there exists an integer $K(0 < K < n)$ such that (see figure 3.1 for $n = 3$ and $K = 1$)*

$$b_{ijk} \geq 0 \quad \text{for} \quad j = 0, 1, \cdots, n-k; \quad k = 0, 1, \cdots, K-1 \tag{7}$$

$$b_{ijk} \leq 0 \quad \text{for} \quad j = 0, 1, \cdots, n-k; \quad k = K+1, \cdots, n \tag{8}$$

and $\sum_{j=0}^{n} b_{n-j,j0} > 0$, $\sum_{j=0}^{n-k} b_{n-j-k,jk} < 0$ for at least one k $(K < k \leq n)$, then for any β that $0 < \beta < 1$, the straight line

$$(\alpha_1, \alpha_2, \alpha_3)(t) = (1-t)(\beta, 1-\beta, 0) + t(0, 0, 1) \tag{9}$$

that passes through v_1 and $\beta\boldsymbol{p}_1 + (1-\beta)\boldsymbol{p}_2$, intersects the curve $F(\alpha_1, \alpha_2, \alpha_3) = 0$ one and only one time(counting multiplicity) in the interior of the triangle $[\boldsymbol{p}_1\boldsymbol{p}_2 v_1]$.

Proof. Substituting $(\alpha_1, \alpha_2, \alpha_3)(t)$ into $F(\alpha_1, \alpha_2, \alpha_3)$ we have

$$\begin{aligned} B_\beta(t) &:= F((1-t)\beta, (1-t)(1-\beta), t) \\ &= \sum_{i+j+k=n} b_{ijk} \frac{n!}{i!j!k!} t^k (1-t)^{i+j} \beta^i (1-\beta)^j \\ &= \sum_{i+j+k=n} b_{ijk} B_k^n(t) B_i^{i+j}(\beta) \\ &= \sum_{k=0}^{n} b_k(\beta) B_k^n(t) \end{aligned} \tag{10}$$

where

$$b_k(\beta) = \sum_{i+j=n-k} b_{ijk} B_i^{n-k}(\beta), \quad B_j^n(t) = \frac{n!}{j!(n-j)!} t^j (1-t)^{n-j}$$

It follows from (7)–(8) that $b_0(\beta) > 0,\ b_k(\beta) \geq 0,\ \ k = 1, ..., K-1,\ \ b_k(\beta) \leq 0,\ \ k = K+1, ..., n.$ If $l (0 \leq l \leq n-K-1)$ is the integer such that $b_n(\beta) = \cdots = b_{n-l+1}(\beta) = 0;\ \ b_{n-l}(\beta) < 0$, then $B_\beta(t)$ can be written as

$$B_\beta(t) = (1-t)^l \sum_{k=0}^{n-l} c_k(\beta) B_k^{n-l}(t)$$

where $c_0 > 0, c_{n-l} < 0$ and the sequence $c_0, c_1, \cdots, c_{n-l}$ has one sign change. By variation diminishing property [53], the equation $B_\beta(t) = 0$ has exactly one root in $(0, 1)$. □

This theorem guarantees that there is one and only one curve segment of $F(\alpha_1, \alpha_2, \alpha_3) = 0$ within the triangle under the given condition. The term **algebraic spline** or ***A*-spline** that we use is a chain of such curve segments with fixed continuity at the join points. We should mention that the curve $F(\alpha_1, \alpha_2, \alpha_3) = 0$ passes through v_1 if $b_{00n} = 0$. However, we do not use this part of the curve. In applications, we usually take b_{00n} to be -1.

Remark 2.2. *Formulas (9) and (10) could be used to evaluate the curve $F(\alpha_1, \alpha_2, \alpha_3) = 0$. That is, for a given sequence of points of $\beta \in (0, 1)$, solve the equation $B_\beta(t) = 0$ for $t \in (0, 1)$. Then substituting (β, t) into (9), we then obtain a sequence of points, in terms of barycentric coordinates, on the curve. For $n \leq 4$, a closed form for the solution of $B_\beta(t) = 0$ exists. For $n \geq 5$, numerical methods have to be employed to solve the equation. However, since the equation has a single root in $(0, 1)$, Newton iterations combined with bisection suffice.*

The next theorem goes further about the smoothness of the curve $F(\alpha_1,\ \alpha_2, \alpha_3) = 0$ and the properties on the boundary of the triangle.

Theorem 2.3. *Let $F(\alpha_1, \alpha_2, \alpha_3)$ be defined as Theorem 2.1, then*

(i) The curve $F(\alpha_1, \alpha_2, \alpha_3) = 0$ is smooth in the interior of the triangle $[\boldsymbol{p}_1 \boldsymbol{p}_2 v_1]$.

(ii) If we further assume $b_{n-k,0k} = 0$ for $k = 0, ..., K$, $b_{n-(K+1),0,K+1} < 0$ and $b_{n-1,10} > 0$, then the curve in the triangle passes through $\boldsymbol{p}_1$, tangent with the line $\alpha_2 = 0$ with multiplicity $K+1$ at $\boldsymbol{p}_1$ and no other intersection with $\alpha_2 = 0$ for $\alpha_1 > 0, \alpha_3 > 0$. Similarly, if $b_{0,n-k,k} = 0$ for $k = 0, ..., K$, $b_{0,n-(K+1),K+1} < 0$, and $b_{1,n-1,0} > 0$, then the curve passes through $\boldsymbol{p}_2$, tangent with the line $\alpha_1 = 0$ with multiplicity $K+1$ at $\boldsymbol{p}_2$ and no other intersection with $\alpha_1 = 0$ for $\alpha_2 > 0, \alpha_3 > 0$.

(iii) If $b_{n00} = b_{n-1,01} = b_{n-1,10} = 0$, then $\boldsymbol{p}_1$ is a singular point of the curve. Similarly, if $b_{0n0} = b_{1,n-1,0} = b_{0,n-1,1} = 0$, then $\boldsymbol{p}_2$ is a singular point of the curve.

Since it is obvious that the quadratic A-spline is convex, we consider now the convexity of the cubic spline. At present, the convexity characterization of the general case for $n \geq 4$ degree A-spline is left as an open problem. Even for the cubic case, the convexity is not always guaranteed. If the curve segment is tangent with the sides of the triangle at $\mathbf{p}_1$ and $\mathbf{p}_2$, i.e., a G^1 A-spline as in Theorem 2.3.ii. , then it is convex. This is of course a special case, but it is the case we most often use.

Theorem 2.4. *The cubic A-spline defined in Theorem 2.3.ii has no inflection point inside its reference triangle.*

2.1.3 G^k A-Splines

In this section, we connect the A-Spline segments to form a piecewise G^k continuous spline curve. For simplicity we assume that we are given a polygon in the plane, that is we have an ordered point set $\{p_i\}_{i=0}^{m+1}$, and additionally a vertex set $\{v_i\}_{i=0}^{m}$ (see figure), such that the three points p_i, v_i and p_{i+1} are affine independent (non-collinear). The G^k continuity of an A-spline is achieved by the following steps:

1. Form a G^1 control polygon $\{p_i\widehat{v_ip_{i+1}}\}$.
2. Compute the first k terms of the local power series expansion of the A-spline at the join points p_i from the given data at these points.
3. Determine the coefficients of F such that $F = 0$ has the same first k terms of local power series at the join points.

Several schemes exist which produce a desired polygon chain from scattered data [51, 99]. To produce a G^1 polygon from a polygonal chain is trivial and amounts to inserting a single additional vertex per polygon edge. We first define the local power series and then compute the coefficients of F. Then we compute the power series for three fitting problems: (a) fit to a parametric curve; (b) fit to discrete data; (c) fit to a higher degree implicit curve.

2.1.4 Coefficients of F from Local Power Series Expansion

We consider first a two segment A-spline curve

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

on triangles $[p_1^{(l)} p_2^{(l)} v_1^{(l)}]$ for $l = 1, 2$ with $p_1^{(1)} = p_2^{(2)}$ as join point (see Figure 5), where $\tilde{b}_{ijk}^{(l)} = \frac{n}{i!j!k!} b_{ijk}^{(l)}$. We want to join these curve segments with the desired smoothness at $p_1^{(1)}$.

In the triangle $[p_1^{(l)} p_2^{(l)} v_1^{(l)}]$, we require our A-spline passing through $p_1^{(1)}$ and tangent with the line $[p_1^{(1)} v_1^{(1)}]$ at $p_1^{(1)}$. Hence we assume $b_{n-1,1,0}^{(1)} > 0$, $b_{1,n-1,0}^{(2)} > 0$. This implies that the curves $F_l(\alpha_1,\alpha_2,\alpha_3) = 0$ are regular at $p_1^{(1)}$. Therefore, the curve $F_1(1-\alpha_2-\alpha_3,\alpha_2,\alpha_3) = 0$ can be represented as a power series at $p_1^{(1)}$

$$\alpha_2 = \sum_{i=0}^{\infty} a_i^{(1)} \alpha_3^i = \sum_{i=0}^{\infty} a_i^{(1)}(p_1^{(1)}) \alpha_3^i, \qquad \alpha_1 = 1-\alpha_2-\alpha_3 \tag{11}$$

with $a_0^{(1)} = a_0^{(1)}(p_1^{(1)}) = 0$, where we relate the coefficients $a_i^{(1)}$ to $p_1^{(1)}$ to emphasize that the expansion is performed at $p_1^{(1)}$. Similarly, $F_2(\alpha_1, 1-\alpha_1-\alpha_3, \alpha_3) = 0$ can be represented as

$$\alpha_1 = \sum_{i=0}^{\infty} a_i^{(2)} \alpha_3^i = \sum_{i=0}^{\infty} a_i^{(2)}(p_2^{(2)}) \alpha_3^i, \qquad \alpha_2 = 1-\alpha_1-\alpha_3 \tag{12}$$

at $p_2^{(2)}$ with $a_0^{(2)} = 0$. It follows from Theorem 2.3 that the curve $F_1 = 0$ is tangent with $[p_1^{(1)} v_1^{(1)}]$ $n-2$ times at $p_1^{(1)}$ if and only if

$$b_{n-k,0,k}^{(1)} = 0, \qquad \text{for } k = 0, 1, \cdots, n-2 \tag{13}$$

or if and only if $a_k^{(1)} = 0$, for $k = 0, 1, \cdots, n-2$. The same is true for the curve $F_2 = 0$ at $p_2^{(2)}$, that is, the curve is tangent with $[p_2^{(2)} v_1^{(2)}]$ at $p_2^{(2)}$ $n-2$ times if and only if

$$b_{0,n-k,k}^{(2)} = 0, \qquad \text{for } k = 0, \cdots, n-2. \tag{14}$$

Now we assume (13) and (14) hold. Hence (11) and (12) become

$$\alpha_2 = \sum_{i=n-1}^{\infty} a_i^{(1)} \alpha_3^i, \quad \alpha_1 = 1-\alpha_2-\alpha_3, \tag{15}$$

$$\alpha_1 = \sum_{i=n-1}^{\infty} a_i^{(2)} \alpha_3^i, \quad \alpha_2 = 1-\alpha_1-\alpha_3 \tag{16}$$

respectively. Substitute (15) into $F_1(1-\alpha_2-\alpha_3,\alpha_2,\alpha_3)=0$, we get

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

From $g_i^{(1)}=0$ for $i=n-1,\cdots,2n-3$, we derive

$$\tilde{b}_{n-1,10}^{(1)}=-\frac{\tilde{b}_{10,n-1}^{(1)}}{a_{n-1}^{(1)}} \tag{17}$$

$$\tilde{b}_{n-2,11}^{(1)}=-\frac{-\tilde{b}_{10n-1}^{(1)}+\tilde{b}_{00n}^{(1)}+\tilde{b}_{n-1,10}^{(1)}\left[a_n^{(1)}-(n-1)a_{n-1}^{(1)}\right]}{a_{n-1}^{(1)}} \tag{18}$$

$$\tilde{b}_{n-i-2,1,i+1}^{(1)}=-\frac{\sum_{j=0}^{i}\tilde{b}_{n-1-j,1,j}^{(1)}\sum_{l-0}^{n-1-j}(-1)^l C_{n-1-j}^{l}a_{n+i-l-j}^{(1)}}{a_{n-1}^{(1)}} \tag{19}$$

for $i=1,2,\cdots,n-3$, where $C_n^k=\frac{n!}{k!(n-k)!}$, $a_j^{(1)}=0$ if $j<n-1$. That is, the coefficients determined by (17) – (19) will lead to the curve $F_1(\alpha_1,\alpha_2,\alpha_3)=0$ matching the power series (15) up to the first $2n-3$ terms. It is noted that, each of the formulas (17) – (19) determines one of the coefficients $b's$, and introduces one of the coefficients $a's$. Among all the coefficients $b's$, there is one degree of freedom.

Since $b_{n-1,10}^{(1)}>0$, $b_{10,n-1}^{(1)}<0$, (17) implies that $a_{n-1}^{(1)}>0$. The correct sign of $b_{n-1-k,1k}^{(1)}$ can be obtained by giving $a_{n+k-1}^{(1)}$ properly.

For the curve $F_2=0$ at $p_2^{(2)}$, we similarly have,

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

From which we have

$$\tilde{b}_{1,n-1,0}^{(2)}=-\frac{\tilde{b}_{01,n-1}^{(2)}}{a_{n-1}^{(2)}} \tag{20}$$

$$\tilde{b}_{1n-2,1}^{(2)}=-\frac{-\tilde{b}_{01,n-1}^{(2)}+\tilde{b}_{00n}^{(2)}+\tilde{b}_{1,n-1,0}^{(2)}\left[a_n^{(2)}-(n-1)a_{n-1}^{(2)}\right]}{a_{n-1}^{(2)}} \tag{21}$$

$$\tilde{b}_{1,n-i-2,i+1}^{(2)}=-\frac{\sum_{j=0}^{i}\tilde{b}_{1,n-1-j,j}^{(2)}\sum_{l=0}^{n-1-j}(-1)^l C_{n-1-j}^{l}a_{n+i-l-j}^{(2)}}{a_{n-1}^{(2)}} \tag{22}$$

for $i=1,2,\cdots,n-3$. If we further assume

$$a_{n-1}^{(1)}=b_{1,0,n-1}^{(1)}=0 \tag{23}$$

then similar to the discussion above, we have

$$\tilde{b}_{n-1,1,0}^{(1)}=-\frac{\tilde{b}_{00n}^{(1)}}{a_n^{(1)}} \tag{24}$$

$$\tilde{b}_{n-i-1,1,i}^{(1)}=-\frac{1}{a_n^{(1)}}\left[\sum_{j=0}^{i-1}\tilde{b}_{n-1-j,1,j}^{(1)}\sum_{l=0}^{n-1-j}(-1)^l C_{n-1-j}^{l}a_{n+i-l-j}^{(1)}\right] \tag{25}$$

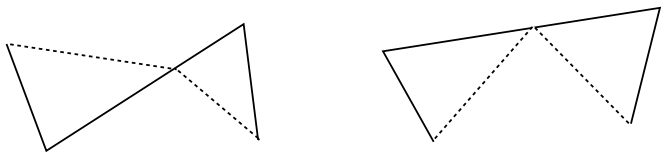

Figure 5: The two different cases of G^1 join polygon segments

for $i = 1, 2, \cdots, n-2$. Similarly, for curve $F_2 = 0$ at $p_2^{(2)}$, if we assume

$$a_{n-1}^{(2)} = b_{0,1,n-1}^{(2)} = 0 \tag{26}$$

then

$$\tilde{b}_{1,n-10}^{(2)} = -\frac{\tilde{b}_{00n}^{(2)}}{a_n^{(2)}}, \tag{27}$$

$$\tilde{b}_{1,n-i-1,i}^{(2)} = -\frac{1}{a_n^{(2)}} \left[\sum_{j=0}^{i-1} \tilde{b}_{1,n-1-j,j}^{(2)} \sum_{l=0}^{n-1-j} (-1)^l C_{n-1-j}^l a_{n+i-l-j}^{(2)} \right] \tag{28}$$

for $i = 1, 2, \cdots, n-2$.

Formulas (23)–(28) match the power series up to the first $2n-2$ terms. If we only fit the first $2n-3$ terms, $b_{1,1,n-2}^{(l)}$ could be free. For the G^3 fitting with cubics in section 4, we choose it to be zero.

Now we explain why we consider both of the cases of $a_{n-1}^{(l)} > 0$ and $a_{n-1}^{(l)} = 0$. As before, let $p_1^{(1)}\widehat{v_1^{(1)}p_2^{(1)}}$ and $p_1^{(2)}\widehat{v_1^{(2)}p_2^{(2)}}$ be two segments of a polygon. If they G^1 join at $p_1^{(1)}$, then there are two join configurations(see Figure 5): *non-convex* join and *convex* join. In the non-convex join, $p_2^{(1)}$ and $p_1^{(2)}$ lie on different sides of the line $[v_1^{(1)}v_1^{(2)}]$, while in the convex join, $p_2^{(1)}$ and $p_1^{(2)}$ lie on the same side of the line $[v_1^{(1)}v_1^{(2)}]$. Since our A-splines are always contained within the triangles considered, if $p_1^{(1)}$ is of a non-convex join, then the curve will be tangent with the line $[v_1^{(1)}v_1^{(2)}]$ an odd number of times, otherwise, it will be tangent with the line $[v_1^{(1)}v_1^{(2)}]$ an even number of times. Therefore,

(i). If $p_1^{(1)}$ is of a non-convex join, n is an even number then $a_{n-1}^{(l)} > 0$ for $l = 1, 2$; if n is an odd number, then $a_{n-1}^{(l)} = 0$.

(ii). If $p_1^{(1)}$ is of a convex join, n is an even number, then $a_{n-1}^{(l)} = 0$ for $l = 1, 2$; if n is an odd number, then $a_{n-1}^{(l)} > 0$.

Theorem 2.5. *The degree n A-spline can achieve G^{2n-3} continuity by fitting locally the given parametric or implicit curve at the join points.*

Note 4.1. If $n > 3$, the coefficients b_{ijk} are free for $i > 1$ and $j > 1$. These degrees $(= \frac{(n-2)(n-3)}{2})$ of freedom can be used to interpolate/approximate points in the triangle, fairing the constructed curve, or to achieve even higher order continuity at the joins.

Note 4.2. If n is an odd/even number and all the points are non-convex/ convex join, then each $b_{1,1,n-2}$ is free. This degree of freedom can be used to interpolate points in the triangle or to achieve G^{2n-2} continuity(see §4.3).

2.1.5 Local Power Series Computation

A. Fitting to a parametric curve.

Suppose we are given a parametric curve $X(t) = [\phi(t), \psi(t)]^T$ in the neighborhood of $p_1^{(1)}$ and assume $X(0) = p_1^{(1)}$. Now we compute $a_i^{(1)}$ defined in (11) ($a_i^{(2)}$ are similarly computed). It follows from (40) that the curve (15) and (16) in Cartesian xy-coordinates can be expressed as

$$\begin{aligned} Y_1(\alpha_3) &= [x_1(\alpha_3), y_1(\alpha_3)]^T \\ &= p_1^{(1)} + \left[v_1^{(1)} - p_1^{(1)}\right]\alpha_3 + \left[p_2^{(1)} - p_1^{(1)}\right]\sum_{i=n-1}^{\infty} a_i^{(1)}(p_1^{(1)})\alpha_3^i \qquad (29) \\ Y_2(\alpha_3) &= [x_2(\alpha_3), y_2(\alpha_3)]^T \\ &= p_2^{(2)} + \left[v_1^{(2)} - p_2^{(2)}\right]\alpha_3 + \left[p_1^{(2)} - p_2^{(2)}\right]\sum_{i=n-1}^{\infty} a_i^{(2)}(p_2^{(2)})\alpha_3^i \qquad (30) \end{aligned}$$

Now we need to determine the so called β-matrix(see [93])

$$C^{(l)} = \begin{bmatrix} \beta_1 & & & & & \\ \beta_2 & \beta_1^2 & & & & \\ \beta_3 & 3\beta_1\beta_2 & \beta_1^3 & & & \\ \beta_4 & 3\beta_2^2 + 4\beta_1 & \beta_3 & 6\beta_1^2\beta_2 & \beta_1^4 & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix} = \left[\beta_{ij}^{(l)}(p_1^{(1)})\right]$$

and $a_i^{(l)}(p_1^{(1)})$ for $l = 1, 2$ and $i = n-1, \cdots$, so that

$$\begin{bmatrix} Y_l'(0) \\ Y_l''(0) \\ \vdots \\ Y_l^k(0) \end{bmatrix} = C^{(l)} \begin{bmatrix} X'(0) \\ X''(0) \\ \vdots \\ X^k(0) \end{bmatrix}, \quad l = 1, 2 \qquad (31)$$

(31) is the condition of G^k continuity between two parametric curves. From (29)–(30) and (31), we have

$$v_1^{(l)} - p_1^{(1)} = \beta_{11}^{(l)}(p_1^{(1)})X'(0)$$

Hence

$$\beta_{11}^{(l)}(p_1^{(1)}) = (-1)^{l-1}\frac{\|v_1^{(l)} - p_1^{(1)}\|}{||X'(0)||}, \quad l = 1, 2$$

Let $n_x = (v_1^{(1)} - p_1^{(1)})/\|v_1^{(1)} - p_1^{(1)}\|$ and n_y be two unit vectors such that $n_x^T n_y = 0$ and $det[n_x, n_y] = 1$. Let

$$X^{(i)}(0) = \gamma_i(p_1^{(1)})n_y + \delta_i(p_1^{(1)})n_x, \qquad i = 1, 2, \cdots \qquad (32)$$

Then, $\gamma_1(p_1^{(1)}) = 0$, $\delta_1(p_1^{(1)}) = \|X'(0)\|$ and

$$\gamma_i(p_1^{(1)}) = \det\left[n_x, X^{(i)}(0)\right], \quad \delta_i(p_1^{(1)}) = \det\left[X^{(i)}(0), n_y\right]$$

and

$$\begin{aligned} Y_l^{(k)}(0) &= \sum_{i=1}^{k} \beta_{ki}^{(l)}(p_1^{(1)})X^{(i)}(0) \\ &= \left[\sum_{i=1}^{k} \beta_{ki}^{(l)}(p_1^{(1)})\gamma_i(p_1^{(1)})\right] n_y + \left[\sum_{i=1}^{k} \beta_{ki}^{(l)}(p_1^{(1)})\delta_i(p_1^{(1)})\right] n_x \end{aligned}$$

where $\beta_{ki}^{(l)}(p_1^{(1)})$ are known for $i = 2, \cdots, k$ and $l = 1, 2$. Let

$$p_2^{(1)} - p_1^{(1)} = s^{(1)} n_x + t^{(1)} n_y, \quad p_1^{(2)} - p_2^{(2)} = s^{(2)} n_x + t^{(2)} n_y$$

It follows from (29) and (30) that

$$Y_l^{(k)}(0) = k! a_k^{(l)}(p_1^{(1)}) \left[s^{(l)} n_x + t^{(l)} n_y \right], \quad k \geq 2, \ l = 1, 2$$

We have

$$\beta_{k1}^{(l)}(p_1^{(1)}) = \frac{\sum_{i=2}^{k} \beta_{ki}^{(l)}(p_1^{(1)}) \left[s^{(l)} \gamma_i(p_1^{(1)}) - t^{(l)} \delta_i(p_1^{(1)}) \right]}{t^{(l)} \| X'(0) \|}$$

$$a_k^{(l)}(p_1^{(1)}) = \frac{1}{k! t^{(l)}} \sum_{i=2}^{k} \beta_{ki}^{(l)}(p_1^{(1)}) \gamma_i(p_1^{(1)})$$

B. Fitting to discrete data

Suppose we are given a set of points $\{p_i\}$. Let

$$\Delta^0 p_i = p_i$$
$$\Delta^{j+1} p_i = \frac{\sigma(\Delta^j p_{i+1} - \Delta^j p_i)}{||p_{i+1} - p_i||} + \frac{(1-\sigma)(\Delta^j p_i - \Delta^j p_{i-1})}{||p_i - p_{i-1}||}$$

where $\sigma = \frac{||p_{i-1} - p_i||}{||p_{i+1} p - p_i|| + ||p_i - p_{i-1}||}$. Then $\Delta^j p_i$ can be an approximation of $X^j(t)$ at p_i with $X(t)$ as an imaginary space curve.

The computation of $a_i^{(j)}$ from $X^j(t)$ is the same as before.

C. Fitting to an implicit curve.

Let $g(x, y) = 0$ be a given implicit curve to be approximated. First compute the singular points and inflection points. These points will divide the curve into smooth and convex segments. For each segment, form a point list by a tracing (see [31]) scheme, such that the normals at two adjacent points have angle $< \frac{\pi}{2}$. Then a G^1 polygon for one segment is formed by the tangent lines at the point list.

For each triangle, say $[p_1 p_2 v_1]$, the curve $g(x, y) = 0$ passes through p_1, p_2 and is tangent with the line $[p_i v_1]$ for $i = 1, 2$. Let $G(\alpha_1, \alpha_2, \alpha_3)$ be the barycentric form of $g(x, y)$ over $[p_1 p_2 v_1]$. Let $G_1(\alpha_2, \alpha_3) = G(1 - \alpha_2 - \alpha_3, \alpha_2, \alpha_3)$, then at p_1, $G_1(\alpha_2, \alpha_3) = 0$ can be expressed as a power series $\alpha_2 = \sum_{i=0}^{\infty} a_i \alpha_3^i$ by the following algorithm for $f(x, y) = 0$. Let

$$f^0(x, y) = f(x, y) = y - a_2 x^2 + a_0^0(x) + a_1^0(x) y + \cdots + a_n^0(x) y^n$$

with $ord(a_0^0) > 2$. As a function of x, $y = y(x)$ has order ≥ 2. Let $y_1 = y - a_2 x^2$. Then the order of $y_1 = y_1(x)$ is ≥ 3. Let

$$\begin{aligned} f^1(x, y_1) &= f^0(x, y_1 + a_2 x^2) \\ &= y_1 - a_3 x^3 + a_0^1(x) + a_1^1(x) y_1 + \cdots + a_n^1(x) y_1^n \end{aligned}$$

then $ord(a_0^1) > 3$. Repeating this procedure, we get $a_2 x^2, a_3 x^3 \cdots$. Then $\sum_{i=2}^{\infty} a_i x^i$ is the power series expansion.

This algorithm is simple and easy to implement. If we want to compute $a_2 x^2, a_3 x^3 \cdots$ up to $a_k x^k$, then the terms in $a_i^j(x)$ with degree $> k - (j+2)^i$ can be deleted during the computation, since these terms have no contribution to $\sum_{i=2}^{k} a_i x^i$. Hence the algorithm is also space effective.

2.1.6 Cubic A-Splines Example

As an example, we describe cubic A-splines in detail. We omit the detail discussion of quadratic A-splines, since it is easier and the conclusions arrived are similar to the ones in the literature [53] and [80].

Suppose we are given parametric data at the join points, that is X^k, $k = 1, 2, \cdots$. We shall determine the coefficients b_{ijk} so that G^{2n-3} continuity is achieved. Furthermore, for the error estimation (see section §5), we require

$$b_{ijk+e_3} < \frac{b_{ijk+e_1}}{2} + \frac{b_{ijk+e_2}}{2} \tag{33}$$

G^3 continuity: Consider a two segments cubic A-spline as in §4.1. Now suppose p_1 is the join point and assume that the coefficients $b^{(l)}_{111} = 0$ for both segments. There are two cases that need to be considered:

1. p_1 **is of non-convex join**.

In this case, we have

$$a^{(1)}_2(p_1) = a^{(2)}_2(p_1) = 0, \qquad \tilde{b}^{(1)}_{210} = \frac{1}{a^{(1)}_3(p_1)}, \qquad a^{(1)}_3(p_1) > 0,$$
$$\tilde{b}^{(2)}_{120} = \frac{1}{a^{(2)}_3(p_1)}, \qquad a^{(2)}_3(p_1) > 0.$$

Since

$$\begin{aligned} a^{(l)}_3 &= \frac{1}{6t^{(l)}(p_1)} \left[\beta^{(l)}_{32}(p_1)\gamma_2(p_1) + \beta^{(l)}_{33}(p_1)\gamma_3(p_1)\right] \\ &= \frac{1}{6t^{(l)}(p_1)} \left[\beta^{(l)}_{33}(p_1)\gamma_3(p_1)\right] = \frac{1}{6t^{(l)}(p_1)} \left[\beta^{(l)}_{11}(p_1))^3\gamma_3(p_1)\right] \end{aligned}$$

and $(-1)^{l-1}\beta^{(1)}_{11}(p_1) > 0$, we have $(-1)^{l-1}t^{(l)}(p_1)\gamma_3(p_1) > 0$. The geometric meaning is $X'''(0)$ and $p_2 - p_1$ are on the same side of the line $< p_1 v_1 >$.

2. p_1 **is of convex join**.

In this case, we have

$$\begin{aligned} \tilde{b}^{(1)}_{210} &= -\frac{\tilde{b}^{(1)}_{102}}{a^{(1)}_2(p_1)}, \qquad \tilde{b}^{(2)}_{120} = -\frac{\tilde{b}^{(2)}_{012}}{a^{(2)}_2(p_1)} \\ \tilde{b}^{(1)}_{111} &= \frac{\tilde{b}^{(1)}_{102}\left[a^{(1)}_3(p_1) - a^{(1)}_2(p_1)\right] + a^{(1)}_2(p_1)}{\left[a^{(1)}_2(p_1)\right]^2} = 0 \qquad (34) \\ \tilde{b}^{(2)}_{111} &= \frac{\tilde{b}^{(2)}_{012}\left[a^{(2)}_3(p_1) - a^{(2)}_2(p_1)\right] + a^{(2)}_2(p_1)}{\left[a^{(2)}_2(p_1)\right]^2} = 0 \qquad (35) \end{aligned}$$

Since $\tilde{b}^{(1)}_{102} < 0$, $\tilde{b}^{(2)}_{012} < 0$, we require $a^{(l)}_2(p_1) > 0$, $l = 1, 2$. Since $a^{(l)}_2 = \beta^{(l)}_{22}(p_1)\gamma_2(p_1)/2t^{(l)}(p_1)$, we need to have $t^{(l)}(p_1)\gamma_2(p_1) > 0$. Hence $X''(0)$ points to the inside of the polygon. It follows from (34) and (35) that

$$\tilde{b}^{(1)}_{102} = -\frac{a^{(1)}_2(p_1)}{a^{(1)}_3(p_1) - a^{(1)}_2(p_1)}, \qquad \tilde{b}^{(2)}_{012} = -\frac{a^{(2)}_2(p_1)}{a^{(2)}_3(p_1) - a^{(2)}_2(p_1)}$$

In order to satisfy (33), we require

$$3a_3^{(l)}(p_1) - 4a_2^{(l)}(p_1) \geq 0, \quad l = 1, 2$$

These two inequalities, which have three unknowns $\gamma_2(p_1), \delta_2(p_1), \gamma_3(p_1)$, will have infinitely many solutions. Therefore, we have proved the following theorem.

Theorem 2.6. *At each non-convex join point, if X', X'' and X''' are given such that $\gamma_1 = \gamma_2 = 0$, $(-1)^{l-1}t^{(l)}\gamma_3 > 0$ and at each convex join point, if X', X'' and X''' are given such that $\gamma_1 = 0$, $t^{(l)}\gamma_2 > 0$ and $3a_3^{(l)} - 4a_2^{(l)} \geq 0, \quad l = 1, 2$, then G^3 continuous cubic A-splines exist that fit the given data X', X'' and X''' (with possibly different magnitudes).*

G^4 **continuity**: In order to achieve G^4 continuity, we assume each join point is a non-convex join. Consider the curve $F = \sum_{i+j+k=3} b_{ijk}B_{ijk}^n = 0$ on the triangle $[p_1p_2v_1]$. All the coefficients, except b_{111} that is free, are determined as in the G^3 continuity case. Now we use the free b_{111} to achieve G^4 continuity. It follows from (24)–(28) that

$$\tilde{b}_{111}^{(1)} = -\frac{a_4^{(1)}(p_1) - 2a_3^{(1)}(p_1)}{\left[a_3^{(1)}(p_1)\right]^2} = -\frac{a_4^{(2)}(p_2) - 2a_3^{(2)}(p_2)}{\left[a_3^{(2)}(p_2)\right]^2} \tag{36}$$

Since

$$a_4^{(l)}(p_l) = \frac{\beta_{43}^{(l)}(p_l)\gamma_3(p_l) + \beta_{44}^{(l)}(p_l)\gamma_4(p_l)}{24t^{(l)}(p_l)}, \qquad l = 1, 2$$

$a_4^{(l)}(p_l)$ depend linearly on $\gamma_4(p_l)$. Hence (36) can be written as

$$c_1\gamma_4(p_1) + c_2\gamma_4(p_2) = c_3 \tag{37}$$

This system of linear equations always have solutions and has one degree of freedom. Therefore, we have

Theorem 2.7. *If each join point is a non-convex join, and the data $X^{(i)}$, $(i = 1, ..., 4)$ at each join point are given such that*

$$\gamma_1 = \gamma_2 = 0, \quad (-1)^{l-1}t^{(l)}\gamma_3 > 0, \quad l = 1, 2$$

and (37) holds, then the G^4 continuous cubic A-splines exist that fits the given points and derivative data.

2.1.7 Parameterization of general algebraic plane curves by A-splines

In general, a degree d curve can be parameterized if it satisfies the Cayley - Riemann criterion. Consider a curve C_d, with a $d-1$ singular point. By sending that point to infinity, we can draw lines which intersect the curve at one point each. The slopes of these pencil of lines obtains the parameterization of the curve.

An A-spline of degree n over the triangle $[\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3]$ is defined by

$$G_n(x, y) := F_n(\boldsymbol{\alpha}) = F_n(\alpha_1, \alpha_2, \alpha_3) = 0, \tag{38}$$

where

$$F_n(\alpha_1, \alpha_2, \alpha_3) = \sum_{i+j+k=n} b_{ijk}B_{ijk}^n(\alpha_1, \alpha_2, \alpha_3), \quad B_{ijk}^n(\alpha_1, \alpha_2, \alpha_3) = \frac{n!}{i!j!k!}\alpha_1^i\alpha_2^j\alpha_3^k,$$

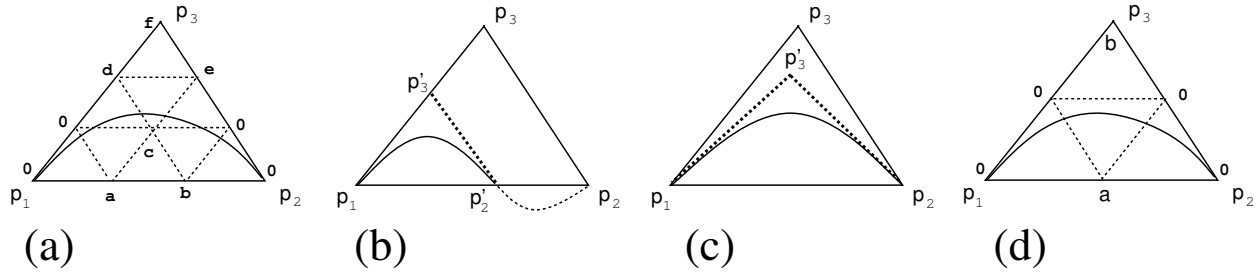

Figure 6: (a): Convex case; (b) Non-convex case; (c) C^0 A-spline; (d) Quadratic A-spline.

and $(x, y)^T$ and $(\alpha_1, \alpha_2, \alpha_3)^T$ are related by

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}. \tag{39}$$

Here the objective is to get an A-spline parameterization of the following form:

$$\mathfrak{X}(t) = \sum_{i=0}^{n} w_i B_i^n(t)\mathbf{b}_i \Big/ \sum_{i=0}^{n} w_i B_i^n(t), \qquad t \in [0,1], \tag{40}$$

where $\mathbf{b}_i \in \mathbb{R}^3$, $w_i \in \mathbb{R}$, and $B_i^n(t) = \{n!/[i!(n-i)!]\}t^i(1-t)^{n-i}$. Without loss of generality, we may assume that $w_0 = 1$ (otherwise we could divide through by t and have a parameterization of one lower degree). Next, under the transformation

$$t = \frac{t' + at'}{1 + at'}, \quad a > -1, \quad t' \in [0,1], \tag{41}$$

the curve (40) will preserve its form, that is

$$\mathfrak{X}(t) = \sum_{i=0}^{n} (1+a)^i w_i B_i^n(t')\mathbf{b}_i \Big/ \sum_{i=0}^{n} (1+a)^i w_i B_i^n(t'), \qquad t' \in [0,1].$$

Therefore, we may assume further that $w_n = 1$ by setting $a = w_n^{-1/n} - 1$, which makes $(1+a)^n w_n = 1$, in the transformation (41).

We consider first convex C^1 continuous A-splines (see Figure 6(a)). An A-spline being C^1 implies that $b_{n00} = b_{0n0} = b_{n-1,01} = b_{0,n-1,1} = 0$, as shown in [30]. The C^0 continuous A-splines on the triangle $[\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3]$ can be made into C^1 continuous A-splines on the triangle $[\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3']$ (see Figure 6(c)) through the use of the subdivision formula (see [53]). In our applications in the parameterization of cubic ($n = 3$) A-patches, the coefficients a, b, c are fixed and d, e, f are parameters to be determined, where

$$a = b_{210}, \quad b = b_{120}, \quad c = b_{111}, \quad d = b_{102}, \quad e = b_{012}, \quad f = b_{003}.$$

The non-convex case (see Figure 6(b)) can be converted to the convex case by first computing the intersection point $\mathbf{p}_2'$, which leads to a linear equation for $n = 3$, and then computing the tangent of the curve at $\mathbf{p}_2'$. Note that this tangent does not depend upon the coefficients d, e, f.

Quadratic A-splines It is not difficult to see that the parametric form of a C^1-continuous quadratic A-spline should have the following form (see Figure 6(d)) since it interpolates the points $\mathbf{p}_1$ and $\mathbf{p}_2$ and is tangent to the lines $[\mathbf{p}_1\mathbf{p}_3]$ and $[\mathbf{p}_2\mathbf{p}_3]$ at the points $\mathbf{p}_1$ and $\mathbf{p}_2$, respectively.

$$\mathfrak{X}(t) = \frac{\mathbf{p}_1 B_0^2(t) + w_1\mathbf{p}_3 B_1^2(t) + \mathbf{p}_2 B_2^2(t)}{B_0^2(t) + w_1 B_1^2(t) + B_2^2(t)}, \qquad t \in [0,1], \tag{42}$$

where w_1 is a parameter to be determined. This is called a (2/2) rational parameterization because the of the numerator and denominator are each of degree 2 in t. In order for the quadratic A-spline to be rationally parameterizable, we must have

$$w_1 = \sqrt{-\frac{b_{110}}{2b_{002}}} \geq 0. \tag{43}$$

Cubic A-splines We first show that an irreducible C^1-continuous cubic A-spline never has a (2/2) rational parameterization. If we substitute the αs defined by (39) into $F_3(\boldsymbol{\alpha}) = 0$, we have $\sum_{i=0}^{6} c_i B_i^6(t) \equiv 0$, where

$$\begin{aligned} &c_0 = b_{300}, \quad c_1 = b_{201}w_1, \quad c_2 = \frac{1}{5}b_{210} + \frac{4}{5}b_{102}w_1^2, \quad c_3 = \frac{3}{5}b_{111}w_1 + \frac{2}{5}b_{003}w_1^3, \\ &c_6 = b_{030}, \quad c_5 = b_{021}w_1, \quad c_4 = \frac{1}{5}b_{120} + \frac{4}{5}b_{012}w_1^2. \end{aligned}$$

Since $B_i^6(t),\ i = 0, \ldots, 6$, are linearly independent, we have $c_i = 0, \quad i = 0, \ldots, 6$. It then follows that

$$a + 4dw_1^2 = 0, \quad 3cw_1 + 2fw_1^3 = 0, \quad b + 4ew_1^2 = 0$$

and hence $w_1 = \sqrt{-a/4d}$. The coefficients of the A-spline must satisfy

$$\frac{d}{a} = \frac{f}{6c} = \frac{e}{b}, \tag{44}$$

where

$$a = b_{210}, \quad b = b_{120}, \quad c = b_{111}, \quad d = b_{102}, \quad e = b_{012}, \quad f = b_{003}.$$

However, the substitutions (44) turn the A-spline $F_3(\boldsymbol{\alpha}) = 3a\alpha_1^2\alpha_2 + 3b\alpha_1\alpha_2^2 + 6c\alpha_1\alpha_2\alpha_3 + 3d\alpha_1\alpha_3^2 + 3e\alpha_2\alpha_3^2 + f\alpha_3^3 = 0$ into $F_3(\boldsymbol{\alpha}) = (\alpha_1\alpha_2 + d/a\ \alpha_3^2)(a\alpha_1 + b\alpha_2 + 2c\alpha_3) = 0$, which is the product of a line and an ellipse. The parameterization covers the ellipse, and is essentially the same as the (2/2) parameterization of a quadratic A-spline.

The (3/3) rational parametric form of a C^1-continuous cubic A-spline should have the following form in order to interpolate the points $\mathbf{p}_1$ and $\mathbf{p}_2$ and be tangent to the lines $[\mathbf{p}_1\mathbf{p}_3]$ and $[\mathbf{p}_2\mathbf{p}_3]$ at $\mathbf{p}_1$ and $\mathbf{p}_2$, respectively:

$$\mathfrak{X}(t) = \frac{\mathbf{p}_1 B_0^3(t) + w_1[\mathbf{p}_1 + \alpha(\mathbf{p}_3 - \mathbf{p}_1)]B_1^3(t) + w_2[\mathbf{p}_2 + \beta(\mathbf{p}_3 - \mathbf{p}_2)]B_2^3(t) + \mathbf{p}_2 B_3^3(t)}{B_0^3(t) + w_1 B_1^3(t) + w_2 B_2^3(t) + B_3^3(t)}, \tag{45}$$

where α, β, w_1, w_2 are parameters to be determined.

We will show that the equation

$$\begin{aligned} G_{[\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3]}(a,b,c,d,e,f) = {} & 48a^3e^3f^2 - 9a^2b^2f^4 + 72a^2bcef^3 - 72a^2bde^2f^2 - 96a^2c^2e^2f^2 \\ & - 288a^2cde^3f + 432a^2d^2e^4 + 72ab^2cdf^3 - 72ab^2d^2ef^2 - 8abc^3f^3 \\ & - 552abc^2def^2 + 1152abcd^2e^2f - 864abd^3e^3 + 48ac^4ef^2 + 576ac^3de^2f \\ & - 864ac^2d^2e^3 + 48b^3d^3f^2 - 96b^2c^2d^2f^2 - 288b^2cd^3ef + 432b^2d^4e^2 \\ & + 48bc^4df^2 + 576bc^3d^2ef - 864bc^2d^3e^2 - 288c^5def + 432c^4d^2e^2 = 0 \end{aligned} \tag{46}$$

gives a condition on the A-spline coefficients that guarantee the A-spline has a rational parameterization. The proof of this is rather technical.

We will wish to construct rationally parameterizable cubic A-splines defined on a triangle $[\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3]$ and passing through $\mathbf{p}_1$ and $\mathbf{p}_2$, that are not necessarily tangent to the edges $[\mathbf{p}_1\mathbf{p}_3]$ and $[\mathbf{p}_2\mathbf{p}_3]$ at $\mathbf{p}_1$ and $\mathbf{p}_2$. This situation is illustrated in Figure 6(c), where the tangent lines at $\mathbf{p}_1$ and $\mathbf{p}_2$ intersect at some other point $\mathbf{p}_3'$. These cubic *A*-splines will have one degree of freedom, the

weight b_{003}, which we will use to satisfy (46). In order to accomplish this we define a coordinate system $\alpha'_1\alpha'_2\alpha'_3$ (where $\alpha'_1+\alpha'_2+\alpha'_3=1$) that has its origin $(0,0,1)$ at $\mathbf{p}'_3$ instead of $\mathbf{p}_3$, while keeping the points $(1,0,0)$ and $(0,1,0)$ fixed.

The general cubic curve passing through $\mathbf{p}_1$ and $\mathbf{p}_2$ is

$$\begin{aligned} &3b_{210}\alpha_1^2\alpha_2+3b_{201}\alpha_1^2\alpha_3+3b_{120}\alpha_1\alpha_2^2+6b_{111}\alpha_1\alpha_2\alpha_3 \\ &+3b_{102}\alpha_1\alpha_3^2+3b_{021}\alpha_2^2\alpha_3+3b_{012}\alpha_2\alpha_3^2+b_{003}\alpha_3^3=0 \ . \end{aligned} \tag{47}$$

The tangent lines to this curve at $\mathbf{p}_1$ and $\mathbf{p}_2$ are

$$\begin{aligned} b_{210}\alpha_2+b_{201}\alpha_3&=0 \\ b_{120}\alpha_1+b_{021}\alpha_3&=0 \ , \end{aligned}$$

and these intersect at the point

$$(\alpha_1,\alpha_2,\alpha_3)=\frac{(b_{210}b_{021},b_{201}b_{120},-b_{210}b_{120})}{b_{210}b_{021}+b_{201}b_{120}-b_{210}b_{120}} \ .$$

The linear transformation that maps
$(\alpha_1,\alpha_2,\alpha_3)=(1,0,0),(0,1,0),(b_{210}b_{021},b_{201}b_{120},-b_{210}b_{120})/(b_{210}b_{021}+b_{201}b_{120}-b_{210}b_{120})$ into $(\alpha'_1,\alpha'_2,\alpha'_3)=(1,0,0),(0,1,0),(0,0,1)$, respectively is

$$\begin{aligned} \alpha_1 &= \alpha'_1+\frac{b_{210}b_{021}}{b_{210}b_{021}+b_{201}b_{120}-b_{210}b_{120}}\,\alpha'_3 \\ \alpha_2 &= \alpha'_2+\frac{b_{201}b_{120}}{b_{210}b_{021}+b_{201}b_{120}-b_{210}b_{120}}\,\alpha'_3 \\ \alpha_3 &= -\frac{b_{210}b_{120}}{b_{210}b_{021}+b_{201}b_{120}-b_{210}b_{120}}\,\alpha'_3 \end{aligned} \tag{48}$$

with the inverse

$$\begin{aligned} \alpha'_1 &= \alpha_1-\frac{b_{021}}{b_{120}}\,\alpha_3 \\ \alpha'_2 &= \alpha_2-\frac{b_{201}}{b_{210}}\,\alpha_3 \\ \alpha'_3 &= -\frac{b_{210}b_{021}+b_{201}b_{120}-b_{210}b_{120}}{b_{210}b_{120}}\,\alpha_3 \end{aligned} \tag{49}$$

Thus the transformation (48) maps (47) into an equation of the form

$$a'{\alpha'_1}^2\alpha'_2+3b'\alpha'_1{\alpha'_2}^2+6c'\alpha'_1\alpha'_2\alpha'_3+3d'\alpha'_1{\alpha'_3}^2+3e'\alpha'_2{\alpha'_3}^2+f'{\alpha'_3}^3=0 \ .$$

Parameterization of algebraic space curves Space curves are projected to planes to obtain plane curves, which are then parameterized by the method described in the previous sections. It is shown that such a projection always exists. We need to obtain a birational map, which maps the space curve to a plane curve which has the same genus.

2.1.8 Inversion of Parameterizations

Parameterizations tell how to map a point in parameter space to a curve. The inversion of this map, called a *chart* in differential topology, tells how to map a point on a curve to its parameter value. Certain assumptions must be made on the parameterization in order for it to have a computable inverse.

Similar to the case of curves, a parametric surface is a very special algebraic variety of dimension 2 in x, y, z, s, t space, since the surface lies in the 3-dimensional subspace defined by x, y, z and furthermore points on the surface can be put in 1-to-1 rational correspondence with points on the 2-dimensional sub-space defined by s, t.

Example parametric (rational algebraic) surfaces are degree two algebraic surfaces (quadrics) and most degree three algebraic surfaces (cubic surfaces). The cylinders of nonsingular cubic curves and the cubic surface cone are of not rational.

Other examples of rational algebraic surfaces are Steiner surfaces which are degree four surfaces with a triple point, and Plücker surfaces which are degree four surfaces with a double curve. In general, a necessary and sufficient condition for the rationality of an algebraic surface of arbitrary degree is given by Castelnuovo's criterion: $P_a = P_2 = 0$, where P_a is the arithmetic genus and P_2 is the second plurigenus [107]. Algorithms for symbolically deriving the parametric equations of degree two and three rational surfaces are given in [4, 5, 92].

Both the parametric and the implicit representation of algebraic curve segments and algebraic surface patches can be represented in either Bernstein-Bézier or B-spline bases.

Rational parametric algebraic surface The canonical representation of a rational parametric algebraic surface patch in x, y, z space are given by

$$\begin{cases} X = P_1(s,t), \\ Y = P_2(s,t), \\ Z = P_3(s,t), \\ W = P_4(s,t). \end{cases}$$

or

$$\begin{cases} x = X/W, \\ y = Y/W, \\ z = Z/W, \end{cases}$$

where the P_i are polynomials in any of the above appropriate bases and the variables/parameters s, and t range over a finite interval (or canonically the unit interval [0,1], see [28]).

The domain of the mapping for rational algebraic parametric surfaces is usually one of the following two kinds:

- Tensor domain: The parameters s, t are defined over the interval [0,1]. ($s \in [0,1], t \in [0,1]$)
- Barycentric domain: This is a triangular domain, with the parameters ranging over a finite interval and satisfying the condition: $0 \leq s, t \leq 1$.

Multi-sided patches:

Base points are isolated pairs of parameter values which satisfy $P_1 = P_2 = P_3 = P_4 = 0$ and hence cause the parametric map to be ill-defined (0/0).

For example, in the hyperboloid of 1 sheet, we see a pair of lines being absent due to the ill-formed mapping.

The image in x, y, z of the base points in the parameter domain, are in general curves, yielding multi-sided patches.

2.2 Standard Rational Representation

Quite often geometric designers and engineers using RBB (Rational Bernstein-Bézier) curve like to have it in a standard form, where the denominator polynomial has only positive coefficients. This assumption is quite strong, but rids the curve of real poles (roots of the denominator polynomial)

and gives the RBB curve its convex hull property. In this gem, we show how to convert a smooth rational curve with no poles in the interval $[a, b]$ into a finite number of C^∞ standard RBB curve segments. We also show that for a degree n smooth rational curve, the number of converted RBB curve segments is bounded above by $\frac{n(n-1)}{2}$.

Without loss of generality, we assume that the given rational curve has no poles in the interval $[a, b]$, since it is straightforward to break the curve first at the poles. An essential step is to transform the rational curve into Bernstein-Bézier (BB) form. Let

$$R(s) = [x(s),\ y(s),\ z(s)]^T/w(s)$$

be a space rational curve on the interval $[a, b]$, where $x(s), y(s), z(s)$ and $w(s)$ are polynomials of degree n. Since

$$t^i = \sum_{j=i}^{n} \frac{C_i^j}{C_i^n} B_j^n(t)$$

where

$$t = \frac{s-a}{b-a} \in [0,1], \quad B_j^n(t) = C_i^n t^j (1-t)^{n-j}, \quad C_i^n = \frac{n!}{i!(n-i)!}$$

we have, for any polynomial $P(s)$ of degree n

$$P(s) = \sum_{i=0}^{n} c_i t^i = \sum_{i=0}^{n} (\sum_{j=0}^{i} \frac{C_j^i}{C_j^n} c_j) B_i^n(t) = \sum_{i=0}^{n} b_i' B_i^n(t)$$

where $b_i' = \sum_{j=0}^{i} \frac{C_j^i}{C_j^n} c_j$. Therefore $R(s)$ can be expressed as a RBB curve over [a,b]

$$R(s) = \sum_{i=0}^{n} w_i b_i B_i^n(t) / \sum_{i=0}^{n} w_i B_i^n(t)$$

where $w_i \in I\!R,\ b_i \in I\!R^3$ are Bézier weights and points, respectively.

2.2.1 Converting a Rational Curve to a Standard Rational Bernstein-Bézier Representation

The remaining step now is to transform the RBB curve into the standard RBB representation, where the denominator polynomial, say $P(t)$, has only positive coefficients.

We now show how the denominator polynomial $P(t) = \sum_{i=0}^{n} w_i B_i^n(t), \quad t \in [0,1]$ is divided over subintervals, say, $0 = t_0 < t_1 < \ldots < t_{l+1} = 1$, such that the BB-form of $P(t)$ on each of the subintervals $P(t)|_{[t_i t_{i+1}]} = P_i(t) = P_i(t_i + (t_{i+1} - t_i)s) = \tilde{P}_i(s) = \sum w_j^i B_i^n(s)$ has positive coefficients. We assume $P(t) > 0$ over [0,1] since the RBB has no poles in [0,1]. Compute the first breakpoint $t_1 = c$ as explained below. The remaining breakpoints can be computed in a similar fashion. By the subdivision formula, $B_i^n(ct) = \sum_{j=0}^{n} B_i^j(c) B_j^n(t)$. We have in [0,c], $(s = ct; \quad t \in [0,1])$

$$\begin{aligned} P(s) = P(ct) &= \sum_{i=0}^{n} w_i B_i^n(ct) \\ &= \sum_{i=0}^{n} w_i \sum_{j=0}^{n} B_i^j(c) B_j^n(t) \\ &= \sum_{j=0}^{n} q_j(c) B_j^n(t) \end{aligned}$$

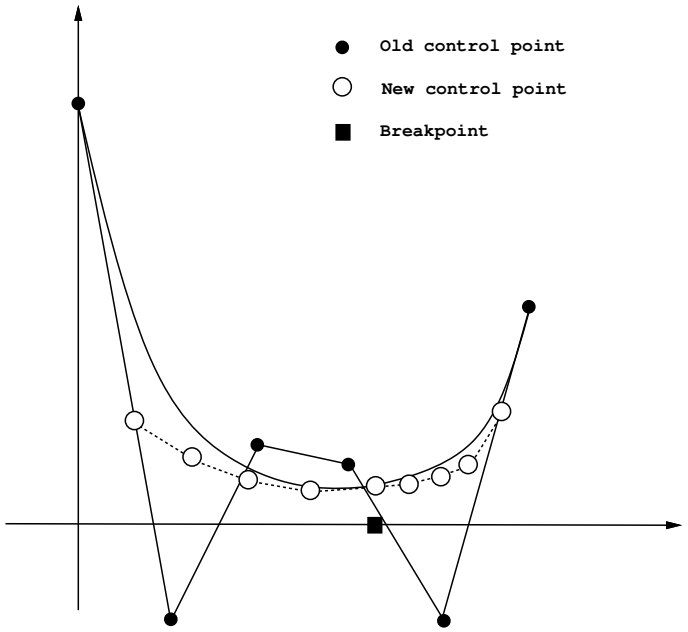

Figure 7: Denominator Polynomial with Positive Bézier Coefficients.

where $q_j(c) = \sum_{i=0}^{j} w_i B_i^j(c)$ is a degree j polynomial in BB form.

Note that the $\lim_{c\to 0}\ q_j(c) = w_0$, since $B_0^j(0) = 1$, $B_i^j(0) = 0$, $i > 0$. But $P(0) = w_0$, and since $P(t) > 0$ for $t \in [0,1]$ we know that $w_0 > 0$. Take $c < \min\{\text{all roots of } \mathrm{q_j(c)} \text{ in } [0,1]\}$ and $c > 0$. This c will guarantee that **all** $q_j(c)$ are positive.

Example 2.8. *Figure 7 shows an example of this conversion for the denominator polynomial* $(1-x)^5 - x(1-x)^4 + 2x^2(1-x)^3 + x^3(1-x)^2 - x^4(1-x) + 0.5x^5$

The initial quintic Bézier coefficients over [0,1] are

$$bb[0] = 1.000000 \quad bb[1] = -0.200000 \quad bb[2] = 0.200000$$
$$bb[3] = 0.100000 \quad bb[4] = -0.200000 \quad bb[5] = 0.500000$$

of which two coefficients are negative. The control points are plotted in Figure 1 with dark colored dots. The above conversion yields two pieces in standard BB form over [0,1] with 0.640072 as the breakpoint. The new coefficients of the two quintic BB pieces are

$$bb[0] = 1.000000 \quad bb[1] = 0.231913 \quad bb[2] = 0.119335$$
$$bb[3] = 0.111575 \quad bb[4] = 0.060781 \quad bb[5] = 0.060781$$

and

$$bb[0] = 0.060781 \quad bb[1] = 0.060781 \quad bb[2] = 0.076842$$
$$bb[3] = 0.125649 \quad bb[4] = 0.248051 \quad bb[5] = 0.500000$$

The new control points are plotted in Figure 1 with circles. The curve itself is, of course, the same.

2.2.2 Upper Bound on RBB pieces

We give an upper bound for the total number of RBB pieces required for a degree n rational curve.

Theorem. *Let* $p(x) = \sum_{i=0}^{n} w_i B_i^n(x), \deg(p) = n$, *and* $p(x) > 0$ *on* $[0,1]$. *Then there exists*

$$0 = x_0 < x_1 < x_1 < \ldots x_\ell < x_{\ell+1} = 1 \tag{1}$$

with

$$\ell \leq \frac{n(n-1)}{2} \tag{2}$$

such that the BB form of $p(x)$ *on* $[x_i, x_{i+1}]$ *has positive and monotonic coefficients.*

Proof. Let $Z = \bigcup_{i=1}^{n-1} \{x : p^{(j)}(x) = 0\}$. Then the cardinality of $Z \leq \frac{n(n-1)}{2}$. Take distinct x_i in $Z \bigcap (0,1)$ and arrange them in increasing order, to obtain (1) and (2). Next subdivide the interval [0,1] into sub-intervals (x_i, x_{i+1}) for $(i = 0, 1, \ldots, \ell)$, such that $p^{(j)}(x)$ has no zero in (x_i, x_{i+1}) for $j = 0, 1, \ldots, n-1$. Let $q_i(t) := p(x_i(1-t) + x_{i+1}t)$. Then $\frac{d^j q_i(t)}{dt^j} = \frac{d^j p(x)}{dx^j}(x_{i+1} - x_i)^j$. Hence $q_i^{(j)}(t)$ has no zero in (0,1) for $j = 0, 1, \ldots, n-1$. Further, $q_i^{(0)}(t)$ has no zero on [0,1] by the earlier assumption.

Now we prove that the BB form $q_i(t) = \sum_{j=0}^{n} w_j^{(i)} B_j^n(t), i = 0, 1, \ldots, \ell$ has positive and monotonic coefficients. In fact we prove a more general conclusion.

Lemma. *If* $q(t)$ *is a polynomial of degree n, and* $q^{(j)}(t)$ *has no zero in the open interval (0,1) for* $j = 0, 1, \ldots, n$, *then the coefficients of the BB form representation* $q^{(j)}(t) = \sum_{i=0}^{n-j} w_i^{(j)} B_i^{n-j}(t)$ *are monotonic and have the same sign for any fixed* j.

Proof. We prove this fact by induction. For $j = n$, $q^{(j)}(t)$ is a nonzero constant and the required conclusion is obviously true. In general, suppose the Lemma is true for $j+1$, then for j we have since $q^{(j)}(t) = \sum_{i=0}^{n-j} w_i^{(j)} B_i^{n-j}(t)$

$$\begin{aligned} q^{(j+1)}(t) &= \sum_{i=0}^{n-j-1} w_i^{(j+1)} B_i^{n-j-1}(t) \\ &= \sum_{i=0}^{n-j-1} \Delta w_i^{(j)} B_i^{n-j-1}(t). \end{aligned}$$

where $\Delta w_i^{(j)} = w_{i+1}^{(j)} - w_i^{(j)}$, i.e., $w_{i+1}^{(j)} - w_i^{(j)} = w_i^{(j+1)}$. Since $w_i^{(j+1)}$ does not change sign, hence $w_i^{(j)}$ is monotonic. But $w_0^{(j)} = q^{(j)}(0)$ and $w_{n-j}^{(j)} = q^{(j)}(1)$ have the same sign. Hence $w_i^{(j)}$ has the same sign and the induction is complete.

Back to the proof of the theorem. We know from the above Lemma that the coefficients $w_j^{(i)}$ of $q_i(t)$ are monotonic for fixed i. Hence they are positive since $w_0^{(i)}$ and $w_n^{(i)}$ are positive.

It should be noted that the partition given in the Theorem guarantees not only positivity but also monotonicity of coefficients. This is often important because this stronger condition on the coefficients prevents the standard RBB representation from having very small positive denominator coefficients.

2.3 Dynamic Curves

2.3.1 Energy Formulations of A-Splines

2.3.2 Arc Length

Recall that cartesian and barycentric coordinates are related by

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} . \tag{50}$$

Equation (50) may be rewritten without the use of α_3 as

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_1 - x_3 & x_2 - x_3 \\ y_1 - y_3 & y_2 - y_3 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} + \begin{bmatrix} x_3 \\ y_3 \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} + \begin{bmatrix} x_3 \\ y_3 \end{bmatrix} \tag{51}$$

$\mathbf{J}$ is the Jacobian of $\boldsymbol{\alpha}$ in terms of x and y:

$$\mathbf{J} = \frac{\partial(\alpha_1, \alpha_2)}{\partial(x, y)} = \frac{1}{\Delta} \begin{bmatrix} y_2 - y_3 & x_3 - x_2 \\ y_3 - y_1 & x_1 - x_3 \end{bmatrix} , \tag{52}$$

where

$$\Delta = \begin{vmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{vmatrix} . \tag{53}$$

Given an implicit function $F(x, y) = 0$, assume that x and y, and thereby the arc length s, are all smooth functions of an independent variable u. Then by the arc length relation $(ds/dx)^2 = 1 + (dy/dx)^2$ and by differentiating

with respect to u to obtain $d\mathbf{p}/du = \mathbf{J}^{-1} d\boldsymbol{\alpha}/du$, we also have

$$\begin{aligned} \left(\frac{ds}{du}\right)^2 &= \left(\frac{dx}{du}\right)^2 + \left(\frac{dy}{du}\right)^2 \\ &= \left(\frac{d\mathbf{p}^T}{du}\right)\left(\frac{d\mathbf{p}}{du}\right) \\ &= \left(\frac{d\boldsymbol{\alpha}^T}{du}\right)\mathbf{J}^{-T}\mathbf{J}^{-1}\left(\frac{d\boldsymbol{\alpha}}{du}\right) \\ &= d_{13}^2\left(\frac{d\alpha_1}{du}\right)^2 + 2c_{12}\left(\frac{d\alpha_1}{du}\right)\left(\frac{d\alpha_2}{du}\right) + d_{23}^2\left(\frac{d\alpha_2}{du}\right)^2 \end{aligned} \tag{54}$$

where

$$\begin{aligned} d_{ij} &= \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \\ c_{12} &= (x_1 - x_3)(x_2 - x_3) + (y_1 - y_3)(y_2 - y_3) \ . \end{aligned}$$

2.3.3 Curvature

Given an implicit function $F(x, y) = 0$, we have $dF = 0$, or

$$F_x dx + F_y dy = 0 \tag{55}$$

Assuming that in a small neighborhood of point (x, y), y is a function of the independent variable x, from (55) we have

$$y_x = -\frac{F_x}{F_y} \tag{56}$$

By differentiating (56) with respect to x, we obtain

$$y_{xx} = -\frac{\partial}{\partial x}\left(\frac{F_x}{F_y}\right) = -\frac{F_{yy}F_x^2 + F_{xx}F_y^2 - 2F_{xy}F_xF_y}{F_y^3} \ .$$

This allows us to express the curvature of the function $F(x,y)=0$, or $S(\alpha_1,\alpha_2)=0$ in BB form, as

$$\begin{aligned}\kappa &= \frac{|y_{xx}|}{(1+y_x^2)^{3/2}} \\ &= \frac{\left|\nabla F^T \mathbf{P}^T \nabla^2 F \mathbf{P} \nabla F\right|}{(\nabla F^T \nabla F)^{3/2}} \\ &= \frac{\left|\nabla S^T \mathbf{J}\mathbf{P}^T \mathbf{J}^T \nabla^2 S \mathbf{J}\mathbf{P}\mathbf{J}^T \nabla S\right|}{(\nabla S^T \mathbf{J}\mathbf{J}^T \nabla S)^{3/2}}\end{aligned} \tag{57}$$

where the permutation matrix $\mathbf{P}$ is defined as

$$\mathbf{P} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \ .$$

Recall that smoothness of certain degrees and local interpolation of certain degrees are enforced by some linear equality constraints

$$\mathbf{b}^T \mathbf{C}(\mathbf{p}) = \mathbf{0} \ , \tag{58}$$

and connectedness of the curve is enforced by additional linear sign inequalities

$$\mathbf{b}^T \mathbf{S} > \mathbf{0} \ . \tag{59}$$

2.3.4 Elastic curves

Let a be the material coordinate of a point on a plane curve C with parameterization $\mathbf{w}(a)$. For a parametric representation $\mathbf{w}(a)$, the elastic potential energy of a deformable curve is as follows:

$$E = \int_C [\beta(a)|\mathbf{w}_a|^2 + \gamma(a)|\mathbf{w}_{aa}|^2]\, da \tag{60}$$

where a is the intrinsic or material coordinate of the curve. Most of the elastic curves that are built on parameterizations other than the intrinsic, are referred to as "elastica" (Jou and Han, 1990a; Mumford, 1994).

The first step of defining the elastic energy of a geometric entity is actually the mapping between the material coordinates and some parameterization of the geometric entity. For a piecewise representation, the joining points of the pieces could be identified as such a parameterization. Namely, each joining point is associated with fixed material coordinates during a deformation process. However, such a parameterization is too coarse. A second level parameterization is needed to describe detail changes, especially when there is substantial freedom within each entity. In the following energy curves, we assume that the material is uniformly distributed along the curve. Thus in place of the material coordinate a we will use the arc length coordinate s, and the elastic potential energy in (60) becomes (Mumford, 1994):

$$E = \int_C (\beta + \gamma\kappa^2)\, ds \ . \tag{61}$$

The terms β and $\gamma\kappa^2$ represent the stretching and bending energy, respectively.

2.3.5 Elastic strain energy model of A-splines

Let $S(\alpha_1,\alpha_2)=0$ be an A-spline defined within a triangle $\Delta\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3$ (see figure 8). The curved piece interpolates $\mathbf{p}_1$ and $\mathbf{p}_3$ and is tangent to $\overline{\mathbf{p}_1\mathbf{p}_2}$ and $\overline{\mathbf{p}_3\mathbf{p}_2}$ at $\mathbf{p}_1$ and $\mathbf{p}_3$ respectively. Let $F(x,y)=0$ be the representation of the spline in Cartesian coordinates. The Cartesian coordinates of $\mathbf{p}_1$,

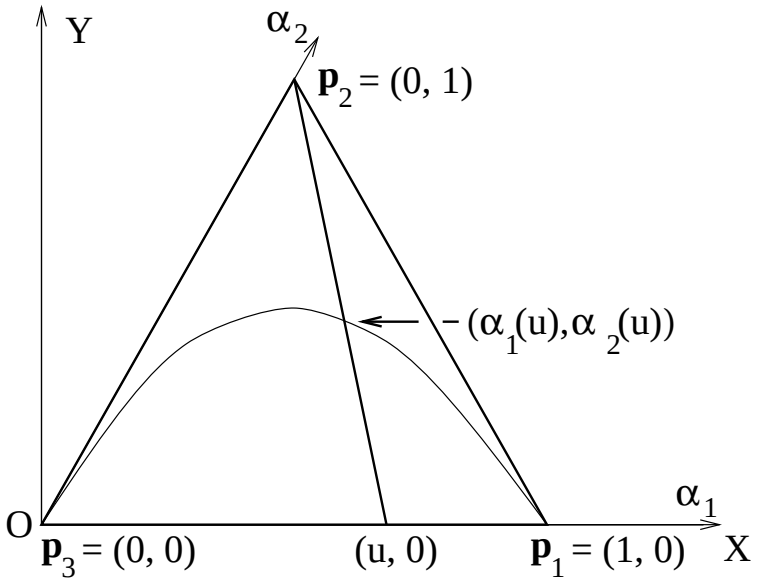

Figure 8: Representation of points in local BB coordinates. Point $\mathbf{p}_i$ has Cartesian coordinates (x_i, y_i), $i = 1, 2, 3$. A point $(\alpha_1(u), \alpha_2(u))$ is given by the intersection of the spline with the line connecting the points $(u, 0)$ and $(0, 1)$ ($\mathbf{p}_2$) in BB coordinates.

$\mathbf{p}_2$ and $\mathbf{p}_3$ are (x_1, y_1), (x_2, y_2) and (x_3, y_3), respectively, and their local barycentric coordinates, suppressing the third coordinate $\alpha_3 = 1 - \alpha_1 - \alpha_2$, are $(1, 0)$, $(0, 1)$, and $(0, 0)$, respectively.

For several purposes, such as computing the energy (61) below, we need to express the spline coordinates as functions of a single parameter, say u. One effective way of doing this is to parametrize the A-spline within a control triangle by finding its point of intersection with a line segment connecting the apex point ($\mathbf{p}_2$) to a point on the base ($\overline{\mathbf{p}_3\mathbf{p}_1}$). It was shown in (Bajaj and Xu, 1992) that such a line segment always intersects the A-spline exactly once when the constraints (58) and (59) are satisfied, and there this technique was used to obtain parameterizations of various quadratic and cubic A-splines. We simply let u parametrize the line segment $\overline{\mathbf{p}_3\mathbf{p}_1}$, so that in BB coordinates, $\overline{\mathbf{p}_3\mathbf{p}_1}$ is given by $(u, 0)$, $0 \leq u \leq 1$.

Now using (61), (57), and (54) we can write the total energy as

$$\begin{aligned} E_{total} &= \int_C (\beta + \gamma\kappa^2)\, ds \\ &= \int_0^1 \left[\left(\frac{d\boldsymbol{\alpha}^T}{du} \right) \mathbf{J}^{-T} \mathbf{J} i \left(\frac{d\boldsymbol{\alpha}}{du} \right) \right]^{1/2} \left(\beta + \gamma\kappa^2 \right)\, du \qquad (62) \\ &= \int_0^1 \left[\left(\frac{d\boldsymbol{\alpha}^T}{du} \right) \mathbf{J}^{-T} \mathbf{J}^{-1} \left(\frac{d\boldsymbol{\alpha}}{du} \right) \right]^{1/2} \left[\beta + \frac{\gamma \left(\nabla S^T \mathbf{J} \mathbf{P}^T \mathbf{J}^T \nabla^2 S \mathbf{J} \mathbf{P} \mathbf{J}^T \nabla S \right)^2}{\left(\nabla S^T \mathbf{J} \mathbf{J}^T \nabla S \right)^3} \right] du \end{aligned}$$

2.3.6 Simplified Elastic Strain Energy Model of A-Splines

Since the integrand, call it $g(u)$, in (62) cannot be integrated symbolically in general, we wish to find an approximation to the total energy that can be computed quickly compared to the time- consuming numerical integration involved with (62). An ideal candidate is apply Simpson's rule. Using just three points in the interest of computational speed, this gives the approximation $E_{total} = [g(0) + 4g(1/2) + g(1)]/6$. However, for many common parameterizations $g(u) \to \infty$ as $u \to 0$ or 1. In this case we make a change of variables to eliminate the singularity, and use a

Simpson's rule approximation to the result.

2.3.7 Energy Optimization

We now consider how to minimize the different energy functions of the earlier section over the constrained degrees of freedom (domain vertices and control weights) of the A-spline curve.

2.3.8 Local Minimization of Total Energy

Here we take into account both the bending and stretching energy as defined in equation (61) and expressed by equation (62). The minimization is obtained locally by varying the free weights of each individual A-spline curve within its triangle.

An energy-minimized setting is a solution to

$$\nabla_{\mathbf{b}} E_{total} = \mathbf{0} \tag{63}$$

or

$$\int_0^1 \nabla_{\mathbf{b}} \left\{ \left[\left(\frac{d\boldsymbol{\alpha}^T}{du} \right) \mathbf{J}^{-T} \mathbf{J}^{-1} \left(\frac{d\boldsymbol{\alpha}}{du} \right) \right]^{1/2} \left[\beta + \frac{\gamma \left(\nabla S^T \mathbf{J} \mathbf{P}^T \mathbf{J}^T \nabla^2 S \mathbf{J} \mathbf{P} \mathbf{J}^T \nabla S \right)^2}{\left(\nabla S^T \mathbf{J} \mathbf{J}^T \nabla S \right)^3} \right] \right\} du = 0 \ .$$

2.3.9 Exact solutions

System 63 is in general a nonlinear system of $\mathbf{b}$. A nonlinear system is not guaranteed to be solvable. However, by restricting the freedom to one variable, we reduce to the system to a univariate nonlinear system, which is easy to solve.

Let $\mathbf{b}$ be a vector function of some parameter t. For an A-spline curve, system 63 is reduced to

$$\frac{dE(t)}{dt} = 0 \ .$$

Let

$$\begin{aligned} f(t) &= \frac{dE(t)}{dt} \\ &= \int_0^1 \frac{d}{dt} \left\{ \left[\left(\frac{d\boldsymbol{\alpha}^T}{du} \right) \mathbf{J}^{-T} \mathbf{J}^{-1} \left(\frac{d\boldsymbol{\alpha}}{du} \right) \right]^{1/2} \left[\beta + \frac{\gamma \left(\nabla S^T \mathbf{J} \mathbf{P}^T \mathbf{J}^T \nabla^2 S \mathbf{J} \mathbf{P} \mathbf{J}^T \nabla S \right)^2}{\left(\nabla S^T \mathbf{J} \mathbf{J}^T \nabla S \right)^3} \right] \right\} du \end{aligned}$$

so that

$$f'(t) = \int_0^1 \frac{d^2}{dt^2} \left\{ \left[\left(\frac{d\boldsymbol{\alpha}^T}{du} \right) \mathbf{J}^{-T} \mathbf{J}^{-1} \left(\frac{d\boldsymbol{\alpha}}{du} \right) \right]^{1/2} \left[\beta + \frac{\gamma \left(\nabla S^T \mathbf{J} \mathbf{P}^T \mathbf{J}^T \nabla^2 S \mathbf{J} \mathbf{P} \mathbf{J}^T \nabla S \right)^2}{\left(\nabla S^T \mathbf{J} \mathbf{J}^T \nabla S \right)^3} \right] \right\} du \ .$$

$f(t)$ and $f'(t)$ are both continuous.

In order to solve these equations, we need a parameterization of the A-spline.

We can use standard methods, such as Newton's method, to solve for the roots. Note that the evaluations of $f(t)$ and $f'(t)$ would involve numerical integrations.

2.3.10 Low degree A-splines

We now proceed to derive parameterizations of linear and quadratic A-splines. Doing the linear case first, we observe that a parameterization of a general line in BB form

$$S(\alpha_1, \alpha_2) = b_{100}\alpha_1 + b_{010}\alpha_2 + b_{001}(1 - \alpha_1 - \alpha_2) = 0 \tag{64}$$

is given by

$$\alpha_1 = u \qquad \alpha_2 = 0 \qquad (\alpha_3 = 1 - u) \ , \tag{65}$$
$$0 \le u \le 1 \ .$$

Then $\nabla S = [0 \ \ 1]^T$ and $\nabla^2 S$ is the 2×2 zero matrix, so $\kappa = 0$ in (57) and (62), which must be the case for a straight line segment. Thus only the stretching energy is present in this formulation. Equation (62) then reduces to

$$E_{total} = \beta d_{13} \ , \tag{66}$$

a multiple of the arc length, here the distance from $\mathbf{p}_1$ to $\mathbf{p}_3$, as expected.

Now we present an example of an energy-minimizing quadratic A-spline with C^1 continuity. First we derive the general equation for such a curve in BB form. The general quadratic spline is $S(\alpha_1, \alpha_2) = b_{200}\alpha_1^2 + 2b_{110}\alpha_1\alpha_2 + 2b_{101}\alpha_1(1-\alpha_1-\alpha_2) + b_{020}\alpha_2^2 + 2b_{011}\alpha_2(1-\alpha_1-\alpha_2) + b_{002}(1-\alpha_1-\alpha_2)^2 = 0$. However, $S(1,0) = 0 \Rightarrow b_{200} = 0$ and $S(0,0) = 0 \Rightarrow b_{002} = 0$. Furthermore, the tangent to $S(\alpha_1, \alpha_2)$ at $(1,0)$ is parallel to the line $\alpha_1 + \alpha_2 = 1$. Therefore $d\alpha_2/d\alpha_1 = -(dS/d\alpha_1)/(dS/\alpha_2) = -1$, or $dS/d\alpha_1 = dS/d\alpha_2$ at that point. This now implies that $b_{110} = 0$. Also, the tangent to $S(\alpha_1, \alpha_2)$ at $(0,0)$ is parallel to the α_2-axis. Therefore $dS/d\alpha_2 = 0$ there, and this implies that $b_{011} = 0$. We now have

$$S(\alpha_1, \alpha_2) = 2b_{101}\alpha_1(1 - \alpha_1 - \alpha_2) + b_{020}\alpha_2^2 = 0 \ , \tag{67}$$

in accordance with Bajaj and Xu (1996).

We now parametrize the quadratic spline as described in Section 2.3.5. Intersecting the curve (67) with the line $\alpha_1 = u(1 - \alpha_2)$ yields (Bajaj and Xu, 1992):

$$\alpha_1(u; \mathbf{b}) = \frac{u}{1 + \sqrt{2b_{101}u(1-u)}} \qquad \alpha_2(u; \mathbf{b}) = \frac{\sqrt{2b_{101}u(1-u)}}{1 + \sqrt{2b_{101}u(1-u)}} \ , \tag{68}$$
$$0 \le u \le 1 \ .$$

Here $\mathbf{b}$ denotes the column vector of the coefficients b_{ijk}, and in this case $\mathbf{b}$ consists of the single element b_{101}.

We find that

$$\mathbf{J}^{-T}\mathbf{J}^{-1} = \begin{bmatrix} d_{13}^2 & c_{12} \\ c_{12} & d_{23}^2 \end{bmatrix} , \qquad \mathbf{JPJ}^T = \begin{bmatrix} 0 & -\frac{1}{\Delta} \\ \frac{1}{\Delta} & 0 \end{bmatrix} ,$$

$$\mathbf{JPJ}^T = \begin{bmatrix} 0 & \frac{1}{\Delta} \\ -\frac{1}{\Delta} & 0 \end{bmatrix} , \qquad \mathbf{JJ}^T = \begin{bmatrix} \frac{d_{23}^2}{\Delta^2} & -\frac{c_{12}}{\Delta^2} \\ -\frac{c_{12}}{\Delta^2} & \frac{d_{13}^2}{\Delta^2} \end{bmatrix} ,$$

$$\left[\left(\frac{d\boldsymbol{\alpha}}{du}^T\right)\mathbf{J}^{-T}\mathbf{J}^{-1}\left(\frac{d\boldsymbol{\alpha}}{du}\right)\right]^{1/2} =$$
$$\left\{4u(1-u)d_{13}^2 + 2b\sqrt{u(1-u)}[(1-2u)(d_{23}^2 - d_{12}^2) + d_{13}^2]\right.$$
$$\left.+b^2[-u(1-2u)d_{12}^2 + u(1-u)d_{13}^2 + (1-u)(1-2u)d_{23}^2]\right\}^{1/2} \Big/ \left\{2\sqrt{u(1-u)}[1 + b\sqrt{u(1-u)}]^2\right\}$$

and that the curvature satisfies

$$\begin{aligned}\kappa^2 &= \frac{\left(\nabla S^T \mathbf{J}\mathbf{P}^T\mathbf{J}^T\nabla^2 S\mathbf{J}\mathbf{P}\mathbf{J}^T\nabla S\right)^2}{\left(\nabla S^T\mathbf{J}\mathbf{J}^T\nabla S\right)^3} \\ &= 4b^2\Delta^2\left[1+b\sqrt{u(1-u)}\right]^6 \Big/ \Big\{4u(1-u)d_{13}^2 + 2b\sqrt{u(1-u)}[(1-2u)(d_{23}^2-d_{12}^2)+d_{13}^2] \\ &\quad + b^2[-u(1-2u)d_{12}^2 + u(1-u)d_{13}^2 + (1-u)(1-2u)d_{23}^2]\Big\}^3 \quad .\end{aligned}$$

Consequently the total energy as given by (62) equals

$$\begin{aligned}E_{total} = \int_0^1 \Big\{\beta\Big(&\{4u(1-u)d_{13}^2 + 2b\sqrt{u(1-u)}[(1-2u)(d_{23}^2-d_{12}^2)+d_{13}^2] \\ &+b^2[-u(1-2u)d_{12}^2 + u(1-u)d_{13}^2 + (1-u)(1-2u)d_{23}^2]\}^{1/2} \Big/ \\ &\left\{2\sqrt{u(1-u)}\left[1+b\sqrt{u(1-u)}\right]^2\right\}\Big) \\ +\gamma\Big[2b^2\Delta^2\left[1+b\sqrt{u(1-u)}\right]^4 \Big/ \Big(&\sqrt{u(1-u)}\,\{4u(1-u)d_{13}^2 \\ &+ 2b\sqrt{u(1-u)}[(1-2u)(d_{23}^2-d_{12}^2)+d_{13}^2] \\ &+b^2[-u(1-2u)d_{12}^2 + u(1-u)d_{13}^2 + (1-u)(1-2u)d_{23}^2]\}^{5/2}\Big)\Big]\Big\}\,du \quad .\end{aligned} \tag{69}$$

Note that as $b \to 0$, the conic spline approaches the line segment from p_1 to p_3, and $E_{total} \to \beta d_{13}$, in accord with (66).

Numerical integration of (69) can be tricky because the integrand, call it $g(b,u)$, goes to infinity as u approaches 0 or 1. It is true that $g(b,u)$ is integrable over $[0,1]$ since infinity is approached as $u^{-1/2}$ as $u \to 0$ and as $(1-u)^{-1/2}$ as $u \to 1$. Numerical evaluation of this integral is made easier with the following changes of variable: Let $u = v^2$ for $u \in [0, 1/2]$ and let $u = 1 - w^2$ for $u \in [1/2, 1]$. These substitutions eliminate the singularities at the endpoints, and we have this equivalent expression for the total energy:

$$
\begin{aligned}
E_{total} \;=\; & \int_0^{1/\sqrt{2}} \Big[\beta\left(\Big\{4v^2(1-v^2)d_{13}^2 + 2bv\sqrt{1-v^2}\left[(1-2v^2)(d_{23}^2-d_{12}^2)+d_{13}^2\right]\right. \\
& \qquad \left. +b^2[-v^2(1-2v^2)d_{12}^2 + v^2(1-v^2)d_{13}^2 + (1-v^2)(1-2v^2)d_{23}^2]\Big\}^{1/2}\Big/ \right. \\
& \qquad \left.\left[\sqrt{1-v^2}\left(1+bv\sqrt{1-v^2}\right)^2\right]\right) \\
& + 4\gamma\Delta^2 b^2\left(\;(1+bv\sqrt{1-v^2})^4\;\Big/\right. \\
& \qquad \left.\sqrt{1-v^2}\Big\{4v^2(1-v^2)d_{13}^2 + 2bv\sqrt{1-v^2}\left[(1-2v^2)(d_{23}^2-d_{12}^2)+d_{13}^2\right]\right. \\
& \qquad \left. +b^2[-v^2(1-2v^2)d_{12}^2 + v^2(1-v^2)d_{13}^2 + (1-v^2)(1-2v^2)d_{23}^2]\Big\}^{5/2}\right)\Big]\,dv \\
+ & \int_0^{1/\sqrt{2}} \Big[\beta\left(\Big\{4w^2(1-w^2)d_{13}^2 + 2bw\sqrt{1-w^2}\left[(1-2w^2)(d_{12}^2-d_{23}^2)+d_{13}^2\right]\right. \\
& \qquad \left. +b^2[(1-w^2)(1-2w^2)d_{12}^2 + w^2(1-w^2)d_{13}^2 - w^2(1-2w^2)d_{23}^2]\Big\}^{1/2}\Big/ \right. \\
& \qquad \left.\left[\sqrt{1-w^2}\left(1+bw\sqrt{1-w^2}\right)^2\right]\right) \\
& + 4\gamma\Delta^2 b^2\left(\;(1+bw\sqrt{1-w^2})^4\;\Big/\right. \\
& \qquad \left.\sqrt{1-w^2}\Big\{4w^2(1-w^2)d_{13}^2 + 2bw\sqrt{1-w^2}\left[(1-2w^2)(d_{12}^2-d_{23}^2)+d_{13}^2\right]\right. \\
& \qquad \left. +b^2[(1-w^2)(1-2w^2)d_{12}^2 + w^2(1-w^2)d_{13}^2 - w^2(1-2w^2)d_{23}^2]\Big\}^{5/2}\right)\Big]\,dw
\end{aligned}
\tag{70}
$$

Case study

Suppose a quadratic curve $F(x,y)=0$ passes through the points $\mathbf{p}_3=(0,0)$ and $\mathbf{p}_1=(1,0)$, and that we are given that the tangent lines at these points have slopes 2 and -3, respectively. Then the intersection of the tangent lines is $\mathbf{p}_2=(3/5,6/5)$. In this case (51) gives $(\alpha_1,\alpha_2)=((2x-y)/2,\,5y/6)$. We take $b_{020}=-1$ as in (Bajaj and Xu, 1992), and take $\beta=1$ and $\gamma=1$. Then $d_{12}=2\sqrt{10}/5$, $d_{13}=1$, $d_{23}=3\sqrt{5}/5$, $\Delta=6/5$, and the integral we wish to minimize is this specialization of (70):

$$
\begin{aligned}
E_{total} \;=\; & \int_0^{1/\sqrt{2}} \left[\left(\left\{4v^2(1-v^2) + 4bv\sqrt{1-v^2}(3-v^2)/5 + b^2[(29v^4-30v^2+9)/5]\right\}^{1/2}\Big/ \right.\right. \\
& \qquad \left.\left.\left[\sqrt{1-v^2}\left(1+bv\sqrt{1-v^2}\right)^2\right]\right) \right. \\
& \left. + (144/25)b^2\left(\;(1+bv\sqrt{1-v^2})^4\;\Big/\right. \right. \\
& \left.\left.\sqrt{1-v^2}\left\{4v^2(1-v^2) + 4bv\sqrt{1-v^2}(3-v^2)/5 + b^2[(29v^4-30v^2+9)/5]\right\}^{5/2}\right)\right]\,dv \\
+ & \int_0^{1/\sqrt{2}} \left[\left(\left\{4w^2(1-w^2) + 4bw\sqrt{1-w^2}(2+w^2)/5 + b^2[(29w^4-28w^2+8)/5]\right\}^{1/2}\Big/ \right.\right. \\
& \qquad \left.\left.\left[\sqrt{1-w^2}\left(1+bw\sqrt{1-w^2}\right)^2\right]\right) \right. \\
& \left. + (144/25)b^2\left(\;(1+bw\sqrt{1-w^2})^4\;\Big/\right. \right. \\
& \left.\left.\sqrt{1-w^2}\left\{4w^2(1-w^2) + 4bw\sqrt{1-w^2}(2+w^2)/5 + b^2[(29w^4-28w^2+8)/5]\right\}^{5/2}\right)\right]\,dw
\end{aligned}
\tag{71}
$$

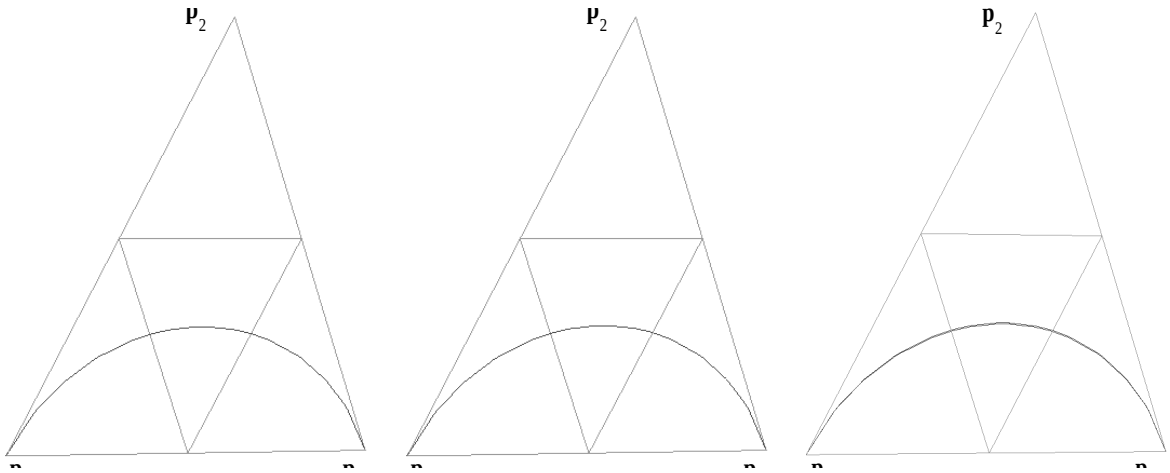

Figure 9: In these figures $\mathbf{p}_1 = (1, 0)$, $\mathbf{p}_2 = (3/5, 6/5)$, and $\mathbf{p}_3 = (0, 0)$ in Cartesian coordinates. (a): The exact energy-minimizing quadratic A-spline (72), with C^1 continuity at the endpoints, obtained when $b_{101} = 0.356$. (b) The simplified energy-minimizing A-spline (75), obtained when $b_{101} = 0.366$. (c) Superposition of the curves in (a) and (b).

The function in (71) could not be integrated symbolically, so it and its derivative with respect to b were integrated numerically for several different values of $b = \sqrt{b_{101}}$. The integral attained its minimum value of 5.63763 at $b = 0.844$, or $b_{101} = 0.356$. This gives an ellipse whose equation in Cartesian coordinates is

$$6.408x^2 - 1.068xy + 5.182y^2 - 6.408x + 3.204y = 0 \; . \tag{72}$$

This elliptic arc is shown in Figure 9(a).

2.3.11 Local Minimization of Simplified Energy

Since the expression (69) for the strain energy is quite complicated, a simplified form of the energy may be desired as indicated in Section 2.3.6. We would like a simple approximation to the integral in (69). This can be tricky because the integrand, call it $g(b, u)$, goes to infinity as u approaches 0 or 1. It is true that $g(b, u)$ is integrable over $[0, 1]$ since infinity is approached as $u^{-1/2}$ as $u \to 0$ and as $(1-u)^{-1/2}$ as $u \to 1$. Numerical evaluation of this integral is made easier with the following changes of variable: Let $u = v^2$ for $u \in [0, 1/2]$ and let $u = (1 - w)^2$ for $u \in [1/2, 1]$. These substitutions eliminate the singularities at the endpoints, and we have this equivalent expression for the total energy:

We now apply Simpson's rule using the three points at v or $w = 0$, $1/2\sqrt{2}$, and $1/\sqrt{2}$. This approximation is best for moderate values of b, say $0.8 \le b \le 4 \leftrightarrow 0.32 \le b_{101} \le 8$, and naturally more accurate approximations may be obtained using more points. We select this approximation

in the interest of computational speed. The result is

$$
\begin{aligned}
E_{simp} = \beta \Bigg(& \frac{b(d_{12}+d_{23})}{6\sqrt{2}} + \frac{2d_{13}}{3(2+b)} \\
& + \frac{32}{3\sqrt{7}(8+\sqrt{7}\,b)^2} \Big\{ [28d_{13}^2 + 4b\sqrt{7}(-3d_{12}^2+4d_{13}^2+3d_{23}^2) + b^2(-6d_{12}^2+7d_{13}^2+42d_{23}^2)]^{1/2} \\
& \qquad + [28d_{13}^2 + 4b\sqrt{7}(3d_{12}^2+4d_{13}^2-3d_{23}^2) + b^2(42d_{12}^2+7d_{13}^2-6d_{23}^2)]^{1/2} \Big\} \Bigg) \\
+ \quad \gamma\Delta^2 \Bigg(& \frac{\sqrt{2}}{3b^3}\left(\frac{1}{d_{12}^5}+\frac{1}{d_{23}^5}\right) + \frac{8b^2}{3(2+b)d_{13}^5} \\
& + \frac{128b^2(8+\sqrt{7}\,b)^4}{3\sqrt{7}} \Big\{ [28d_{13}^2 + 4b\sqrt{7}(-3d_{12}^2+4d_{13}^2+3d_{23}^2) + b^2(-6d_{12}^2+7d_{13}^2+42d_{23}^2)]^{-5/2} \\
& \qquad + [28d_{13}^2 + 4b\sqrt{7}(3d_{12}^2+4d_{13}^2-3d_{23}^2) + b^2(42d_{12}^2+7d_{13}^2-6d_{23}^2)]^{-5/2} \Big\} \Bigg).
\end{aligned}
\tag{73}
$$

To illustrate the accuracy of the simplified energy, we present a couple examples, one with an equilateral control triangle and one where angle $\angle \mathbf{p}_1\mathbf{p}_2\mathbf{p}_3$ is obtuse. The first example has $d_{12} = d_{13} = d_{23} = 1$, $\Delta = \sqrt{3}/2$. We obtain

		True	Simplified	True	Simplified
b	b_{101}	Stretching Energy		Bending Energy	
0.6	0.180	1.120	1.126	5.000	5.419
0.8	0.320	1.166	1.172	3.941	3.984
1.0	0.500	1.209	1.215	3.628	3.646
1.5	1.125	1.304	1.316	3.841	3.792
2.0	2.000	1.380	1.408	4.500	4.297
3.0	4.500	1.494	1.580	6.158	5.680
4.0	8.000	1.573	1.749	7.958	7.323

The simplified stretching and bending energies are within approximately 10% of the true energies for $0.6 < b < 4.0$, or $0.18 < b_{101} < 8.00$. This covers most of the splines occurring in actual practice.

In the second example, $d_{12} = d_{23} = 1$, $d_{13} = \sqrt{3}$, $\Delta = \sqrt{3}/2$. Here we get

		True	Simplified	True	Simplified
b	b_{101}	Stretching Energy		Bending Energy	
0.8	0.320	1.770	1.775	1.043	1.613
1.0	0.500	1.780	1.785	0.804	1.014
1.5	1.125	1.804	1.813	0.619	0.635
2.0	2.000	1.824	1.846	0.611	0.613
3.0	4.500	1.854	1.931	0.725	0.727
4.0	8.000	1.876	2.041	0.890	0.879

In this example the simplified energies are accurate to within 20% for $1 \le b \le 4$, or $0.5 \le b_{101} \le 8$.

Case Study

Using the same setup as in the example in Section 2.3.10, we find that we need to minimize this

specialization of (73):

$$\begin{aligned} & \frac{b(4+3\sqrt{2})}{12\sqrt{5}} + \frac{2}{3(2+b)} + \frac{32\left[(140+92\sqrt{7}\,b+365b^2)^{1/2}+(140+68\sqrt{7}\,b+317b^2)^{1/2}\right]}{3\sqrt{35}(8+\sqrt{7}\,b)^2} \\ + \quad & \frac{12}{25}\left\{\frac{25\sqrt{5}}{b^3}\left(\frac{1}{128}+\frac{\sqrt{2}}{243}\right)+\frac{8b^2}{2+b}\right. \\ & \left. + \frac{128b^2(8+\sqrt{7}\,b)^4}{\sqrt{35}}\left[\frac{1}{(140+92\sqrt{7}\,b+365b^2)^{5/2}}+\frac{1}{(140+68\sqrt{7}\,b+317b^2)^{5/2}}\right]\right\} \quad . \end{aligned} \tag{74}$$

This function is minimized when $b = 0.856$, or $b_{101} = 0.366$, and the minimum value of the simplified energy is 5.66611. This gives an ellipse whose equation in Cartesian coordinates is

$$6.588x^2 - 1.098xy + 5.152y^2 - 6.588x + 3.294y = 0 \; . \tag{75}$$

This elliptic arc is shown with in Figure 9(b), and is quite close to the arc obtained by using the exact representation, as is evidenced by the superposition of the two curves in Figure 9(c).

2.3.12 Global Minimization of Simplified Energy

Here we consider the minimization of the simplified energy by varying the domain endpoint vertices of a chain of quadratic A-spline curves. With the chain we are minimizing a sum of expressions (73) instead of just a single one. Suppose we have $n+1$ junction points on a curve, and we wish to pass a spline through all these points. Label them $\mathbf{p}_1$, $\mathbf{p}_3 = \mathbf{p}_1'$, $\mathbf{p}_3' = \mathbf{p}_1''$, $\ldots$, $\mathbf{p}_3^{(n-1)} = \mathbf{p}_1^{(n)}$, $\mathbf{p}_3^{(n)}$, where superscript ^{i} denotes i primes and $\mathbf{p}_3^{(n)} = \mathbf{p}_1$ if and only if the curve is to be a closed contour. The apex points $\mathbf{p}_2^{(i)}$ will be the intersections of the tangent lines through $\mathbf{p}_1^{(i)}$ and $\mathbf{p}_3^{(i)}$. Recognizing that the d_{jk} and Δ are functions of the coordinates of $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$, we can express the simplified energy in (73) as

$$E_{simp}^{(i)} = E_{simp}(b_i, \mathbf{p}_1^{(i)}, \mathbf{p}_2^{(i)}, \mathbf{p}_3^{(i)}; \beta, \gamma) \quad . \tag{76}$$

Thus our objective is to minimize

$$E_{simp} = \sum_{i=0}^{n-1} E_{simp}^{(i)}$$

for an open contour and

$$E_{simp} = \sum_{i=0}^{n} E_{simp}^{(i)}$$

for a closed one over all possible locations of the $\mathbf{p}_j^{(i)}$ and values of b_i.

Case Study

As a simple example, suppose we have $(1,0)$ and $(-1,0)$ as two fixed points on the unit circle, and we wish to find the point (x_0, y_0) on the unit upper semicircle such that the simplified energy is minimized. This will require the sum of two components, the first of which has $\mathbf{p}_1 = (x_1, y_1) = (1,0)$ and $\mathbf{p}_3 = (x_0, y_0)$, and the second with $\mathbf{p}_1' = (x_1', y_1') = (x_0, y_0)$ and $\mathbf{p}_3' = (-1,0)$ (see Figure 10).

The points $\mathbf{p}_2 = (x_2, y_2)$ and $\mathbf{p}_2' = (x_2', y_2')$ are the intersections of the tangent lines to the circle through $\mathbf{p}_1$ and $\mathbf{p}_3$ and through $\mathbf{p}_1'$ and $\mathbf{p}_3'$, respectively; these will be $(1, (1-x_0)/y_0)$ and $(-1, (1+x_0)/y_0)$ in the two cases. Since all the points can be expressed in terms of x_0 via $y_0 = (1-x_0^2)^{1/2}$, this problem is a minimization over the three remaining unknowns x_0, b, and b', with the restrictions $|x_0| < 1$ and $b, b' > 0$.

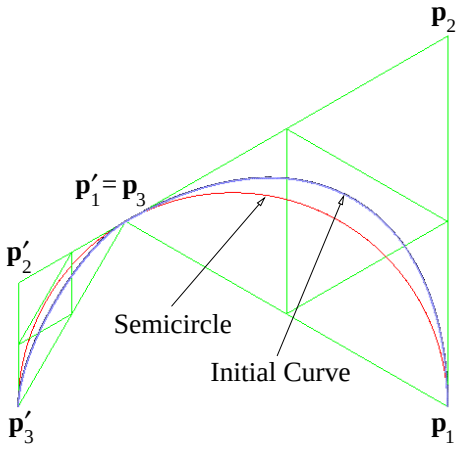

Figure 10: Initial configuration. The point $\mathbf{p}_3 = \mathbf{p}_1'$ is allowed to slide along the semicircle. The spline labeled "initial curve" is a typical minimal-energy spline if $\mathbf{p}_3 = \mathbf{p}_1'$ is at the location indicated.

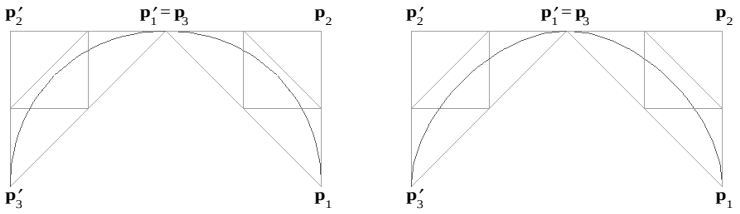

Figure 11: Optimal configurations for (a): $\beta = 1/12, \gamma = 1$; (b): $\beta = 1, \gamma = 1/12$.

We illustrate the results for $\beta = 1/12$, $\gamma = 1$ and for $\beta = 1$, $\gamma = 1/12$. In the first case, the global minimum occurs when $x_0 = 0$ and $b = b' = 1.501$ ($b_{101} = 1.126$), yielding a total energy of 3.414. For $\Delta\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3$, we have $\alpha_1 = 1 - y$ and $\alpha_2 = x + y - 1$ (from (51)), and the Cartesian equation of this portion of the arc, that of an ellipse, is given by

$$x^2 - 0.252xy + y^2 + 0.252x + 0.252y - 1.252 = 0 \ .$$

The second part of the spline, the piece within $\Delta\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3$, is the reflection of the first piece across the y-axis. In the second case, the global minimum is at $x_0 = 0$ and $b = b' = 1.077$ ($b_{101} = 0.580$), yielding a total energy of 3.382. The Cartesian equation of the part of the arc in the first quadrant is

$$x^2 + 0.840xy + y^2 - 0.840x - 0.840y - 0.160 = 0 \ .$$

These two cases are shown in Figure 11. Note that as β increases with respect to γ, the stretching component of the total energy becomes more important than the bending component, and as a result the length of the simplified energy minimizing spline in 11(b) is less than that in 11(a).

Case Study

For another example, we consider the problem of minimizing the total simplified energy of a closed contour with one point on each of the sides of the triangle $\triangle\mathbf{p}_2\mathbf{p}_2'\mathbf{p}_2''$, where $\mathbf{p}_2 = (0, 0)$, $\mathbf{p}_2' = (7, 0)$,

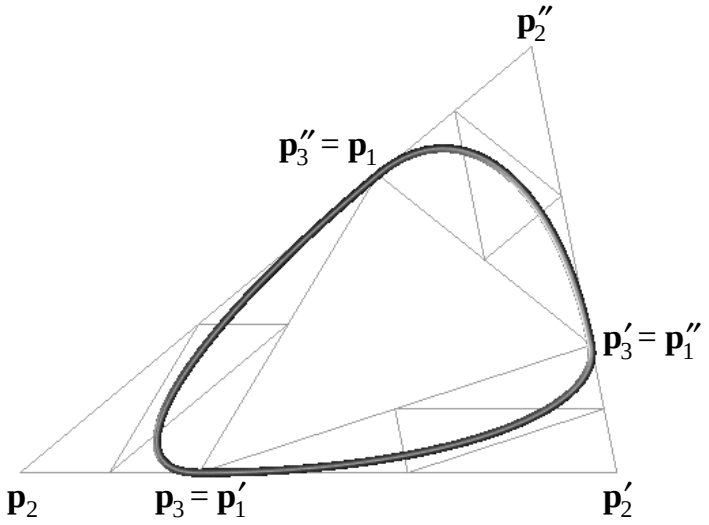

Figure 12: Initial configuration. The points $\mathbf{p}_1$, $\mathbf{p}_1'$, and $\mathbf{p}_1''$ slide along the edges of $\triangle \mathbf{p}_2\mathbf{p}_2'\mathbf{p}_2''$ in such a way that $\overline{\mathbf{p}_2\mathbf{p}_1'}/\overline{\mathbf{p}_2\mathbf{p}_2'} = \overline{\mathbf{p}_2'\mathbf{p}_1''}/\overline{\mathbf{p}_2'\mathbf{p}_2''} = \overline{\mathbf{p}_2''\mathbf{p}_1}/\overline{\mathbf{p}_2''\mathbf{p}_2} = u$ for some u in $[0, 1]$.

and $\mathbf{p}_2'' = (6, 5)$. The three points on the sides of $\triangle \mathbf{p}_2\mathbf{p}_2'\mathbf{p}_2''$ will be denoted by $\mathbf{p}_3 = \mathbf{p}_1'$, $\mathbf{p}_3' = \mathbf{p}_1''$, and $\mathbf{p}_3'' = \mathbf{p}_1$, as in Figure 12. We will also impose the condition that each of $\mathbf{p}_1$, $\mathbf{p}_1'$, and $\mathbf{p}_1''$ is the same fraction along the way of their respective edges. That is, $\overline{\mathbf{p}_2\mathbf{p}_1'}/\overline{\mathbf{p}_2\mathbf{p}_2'} = \overline{\mathbf{p}_2'\mathbf{p}_1''}/\overline{\mathbf{p}_2'\mathbf{p}_2''} = \overline{\mathbf{p}_2''\mathbf{p}_1}/\overline{\mathbf{p}_2''\mathbf{p}_2} = u$ for some u in $[0, 1]$. With this condition the points on the edges of $\triangle \mathbf{p}_2\mathbf{p}_2'\mathbf{p}_2''$ are found to be $\mathbf{p}_1 = (6 - 6u, 5 - 5u)$, $\mathbf{p}_1' = (7u, 0)$, and $\mathbf{p}_1'' = (7 - u, 5u)$. With all points expressed in terms of u, minimizing the total simplified energy amounts to minimizing (76) over the four unknowns u, b, b', and b'', where $0 < u < 1$ and the $b^{(i)} > 0$.

We illustrate the results for various values of β/γ:

β/γ	1/10	1/3	1	3	10
u	0.5111	0.5109	0.5092	0.5057	0.5030
b_{101}	0.318	0.264	0.189	0.123	0.074
b'_{101}	0.916	0.789	0.604	0.409	0.243
b''_{101}	0.627	0.564	0.456	0.322	0.196

Figure 13 shows the resulting closed contour for these five cases. Here we have the same phenomenon which occurred in the previous example: as β/γ increases, the stretching energy component becomes more important as compared to the bending component, and the length of the closed contour for the simplified energy minimizing spline decreases. Also, for each value of β/γ we have $b_{101} < b'_{101} < b''_{101}$, so that the ordering of these coefficients is the same as the order of the size of the angle of the apexes of their corresponding triangles ($\angle \mathbf{p}_1^{(i)}\mathbf{p}_2^{(i)}\mathbf{p}_3^{(i)}$).

2.3.13 Conclusion

Several elastic models using A-splines have been proposed, each of which has its own advantages and shortcomings. Besides the traditional energy model adapted from theory of elasticity, we give several different simplified models that take advantage of the A-spline formulation and also yield

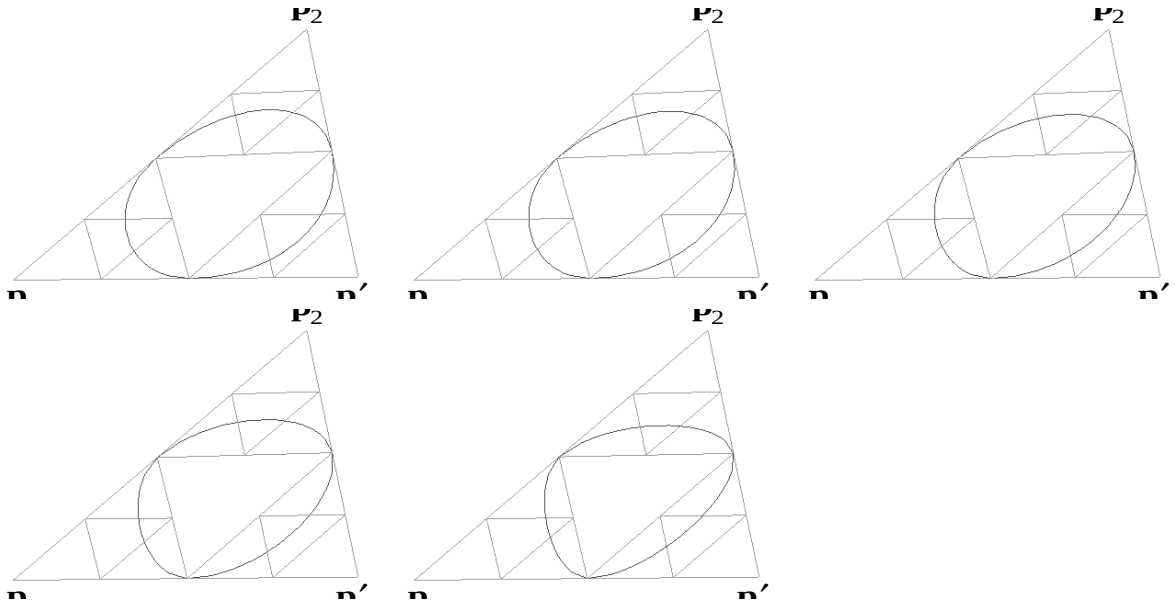

Figure 13: Optimal configurations for (a): $\beta = \gamma/10$; (b): $\beta = \gamma/3$; (c): $\beta = \gamma$; (d): $\beta = 3\gamma$; (e): $\beta = 10\gamma$.

efficient computation of the minimum energy solution. A subsequent paper will report the use of these energy splines in image processing applications.

One problem of great interest is the efficient computation of the minimum energy for the exact model presented in Section 2.3.9. If such a problem can be restricted to having just one control weight being free, then the energy-minimizing problem reduces to a non-linear univariate equation. In the much more common situation where this is not possible, one possibility is to generate a piecewise linear approximation of the A-spline such that the denominator in the expression (62), $(\nabla S^T \mathbf{J}\mathbf{J}^T \nabla S)^3$, over each piece is nearly constant. While the resulting system is sparse and may be solved iteratively, generally a large number of pieces will be required. For example, in the case study in Section 2.3.10, the denominator ranges from 0.254 when $\alpha_1 = 0$ down to 0.110 at $\alpha_1 = 0.535$ and back up to 0.719 when $\alpha_1 = 1$. Thus if we wanted the denominator to vary by at most 1 percent over each subinterval and were able to divide the interval $[0, 1]$ of the α_1-axis at precisely the right points, we would need 133 subintervals. If we were willing to relax the condition to a 5 percent variance, we would still need 27 subintervals. Furthermore, finding the values of α_1 at which the break points should be located is in itself a significant problem. In practice these locations will not be known beforehand, and one may have to make a conservative subdivision of the interval $\alpha_1 \in [0, 1]$ to ensure the desired accuracy.

2.4 Interpolation with Cubic Algebraic Curves

In this section, we focus on implicitly defined cubic algebraic curves, and give conditions on the coefficients of cubic algebraic curves that guarantee nice properties inside regions bounded by triangles. These conditions can be equally applied to cubic curves in the restricted or the general basis.

Paluszny and Patterson [78] considered a special family of implicit cubic curves which yields tangent continuous cubic splines. Our method here differs in that both tangents and curvatures are specified and the splines are not limited to be convex inside the bounding triangles. Bajaj and Xu [29] show how to construct C^3 continuous cubic algebriac splines, however their method is not

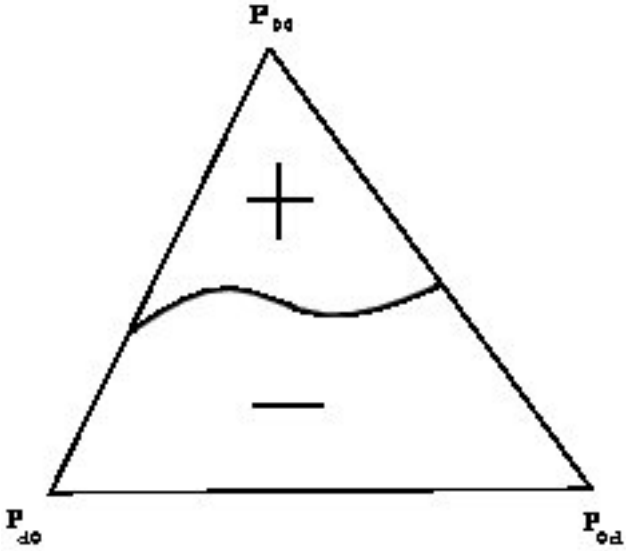

Figure 14: An Effective Spline Curve

directly applicable for symmetric restricted bases.

A *general* [1] cubic algebraic curve in the Bernstein basis is defined as $B^3(u,v) = \sum_{i+j\leq 3} w_{ij} B^3_{ij}(u,v) = 0$. (For introduction to barycentric coordinates, see [52].) Sederberg [89] proposed to view an algebraic curve as the intersection of the explicit surface $w = B^d(u,v)$ with the plane $w = 0$, hoping to associate geometric meanings to the coefficients of the polynomial. Especially, the coefficients in the polynomial are considered as the w coordinates of the control net of a triangular Bernstein-Bézier surface patch, where the coefficient w_{ij} corresponds to the control point $b_{ij} = (\frac{i}{3}, \frac{j}{3})$ in the Bernstein basis. The coefficients w_{ij} is relative to selection of a control triangle $\mathcal{T} = (P_{00}, P_{30}, P_{03})$ in the power basis. There are ten coefficients, and since dividing the equation out by a nonzero number would not change the algebraic curve, we see that there are nine degrees of freedom. For symmetric *restricted* cubic algebraic curves in the Bernstein basis there are only five degrees of freedom. Hence, three degrees of freedom are left after C^2 interpolation with general cubic algebraic curves, and one for C^1 interpolation with restricted cubics.

Computation of Effective Cubic Algebraic Spline Curves We describe in some detail the case of C^2 continuous general algebraic cubic splines. Computation of C^1 continuous restricted algebraic cubic splines can be achieved along similar lines. Let $C_{B_0}(t)$ and $C_{B_1}(t)$ be two truncated power series of degree two that describe geometric properties at two points π_0 and π_1, respectively. One of goals we try to accomplish is to find a triangle within which a single connected smooth segment of a cubic algebraic curve is confined such that the curve segment achieves C^2 continuity at π_0 and π_1 and subdivides the triangle into a positive and a negative space. (See Figure 2.4.)

Definition 2.9. *Let $\mathcal{T}$ be a triangle made of three vertices P_{00}, P_{d0}, P_{0d}. Consider a smooth curve segment of degree n on $B^d(u,v) = 0$ whose two end points are on the two sides $\overline{P_{00}P_{d0}}$ and $\overline{P_{00}P_{0d}}$. The curve segment is called* an effective algebraic spline associated with the bounding triangle $\mathcal{T}$ *if the curve segment intersects exactly once a line segment connecting P_{00} and any point on the side $\overline{P_{d0}P_{0d}}$.*

The restriction imposed in the definition of an effective spline reomves disconnected curve segments, loops, unwanted extra pieces and singularities from within the bounded triangle. It also

[1] We use the adjectives *general* and *restricted* to distinguish cubic algebraic curves in the general and the restricted bases, respectively.

forces the spline curve segment to subdivide a bounding triangle into a positive and a negative space. The ability of finding an effective spline with a proper bounding triangle is essential in that it allows easy implementations of many geometric modeling operations [11]. A point can be easily classified as in, out, or on the boundary of an object that is made of several algebraic splines. This point-classification operation is a primitive operation to high level geometric modeling operations.

For a spline curve segment that is C^2 continuous at the end points π_0 and π_1 within the triangle $\mathcal{T}$, interpolation of the respective truncated power series at these points with a cubic polynomial generates six constraints, leaving three degrees of freedom. After solving the homogeneous linear system with ten unknowns, and six linearly independent constraints, the ten coefficients can be expressed in terms of linear functions in four free parameters λ_0, λ_1, λ_2, and λ_3. We next set up constraints on these free parameters such that for feasible values of λ_i, $i = 0, 1, 2, 3$, the curve segment is a single piece within $\mathcal{T}$. Note that the feasible values of λ_i, $i = 0, 1, 2, 3$, are those for which the triangular Bernstein-Bezier surface patch corresponding to $\mathcal{T}$ intersects the plane $w = 0$ within $\mathcal{T}$ *exactly once* and as shown in Figure 15.

Lemma 2.10. *Let ten coefficients w_{ij} of $B^3(u, v)$ be expressed linearly in terms of λ_j, $j = 0, 1, 2, 3$ after C^2 interpolation of $C_{B_0}(t)$ and $C_{B_1}(t)$ at π_0 and π_1, respectively, with respect to a control triangle $\mathcal{T}$. Then, there exists an effective cubic algebraic spline associated with $\mathcal{T}$ if and only if there exists some λ_j, $j = 0, 1, 2, 3$ such that the univariate cubic polynomial $G(x) \stackrel{\text{def}}{=} B^3((1-\alpha)x, \alpha x) = g_3(\alpha)x^3 + g_2(\alpha)x^2 + g_1(\alpha)x + g_0(\alpha)$ has one and only one root in $0 \le x \le 1$ for all $\alpha \in [0, 1]$. The $g_i(\alpha)$, $(i = 0, 1, 2, 3)$ are polynomials of degree i in α with coefficients which are linear relations on w_{ij} and hence of the free parameters λ_j, $(j = 0, 1, 2, 3)$.*

PROOF : See [20]. □

Due to the limited space, we now present only the final results Details can be found in [20]. Consider the three cases where $h_i(\alpha)$, $(i = 0, 1, 2, 3)$ is a degree $3 - i$ polynomial in α and a linear combination of the above $g_i(\alpha)$ polynomials. The coefficients of $h_i(\alpha)$ are linear combinations of the free parameters λ_j, $(j = 1, 2, 3)$:

- `[CASE 1]` $h_3(\alpha) = 1 > 0$, $h_2(\alpha)^2 - 3h_3(\alpha)h_1(\alpha) \le 0$, $h_0(\alpha) < 0$
- `[CASE 2]` $h_3(\alpha) = 1 > 0$, (either $h_2(\alpha) \ge 0$ or $h_1(\alpha) \le 0$), $h_0(\alpha) < 0$
- `[CASE 3]` $h_3(\alpha) = 1 > 0$, $h_2(\alpha) < 0$, $h_1(\alpha) > 0$, $h_0(\alpha) < 0$, $h_2(\alpha)^2 - 3h_3(\alpha)h_1(\alpha) > 0$, $(-27h_0(\alpha)h_3(\alpha)^2 + 9h_1(\alpha)h_2(\alpha)h_3(\alpha) - 2h_2(\alpha)^3) > 0$, $(27h_0(\alpha)^2h_3(\alpha)^2 - 18h_0(\alpha)h_1(\alpha)h_2(\alpha)h_3(\alpha) + 4h_1(\alpha)^3h_3(\alpha) + 4h_0(\alpha)h_2(\alpha)^3 - h_1(\alpha)^2h_2(\alpha)^2) > 0$

Theorem 2.11. *Let ten coefficients w_{ij} of $B^3(u, v)$ be expressed linearly in terms of λ_j, $j = 1, 2, 3$ with $w_{00} = 1$ after C^2 interpolation of $C_{B_0}(t)$ and $C_{B_1}(t)$ at π_0 and π_1, respectively, with respect to a control triangle $\mathcal{T}$. Then, there exists an effective cubic algebraic spline associated with $\mathcal{T}$ if and only if there exists some λ_j, $j = 1, 2, 3$ such that, for all $\alpha \in [0, 1]$, either* `[CASE 1]`, `[CASE 2]`, *or* `[CASE 3]` *is satisfied.*

Theorem 2.11 generates inequality constraints whose expressions are linear, quadratic, cubic, and quartic in λ_1, λ_2, λ_3. Hence, all the feasible solutions (λ_1, λ_2, λ_3) of those constraints comprise a union of subspaces in the three dimensional $\lambda_1\lambda_2\lambda_3$ solution space bounded by linear, quadratic, cubic, or quartic algebraic surfaces. Choosing an effective cubic algebraic spline associated with a bounding triangle becomes equivalent to finding feasible points in these subspaces. In our implementation we currently use standard nonlinear numerical optimization techniques to compute feasible solutions. Given the low dimensionality of the solution space and the bounded degree of the constraints, we are currently experimenting with symbolic methods which yield a cell decomposition of the feasible region for easy solution point generation and navigation.

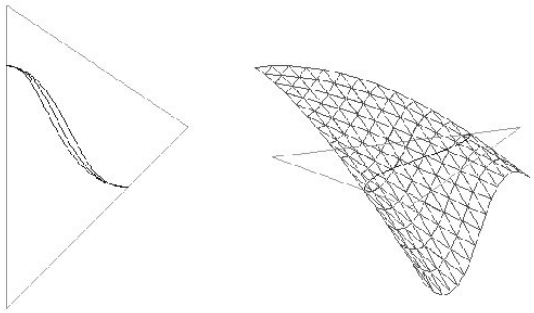

Figure 15: C^2 Continuous Cubic Algebraic Spline Curves

Example 2.12. *Figure 15(a), shows three instances of cubic algebraic curves that C^2 interpolate the two endpoint truncated power series $C_0(t) = (1+t, t^2)$ and $C_1(t) = (t, 1-2t^2)$ with respect to $\mathcal{T} = ((0.0, -1.0), (1.5, 0.5), (0.0, 1.5))$. The three curves chosen from the four dimensional space are $f_0(x,y) = 0.757333x^3 - 1.19933x^2y - 0.768667x^2 + 0.534667xy^2 + 0.2xy - 0.734667x + 0.004y^3 - 0.246y^2 - 0.504y + 0.746$, $f_1(x,y) = 4.08x^3 - 7.37x^2y - 5.99x^2 + 0.06xy^2 + 0.2xy - 0.26x - 1.42y^3 - 1.67y^2 + 0.92y + 2.17$, and $f_2(x,y) = 0.421333x^3 - 0.575333x^2y - 0.240667x^2 + 0.582667xy^2 + 0.2xy - 0.782667x + 0.148y^3 - 0.102y^2 - 0.648y + 0.602$. As C^2 continuity implies, $f_i(C_j(t)) = O(t^3)$, $i = 0, 1, 2$, $j = 0, 1$. Figure 15(b) illustrates how a cubic Bernstein surface patch intersects once with the bounding triangle to produce an effective cubic algebraic spline.*

3 Operations on Spline Curve Geometric Models

3.1 Geometric Models

Algebraic Boundary Model In a boundary representation an object with general algebraic surfaces consists of the following:

- A finite set of vertices usually specified by Cartesian coordinates.
- A finite set of directed edges, where each edge is incident to two vertices. Typically, an edge is specified by the intersection of two faces, one on the left and one on the right. Here left and right are defined relative to the edge direction as seen from the exterior of the object. Further an interior point is also provided on each edge which helps remove any geometric ambiguity in the representation for high degree algebraic curves. Geometric disambiguation may also be achieved by adding tangent and higher derivative information at singular vertices.
- A finite set of faces, where each face is bounded by a single oriented cycle of edges. Each face also has a surface equation, represented either in implicit or in parametric form. The surface equation has been chosen such that the gradient vector points to the exterior of the object.

In addition edge and face adjacency information is provided. Additional conventional assumptions are also made, e.g., edges and faces are non-singular, two distinct faces intersect only in edges, an auxiliary surface is specified for each edge where adjacent faces meet tangentially, etc. The objects and obstacles that we consider are *solids* and are assumed to enclose non-zero finite volume. Hence non-regularities such as dangling edges and dangling faces which depending on one's viewpoint enclose zero or infinite volume, are not permitted. The *C-spaces* that we construct are also regularized in this fashion and assumed to be solids enclosing non-zero finite volume.

Gaussian Model Let S^2 be the unit sphere in R^3, and $Bdr(T)$ be the boundary of a convex set $T \subset R^3$. For any set $K \subset Bdr(T)$, we shall define a set $N(T, K) \subset S^2$ as follows. A point

$e \in S^2$ belongs to $N(T,K)$ if there exists a point $p \in K$ and a supporting plane L_p at p such that e is an exterior normal to L_p. This set $N(T,K)$ is called the *Gaussian Image* of K. The function $N(T,\cdot) : P(Bdr(T)) \to P(S^2)$ is called the *Gaussian Map* of T, where $P(Bdr(T))$ and $P(S^2)$ are the power sets of $Bdr(T)$ and S^2. It is a bijective map and its inverse $N^{-1}(T,\cdot) : P(S^2) \to P(Bdr(T))$ is called the *Inverse Gaussian Map* of T. For any set $G \subset S^2$, the *Inverse Gaussian Image* of G is defined by $N^{-1}(T,G)$. The *Gaussian Curvature* of $p \in Bdr(T)$ is the limit of the ratio (Area of $N(T,K)$) / (Area of K) as K shrinks to the point p, see [65].

Gaussian Image of Faces, Edges and Vertices Since all faces are patches of algebraic surfaces, we may assume that each face of a convex object is either a strictly convex face (*Gaussian Curvature* is positive at each point), a convex ruled surface patch, or a planar patch. The Gaussian Model of a curved object then consists of a finite set of vertices, edges and faces on the surface of a unit sphere as follows.

1. For a strictly convex face F, the Gaussian Image $N(T,F)$ is a patch of S^2 with its boundary curves determined by the normals to the tangent planes of F at the boundary. That is, the boundary of $N(T,F)$ consists of the set of points $\frac{\nabla f(p)}{\|\nabla f(p)\|}$ for $p \in \cup_{E\in\Gamma} E$, where $f = 0$ is the surface equation of F and Γ is the set of boundary edges of F. For a ruled surface patch F, $N(T,F)$ is a degenerate curve on S^2. And for a planar patch F, $N(T,F)$ is a degenerate point on S^2.

2. Consider an edge E defined by two intersecting faces F and G, where F and G meet either transversally or tangentially along E. When F and G meet transversally along E, each point $p \in E$ determines two different points n_F and n_G on S^2 determined by the exterior normals of the tangent planes of F and G at p. Then $N(T,p)$ is the geodesic arc γ_p connecting n_F and n_G on S^2 and $N(T,E) = \cup_{p\in E}\gamma_p$ is a patch of S^2. The set $N(T,E)$ has 4 boundary curves given by the set of points $\frac{\nabla f(p)}{\|\nabla f(p)\|}$ for $p \in E$, the set of points $\frac{\nabla g(p)}{\|\nabla g(p)\|}$ for $p \in E$, and the geodesic arcs γ_{p_S} and γ_{p_E}, where $f = 0$ and $g = 0$ are the surface equations of F and G, and p_S and p_E are the starting and ending vertices of E. When F and G meet tangentially along E, $N(T,E)$ is a degenerate curve on S^2. In particular, $N(T,E)$ is the common boundary curve of $N(T,F)$ and $N(T,G)$. That is, it is the set of points $\frac{\nabla f(p)}{\|\nabla f(p)\|} = \frac{\nabla g(p)}{\|\nabla g(p)\|}$ for $p \in E$. When F and G are planar patches, E is a linear edge and $N(T,E)$ is a degenerate geodesic arc γ connecting n_F and n_G on S^2, where n_F and n_G are the exterior normals of F and G.

3. Consider next a vertex p defined by k adjacent faces $F_1, F_2, \ldots, F_k$ intersecting at p (ordered via their normals at p in a counter–clockwise direction). Each face F_i determines a point n_i on S^2 determined by the normal of F_i at p. Let γ_i $(i = 1, \ldots, k)$ be the geodesic arc (great circle) on S^2 connecting n_i and n_{i+1} where $n_{k+1} = n_1$. Then $N(T,p)$ is the convex patch on S^2 bounded by the cycle of geodesic arcs $\gamma_1, \gamma_2, \ldots, \gamma_k$. When F_i and F_{i+1} are tangent at p, γ_i is a degenerate point $n_i = n_{i+1}$. In the special case of all k faces being tangent at p, the entire set $N(T,p)$ is a degenerate point. The set $N(T,p)$ can also be a degenerate geodesic arc on S^2 when $Bdr(T)$ is locally smooth at p except along a curve which is tangent at p.

Topology of Gaussian Model The Gaussian Image of $Bdr(T)$ covers S^2 completely and partitions S^2 into a set of generic faces (surface patches) as described above. Certain generic faces on S^2 degenerate to curves and points and are tagged appropriately. By using the adjacency graph of vertices, edges and faces of $Bdr(T)$ we connect the generic faces on S^2 with the same topology. Hence we construct a face adjacency graph on S^2 with degenerate faces tagged as curves and points.

3.2 Convex Hulls of Objects Bounded by Algebraic Curves

The convex hull computation is a fundamental one in computational geometry. There are numerous applications in which the convex hulls of complex objects can be used effectively to make certain geometric decisions easier. For example, a null intersection between the convex hulls of two objects implies a null intersection between the original objects. Since intersection testing for convex objects is easier than for non-convex objects, convex hulls are used as an efficient first test in a general object intersection detection algorithm. Additional motivation arises from the use of convex hulls of moving object and obstacles, for heuristic collision-free motion planning.

We present an $O(n \cdot d^{O(1)})$ algorithm to compute the convex hull of a curved object bounded by $O(n)$ algebraic curve segments of maximum degree d.

Several linear–time algorithms for computing the convex hull of simple planar polygons are known. These algorithms achieve the more efficient $O(n)$ bound whereas the $\Omega(n \log n)$ lower bound applies to general problem of computing the convex hull of n points in the plane , see [81]. The above algorithms for planar polygons are iterative and vertex–based, i.e., the computation in each step depends on the region where the next vertex lies. It is not entirely obvious how to modify this to deal with curved planar objects with piecewise algebraic boundary curves. By generalizing [58] to an edge–based algorithm Schäffer and Van Wyk [86] extended the planar polygon results to a linear–time algorithm for curved objects bounded by piecewise–smooth Jordan curves. Souvaine [49] also suggests a linear–time convex hull algorithm based on *bounding polygon approach* for a class of curved objects (termed *splinegons*). We suggest another linear–time algorithm which reflects the practical considerations discussed in the following. Further, our algorithm is simple and flexible to further modifications.

There is a difference between simple polygons and curved objects, where a single edge may intersect two different *pockets*. In the polygonal case, when an edge exits from a *pocket*, the ending vertex should be outside of all the current pockets and becomes a vertex on the updated convex hull boundary. Further, the interior of this edge is totally contained in the convex hull interior. We can not assume this fact for planar curved objects; the ending vertex of an exiting edge may lie in a pocket, and an interior point of this edge may lie on the updated convex hull boundary. To determine the exact portion of an exiting edge which appears on the new convex hull boundary, [86, 49] compute a common tangent line between this edge and the previous convex hull boundary. As we will see in §2, common tangent computation is the most expensive geometric operation among those required to compute the convex hull of an object. In practice the time spent in the common tangent computations may dominate the overall convex hull computation. Though common tangents should be computed for those appearing on the final output, the common tangents computed for the intermediate steps but not on the final output are wasted. Avoiding these wasted expensive computations is crucial for the design of an efficient convex hull algorithm for general planar curved object. We suggest two ways of eliminating this problem. First, we do not compute common tangents with concave edges.

This invariant has been used in the convex hull algorithms for polygons, too. We suggest a new invariant, see $(*)$ in §3.1, and define a *pocket* which can intersect with itself. Using these new techniques, we can eliminate the common tangent computations with concave edges. Further, we give a simple correctness proof and a simple algorithm is designed. Another important factor to be considered in computing with curved objects is how to precede the algorithm with approximate solutions in the intermediate steps. Certain intermediate decisions could be made with only approximate solutions. This is an important issue in practice to save computing time. We show our algorithm is robust for this purpose. Consider when two tangent lines have slopes very close to each other. To compare these slopes exactly, one has to refine the approximations further. Since both tangent lines may turn out to be useless later, this exact slope comparison may be wasted. A better strategy is keeping both tangent lines on the stack by assuming the slope of the lower one is smaller than that of the upper one. When these two tangents remain on the final output, we

can apply expensive refinement computations to make an exact decision. In our algorithm keeping both tangents is easy since we update the convex hull boundary only for convex edges or points. However, concave edges make trouble here, too. One can not keep two tangents simultaneously.To keep the lower one, one has to delete the upper one, and vice versa.

Though we can save computation time in the cases mentioned above, there are cases where [86, 49] performs better. There are also dual examples where ours performs better.A slight modification of our algorithm can almost eliminate these problems. For example, we add an extra vertex to the concave edge C_j and update the convex hull with respect to this extra vertex. Since we already know this extra vertex is not on the final output, we even do not need to compute a tangent line from this vertex. A line connecting this extra vertex and an appropriate vertex on the previous convex hull boundary can also be used for the same purpose for which the common tangent line are used, i.e. to detect the next *event edge*. Further modifications of our algorithm can easily be adapted as long as one keep the invariant $(*)$. We concentrate on presenting an algorithm demonstrating all the new techniques discussed above. Depending on each application, one can easily incorporate various other heuristics, define a new sequence of *event edges* keeping the invariant $(*)$, and modify our algorithm to a more efficient one. To simplify the design of our algorithm, we introduce a pre-processing step which divides each edge into monotone subedges. With monotone edges, vertex coordinate comparisons can make many geometric decisions very easily.

The rest of this paper is organized as follows. §2 describes certain preliminary informations of use in later sections. In §2.1 we describe the boundary representation for a planar geometric model with algebraic boundary curves. In §2.2 we present a monotone curve segmentation of boundary curve segments (a pre-processing step of our algorithm) and basic operations on these monotone curve segments. Algebraic curves are treated in each of two internal representations; namely, the implicit and the parametric forms, [4, 102]. In §3 we present an $O(n \cdot d^6 \log d)$ (resp. $O(n \cdot (d^{12} \log d + T(d)))$) time algorithm to compute the convex hull of planar curved objects with parametric (resp. implicit) boundary curves. Here $T(d)$ is the worst case time taken to trace an algebraic curve segment of degree d, [17].

Preliminaries In this section, we describe the algebraic boundary model of the planar curved object, and consider monotone curve segmentations and other related geometric operations on monotone curve segments.

Planar Geometric Model A planar geometric model with algebraic boundary curves has the following boundary representation. A single simple oriented cycle of algebraic curve edges, where each edge is directed and incident to two vertices. Each edge also has curve equations, which are implicit and/or rational parametric equations of algebraic curves. An algebraic curve is implicitly defined by a single polynomial equation $f(x, y) = 0$ and parametrically defined by the pair $(x = \frac{f_1(t)}{f_3(t)}, y = \frac{f_2(t)}{f_3(t)})$, where f_1, f_2 and f_3 are polynomials. Further an interior point is also provided on each implicitly defined edge which helps remove any geometric ambiguity in the case of vertices which are singularities of the algebraic curve, [82]. Finally, each vertex is exactly specified by Cartesian coordinates.

The curve equations for each edge are chosen such that the direction of the normal at each point of the edge is towards the exterior of the object. For a simple point on the curve the normal is defined as the vector of partials to the curve evaluated at that point. For a singular point on the curve we associate a range of normal directions determined by normals to the tangents at the singular point. Further the orientation of the cycle of edges is such that the interior of the object is to the left when the edges are traversed.

Computations with Algebraic Curves We assume some primitive geometric algorithms to manipulate algebraic curve segments, [4, 10, 17, 22, 23, 38, 66]. Prior work has considered the

generation of rational parametric equations for certain implicitly defined algebraic plane curves, [4], the generation of implicit equations for parametrically defined algebraic curves, [10], as well as the robust tracing of algebraic curve segments with correct connectivity, [17]. Tracing for instance is very useful in determining when a given point lies within a general algebraic curve segment. For this last problem the method of sorting along the curve [66], also provides an efficient solution for low degree algebraic curves.

Monotone Segmentation We consider the monotone segmentation of a geometric model boundary and other geometric operations on monotone curve segments. We show the time complexities of these operations which are of relevance to timing analysis of the algorithms described in later sections. Our model of computation is the arithmetic model where arithmetic operations have unit time cost, see [6]. We first define monotone edges.

Definition 3.1. *Let C be a directed boundary edge without any inflection or singular point. Then*
(1) *C is* convex $\Longleftrightarrow$ *the gradient of C turns counter-clockwise along C*
(2) *C is* concave $\Longleftrightarrow$ *the gradient of C turns clockwise along C*
(3) *C is* monotone $\Longleftrightarrow$ *C is either* convex*,* concave *or* linear*, and the interior of C doesn't include any extreme point along the x or y directions.*

We analyze the time complexity of solving systems of polynomial equations.

Lemma 3.2. (I) *All the roots of a univariate polynomial equation of degree $O(d)$ can be computed in $O(d^3 \log d)$ time.*
(II) *The common solutions of two polynomial equations of degree $O(d)$ in two variables can be computed in $O(d^6 \log d)$ time.*
(III) *The common solutions of three (resp. four) polynomial equations of degree $O(d)$ in three (resp. four) variables can be computed in $O(d^{12})$ (resp. $O(d^{16})$) time.*

Proof : (I) The squarefree part of a univariate polynomial can be calculated in $O(d \log^2 d)$ time using fast techniques for the required GCD computation and division steps, [6], and further all roots can be computed using root isolations in $O(d^3 \log d)$ time, see [87].
(II) We can eliminate one variable from two polynomial equations using the Sylvester resultant in $O(d^4 \log^3 d)$ time, see [23], and then compute the roots of the resulting univariate polynomial equation of degree $O(d^2)$ in $O(d^6 \log d)$ time. Doing this twice for each variable in turn together with the pairwise substitutions then allows computing the common solutions in overall $O(d^6 \log d)$ time.
(III) We can eliminate three (resp. four) variables from three (resp. four) polynomial equations using the *u–resultant* in $O(d^9)$ (resp. $O(d^{12})$) time, see [38, 100], and solve resulting equations of degree $O(d^3)$ (resp. $O(d^4)$) in one variable in $O(d^9 \log d)$ (resp. $O(d^{12} \log d)$) time. Though the resultant computation takes naively $O(d^9)$ (resp. $O(d^{12})$) time and solving real root solutions takes $O(d^9 \log d)$ (resp. $O(d^{12} \log d)$) time, the overall time is bounded by the pairwise substitution which takes $O(d^{12})$ (resp. $O(d^{16})$) time. □

The monotone segmentation requires adding singular points, inflection points and extreme points on the curve as extra vertices. First we take care of singular points on curved edges. Singularities are determined for each curved edge and are computed by using Lemma 2.1: I (i) and II (i). The boundary of the object is next modified such that nonsingular edges are either *convex*, *concave* or *linear* segments. Such conditions are easily met by adding extra vertices to inflection points of curved edges. Inflection points of curves can be obtained and the edges are marked *convex*, *concave* or *linear* respectively by using Lemma 2.1: I (iv), (v), (vi) and II (iv), (v), (vi). We may also assume edges are further segmented so that extreme points along x or y directions added as vertices. These extreme points are computed by using Lemma 2.1: I (ii), (iii) and II (ii), (iii).

Lemma 3.3. (I) *Let $C : (a, b) \to R^2$ be a curve parametrized by $t \in (a, b)$ and $p = C(t) = (c_1(t),\ c_2(t))$ be a point on this curve. Then*
(i) p is a (non-self-intersecting) singular point $\iff c_1'(t) = c_2'(t) = 0$,
(ii) p is a non-singular x-extreme point $\iff c_2'(t) = 0$ and $c_1'(t) \neq 0$,
(iii) p is a non-singular y-extreme point $\iff c_1'(t) = 0$ and $c_2'(t) \neq 0$, and
(iv) p is an inflection point of the curve $C \iff c_1'(t) \cdot c_2''(t) - c_2'(t) \cdot c_1''(t) = 0$.
If C has no inflection point, then
(v) C is convex $\iff c_1'(t) \cdot c_2''(t) - c_2'(t) \cdot c_1''(t) > 0$, *and*
(vi) C is concave $\iff c_1'(t) \cdot c_2''(t) - c_2'(t) \cdot c_1''(t) < 0$.

(II) *Let C be a curve implicitly defined by $f(x, y) = 0$ and $p = (x, y)$ be a point on the curve C. Then*
(i) p is a singular point $\iff f = f_x = f_y = 0$,
(ii) p is a non-singular x-extreme point $\iff f = f_y = 0$ and $f_x \neq 0$,
(iii) p is a y-extreme point $\iff f = f_x = 0$ and $f_y \neq 0$, and
(iv) p is an inflection point $\iff f = f_{xx} \cdot f_y^2 - 2f_{xy} \cdot f_x f_y + f_{yy} \cdot f_x^2 = 0$.
If C has no inflection point, then
(v) C is convex $\iff f_{xx} \cdot f_y^2 - 2f_{xy} \cdot f_x f_y + f_{yy} \cdot f_x^2 > 0$, *and*
(vi) C is concave $\iff f_{xx} \cdot f_y^2 - 2f_{xy} \cdot f_x f_y + f_{yy} \cdot f_x^2 < 0$.

Proof : Most of these results are classical, see [102]. □

Lemma 3.4. (I) *For a parametric curve segment C of degree d, a monotone segmentation can be obtained in $O(d^3 \log d)$ time.*
(II) *For an implicit algebraic curve segment C of degree d, a monotone segmentation can be obtained in $O(d^6 \log d + T(d))$ time, where $T(d)$ is the time required for the curve segment tracing.*

Proof : (I) The equations in Lemma 2.1 (I) are of degree $O(d)$ in a single variable t.
(II) The equations in Lemma 2.1 (II) are two simultaneous polynomial equations of degree $O(d)$ in two variables x and y. □

Basic Operations on Monotone Curve Segments We consider primitive operations on monotone curve segments C and D, and a line segment L.

1. The intersection of C and L,
2. The containment of C in the upper–left halfplane $H^{UL}(L)$ of L,
3. The tangent line L of C from a point q,
4. The common tangent line L of C and D.

Line–Curve Segments Intersection Suppose C is a monotone curve segment and L is a line segment. Let $R(C)$ and $R(L)$ be the minimal rectangles with sides parallel to coordinate axes and containing C and L respectively, and $T(C)$ be the minimal triangle defined by the line connecting both end points of C and two tangent lines of C at both end points of C. The intersection of C and L is computed as follows.

if $(R(C) \cap R(L) = \emptyset$ or $T(C) \cap L = \emptyset)$ **then** $C \cap L = \emptyset$;
else Let $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ be the starting and ending points of $L' = T(C) \cap L$, then the intersection point(s) of C and L' is computed by Lemma 2.4;

Lemma 3.5. (I) *If C is a parametric curve segment given by $C(s) = (x(s), y(s))$ with $a \le s \le b$, then C intersects with L at a point $p = C(s) = t \cdot p_1 + (1-t) \cdot p_2$* if and only if *$s$ and t satisfy*

$$\begin{cases} a \le s \le b, \text{ and } 0 \le t \le 1 & (1) \\ x(s) = t \cdot x_1 + (1-t) \cdot x_2 & (2) \\ y(s) = t \cdot y_1 + (1-t) \cdot y_2 & (3) \end{cases}$$

(II) *If C is an implicit algebraic curve segment given by $f(x, y) = 0$, then C intersects with L' at a point $p = t \cdot p_1 + (1-t) \cdot p_2$* if and only if *$t$ satisfies*

$$\begin{cases} p \in C, \text{ and } 0 \le t \le 1 & (1) \\ f(t \cdot x_1 + (1-t) \cdot x_2, t \cdot y_1 + (1-t) \cdot y_2) = 0 & (2) \end{cases}$$

Proof : Straightforward. □

Lemma 3.6. (I) *For a parametric curve segment C, the curve–line segments intersection is computed in $O(d^3 \log d)$ time.*
(II) *For an implicit algebraic curve segment C, the curve–line segments intersection is computed in $O(d^3 \log d + T(d))$ time.*

Proof : (I) The elimination of t can be done within constant time resulting in a single polynomial of degree d in a single variable s. This polynomial can be solved in $O(d^3 \log d)$ time. There are at most d solutions for s with $a \le s \le b$ and the corresponding t to each s can be solved within constant time.
(II) When we expand the equation (2) in an increasing order of t, it gives a polynomial of degree d in a single variable t. The expansion can be done in $O(d^2)$ time and the polynomial can be solved in $O(d^3 \log d)$ time. Finally, we need to trace along the curve segment C to check whether these solutions are on the curve segment C in $T(d)$ time. □

Containment in a Halfplane The halfplane containment for points and line segments can be done within constant time. Suppose C is a *convex* monotone edge along which x and y-coordinates are strictly increasing, L is an infinite line with slope $m \ge 0$, and $H^{UL}(L)$ is the upper–left closed halfplane of the line L. These are the only types of curved edges and halfplanes considered in §3. Then, $C \subset H^{UL}(L) \iff p_S$ and $p_E \in H^{UL}(L)$, and $C \cap L' \neq \emptyset$, where p_S and p_E are the starting and ending points of C respectively, and $L' = T(C) \cap L$. Hence, the time complexity of halfplane containment testing is $O(d^3 \log d)$ (resp. $O(d^3 \log d + T(d))$), the same as that of line–curve segments intersection.

Common Supporting Line of a Curve Segment and a Point Suppose L is a common supporting line of a monotone curve segment C and a point $q = (\alpha, \beta) \notin C$ such that $C \cup \{q\} \subset H^{UL}(L)$. Then the supporting point p of L at C is given by the following Lemma.

Lemma 3.7. (I) *If C is given by a parametric curve $C(t) = (x(t), y(t))$ with $a \le t \le b$, then $p = (x(t), y(t))$ is given by*

$$\begin{cases} a \le t \le b & (1) \\ (x(t) - \alpha) \cdot y'(t) - (y(t) - \beta) \cdot x'(t) = 0 & (2) \end{cases}$$

(II) *If C is given by an implicit curve $f(x, y) = 0$, then the point $p = (x, y)$ is given by*

$$\begin{cases} f(x, y) = 0, \text{ and } p = (x, y) \in C & (1) \\ (x - \alpha) \cdot f_x + (y - \beta) \cdot f_y = 0 & (2) \end{cases}$$

Proof : Straightforward. □

Lemma 3.8. (I) *For a parametric curve segment C, the common supporting line of C and q is computed in $O(d^3 \log d)$ time.*
(II) *For an implicit algebraic curve segment C, the common supporting line of C and q is computed in $O(d^6 \log d + T(d))$ time.*

Proof : Similar to Lemma 3.1.2. □

Common Supporting Line of Two Curve Segments Suppose L is a common supporting line of two disjoint monotone curve segments C and D such that $C \cup D \subset H^{UL}(L)$. Then the supporting points $p = (x, y)$ and $q = (\alpha, \beta)$ of L at C and D respectively are given by the following Lemma.

Lemma 3.9. (I) *If C and D are given by parametric curves $C(s) = (x(s), y(s))$ with $a \leq s \leq b$ and $D(t) = (\alpha(t), \beta(t))$ with $c \leq t \leq d$, then $p = (x(s), y(s))$ and $q = (\alpha(t), \beta(t))$ are given by*

$$\begin{cases} a \leq s \leq b, \text{ and } c \leq t \leq d & (1) \\ (x(s) - \alpha(t)) \cdot y'(s) - (y(s) - \beta(t)) \cdot x'(s) = 0 & (2) \\ (x(s) - \alpha(t)) \cdot \beta'(t) - (y(s) - \beta(t)) \cdot \alpha'(t) = 0 & (3) \end{cases}$$

(II) *If C is given by a parametric curve $C(s) = (x(s), y(s))$ with $a \leq s \leq b$ and D is given by an implicit curve $g(\alpha, \beta) = 0$, then $p = (x(s), y(s))$ and $q = (\alpha, \beta)$ are given by*

$$\begin{cases} a \leq s \leq b & (1) \\ g(\alpha, \beta) = 0, \text{ and } q = (\alpha, \beta) \in D & (2) \\ (x(s) - \alpha) \cdot y'(s) - (y(s) - \beta) \cdot x'(s) = 0 & (3) \\ (x(s) - \alpha) \cdot g_\alpha + (y(s) - \beta) \cdot g_\beta = 0 & (4) \end{cases}$$

(III) *If C and D are given by implicit curves $f(x, y) = 0$ and $g(\alpha, \beta) = 0$, then $p = (x, y)$ and $q = (\alpha, \beta)$ are given by*

$$\begin{cases} f(x, y) = 0, \text{ and } p = (x, y) \in C & (1) \\ g(\alpha, \beta) = 0, \text{ and } q = (\alpha, \beta) \in D & (2) \\ (x - \alpha) \cdot f_x + (y - \beta) \cdot f_y = 0 & (3) \\ (x - \alpha) \cdot g_\alpha + (y - \beta) \cdot g_\beta = 0 & (4) \end{cases}$$

Proof : Straightforward. □

Lemma 3.10. (I) *For parametric curve segments C and D, the common supporting line of C and D is computed in $O(d^6 \log d)$ time.*
(II) *For a parametric curve segment C and an implicit algebraic curve segment D, the common supporting line of C and D is computed in $O(d^9 \log d + T(d))$ time.*
(III) *For implicit algebraic curve segments C and D, the common supporting line of C and D is computed in $O(d^1 2 \log d + T(d))$ time.*

Proof : Though the common solutions of three (resp. four) polynomial equations as stated in Lemma 2.1 take $O(d^{12})$ (resp. $O(d^{16})$) time, for monotone curve segments and the application of common tangents, this can be reduced to $O(d^9 \log d + T(d))$ (resp. $O(d^{12} \log d + T(d))$) time. □

Now, initially assume there are $O(m)$ algebraic curve segments of maximum degree d on the model boundary. Then the monotone segmentation preprocessing can be done in $O(m \cdot d^3 \log d)$ (resp. $O(m \cdot (d^6 \log d + T(d)))$) time if all the boundary curve segments are parametric (resp. implicit). By Bezout Theorem a single polynomial of degree d in one variable t can have at most d

solutions, and a system of two polynomial equations of degree d in two variables x and y can have at most d^2 solutions. Thus, Lemma 2.1 implies that each parametric (resp. implicit) algebraic curve segment of degree d can be segmented into $O(d)$ (resp. $O(d^2)$) monotone curve segments by adding extra vertices into singular points, inflection points and extreme points. After this preprocessing step of monotone segmentation, we let the total number of boundary edges be n, which is $O(m \cdot d)$ (resp. $O(m \cdot d^2)$) for parametric (resp. implicit) curves. In the following, we assume the object boundary is already segmented into $O(n)$ monotone curve segments and the timing analysis is given in terms of n.

Convex Hull of Geometric Model In this section we present an algorithm to compute the convex hull of a planar geometric model bounded by $O(n)$ monotone curve segments. The algorithm runs in $O(n)$ steps, where each step takes polynomial time in the degree d. In the following we consider the construction of the lower–right subpart of the convex hull boundary which lies between the bottommost vertex p_0 and the rightmost vertex p_M of the original object. The entire convex hull is obtained by applying the same algorithm to the remaining three subparts. W.l.o.g. we may assume there are unique bottommost and rightmost vertices.

In the following, let $C_1, C_2, \ldots, C_M$ be a connected sequence of edges from p_0 to p_M, where each C_i has p_{i-1} and p_i as its starting and ending vertices resectively. For a point $p_{\#}$ (resp. $p^{\#}$) we denote its x and y-coordinates by $x_{\#}$ and $y_{\#}$ (resp. $x^{\#}$ and $y^{\#}$). We also denote the line segment connecting two points p and q by $L(p, q)$ and the path from p to q along the boundary curve by $\gamma(p, q)$. Also let $A \sim B$ denote everything in A which is not in B.

Sequences of Event Edge and Current Hull We give a constructive definition of a sequence of *event edges* $\{C_{i_k}\}_{k=1}^{N}$ with $C_{i_N} = C_M$ and a sequence of *current hulls* $\{CH_k\}_{k=1}^{N}$. Further, we show that the N-th *current hull* CH_N is the lower–right subpart of the convex hull boundary between p_0 and p_M.

Definition of C_{i_k} and CH_k Let $i_0 = 0$ and $CH_0 = \{p_0\}$. Assume that the index i_k and the k-th *current hull* CH_k $(0 \le k < N)$ have been defined. We define the (k+1)-th *event edge* $C_{i_{k+1}}$ and the (k+1)-th *event component* $EC_{k+1} \subset C_{i_{k+1}}$ in terms of i_k and CH_k as follows. The terms *lid* and *pocket*, having similar meanings as in [81], are first used without definition and are rigorously defined later.

1. If $x_{i_k+1} \le x_{i_k}$ and the inner angle of $p_{i_k} < \pi$, then $i_{k+1} = \min\{j \mid j > i_k \text{ and } x_j > x_k\}$.
   (a) $EC_{k+1} = C_{i_{k+1}}$ if $y_{i_{k+1}-1} < y_{i_{k+1}}$ and $C_{i_{k+1}}$ is *convex*,
   (b) $EC_{k+1} = p_{i_{k+1}}$ otherwise.
2. If $x_{i_k} < x_{i_k+1}, y_{i_k} < y_{i_k+1}$, and the inner angle of $p_{i_k} < \frac{3\pi}{2}$, then $i_{k+1} = i_k + 1$.
   (a) $EC_{k+1} = C_{i_{k+1}}$ if $C_{i_{k+1}}$ is *convex*,
   (b) $EC_{k+1} = p_{i_{k+1}}$ otherwise.
3. Otherwise, let j_0 be the smallest j such that $j > i_k + 1$, and either (p_{j-1} is not inside of any pocket of CH_k with $y_{j-1} < y_j$) or (C_j intersects with a lid $L(p', p'')$ at a point p_{**}, but it is not totally inside of the pocket implied by $L(p', p'')$, and further $x_{j-1} < x_j$, $y_{j-1} < y_j$, and $y_{**} < \min\{y_* \mid p_* \in \gamma(p_{i_k}, p_{j-1}) \cap L(p', p'')\}$.
   (a) If p_{j_0-1} is not inside of any pocket of CH_k and either the inner angle of $p_{j_0-1} > \pi$ or $x_{j_0} < x_{j_0-1}$, then $i_{k+1} = j_0 - 1$ and $EC_{k+1} = p_{i_{k+1}}$.
   (b) Otherwise, $i_{k+1} = j_0$ and

i. $EC_{k+1} = C_{i_{k+1}}$ if $C_{i_{k+1}}$ is *convex*, and
ii. $EC_{k+1} = L(p_{i_{k+1}-1}, p_{i_{k+1}})$ otherwise.

Next we inductively define the (k+1)-th *current hull* CH_{k+1}. It is easy to show there is a unique common tangent line $L_{p',p''}$ of CH_k and EC_{k+1} (with $x' < x''$ and $y' < y''$) such that $CH_k \cup EC_{k+1} \subset H^{UL}(L)$. If there is more than one choice of p' (resp. p''), we choose p' (resp. p'') so that the distance between p' and p'' is minimal. Further, let $FRONT\_CH_{k+1}$ denote the subarc of CH_k between the points p_0 and p', and $REAR\_CH_{k+1}$ denote the subarc of EC_{k+1} between the points p'' and $p_{i_{k+1}}$. The (k+1)-th *current hull* CH_{k+1} is defined as the connected union $FRONT\_CH_{k+1} \cup L(p', p'') \cup REAR\_CH_{k+1}$. CH_{k+1} is a convex arc along which both x and y-coordinates are strictly increasing. $L(p', p'')$ is called the *lid* determined by p' and p''. Let $\bar{\gamma}$ be the closed path given as $\gamma(p', p'')$ followed by a path from p'' to p' along $L(p', p'')$. If $\bar{\gamma}$ has no self-intersection, the region bounded by $\bar{\gamma}$ is called the *pocket* determined by the lid $L(p', p'')$. Otherwise, $\gamma(p', p'')$ has an even number of intersections with the lid $L(p', p'')$ counting intersections with multiplicities and $\bar{\gamma}$ divides the plane into a finite number of connected regions. The union of all the regions which are to the right of $\bar{\gamma}$ is the *pocket* implied by $L(p', p'')$.

Properties of CH_k We prove two important properties of CH_k in the following Lemmas 3.1–3.2.

Lemma 3.11. *If a point $p \in C_i$ $(1 \le i \le i_k)$ is on the convex hull boundary, then $p \in CH_k$.*

Proof : Using induction we can easily show that the interior of the path $\gamma(p_{i_k}, p_{i_{k+1}-1})$, the arc $C_{i_{k+1}} \sim EC_{k+1}$, and $(CH_k \cup C_{i_{k+1}}) \sim CH_{k+1}$ are in the convex hull interior. □

Lemma 3.12. *If a point $p \in CH_k$ is on the convex hull boundary, then the subarc of CH_k between p_0 and p is contained entirely in the convex hull boundary.*

Proof : The case $k = 1$ is easy to show. By induction, we assume for k $(1 \le k < N)$ and consider $k + 1$. Suppose a point $p \in CH_{k+1}$ is on the convex hull boundary. (a) If $p \in FRONT\_CH_{k+1} \subset CH_k$, then the statement follows by induction. (b) If $p \in L(p', p'')$, then $L(p', p'')$ is also on the convex hull boundary. Further, $FRONT\_CH_{k+1}$ is on the convex hull boundary by induction. (c) If $p \in REAR\_CH_{k+1}$, then there is a supporting line L_p at p. We now prove that the lid $L(p', p'')$ is on the convex hull boundary. Suppose there is a boundary point q in the region R_1. We may assume q is extreme to the outward normal direction of the lid and thus on the convex hull boundary. (i) If $q \in C_j$ $(1 \le j \le i_{k+1})$, then Lemma 3.1 implies $q \in CH_{k+1}$, however, this is impossible since CH_{k+1} is convex. (ii) Otherwise, there is a continuous path from $p_{i_{k+1}}$ to q. This path should pass through either the region R_2 or R_3, however, both are impossible. Hence the lid $L(p', p'')$ is on the convex hull boundary, and by induction $FRONT\_CH_{k+1}$ is also on the convex hull boundary. Similarly one can show that the subarc of $REAR\_CH_{k+1}$ between p'' and p is on the convex hull boundary. □

Since p_M is on the convex hull boundary and it is the end point of CH_N, Theorem 3.1 below follows easily from Lemmas 3.1–3.2.

Theorem 3.13. *CH_N is the lower–right part of the convex hull boundary between p_0 and p_M.*

Description of Algorithm We describe an algorithm to compute the sequences of *event edges* $\{C_{i_k}\}_{k=1}^{N}$ and *current hulls* $\{CH_k\}_{k=1}^{N}$ by using a single stack CH. CH contains segments of the k-th *current hull* CH_k which are subarcs of some *convex* edges, some *linear* edges and the lids of pockets. Consecutive elements on the stack share a common end point and the connected sequence of elements on the stack CH generate the *current hull* CH_k (we say then that "the stack CH *implies* the *current hull* CH_k").

Computing Event Edges We start by pushing a single point interval $[p_0, p_0]$ into an empty stack CH. The stack CH implies the *current hull* $CH_0 = \{p_0\}$. Assume i_k is detected and the stack CH implies the k-th *current hull* CH_k $(0 \leq k < N)$. We consider how to detect i_{k+1} using the stack CH. Since the cases 1 and 2 of §3.1.1 are easy, we consider only case 3. We detect j_0 by using a loop variable j initialized to i_k and maintaining the following invariant for j at the beginning of each loop :

"p_j is outside (including the boundary) of any pocket of CH_k and TOP(CH) is not strictly below the horizontal line $y = y_j$."

In each loop, j is first incremented by 1. Then, (1) If $y_{j-1} \leq y_j$, then $j_0 = j$. (2) If $y_{j-1} > y_j$, then pop all the stack elements until (a) TOP(CH) which does not intersect with C_j and is not strictly above the line $y = y_j$, or (b) TOP(CH) which is a lid intersecting with C_j. In the case (a), repeat the j–loop. In the case (b), initially let p_* be the first intersection of C_j with $L(p', p'')$. Next, walk along the path $\gamma(p_{j-1}, p_M)$ and count the number of right–to–left cuts and left–to–right cuts this path makes with the lid $L(p', p'')$. At each intersection point, say p_{**}, make $p_* = p_{**}$ if $y_{**} \leq y_*$. While the number of right–to–left cuts is larger than left–to-right cuts, we are inside of the pocket implied by $L(p', p'')$. Consider when a path $\gamma(p_*, p_j)$ is totally contained in a self–intersecting pocket and has its first interior intersection with a lid $L(p', p'')$ in a right–to–left direction. When $\gamma(p_*, p_j)$ comes out of a pocket through a point p_{**}, the total cuts in both directions are equal. If these two numbers become equal at a point $p_{**} \in C_{j'}$, compare y_{**} with y_*. If $y_{**} > y_*$, then continue the walking along the path $\gamma(p_{**}, p_M)$. If $y_{**} \leq y_*$, then (i) let $j_0 = j'$ if $y_{j'-1} \leq y_{j'}$, and (ii) otherwise, let $j = j'$ and further treat p_{**} as p_j and repeat the j-loop. It is easy to show that the j–loop invariant holds everytime the loop is repeated and j_0 is detected correctly.

procedure *Detect-Event-Edge* (CH,i_k,i_{k+1},EC);
begin
if ($x_{i_k+1} \leq x_{i_k}$ and the inner angle of $p_{i_k} < \pi$) **then begin**
  $i_{k+1} = \min\{j \mid j > i_k + 1$ and $x_j > x_{i_k}\}$;
  if ($y_{i_{k+1}-1} < y_{i_{k+1}}$ and $C_{i_{k+1}}$ is *convex*) **then** $EC = C_{i_{k+1}}$ **else** $EC = p_{i_{k+1}}$ **end**;
else if ($x_{i_k} < x_{i_k+1}$, $y_{i_k} < y_{i_k+1}$, and the inner angle of $p_{i_k} < \frac{3\pi}{2}$) **then begin**
  $i_{k+1} = i_k + 1$; **if** ($C_{i_{k+1}}$ is *convex*) **then** $EC = C_{i_{k+1}}$ **else** $EC = p_{i_{k+1}}$ **end**;
else begin
  $j = i_k$; *FOUND* = false;
  while (not *FOUND*) **do begin**
    $j = j + 1$;
    if ($y_{j-1} < y_j$ and $j - 1 \neq i_k$) **then begin**
      FOUND = true;
      if ($x_{j-1} < x_j$ and the inner angle of $p_{j-1} < \pi$) **then begin**
        $i_{k+1} = j$; **if** ($C_{i_{k+1}}$ is *convex*) **then** $EC = C_{i_{k+1}}$ **else** $EC = L(p_{i_{k+1}-1}, p_{i_{k+1}})$ **end**;
      else begin $i_{k+1} = j - 1$; $EC = p_{i_{k+1}}$; **end**
    else begin
      Pop all the stack elements until (a) a lid $L(p', p'')$ which contains p_j in the interior of the pocket bounded by the lid $L(p', p'')$ or (b) a stack element which is not strictly above the horizontal line $y = y_j$;
      if (p_j is an interior point of the pocket bounded by the lid $L(p', p'')$ which intersects with C_j at p_*) **then begin**
        DONE = false; *RightLeftCut* = 1; *LeftRightCut* = 0; *YMIN* = y_*;
        repeat
          Skip all the subsequent edges until an edge $C_{j'}$ which transversally intersects with $L(p', p'')$;

for (each transversal intersection point p_{**} on $C_{j'}$) **do begin**
if (the intersection is a left-to-right cut)
then *LeftRightCut* = *LeftRightCut* + 1;
else *RightLeftCut* = *RightLeftCut* + 1;
if ($y_{**} <$ *YMIN* and *RightLeftCut* = *LeftRightCut*)
then *DONE* = true;
else *YMIN* = min (*YMIN*, y_{**}); **end**; ($*$ for $*$)
until (*DONE*);
if ($y_{j'-1} < y_{j'}$) **then begin** $i_{k+1} = j'$; *FOUND* = true **end**
else $j = j'$ **end**; **end**; **end**; **end**;
end; ($*$ *Detect-Event-Edge* $*$)

Computing Current Hulls We consider next how to construct CH_{k+1} from CH_k and EC_{k+1} by using the stack *CH*. Though we have popped some stack elements from *CH*, the elements on the stack *CH* imply a convex arc Γ which contains $FRONT\_CH_{k+1}$. To remove redundant elements $\Gamma \sim FRONT\_CH_{k+1}$ from *CH*, we consider the top stack element TOP(*CH*). We check whether TOP(*CH*) contains the common tangent point p' for the new lid $L(p', p'')$ of CH_{k+1}. Since EC_{k+1} is not strictly below $y = y_S$, we have (1) $p' = p_E$ if $EC_{k+1} \subset H^{UL}(L_{p_E})$ and EC_{k+1} is not strictly below the horizontal line $y = y_E$; otherwise, (2) $p' \in$ TOP(*CH*) if $EC_{k+1} \subset H^{UL}(L_{p_S})$, and (3) $p' \notin$ TOP(*CH*) otherwise. Here p_S and p_E are the starting and ending points of TOP(*CH*), and L_{p_S} and L_{p_E} are the tangent lines of TOP(*CH*) at p_S and p_E respectively. In the cases (1) and (2), p' and p'' can be computed by using Lemmas 2.3–2.4. In the case (3), we pop TOP(*CH*) and repeat the same procedure. Once we have computed p' and p'', we can adjust the stack appropriately to imply CH_{k+1}.

procedure *Update-Current-Hull* (*CH*,*EC*);
begin
DONE = false;
while (not *DONE*) **do begin**
Let p_S and p_E be the starting and ending points of TOP(*CH*);
Let L_{p_S} and L_{p_E} be the tangent lines of TOP(*CH*) at p_S and p_E respectively;
if (*EC* $\subset H^{UL}(L_{p_E})$ and *EC* is not strictly below the horizontal line $y = y_E$) **then begin**
if (*EC* is a point $p_{i_{k+1}}$) **then** $p'' = p_{i_{k+1}}$;
else if (*EC* is a convex edge $C_{i_{k+1}}$) **then** p'' = SUPP2 (p_E, $C_{i_{k+1}}$);
else if (*EC* is a line segment $L(p_{i_{k+1}-1}, p_{i_{k+1}})$) **then**
if ($p_{i_{k+1}-1} \in H^{UL}(L(p_E, p_{i_{k+1}}))$) **then** $p'' = p_{i_{k+1}}$ **else** $p'' = p_{i_{k+1}-1}$;
Push the lid $L(p_E, p'')$ into *CH*;
if ($p'' \neq p_{i_{k+1}}$) **then**
if ($EC = C_{i_{k+1}}$) **then** Push the subsegment of $C_{i_{k+1}}$ between p'' and $p_{i_{k+1}}$ into *CH*;
else if ($EC = L(p_{i_{k+1}-1}, p_{i_{k+1}})$) **then** Push the lid $L(p_{i_{k+1}-1}, p_{i_{k+1}})$ into *CH*;
DONE = true **end**;
else if (*EC* $\subset H^{UL}(L_{p_S})$) **then begin**
if (*EC* is a point $p_{i_{k+1}}$) **then** $p'' = p_{i_{k+1}}$ and p' = SUPP1 (TOP(*CH*),$p_{i_{k+1}}$);
else if (*EC* is a *convex* edge $C_{i_{k+1}}$) **then**
p' = SUPP1 (TOP(*CH*),$C_{i_{k+1}}$); p'' = SUPP2 (TOP(*CH*), $C_{i_{k+1}}$);
else if (*EC* is a line segment $L(p_{i_{k+1}-1}, p_{i_{k+1}})$) **then begin**
p' = SUPP1 (TOP(*CH*), $p_{i_{k+1}}$);
if ($p_{i_{k+1}-1} \in H^{UL}(L(p', p_{i_{k+1}}))$) **then** $p'' = p_{i_{k+1}}$;
else $p'' = p_{i_{k+1}-1}$; p' = SUPP1 (TOP(*CH*),$p_{i_{k+1}-1}$) **end**; ($*$ **else if** $*$)

Let C = the subsegment of TOP(CH) between p_S and p';
Pop TOP(CH) from CH, and push C and the lid $L(p', p'')$ into CH;
if $(p'' \neq p_{i_{k+1}})$ **then**
if $(EC = C_{i_{k+1}})$ **then** Push the subsegment of $C_{i_{k+1}}$ between p'' and $p_{i_{k+1}}$ into CH;
else Push the lid $L(p_{i_{k+1}-1}, p_{i_{k+1}})$ into CH;
$DONE$ = true **end**;
else Pop TOP(CH) from the stack CH **end**; ($*$ while $*$)
end; ($*$ *Update-Current-Hull* $*$)

Timing Analysis There are basically two types of operations in this algorithm. Operations to manipulate the edge sequence itself or operations to manipulate the stack. Each edge as input is processed within a constant number of steps before it is determined whether it is an event edge or not. If it is an event edge, a segment of this edge is pushed on the stack. While an edge is on the stack, it may be used to process other edges and/or be changed into a shorter subsegment. Finally, it is popped if it is not on the convex hull boundary or it appears in the final output as an edge of the convex hull boundary.

Before an edge is determined to be an event edge or not, we apply to this edge such computations like coordinate comparisons, an inner angle computation, and intersections with stack elements. If a stack element is popped after certain computations, we charge the cost of these computations to this popped stack element. Since the popping occurs at most once to an edge, the cost at the popping time will be charged at most once to each edge. Further, there is at most one stack element which is involved in the operation with the input edge and still remains on the stack. We can charge this cost and other trivial computation costs to the input edge. The most expensive computation cost here is line–curve segment intersection which is $O(d^3 \log d)$ (resp. $O(d^3 \log d + T(d))$) time for a parametric (resp. implicit) curve segment. Since there are totally $O(n)$ input edges and popped edges, the total cost for event edge detections is $O(n \cdot d^3 \log d)$ (resp. $O(n \cdot (d^3 \log d + T(d)))$) time.

After an event edge is detected, the stack is modified to imply the new current hull. The stack elements are popped after two halfplane containment testings which cost $O(d^3 \log d)$ (resp. $O(d^3 \log d + T(d))$) time. The new stack is constructed by computing the lid $L(p', p'')$ and modifying itself appropriately, which costs at most $O(d^6 \log d)$ (resp. $O(d^{12} \log d + T(d))$) time for common tangent computations. Note that this last operation takes constant time if both p' and p'' are known, and $O(d^3 \log d)$ ($O(d^6 \log d + T(d))$) time if one of p' and p'' is known. We charge this cost to the event edge. Thus, the total cost for the current hull computation is $O(n \cdot d^6 \log d)$ (resp. $O(n \cdot (d^{12} \log d + T(d)))$) time, which is also the overall time complexity of the convex hull algorithm.

We suggested an $O(n \cdot d^6 \log d)$ (resp. $O(n \cdot (d^{12} \log d + T(d)))$) time algorithm to compute the convex hull of planar curved object bounded by $O(n)$ rational (resp. non–rational) algebraic monotone curve segments. Though within the same asymptotic time complexity, this improves [86, 49] in the case of planar curved objects bounded by arbitrary algebraic curve segments. Main differences between this algorithm and [86, 49] are as follows: (1) the boundary curves are divided into monotone curve segments by adding inflection and extreme points as extra vertices in a preprocessing step, (2) simple coordinate comparisons between vertex coordinates make many geometric decisions very easily, (3) this algorithm reduces the number of common tangent computation by detecting next event edges with a correct orientation, and (4) this algorithm is robust when used with approximations to save expensive computations in the intermediate steps.

3.3 Decomposition

In this section we present an algorithm to compute the convex hull of a planar curved object bounded by $O(n)$ monotone curve segments. The algorithm runs in $O(n)$ steps, where each step takes polynomial time in the degree d. In the following we consider the construction of the lower–right subpart of the convex hull boundary which lies between the bottommost vertex p_0 and the

rightmost vertex p_M of the original object. The entire convex hull is obtained by applying the same algorithm to the remaining three subparts. W.l.o.g. we may assume there are unique bottommost and rightmost vertices.

In the following, let $C_1, C_2, \ldots, C_M$ be a connected sequence of edges from p_0 to p_M, where each C_i has p_{i-1} and p_i as its starting and ending vertices resectively. For a point $p_\#$ (resp. $p^\#$) we denote its x and y-coordinates by $x_\#$ and $y_\#$ (resp. $x^\#$ and $y^\#$). We also denote the line segment connecting two points p and q by $L(p,q)$ and the path from p to q along the boundary curve by $\gamma(p,q)$.

Sequences of Event Edge and Current Hull We give a constructive definition of a sequence of *event edges* $\{C_{i_k}\}_{k=1}^N$ with $C_{i_N} = C_M$ and a sequence of *current hulls* $\{CH_k\}_{k=1}^N$. Further, we show that the N-th *current hull* CH_N is the lower–right subpart of the convex hull boundary between p_0 and p_M.

Definition of C_{i_k} and CH_k Let $i_0 = 0$ and $CH_0 = \{p_0\}$. Assume that the index i_k and the k-th *current hull* CH_k $(0 \le k < N)$ have been defined. We define the (k+1)-th *event edge* $C_{i_{k+1}}$ and the (k+1)-th *event component* $EC_{k+1} \subset C_{i_{k+1}}$ in terms of i_k and CH_k as follows, see Figures 4.1.1−4.1.3.

1. If $x_{i_k+1} \le x_{i_k}$ and the inner angle of $p_{i_k} < \pi$, then $i_{k+1} = \min\{j \mid j > i_k \text{ and } x_j > x_k\}$.
   (a) $EC_{k+1} = C_{i_{k+1}}$ if $y_{i_{k+1}-1} < y_{i_{k+1}}$ and $C_{i_{k+1}}$ is *convex*,
   (b) $EC_{k+1} = p_{i_{k+1}}$ otherwise.
2. If $x_{i_k} < x_{i_k+1}$ and $y_{i_k} < y_{i_k+1}$, then $i_{k+1} = i_k + 1$.
   (a) $EC_{k+1} = C_{i_{k+1}}$ if $C_{i_{k+1}}$ is *convex*,
   (b) $EC_{k+1} = p_{i_{k+1}}$ otherwise.
3. Otherwise, let j_0 be the smallest j such that $j > i_k + 1$, and either (p_{j-1} is not inside of any pocket of CH_k with $y_{j-1} < y_j$) or (C_j intersects with a lid $L(p', p'')$ at a point p_{**}, but it is not totally inside of the pocket implied by $L(p', p'')$, and further $x_{j-1} < x_j$, $y_{j-1} < y_j$, and $y_{**} < \min\{y_* \mid p_* \in \gamma(p_{i_k}, p_{j-1}) \cap L(p', p'')\}$.
   (a) If p_{j_0-1} is not inside of any pocket of CH_k and the inner angle of $p_{j_0-1} > \pi$, then $i_{k+1} = j_0 - 1$ and $EC_{k+1} = p_{i_{k+1}}$.
   (b) Otherwise, $i_{k+1} = j_0$ and
      i. $EC_{k+1} = C_{i_{k+1}}$ if $C_{i_{k+1}}$ is *convex*, and
      ii. $EC_{k+1} = L(p_{i_{k+1}-1}, p_{i_{k+1}})$ otherwise.

Next we inductively define the (k+1)-th *current hull* CH_{k+1}. It is easy to show there is a unique common tangent line $L_{p',p''}$ of CH_k and EC_{k+1} (with $x' < x''$ and $y' < y''$) such that $CH_k \cup EC_{k+1} \subset H^{UL}(L)$. If there is more than one choice of p' (resp. p''), we choose p' (resp. p'') so that the distance between p' and p'' is minimal. Further, let $FRONT\_CH_{k+1}$ denote the front subarc of CH_k between the points p_0 and p', and $REAR\_CH_{k+1}$ denote the rear subarc of EC_{k+1} between the points p'' and $p_{i_{k+1}}$. The (k+1)-th *current hull* CH_{k+1} is defined as the connected union $FRONT\_CH_{k+1} \cup L(p', p'') \cup REAR\_CH_{k+1}$, see Figure 4.1.4. CH_{k+1} is a convex arc along which both x and y-coordinates are strictly increasing. $L(p', p'')$ is called as the *lid* determined by p' and p''. Let $\bar{\gamma}$ be the closed path given as $\gamma(p', p'')$ followed by a path from p'' to p' along $L(p', p'')$. If $\bar{\gamma}$ has no self-intersection, the region bounded by $\bar{\gamma}$ is called as the *pocket* determined by the lid $L(p', p'')$. Otherwise, $\gamma(p', p'')$ has an even number of intersections with the lid $L(p', p'')$

counting intersections with multiplicities and $\bar{\gamma}$ divides the plane into a finite number of connected regions. The union of all the regions which are to the right of $\bar{\gamma}$ is the *pocket* implied by $L(p', p'')$, see Figure 4.1.5.

Properties of CH_k We prove two important properties of CH_k in the following Lemmas 3.1–3.2.

Lemma 3.14. *If a point $p \in C_i$ $(1 \leq i \leq i_k)$ is on the convex hull boundary, then $p \in CH_k$.*

Proof : Using induction we can easily show that the interior of the path $\gamma(p_{i_k}, p_{i_{k+1}-1})$, the arc $C_{i_{k+1}} \sim EC_{k+1}$, and $(CH_k \cup C_{i_{k+1}}) \sim CH_{k+1}$ are in the convex hull interior. □

Lemma 3.15. *If a point $p \in CH_k$ is on the convex hull boundary, then the front subarc of CH_k between p_0 and p is entirely contained in the convex hull boundary.*

Proof : The case $k = 1$ is easy to show. By induction, we assume for k $(1 \leq k < N)$ and consider $k + 1$. Suppose a point $p \in CH_{k+1}$ is on the convex hull boundary. (a) If $p \in FRONT\_CH_{k+1} \subset CH_k$, then the statement follows by induction. (b) If $p \in L(p', p'')$, then $L(p', p'')$ is also on the convex hull boundary. Further, $FRONT\_CH_{k+1}$ is on the convex hull boundary by induction. (c) If $p \in REAR\_CH_{k+1}$, then there is a supporting line L_p at p. We prove the lid $L(p', p'')$ is on the convex hull boundary. Suppose there is a boundary point q in the region R_1, see Figure 4.1.6. We may assume q is extreme to the outward normal direction of the lid and thus on the convex hull boundary. (i) If $q \in C_j$ $(1 \leq j \leq i_{k+1})$, then Lemma 4.1.2 implies $q \in CH_{k+1}$, however, this is impossible since CH_{k+1} is convex. (ii) Otherwise, there is a continuous path from $p_{i_{k+1}}$ to q. This path should pass through either the region R_2 or R_3, however, both are impossible. Hence, the lid $L(p', p'')$ is on the convex hull boundary, and by induction $FRONT\_CH_{k+1}$ is also on the convex hull boundary. Similarly one can show that the subarc of $REAR\_CH_{k+1}$ between p'' and p is on the convex hull boundary. □

Since p_M is on the convex hull boundary and the end point of CH_N, Theorem 3.1 follows easily from Lemmas 3.1–3.2.

Theorem 3.16. *CH_N is the lower–right part of the convex hull boundary between p_0 and p_M.*

Description of Algorithm We describe an algorithm to compute the sequences of *event edges* $\{C_{i_k}\}_{k=1}^{N}$ and *current hulls* $\{CH_k\}_{k=1}^{N}$ by using a single stack CH. CH contains segments of the k-th *current hull* CH_k which are subarcs of some *convex* edges, some *linear* edges and the lids of pockets. Adjacent elements on the stack share a common end point and the connected sequence of elements on the stack CH generates the *current hull* CH_k (we say then that "the stack CH *implies* the *current hull* CH_k").

Computing Event Edges We start with pushing a single point interval $[p_0, p_0]$ into an empty stack CH. The stack CH implies the *current hull* $CH_0 = \{p_0\}$. Assume i_k is detected and the stack CH implies the k-th *current hull* CH_k $(0 \leq k < N)$. We consider how to detect i_{k+1} using the stack CH. Since the cases 1 and 2 of §3.1.1 are easy, we consider only the case 3. We detect $j_0 = \min \Omega_k$ by using a loop variable j initialized to i_k and maintaining the following invariant for j at the beginning of each loop.
"p_j is outside (including the boundary) of any pocket of CH_k, and TOP(CH) is not strictly below the horizontal line $y = y_j$."
In each loop, j is first incremented by 1. (1) If $y_{j-1} \leq y_j$, then $j_0 = j$. (2) If $y_{j-1} > y_j$, then pop all the stack elements until (a) TOP(CH) which does not intersect with C_j and is not strictly above the line $y = y_j$, or (b) TOP(CH) which is a lid intersecting with C_j. In the case (a), repeat the j–loop. In the case (b), let p_* be the first intersection of C_j with $L(p', p'')$, walk along the path $\gamma(p_{j-1}, p_M)$ and count the numbers of Left–to–right cuts and Right–to–left cuts this path makes

with the lid $L(p', p'')$. At each intersection point p_{**}, let $p_* = p_{**}$ if $y_{**} \leq y_*$. While there are more Left–to–right cuts than Right–to-left cuts, we are inside of the pocket implied by $L(p', p'')$. If these two numbers become equal at a point $p_{**} \in C_{j'}$, compare y_{**} with $y_* = ymin_{j'}(L(p', p''))$. If $y_{**} > y_*$, then continue the walking along the path $\gamma(p_{**}, p_M)$. If $y_{**} \leq y_*$, then (i) let $j_0 = j'$ if $y_{j'-1} \leq y_{j'}$, and (ii) let $j = j'$, pretend p_** as p_j and repeat the j-loop otherwise. It is easy to show that the j–loop invariant holds everytime the loop is repeated and j_0 is detected correctly.

Computing Current Hulls We consider how to construct CH_{k+1} from CH_k and EC_{k+1} by using the stack CH. Though we have popped some stack elements from CH, the elements on the stack CH imply a convex arc Γ which contains $FRONT\_CH_{k+1}$ as its front subarc. To remove redundant elements $\Gamma \sim FRONT\_CH_{k+1}$ from CH, we consider the top stack element TOP(CH). We check whether TOP(CH) contains the common tangent point p' for the new lid $L(p'.p'')$ of CH_{k+1}. Since EC_{k+1} is not strictly below $y = y_S$, we have (a) $p' = p_E$ if and only if $EC_{k+1} \subset H^{UL}(L_{p_E})$ and EC_{k+1} is not strictly below the horizontal line $y = y_E$, (b) $p' \in$ TOP(CH) if and only if $EC_{k+1} \subset H^{UL}(L_{p_S})$, and (c) $p' \notin$ TOP(CH) otherwise, where p_S and p_E are the starting and ending points of TOP(CH), and L_{p_S} and L_{p_E} are the tangent lines of TOP(CH) at p_S and p_E respectively. In the cases (a) and (b), p' and p'' can be computed by using Lemmas 2.3–2.4. In the case (c), we pop TOP(CH) and repeat the same procedure. Once we have computed p' and p'', we can adjust the stack appropriately to imply CH_{k+1}.

Timing Analysis There are basically two types of operations in this algorithm, either operations to manipulate the edge sequence itself or operations to manipulate the stack. Each edge as input is processed within a constant number of steps before it is decided to be an event edge or not. If it is an event edge, a piece of this edge is pushed on the stack. While an edge is on the stack, it may be used to process other edges and/or changed into a shorter subsegment. Finally, it is popped if it is not on the convex hull boundary, or it appears on the final output if it is.

Before an edge is decided to be an event edge or not, we apply to this edge such operations like coordinate comparisons, an inner angle computation, and intersections with stack elements. If a stack element is popped after certain computations, we charge the cost to this popped stack element. Since the popping occurs at most once to an edge, the cost at the popping time will be charged at most once to each edge. Further, there is at most one stack element which is involved in the operation with the input edge and still remains on the stack. We can charge this cost and other trivial costs to the input edge. So far, the most expensive cost is line–curve segment intersection which is $O(d^3 \log d)$ (resp. $O(d^3 \log d + T(d))$) time for a parametric (resp. implicit) curve segment. Since there are total $O(n)$ input edges and popped edges, the total cost for event edge detections is $O(n \cdot d^3 \log d)$ (resp. $O(n \cdot (d^3 \log d + T(d)))$) time.

After an event edge is detected, the stack is modified to imply the new current hull. The stack elements are popped after two halfplane containment testings which cost $O(d^3 \log d)$ (resp. $O(d^3 \log d + T(d))$) time. The new stack is constructed by computing the lid $L(p', p'')$ and modifying itself appropriately, which costs at most $O(d^6 \log d)$ (resp. $O(d^{12} \log d + T(d))$) time for common tangent computation. Note that this last operation takes $O(1)$ time if both p' and p'' are known, and $O(d^3 \log d)$ ($O(d^6 \log d + T(d))$) time if one of p' and p'' is known. We charge this cost to the event edge. Thus, the total cost for current hull computation is $O(d^6 \log d)$ (resp. $O(d^{12} \log d + T(d))$) time, which is also the overall time complexity of this algorithm.

Decomposition of Monotone Boundary Edges In this section, we consider how to construct a simple carrier polygon of a geometric model object with at most $O(n^2)$ edges, which is optimal since this number is shown to be $\Omega(n^2)$ in [50]. Further, we consider how to construct a simple *characteristic* carrier polygon, an *inner* polygon and an *outer* polygon of the geometric model. We

assume the model boundary has been decomposed into $O(n)$ *monotone* edges in a preprocessing step as discussed in §2.

Simple Carrier Polygon Consider the horizontal vertex visibility partition of both the interior and exterior of a geometric model, see [50, 97], where the exterior is partitioned within a box enclosing the object. Let H be a visibility cell of the partition, and the right and left sides of H be bounded by the edges C and D respectively.Note that each side of H may be a proper subsegment of C or D. Let us assume H is in the interior of the geometric model and C is a convex edge. The cases of H being in the exterior and/or C being a concave edge can be handled in similar ways. Let p_B and p_T be the bottom and top points of the right side of H, and $\bar{C}$ be the subsegment of C between p_B and p_T. Further, let q_B and q_T be the bottom and top points of the left side of H, and $\bar{D}$ be the subsegment of D between q_B and q_T. To make the construction easier, we add p_B and p_T (resp. q_B and q_T) as extra vertices to C (resp. D). Since there are only $O(n)$ such extra vertices, the total number of edges after this decomposition is still $O(n)$.

We can add extra vertices $p_1, p_2, \ldots, p_{k_{\bar{C}}}$ to the edge $\bar{C}$ so that the *carrier polygonal arc* $P_{\bar{C}}$ connecting the vertices $p_B, p_1, \ldots, p_{k_{\bar{C}}}, p_T$ does not intersect with any other part of the carrier polygon except at p_B and p_T and also with the extra vertices the carrier polygon has at most $O(n^2)$ edges. This is achieved by the following construction. At each vertex p, construct a horizontal line L containing p and parallel to the x-axis. Let $p_1, p_2, \ldots, p_{k_{\bar{C}}}$ (resp. $q_1, q_2, \ldots, q_{k_{\bar{D}}}$) be the intersection points of $\bar{C}$ (resp. $\bar{D}$) with these horizontal lines. Then the corresponding carrier polygonal arcs $P_{\bar{C}}$ and $P_{\bar{D}}$ do not intersect except at the end points and further it follows that the carrier polygon $\cup P_{\bar{C}}$ is simple. Since there are $O(n)$ such horizontal lines and boundary edges, and $k_{\bar{C}} = O(n)$, the carrier polygon has $O(n^2)$ edges. Though $O(n^2)$ is optimal asymptotically, the above construction does not generate the minimal number of extra vertices. Steps can be taken to reduce the number of extra vertices added. For example, when the chord $L(p_B, p_T)$ of $\bar{C}$ does not intersect with $\bar{D}$ except at p_B and p_T, we do not need to add any extra vertices. Thus $P_{\bar{C}}$ is $L(p_B, p_T)$ with $k_{\bar{C}} = 0$. Further, when $L(p_B, p_T)$ intersects with $\bar{D}$ at a point other than p_B and p_T, we construct $P_{\bar{C}}$ so that $P_{\bar{C}}$ does not intersect with $P_{\bar{D}}$ except at p_B and p_T though $P_{\bar{C}}$ may intersect with the edge $\bar{D}$. Thus, we construct $P_{\bar{D}}$ recursively. Assume we have constructed $P_{\bar{D}}$ by adding a small number of extra vertices to $\bar{D}$. Then by scanning H from the top to the bottom, we can add at most $k_{\bar{D}}$ extra vertices to the edge $\bar{C}$ to make the corresponding carrier polygonal arc $P_{\bar{C}}$ simple. Thus we have the relation $k_{\bar{C}} \leq k_{\bar{D}}$. Though for simplicity we assume each boundary edge is segmented into monotone edges and each monotone edge is further decomposed so that each side of H is an edge, we can easily modify the above construction so that we may need to add extra vertices only to y-extreme points and apply the same recursive construction to add a minimal number of extra vertices to each y-monotone edge.

Theorem 3.17. *Assume all the monotone edges are parametric (resp. implicit) algebraic curve segments. A simple carrier polygon of an object with at most $O(n^2)$ edges can be constructed in $O((n \log\log n + k) \cdot d^3 \log d)$ (resp. $O((n \log\log n + k) \cdot (d^3 \log d + T(d)))$) time, where k is the number of edges in the simple carrier polygon.*

Proof: The horizontal vertex visibility partition of both the interior and exterior of an object can be constructed in $O(n \log\log n \cdot d^3 \log d)$ (resp. $O(n \log\log n \cdot (d^3 \log d + T(d)))$) time. A simple carrier polygon can be constructed in $O(k \cdot d^3 \log d)$ (resp. $O(k \cdot (d^3 \log d + T(d)))$) time by computing $O(k)$ line–curve segment intersections. □

Simple Characteristic Carrier Polygon A carrier polygon is *characteristic* if, for each edge E, $S(E)$ is totally contained either in the interior of the geometric model or in the exterior of the model, where $S(E)$ is the convex region bounded by the edge E and its chord. If E is a convex (resp. a concave) edge, then $S(E)$ is totally contained in the interior (resp. in the exterior) of the

model and is called an *additive* (resp. a *subtractive*) convex region. Assume C and D are the same as $\bar{C}$ and $\bar{D}$ of §3.1 respectively. We can add extra vertices p_1, p_2, ..., p_{k_C} to the edge C so that the convex regions of the decomposed subsegments of C are *additive* convex regions. The case of C being a concave edge can be handled in a similar way and the convex regions of the decomposed subsegments of C become *subtractive* convex regions in this case. This decomposition is achieved as follows. If $S(C)$ and $S(D)$ do not intersect, we do not add any extra vertex to C and $S(C)$ is an additive convex region, thus $k_C = 0$. Otherwise, let L_1 be the tangent line from p_B to D, p_1 be the intersection point of L_1 with C, and C_1 be the subsegment of C between p_B and p_1. Then, $S(C_1)$ is an additive convex region. Let $\bar{C}$ be the subsegment of C between p_1 and p_T. If $S(\bar{C})$ intersects with $S(D)$, then we compute a tangent line L_2 from p_1 to D and the intersection point p_2 of L_2 and C, and repeat the same procedure. Otherwise, $S(\bar{C})$ is an additive convex region. Now, we prove the decomposition of C terminates within a finite number of steps.

Theorem 3.18. *Assume no vertex has its inner angle as 0 or 2π. Each edge C can be decomposed into a finite number of subedges $C_1, C_2, \ldots, C_{k_C}$ so that the convex regions are additive.*

Proof Suppose there is an infinite sequence of C_i's (i = 1, 2, ...) constructed as above. Let p_i be the end point of C_i, then $x_S < x_i < x_{i+1} < x_E$ for all i. Since x_i is a strictly increasing sequence bounded above, $x_i \to x$ for some $x \leq x_E$. Let $p_L \in C$ be such that $x_L = x$, then $p_i \to p_L$. In an arbitrary small neighborhood U of p_L, there is an integer N such that $p_i \in U$ and $C_i \subset U$ for all $i \geq N$. Let q_i be the tangent point of $L(p_{i-1}, p_i)$ with D, then $q_i \in U$ for all $i \geq N$. Since q_i's are on D and in the interior of $S(C)$, this means that p_L is a limit point of D which are in the interior of $S(C)$. We can easily show that D is a concave edge and p_L is a common vertex of C and D, and further the inner angle at $p_L > 3\pi/2$ (resp. $< \pi/2$). Since $L(p_{i-1}, p_i)$ is tangent to D at q_i with $p_i \to p_L$ and $q_i \to p_L$, C and D have the same tangent line at p_L. Hence, the inner angle of the model at p_L is 2π (resp. 0). It is impossible since we assume no such vertex on the geometric model boundary. □

We call the polygonal arc P_C^χ connecting the vertices $p_S, p_1, \ldots, p_{k_C}, p_E$ as the *first characteristic polygonal arc* of C, the union $\cup P_C^\chi$ as the *first characteristic polygon* of the geometric model, and $K = \sum(k_C + 1)$ as the *characteristic number* of the object. It is easy to show that the edges of characteristic polygon do not intersect each other transversally and two different vertices do not overlap. However, a vertex may lie on the interior of some characteristic polygon edge. By decomposing each edge with a vertex on its chord interior into two subedges, we can make the carrier polygon simple. Thus, using at most $2K$ edges we can construct a simple characteristic carrier polygon.

Theorem 3.19. *Assume all the monotone edges are parametric (resp. implicit). A simple characteristic carrier polygon of an object with at most $2K$ edges can be computed in $O((n \log\log n + K) \cdot d^3 \log d)$ (resp. $O((n \log\log n + K) \cdot (d^6 \cdot \log d + T(d)))$) time.*

Proof: After the horizontal vertex visibility partition is computed, a simple characteristic carrier polygon can be constructed in $O(K \cdot d^3 \log d)$ (resp. $O(K \cdot (d^3 \log d + T(d)))$) time by computing $O(K)$ tangent lines from given points. □

Inner and Outer Polygons Let C be a monotone edge with p_S as its starting point and p_E as its ending point. For any two different points p and q on C, $L(p, q)$ denotes the line segment connecting p and q, and L_p denotes the tangent line of C at p. Let p^* be the intersection point of two tangent lines L_{p_S} and L_{p_E}. We call the line segment $L(p_S, p_E)$ as the *chord* of C and the polygonal arc $\wedge(C) = L(p_S, p^*) \cup L(p^*, p_E)$ as the *wedge* of C. Let p_S, p_1, p_2, ..., p_k, p_E be a sequence of points on C in the order they appear along C, then the polygonal arc $P_C^{chord}(p_S, p_1, p_2, \ldots, p_k, p_E)$ $= L(p_S, p_1) \cup L(p_1, p_2) \cup \ldots \cup L(p_k, p_E)$ is called as the *chordal* polygonal arc of C determined by

the sequence p_S, p_1, p_2, ..., p_k, p_E. Let p_i^* be the intersection point of $L_{p_{i-1}}$ and L_{p_i} ($i = 1, \ldots, k+1$), where $p_0 = p_S$ and $p_{k+1} = p_E$. Then the polygonal arc $P_C^{tangent}(p_S, p_1, p_2, \ldots, p_k, p_E) = L(p_S, p_1^*) \cup L(p_1^*, p_2^*) \cup \ldots \cup L(p_k^*, p_{k+1}^*) \cup L(p_{k+1}^*, p_E)$ is called as the *tangential* polygonal arc of C determined by the sequence p_S, p_1, p_2, ..., p_k, p_E.

If we decompose the geometric model boundary so that the chords of convex (resp. concave) decomposed edges and the wedges of concave (resp. convex) decomposed edges are totally contained in the interior (resp. the exterior) of the model, then the union of these chords and wedges defines an *inner* (resp. *outer*) polygon which is totally contained in the interior (resp. the exterior) of the model. In the following, we will consider the construction of *inner* polygonal arcs P_C^{chord} of edges C. The *outer* polygonal arcs $P_C^{tangent}$ of edges C can be constructed in a similar way.

We first consider the case of C being a convex edge. Let P_C^χ be the first characteristic polygonal arc of C, then C is to the right of P_C^χ and D is to the left of P_C^χ. We may assume $p_B \neq q_B$ or $p_T \neq q_T$. If D is a convex edge, then P_C^χ is the chord $L(p_B, p_T)$ of C and the chords of C and D do not intersect except at an end point. Further, the chords of C and D are contained in the interior of the object. We call $L(p_B, p_T)$ and $L(q_B, q_T)$ as the *inner* polygonal arcs of C and D respectively. Now, if D is a concave edge, there are points q_1, q_2, ..., q_{k_C} on D which are tangent to some lines segments on P_C^χ. Let p_B, p_1, p_2, ..., p_{k_C}, p_T be the vertices of P_C^χ in the order they appear along C. Note that $q_1 = q_B$ if $p_B = q_B$, and $k_C = 0$ if $D \cap P_C^\chi = \emptyset$. Further, q_i is the tangent point of $L(p_{i-1}, p_i)$ on D ($i = 1, \ldots, k_C$), where $p_0 = p_B$ and $p_{k_C+1} = p_T$. We add an extra vertex p'_i to each subedge of C between p_{i-1} and p_i ($i = 1, \ldots, k_C + 1$). We call the polygon P_C^{inner} connecting $p_B, p'_1, p_1, \ldots, p_{k_C}, p'_{k_C+1}, p_T$ as the *inner* polygonal arc of C. P_C^{inner} is strictly to the right of P_C^χ except the points $p_B, p_1, p_2, \ldots, p_{k_C}, p_T$. Further, we add an extra vertex q'_i to each subedge of D between q_{i-1} and q_i ($i = 1, \ldots, k_C + 1$), where $q_0 = q_B$ and $q_{k_C+1} = q_T$. We choose q'_{k_C+1} so that the tangent line of D at this point is parallel to the line segment $L(p_{k_C-1}, p_{k_C})$. Note that $q_B = q'_1 = q_1$ if $p_B = q_B$, and $q_T = q'_{k_C+1}$ if $p_T = q_T$. We call the tangential polygonal arc $P_D^{tangent}(q_B, q'_1, q_1, \ldots, q_{k_C}, q'_{k_C+1}, q_T)$ as the inner polygonal arc P_D^{inner} of D.

We consider the case of C being a concave edge. Since the case of D being convex can be handled in a similar way as above, we assume D is concave. There are two common tangent lines L_1 and L_2 of C and D. Let p^* be the intersection point of L_1 and L_2, and L be a line containing the point p^* and having its slope strictly between the slopes of L_1 and L_2. Let p' (resp. q') be a point on C (resp. D) at which C (resp. D) has a tangent line parallel to the line L. We call the tangential polygonal arc $P_C^{tangent}(p_B, p', p_T)$ (resp. $P_D^{tangent}(q_B, q', q_T)$) as the *inner* polygonal arc P_C^{inner} of C (resp. P_D^{inner} of D). Since P_C^{inner} (resp. P_D^{inner}) is strictly to the right (resp. to the left) of L, $P_C^{inner} \cap P_D^{inner} = \emptyset$.

Theorem 3.20. *Assume all the monotone edges are parametric (resp. implicit). Both inner and outer carrier polygons of an object with at most $2K$ edges can be computed in $O((n \log\log n + K) \cdot d^3 \log d)$ (resp. $O((n \log\log n + K) \cdot (d^6 \cdot \log d + T(d)))$) time.*

Proof: Similar to Theorem 3.2. □

Object Decompositions We consider various applications of the polygons P^χ and P^{inner} constructed in §3 to the object decompositions. Dobkin, Souvaine, and Van Wyk [50] shows an $O(n \log\log n)$ algorithm to decompose a simple splinegon into a union of differences of unions of possibly overlapping convex pieces. Our decomposotion may involve $O(K)$ where K is often linear in practice. Further, the decomposition structure in terms of unions and differences is simpler than that of [50]. However, when the simple characteristic polygon has small number of edges like K is almost linear, then our decomposition method may be more useful since this decomposition has nicer structure.

Convex Decomposition We can decompose the simple characteristic polygon P^{χ} into unions of disjoint convex polygons $\cup_i P_i$, (see [70] for a survey on convex decomposition algorithms for simple polygons). Let $\cup_j U_j$ (resp. $\cup_k V_k$) be the union of all the *additive* (resp. *subtractive*) convex regions. Then, the original object can be represented as $(\cup_i P_i) \cup (\cup_j U_j) \sim (\cup_k V_k)$. Further, the interiors of P_i's and U_j's are disjoint, and the interiors of U_j's and V_k's are disjoint, however, the interiors of P_i's and V_k's may have non-empty intersections. Thus the orders of union operations in the unions $(\cup_i P_i) \cup (\cup_j U_j)$ and $\cup_k V_k$ are interchangeable. Further, as long as $\cup_i P_i$ has been computed first, the order of adding each U_j and subtracting each V_k is interchangeable. The construction of P^{χ} is highly parallel and would be useful in a parallel implementation of the model decomposition algorithm.

Primitive Decomposition The main purpose of geometric model decomposition is to simplify a problem for complex geometric model into a number of simpler subproblems dealing with "nice" boundary. In most of the cases a decomposition in terms of a union of disjoint convex pieces is useful and this is always possible for simple polygons. However, this fact is certainly not true for planar geometric models. In §4.1 we thus considered an alternative way, namely in decomposing an object into unions and differences of convex objects. However, in some applications involving a Minkowski operation (i.e. convolution) which commutes with union we may consider decomposing a geometric model into a disjoint union of certain primitive objects.

We can decompose an inner polygon P^{inner} into a union of disjoint convex polygons $\cup_i P_i^{inner}$. The difference between the model and the inner polygon P^{inner} is the union $(\cup_j U_j^{inner}) \cup (\cup_k V_k^{inner})$, where each U_j^{inner} is an additive convex region bounded by a convex edge and its chord and each V_k^{inner} is a region bounded by a concave edge and its wedge. We can thus represent the original model as a disjoint union $(\cup_i P_i^{inner}) \cup (\cup_j U_j^{inner}) \cup (\cup_k V_k^{inner})$.

Now, consider the application of this simple decomposition to Minkowski operation. For objects A and B, the Minkowski addition $A \oplus B = \{a+b \mid a \in A \text{ and } b \in B\}$ and the Minkowski subtraction $A \ominus B = \{a - b \mid a \in A \text{ and } b \in B\}$. Let $A = \cup_i S_i$ and $B = \cup_{i'} S'_{i'}$, where S_i and $S'_{i'}$ are simple objects. Then $A \oplus B = \cup(S_i \oplus S'_{i'})$ and $A \ominus B = \cup(S_i \ominus S'_{i'})$.

3.4 Offsets and Convolutions

Using configuration space (*C-space*) to plan motion for a single rigid object among physical obstacles, reduces the problem to planning motion for a mathematical point among "grown" configuration space obstacles, (the points in *C-space* which correspond to the object overlapping one or more obstacles). For example, a rigid polyhedral object in compliant motion, viz., in continuous contact with the boundary of obstacles in three-dimensional space can be represented as a point constrained to move on the boundary of grown obstacles embedded in six-dimensional *C-space*, see also [38]. The compliant motion technique thus initially relies in efficiently generating the boundary of *C-space obstacles*. There exists numerous applications such as automated assembly, numerical machining and part tolerancing, where motion planning in *C-space* has proved useful in generating robust compliant and fine motion strategies

The only efficient algorithms known for generating *C-space* obstacles have been for polyhedral (degree 1) surface objects and obstacles, using methods for efficiently computing convex hulls, and recently efficient convolution algorithms for Minkowski sum. The methods based on generating a cylindrical cell decomposition of free *C-space*, though applicable for general objects and obstacles defined by semi-algebraic sets, are computationally too restrictive.Thus, often the object representations that have been considered for planning motion in three-dimensional space, have been polyhedral approximations to the curved object. However, if we approximate object and obstacle by polyhedra, then for example, the compliant motion paths obtained are "jumpy" or provide discrete contact motion. One solution to obtaining a continuous compliant motion includes generation

of the curved surface boundary of the *C-space* obstacle. Further, it has progressively become easier for geometric modeling systems to deal with objects that are defined by quadrics and higher degree surfaces.Motion planning in these sophisticated modeling environments, for product prototyping and simulation, suggests the need to efficiently generate the surface boundary of *C-space obstacles* arising from the motion of objects among obstacles with algebraic surface boundaries. As we show, one is required to generate only the specific part of the *C-space* obstacle boundary which contains the desired compliant path. Piecewise algebraic, approximate shortest paths (approximate geodesics) can then also be obtained by projecting on the curved *C-space* obstacle the shortest path obtained on approximated *C-space* obstacles.

The main contributions of this paper are as follows. In §3 we very briefly show that the boundary of *C-space obstacles* for general convex curved objects moving with fixed orientation can be viewed as the convolution between the obstacle boundary and the reversed object boundary (reversed with respect to a reference point on the object). In §4 we present algebraic algorithms to generate the curves and surfaces which make up the boundary of the three-dimensional *C-space obstacles*. Here again we only consider objects and obstacles which are convex. These objects and obstacles are represented by a general algebraic boundary representation model discussed in §2. Crucial to the algorithm time complexity and the resulting algebraic surface degree of the *C-space* obstacles is the internal representation of object curves and surfaces, i.e., whether they are parametrically or implicitly defined.[2] We present algorithms for both these internal representations. Next in §5 we show how to construct the topology of the *C-space* obstacle boundary. Use is made of a Gaussian model discussed in §2. In §6 we consider motion of a point on the boundary of *C-space* obstacles. With only translational parameters for the moving object one essentially considers compliant motion wherein the contact points between object and obstacle change during motion. Requiring the contact points to remain the same during compliant motion necessitates the introduction of rotational parameters and thus higher dimensional *C-spaces*. For compliant motion we present algebraic algorithms to compute various paths on curved algebraic surfaces in three dimensions. Locally shortest or geodesic curves on algebraic surfaces are also considered. Exact algebraic algorithms for geodesics are impossible in general because of the existence of non-algebraic or transcendental geodesic curves. However, here we introduce a Gaussian polyhedral approximation model which allows efficient piecewise algebraic approximations of geodesic paths on curved surfaces.

3.4.1 C-space Obstacles and Convolution

Let A be a moving object with its reference point at the origin and B be a fixed obstacle, both compact subsets of R^3. A and B are modeled by the boundary representations of §2. We consider object A to be free to move with fixed orientation. In this case configuration space is also three-dimensional. We make the following definitions.
(1) *CO(A,B)* = "*C-space* obstacle due to A and B" $= \{\bar{p} \in R^3 \mid A_{\bar{p}} \cap B \neq \emptyset\}$, where $A_{\bar{p}} = \{\bar{p} + q \mid q \in A\}$.
(2) $\textsc{Convolution}(G_{-A}, K_B)$ = "Convolution of G_{-A} and K_B" $= \{\bar{p} \in R^3 \mid \bar{p} = p - q$ where $p \in K_B$ and $q \in G_{-A}$ and B has an outward normal direction at p exactly opposite to an outward normal A has at $q\}$, where $-A = \{-p \mid p \in A\}$, $K_B \subset$ *Bdr(B)* and $G_{-A} \subset Bdr(-A)$.
We then note the following.

Theorem 3.21. *For convex A and B, we have $Bdr(CO(A,B))$ = Convolution($Bdr(-A)$,$Bdr(B)$).*

Proof : See [22]. □

[2] For example, a unit sphere is implicitly given as $x^2 + y^2 + z^2 - 1 = 0$ and in rational parametric form as $x = \frac{(1-s^2-t^2)}{(1+s^2+t^2)}, y = \frac{2s}{(1+s^2+t^2)}$ and $z = \frac{2t}{(1+s^2+t^2)}$.

The above Theorem may suggest a natural method for handling *non–convex* object and obstacle shapes. One first obtains a convex decomposition consisting of the union of convex pieces and then generates the *C-space* obstacle as the union of *C-space* obstacles for convex object and obstacle pairs, (since "convolution" commutes with "union"). Disjoint convex decompositions are possible for polyhedral objects, [40]. However not all objects with algebraic curve and surface boundaries permit decompositions consisting of the union of convex pieces, see [21]. For example a complete toroidal surface cannot be decomposed into a finite union of convex pieces. To obtain convex decompositions of general curved solid objects, when possible, say in terms of union, intersection and difference, is a difficult and as yet unresolved problem. We restrict our attention to convex shaped moving objects and obstacles.

3.4.2 Generating the Boundary of C-space Obstacles

Let T be $-A$ or B, $p \in Bdr(T)$ be a boundary point, $E \subset Bdr(T)$ be an edge, and $F \subset Bdr(T)$ be a face. Let (F_T, N_{F_T}) be a pair such that $F_T \subset Bdr(T)$ is a face and $N_{F_T} = N(T, F_T)$, where $N(T, \cdot)$ is the *Gaussian Map* of T. Let (E_T, N_{E_T}) be a pair such that $E_T \subset Bdr(T)$ is an edge and $N_{E_T} \subset N(T, E_T)$ with $N_{E_T} \cap N(T, p) \neq \emptyset$ for all $p \in E_T$. Let (p_T, N_{p_T}) be a pair such that $p_T \in Bdr(T)$ is a vertex and $N_{p_T} \subset N(T, p_T)$ with $N_{p_T} \neq \emptyset$. Further let K_B be F_B, E_B or p_B, and let G_{-A} be F_{-A}, E_{-A} or p_{-A}. There are nine possible (K_B, G_{-A}) pairs. We define *sub-compatible* and *compatible* pairs as follows.

1. K_B and G_{-A} are *sub-compatible* $\Longleftrightarrow N(B, K_B) \cap N(-A, G_{-A}) \neq \emptyset$
2. (K_B, N_{K_B}) and $(G_{-A}, N_{G_{-A}})$ are *compatible* $\Longleftrightarrow N_{K_B} = N_{G_{-A}}$

Further denote by $K_B \sim G_{-A}$ the fact that K_B and G_{-A} are *sub-compatible*. Only sub-compatible pairs can contribute to the CONVOLUTION, one can show that $Convolution(Bdr(-A), Bdr(B)) = \cup_{K_B \sim G_{-A}}\ Convolution(G_{-A}, K_B)$. We can further refine the right-hand side to be a union of only the *compatible* pairs as follows. For a sub-compatible (K_B, G_{-A}) pair, let $N(K_B, G_{-A}) = N(B, K_B) \cap N(-A, G_{-A})$ be the nonempty intersection of two *Gaussian Images* of K_B and G_{-A}. $K(K_B, G_{-A}) = N^{-1}(B, N(K_B, G_{-A})) \subset K_B$ and $G(K_B, G_{-A}) = N^{-1}(-A, N(K_B, G_{-A})) \subset G_{-A}$ be the *Inverse Gaussian Images* of $N(K_B, G_{-A})$. Then $(K(K_B, G_{-A}), N(K_B, G_{-A}))$ and $(G(K_B, G_{-A}), N(K_B, G_{-A}))$ are *compatible*. Hence, *Convolution* $(Bdr(-A), Bdr(B)) = \cup_{K_B \sim G_{-A}}\ Convolution(G(K_B, G_{-A}), K(K_B, G_{-A}))$ and we only need to consider *compatible* pairs to generate the CONVOLUTION.

When (K_B, N_{K_B}) and $(G_{-A}, N_{G_{-A}})$ are *compatible* with at least one of K_B or G_{-A} being a vertex, the CONVOLUTION generation is especially easy, i.e. $Convolution(G_{-A}, K_B) = K_B \oplus G_{-A}$, where $T_1 \oplus T_2$ = "the Minkowski sum of T_1 and T_2" $= \{p_1 + p_2 \mid p_1 \in T_1 \text{ and } p_2 \in T_2\}$ for sets T_1 and $T_2 \subset R^3$. Let $Ch(p)$ = "the *characteristic set* of p" $= \{\bar{p} = p + q \mid N(B, p) \cap N(-A, q) \neq \emptyset\}$. $Ch(E) = \cup_{p \in E} Ch(p)$ is called the *characteristic set* of E, and $Ch(F) = \cup_{p \in F} Ch(p)$ is called the *characteristic set* of F. One can easily show that $Convolution(Bdr(-A), Bdr(B)) = (\cup_{F \in \Gamma_1} Ch(F)) \cup (\cup_{E \in \Gamma_2} Ch(E)) \cup (\cup_{p \in \Gamma_3} Ch(p))$, where Γ_1 is the set of all faces of $Bdr(B)$, Γ_2 is the set of all edges of $Bdr(B)$, and Γ_3 is the set of all vertices of $Bdr(B)$.

In §4.1–4.3 we consider both the implicit and rational parametric representation of curves and surfaces since not all algebraic curves and surfaces have rational parametrization, [107]. For the class of *rational* algebraic curves and surfaces (which have a rational parametric form), algebraic algorithms also exist for converting between the implicit and parametric representations. However their efficiency are limited to curves and surfaces of low degree, see [3].

Growing Faces

For a face $F \subset Bdr(B)$, one can easily show that $Ch(F) = (\cup_{F' \sim F} Convolution(G(F, F'), K(F, F')))$

$\cup(\cup_{E\sim F} Convolution(G(F,E),K(F,E))) \cup (\cup_{q\sim F} Convolution(q,K(F,q)))$. One can use Theorem 3.22 of §4.1 to compute $Convolution(G(F,F'),K(F,F'))$ and Theorem 3.26 of §4.2 to compute $Convolution(G(F,E),K(F,E))$, while directly computing $Convolution(G(F,q),K(F,q)) = K(F,q)\oplus\{q\}$ as a simply translated surface patch.

Growing Edges

For an edge $E \in Bdr(B)$, one can easily show that $Ch(E) = (\cup_{F\sim E} Convolution(G(E,F),K(E,F))) \cup(\cup_{E'\sim E} Convolution(G(E,E'),K(E,E'))) \cup (\cup_{q\sim E} Convolution(q,K(E,q)))$. One can use Theorem 3.26 of §4.2 to compute $Convolution(G(E,F),K(E,F))$, and Theorem 3.29 of §4.3 to compute $Convolution(G(E,E'),K(E,E'))$, while directly computing $Convolution(q,K(E,q)) = \{q\}\oplus K(E,q)$ as a simply translated edge segment.

Growing Vertices

For a vertex $p \in Bdr(B)$, one can easily show that $Ch(p) = (\cup_{F\sim p} Convolution(G(p,F),p)) \cup (\cup_{E\sim p} Convolution(G(p,E),p)) \cup (\cup_{q\sim p} Convolution(q,p))$. Since one has $Convolution(G(p,F),p) = G(p,F)\oplus\{p\}$, $Convolution(G(p,E),p) = G(p,E)\oplus\{p\}$, and $Convolution(q,p) = \{q+p\}$, computing $Ch(p)$ is easy.

Note: (1) For a non–smooth edge E and a non–smooth vertex p the convolution edge *Convolution* $(G(p,E),p) = G(p,E)\oplus\{p\}$ is a non–smooth edge, and (2) for non–smooth vertices p and q the convolution vertex $Convolution(q,p) = \{q+p\}$ is a non–smooth vertex. As we shall see in §4.3, (3) we can also have a non–smooth convolution edge $\textsc{Convolution}(E_{-A},E_B)$ for a *parallel* edge pair E_{-A} and E_B. These are all the non–smooth edges and vertices that may arise on the *C-space* obstacle boundary. Note in this classification that all non–smooth edges and vertices on the *C-space* obstacle boundary result from very special orientations between the non–smooth edges and vertices of A and B. In general the non–smoothness in the boundaries of A and B are removed while generating the *C-space* obstacle boundary. This smoothing effect of convolution, however, raises another question viz., how to specify the edge boundary of a convolution surface patch. Since most of the adjacent convolution faces meet tangentially to each other, computation of the intersecting edge may be quite unstable. One solution is to additionally determine auxiliary surfaces which intersect transversally with the convolution surfaces.

Generating Convolution(F_{-A},F_B) In this section, we consider how to generate the surfaces, edges and vertices of a convolution surface patch $\textsc{Convolution}(F_{-A},F_B)$. We can use Theorem 3.22 for the case of F_{-A} and F_B being implicitly defined algebraic surfaces. Corollary 3.23 is useful when F_{-A} is implicit and F_B is parametric, or the other way around. Corollary 3.24 is useful when both F_{-A} and F_B are parametrically defined. For *sub-compatible* F_B and F_{-A}, we are using the notations $N(F_B,F_{-A}) = N(B,F_B)\cap N(-A,F_{-A})$, $K(F_B,F_{-A}) = N^{-1}(B,N(F_B,F_{-A})) \subset F_B$, and $G(F_B,F_{-A}) = N^{-1}(-A,N(F_B,F_{-A})) \subset F_{-A}$.

Theorem 3.22. *Let $F_B \subset Bdr(B)$ be a patch of an algebraic surface $f=0$ with gradients ∇f. Further let $F_{-A} \subset Bdr(-A)$ be a patch of an algebraic surface $g=0$ with gradients ∇g, and suppose that F_B and F_{-A} are* sub-compatible. *Then $Convolution(F_{-A},F_B) = Convolution(G(F_B,F_{-A}), K(F_B,F_{-A}))$ is the set of points $\bar{p} = (\bar{x},\bar{y},\bar{z}) = p+q = (x+\alpha, y+\beta, z+\gamma)$ such that*

$$\begin{cases} f(x,y,z)=0 \text{ and } p=(x,y,z)\in K(F_B,F_{-A}) & (1)\\ g(\alpha,\beta,\gamma)=0 \text{ and } q=(\alpha,\beta,\gamma)\in G(F_B,F_{-A}) & (2)\\ \nabla f\times\nabla g = 0 & (3)\\ \nabla f\cdot\nabla g > 0 & (4)\end{cases}$$

Proof : Since (3)–(4) imply ∇f and ∇g are in the same direction, (3)–(4) are equivalent to the outward normal direction of B at p to be the same as that of $-A$ at q. $\square$

We use Theorem 3.22 as follows. First substitute $x = \bar{x} - \alpha$, $y = \bar{y} - \beta$ and $z = \bar{z} - \gamma$ in the above equations (1) and (3). Then one obtains the implicit algebraic equation of the $Convolution(F_{-A}, F_B)$ in terms of $\bar{x}, \bar{y}$ and $\bar{z}$ by eliminating α, β and γ from the systems of equations (1)–(3). The vector equation (3) yields three polynomial equations. One of these equations is redundant, leaving two independent equations from (3). Hence, from four non-homogeneous polynomial equations we eliminate three variables α, β, γ to construct the implicit equation in terms of $\bar{x}, \bar{y}, \bar{z}$. Formulas for obtaining a single multivariate resultant polynomial whose vanishing is a necessary and sufficient condition that a system of n homogeneous polynomials in n variables has nontrivial solutions, have been given by Cayley (see [85]), Hurwirtz (see [100]) and Macaulay [71]. The multivariate resultant is a homogeneous polynomial in the coefficients of the system of polynomials. In our case, we have a system of $n = 4$ polynomial equations in variables α, β, γ and ω (a homogenizing variable introduced to make the polynomial equations homogeneous).

In computing the implicit equation of the *C-space* surfaces a time complexity analysis may be done as follows. On substituting $x = \bar{x} - \alpha$, $y = \bar{y} - \beta$ and $z = \bar{z} - \gamma$ in equations (1) and (3) one has to expand each term $c_{ijk} \cdot x^{d_i} \cdot y^{d_j} \cdot z^{d_k} = c_{ijk} \cdot (\bar{x} - \alpha)^{d_i} \cdot (\bar{y} - \beta)^{d_j} \cdot (\bar{z} - \gamma)^{d_k}$ where $d_i + d_j + d_k \leq d$. This is necessary because in computing resultants to eliminate α, β, γ one needs to simplify the equations to be polynomials in α, β, γ with coefficients in $\bar{x}, \bar{y}, \bar{z}$. The polynomial f and the component polynomials of $\nabla f \times \nabla g$ have $O(d^6)$ terms of the form $c_{ijklmn} \cdot (\bar{x} - \alpha)^{d_i} \cdot (\bar{y} - \beta)^{d_j} \cdot (\bar{z} - \gamma)^{d_k} \cdot \alpha^{d_l} \cdot \beta^{d_m} \cdot \gamma^{d_n}$ ($d_i, d_j, d_k \leq d$ and $d_l, d_m, d_n \leq d$). And each of these terms will be expanded into $O(d^3)$ terms of $c'_{ijklmn} \cdot \bar{x}^{d'_i} \cdot \bar{y}^{d'_j} \cdot \bar{z}^{d'_k} \cdot \alpha^{d'_l} \cdot \beta^{d'_m} \cdot \gamma^{d'_n}$ ($d'_i, d'_j, d'_k \leq d$ and $d'_l, d'_m, d'_n \leq 2d$). Hence, the overall time complexity for expansion and simplification is $O(d^9)$. For an efficient computation of the multivariate resultant for a system of n homogeneous polynomials of degree d, we rely on the $O(N^2)$ algorithm of [38], where $N = \begin{pmatrix} nd \\ n-1 \end{pmatrix} = O((nd)^{n-1})$. In our case of $n = 4$, this yields an overall time bound of $O(d^6)$ to compute the implicit equation of the convolution face. The degree of the multivariate resultant in terms of the coefficients of any of the equations is equal to the product of the degrees of the remaining equations, see [71]. In our four equations the powers of the variables α, β, γ and the coefficients $\bar{x}, \bar{y}, \bar{z}$ are all $O(d)$ which give an overall degree bound for the convolution faces to be at most $O(d^4)$.

Corollary 3.23. *Let $F_B \subset Bdr(B)$ be a patch of an algebraic surface $f = 0$ with gradients ∇f. Further let $F_{-A} \subset Bdr(-A)$ be a parametric surface patch $G(u, v) = (\alpha(u, v), \beta(u, v), \gamma(u, v))$ with gradients $G_u \times G_v$, and suppose that F_B and F_{-A} are* sub-compatible. *Then $Convolution(F_{-A}, F_B) = Convolution(G(F_B, F_{-A}), K(F_B, F_{-A}))$ is the set of points $\bar{p} = (\bar{x}, \bar{y}, \bar{z}) = p + q = (x + \alpha(u, v), y + \beta(u, v), z + \gamma(u, v))$ such that*

$$\begin{cases} f(x, y, z) = 0 \text{ and } p = (x, y, z) \in K(F_B, F_{-A}) & (1) \\ q = (\alpha(u, v), \beta(u, v), \gamma(u, v)) \in G(F_B, F_{-A}) & (2) \\ \nabla f \times (G_u \times G_v) = 0 & (3) \\ \nabla f \cdot (G_u \times G_v) > 0 & (4) \end{cases}$$

First substitute $x = \bar{x} - \alpha(u, v)$, $y = \bar{y} - \beta(u, v)$ and $z = \bar{z} - \gamma(u, v)$ in the above equations (1) and (3). Then one can obtain the implicit algebraic equation of the $Convolution(F_{-A}, F_B)$ in terms of $\bar{x}, \bar{y}$ and $\bar{z}$ by eliminating u and v from the equations (1) and (3) by computing the multivariate resultant. Since (3) yields two independent equations, we have three polynomial equations and eliminate two variables u and v to construct the implicit equation.

Since $G(u, v)$ is a rational parametric surface, we have $\alpha(u, v) = p(u, v)/w(u, v)$, $\beta(u, v) = q(u, v)/w(u, v)$ and $\gamma(u, v) = r(u, v)/w(u, v)$ for polynomials $p(u, v), q(u, v), r(u, v)$ and $w(u, v)$ of maximum degree d. At this time the expansion of each term $c_{ijk} \cdot x^{d_i} \cdot y^{d_j} \cdot z^{d_k} = c_{ijk} \cdot$

$w(u,v)^{d-d_i-d_j-d_k} \cdot (w(u,v) \cdot \bar{x} - p(u,v))^{d_i} \cdot (w(u,v) \cdot \bar{y} - q(u,v))^{d_j} \cdot (w(u,v) \cdot \bar{z} - r(u,v))^{d_k} / w(u,v)^d$ is harder than the case of Theorem 3.22. The polynomials f, f_x, f_y and f_z have $O(d^3)$ terms of this form. Each $(w(u,v) \cdot \bar{x} - p(u,v))^{d_i} \cdot (w(u,v) \cdot \bar{y} - q(u,v))^{d_j} \cdot (w(u,v) \cdot \bar{z} - r(u,v))^{d_k}$ can be expanded into $(w(u,v)^{d_i+d_j+d_k} \cdot \bar{x}^{d_i} \cdot \bar{y}^{d_j} \cdot \bar{z}^{d_k} + \ldots + (-1)^{d_i+d_j+d_k} \cdot p(u,v)^{d_i} \cdot q(u,v)^{d_j} \cdot r(u,v)^{d_k})$ in $O(d^3)$ time using binomial expansion. Similarly, f and the component polynomials of $\nabla f \times (G_u \times G_v)$ can be expanded into $O(d^6)$ terms of the form $(*)$ $c \cdot {p^{d_1}} \cdot {q^{d_2}} \cdot {r^{d_3}} \cdot {w^{d_4}} \cdot {p_u}^{d_5} \cdot {p_v}^{d_6} \cdot {q_u}^{d_7} \cdot {q_v}^{d_8} \cdot {r_u}^{d_9} \cdot {r_v}^{d_{10}} \cdot {w_u}^{d_{11}} \cdot {w_v}^{d_{12}} \cdot \bar{x}^{d_{13}} \cdot \bar{y}^{d_{14}} \cdot \bar{z}^{d_{15}} / w^{d+4}$ ($d_1, d_2, d_3, d_4 \leq d+2$, $d_5, \ldots, d_{12} \leq 2$, and $d_{13}, d_{14}, d_{15} \leq d$) within $O(d^6)$ time. There are $O(d^4)$ terms of the form $(**) p^{d_1} \cdot q^{d_2} \cdot r^{d_3} \cdot w^{d_4} \cdot {p_u}^{d_5} \cdot {p_v}^{d_6} \cdot {q_u}^{d_7} \cdot {q_v}^{d_8} \cdot {r_u}^{d_9} \cdot {r_v}^{d_{10}} \cdot {w_u}^{d_{11}} \cdot {w_v}^{d_{12}}$, and since the polynomials p, q, r, w have $O(d^2)$ terms, each term of the form $(**)$ can be expanded into $O(d^4)$ terms of the form $(***)c' \cdot u^{d_{16}} \cdot v^{d_{17}}$ ($d_{16}, d_{17} \leq 5d^2$) within $O(d^6)$ time by using by using the dynamic programming technique. Thus we can expand the overall $O(d^4)$ terms of $(**)$ into $O(d^4)$ terms of $(***)$ within $O(d^{10})$ time. The final conversion of terms of $(*)$ to the terms of $(****)c' \cdot \bar{x}^{d_{13}} \cdot \bar{y}^{d_{14}} \cdot \bar{z}^{d_{15}} \cdot u^{d_{16}} \cdot v^{d_{17}} / w^{d+4}$ ($d_{13}, d_{14}, d_{15} \leq d$ and $d_{16}, d_{17} \leq 5d^2$) can be done by multiplying $O(d^4)$ terms of $(***)$ with $O(d^3)$ terms of $\bar{x}^{d_{13}} \cdot \bar{y}^{d_{14}} \cdot \bar{z}^{d_{15}}$ with $O(d^7)$ time. Hence, the overall time complexity for expansion and simplification is $O(d^{10})$. After the expansion, the power of the variables u and v is $O(d^2)$ and the power of the coefficients $\bar{x}, \bar{y}, \bar{z}$ is $O(d)$. The multivariate resultant for a system of 3 homegenous polynomials of degree $O(d^2)$ can be constructed in $O(d^8)$ time and the degree of the convolution faces can be at most $O(d^6)$.

The case of F_B being a parametric surface and F_{-A} being an algebraic surface is similar to Corollary 3.23.

Corollary 3.24. *Let $F_B \subset Bdr(B)$ be a parametric surface patch $F(s,t) = (x(s,t), y(s,t), z(s,t))$ with gradients $F_s \times F_t$. Further let $F_{-A} \subset Bdr(-A)$ be a parametric surface patch $G(u,v) = (\alpha(u,v), \beta(u,v), \gamma(u,v))$ with gradients $G_u \times G_v$, and suppose that F_B and F_{-A} are* sub-compatible. *Then Convolution(F_{-A}, F_B) = Convolution$(G(F_B, F_{-A}), K(F_B, F_{-A}))$ is the set of points $\bar{p} = (\bar{x}, \bar{y}, \bar{z}) = p + q = (x(s,t) + \alpha(u,v), y(s,t) + \beta(u,v), z(s,t) + \gamma(u,v))$ such that*

$$\begin{cases} p = (x(s,t), y(s,t), z(s,t)) \in K(F_B, F_{-A}) & (1) \\ q = (\alpha(u,v), \beta(u,v), \gamma(u,v)) \in G(F_B, F_{-A}) & (2) \\ (F_s \times F_t) \times (G_u \times G_v) = 0 & (3) \\ (F_s \times F_t) \cdot (G_u \times G_v) > 0 & (4) \end{cases}$$

One can obtain the implicit algebraic equation of the *Convolution*(F_{-A}, F_B) by eliminating s, t, u and v from the equations $\bar{x} = x(s,t) + \alpha(u,v)$, $\bar{y} = y(s,t) + \beta(u,v)$, $\bar{z} = z(s,t) + \gamma(u,v)$ and the above equation (3). Since (3) gives two independent equations, we have 5 equations and need to eliminate 4 variables s, t, u, v to get an implicit equation.

Each component polynomial of $(F_s \times F_t) \times (G_u \times G_v)$ has 8 terms of the form $x_s \cdot y_t \cdot \beta_u \cdot \gamma_v$ or $x_s \cdot y_t \cdot \gamma_u \cdot \beta_v$, etc., and can be expanded into $O(d^4)$ terms of the form $c \cdot s^{d_i} \cdot t^{d_j} \cdot u^{d_k} \cdot v^{d_l}$ (where $d_i, d_j, d_k, d_k \leq 2d$) in $O(d^4)$ time. The multivariate resultant can be constructed in $O(d^8)$ time and the degree of the convolution faces can be at most $O(d^5)$.

Boundary Edges of *Convolution*(F_{-A}, F_B) For *sub-compatible* face pairs F_B and F_{-A} which are relatively open with respect to $Bdr(B)$ and $Bdr(-A)$, each boundary edge E_N of $N(F_B, F_{-A})$ ($= N(B, F_B) \cap N(-A, F_{-A})$) is either a segment of a boundary edge of $N(B, F_B)$ or a segment of a boundary edge of $N(-A, F_{-A})$. Further E_N is either (a) a segment of the common boundary edge of $N(B, F_B)$ and $N(B, E_B)$ for some edge E_B of F_B, or (b) a segment of the common boundary edge of $N(-A, F_{-A})$ and $N(-A, E_{-A})$ for some edge E_{-A} of F_{-A}. Similarly, each boundary edge $E_{CO(A,B)}$ of the surface patch *Convolution*(F_{-A}, F_B) is either (a) a segment of the common boundary edge of *Convolution*(F_{-A}, F_B) and *Convolution*$(Cl(F_{-A}), E_B)$, or (b) a segment of the common boundary edge of *Convolution*(F_{-A}, F_B) and *Convolution*$(E_{-A}, Cl(F_B))$, where $Cl(F_B)$ and $Cl(F_{-A})$ are the closures of F_B and F_{-A} with respect to $Bdr(B)$ and $Bdr(-A)$. This degeneracy can be eliminated

by perturbing A and B slightly. Edges of type (a) are described in Theorem 3.25, edges of type (b) can be described similarly. Let $\textit{sub-Convolution}_{T_N}(G_{-A}, K_B)$ = "sub-Convolution of G_{-A} and K_B restricted to the normal directions T_N" $= \{\bar{p} \in R^3 \mid \bar{p} = p + q$ where $p \in K_B$ and $q \in G_{-A}$, and B has a unit outward normal direction n_p at p which is the same as a unit outward normal A has at q where $n_p \in T_N\}$. Since the Gaussian Image of $E_{CO(A,B)}$ is some edge E_N of $N(F_B, F_{-A})$, one can easily show $E_{CO(A,B)} = \textit{sub-Convolution}_{E_N}(Cl(F_{-A}), Cl(F_B))$.

Theorem 3.25. *Let F_B and F_{-A} be a* sub-compatible *face pair, E_B be an edge of F_B and E_N be a boundary edge of $N(F_B, F_{-A})$ such that E_N is a segment of the common edge $N(B, E_B) \cap Cl(N(B, F_B))$. Suppose E_B is the common edge of two surface patches F_B and $\hat{F}_B$, where F_B is a patch of an algebraic surface $f = 0$ with gradients ∇f, and $\hat{F}_B$ is a patch of an algebraic surface $\hat{f} = 0$ with gradients $\nabla \hat{f}$. Then*
(A) the convolution edge $E_{CO(A,B)} = \textit{sub-Convolution}_{E_N}(Cl(F_{-A}), Cl(F_B))$ due to the normal directions E_N is the set of points $\bar{p} = (\bar{x}, \bar{y}, \bar{z}) = p + q = (x + \alpha, y + \beta, z + \gamma)$ such that

$$\begin{cases} f(x,y,z) = 0 \textit{ and } p = (x,y,z) \in Cl(K(F_B, F_{-A})) & (1) \\ \hat{f}(x,y,z) = 0 \textit{ and } p = (x,y,z) \in Cl(\hat{F}_B) & (2) \\ g(\alpha,\beta,\gamma) = 0 \textit{ and } q = (\alpha,\beta,\gamma) \in Cl(G(F_B, F_{-A})) & (3) \\ \nabla f \times \nabla g = 0 & (4) \\ \nabla f \cdot \nabla g > 0 & (5) \end{cases}$$

(B) The surface patch defined by (1) and (3)–(5) and the surface patch defined by (2)–(5) intersect along the convolution edge $E_{CO(A,B)}$.

Proof (A) The surface patch defined by (1) and (3)–(5) is the face $\textit{Convolution}(F_{-A}, F_B)$ and all its boundary edges and vertices. Since (1)–(2) restrict the set of points p to the subsegment E'_B of E_B such that $E'_B = N^{-1}(B, E_N)$, (1)–(5) define the convolution edge $E_{CO(A,B)}$.
(B) Since $E_{CO(A,B)}$ is the common solution of (1)–(5), $E_{CO(A,B)}$ is the common edge of the surface patch defined by (1) and (3)–(5) and the surface patch defined by (2)–(5). □

For each point $p \in \hat{F}_B$, $f(p) = c$ for some level c and the corresponding point $\bar{p} = p + q$ is the translation of p by q defined by (2)–(5).

The case of a boundary edge E_{-A} of F_{-A} being defined by two transversally intersecting surface patches gives a similar result. Further the cases of F_B, $\hat{F}_B$, F_{-A}, or $\hat{F}_{-A}$ being parametric surfaces give similar results. Also the time and degree complexity analyses are similar to those of Theorem 3.22 and Corollaries 3.23–3.24.

Boundary Vertices of $\textit{Convolution}(F_{-A}, F_B)$ For a *sub-compatible* face pair F_B and F_{-A} which are relatively open with respect to $Bdr(B)$ and $Bdr(-A)$, each boundary vertex e_N of $N(F_B, F_{-A})$ $(= N(B, F_B) \cap N(-A, F_{-A}))$ is either (a) a boundary vertex of $N(B, F_B)$, (b) a boundary vertex of $N(-A, F_{-A})$, or (c) the intersection of one edge of $N(B, F_B)$ with another edge of $N(-A, F_{-A})$. In the case of (a), suppose p is the vertex of F_B and q is a point of F_{-A} such that $p \in N^{-1}(B, e_N)$ and $q \in N^{-1}(-A, e_N)$, then the point $p + q$ is the vertex of $\textit{Convolution}(F_{-A}, F_B)$ such that $p + q \in N^{-1}(CO(A, B), e_N)$. $q \in F_{-A}$ can be computed by solving $g = 0$ and $\frac{\nabla g}{\|\nabla g\|} = e_N$. The case of (b) is similar to the case of (a). In the case of (c), the intersection e_N of one edge of $N(B, F_B)$ with another edge of $N(-A, F_{-A})$ can be computed by Theorems 3.31–3.33. Suppose $p \in Bdr(F_B)$ and $q \in Bdr(F_{-A})$ be such that $p \in N^{-1}(B, e_N)$ and $q \in N^{-1}(-A, e_N)$ where $Bdr(F_B)$ and $Bdr(F_{-A})$ are the boundaries of F_B and F_{-A} with respect to $Bdr(B)$ and $Bdr(-A)$, then $p + q$ is the vertex of $\textit{Convolution}(F_{-A}, F_B)$ such that $p + q \in N^{-1}(CO(A, B), e_N)$. $p \in F_B$ can be computed by solving $f = 0$ and $\frac{\nabla f}{\|\nabla f\|} = e_N$ and $q \in F_{-A}$ can be computed by solving $g = 0$ and $\frac{\nabla g}{\|\nabla g\|} = e_N$.

Generating *Convolution*(F_{-A}, E_B) **and** *Convolution*(E_{-A}, F_B) In this section, we consider how to generate the algebraic surface equations, edges and vertices of convolution surface patches *Convolution*(F_{-A}, E_B) and *Convolution*(E_{-A}, F_B). We can use Theorem 3.26 for the case of E_B being defined by the intersection of two implicit algebraic surfaces and F_{-A} being an implicit algebraic surface. The other combinations of implicit and parametric surfaces defining E_B and F_{-A} have similar results as easy Corollaries of Theorem 3.26. Similar results hold for generating *Convolution*(E_{-A}, F_B).

Theorem 3.26. *Let $E_B \subset Bdr(B)$ be the common edge of two faces F_B and $\hat{F}_B$, where F_B and $\hat{F}_B \subset Bdr(B)$ are patches of algebraic surfaces $f = 0$ with gradients ∇f and $\hat{f} = 0$ with gradients $\nabla \hat{f}$. Further let $F_{-A} \subset Bdr(-A)$ be a patch of an algebraic surface $g = 0$ with gradients ∇g. Suppose that E_B and F_{-A} are* sub-compatible. *Then Convolution(F_{-A}, E_B) = Convolution$(G(E_B, F_{-A}), K(E_B, F_{-A}))$ is the set of points $\bar{p} = (\bar{x}, \bar{y}, \bar{z}) = p+q = (x+\alpha, y+\beta, z+\gamma)$ such that*

$$\begin{cases} f(x,y,z) = \hat{f}(x,y,z) = 0 \text{ and } p = (x,y,z) \in K(E_B, F_{-A}) & (1) \\ g(\alpha,\beta,\gamma) = 0 \text{ and } q = (\alpha,\beta,\gamma) \in G(E_B, F_{-A}) & (2) \\ \nabla g \cdot (\nabla f \times \nabla \hat{f}) = 0 \text{ and } \frac{\nabla g}{\|\nabla g\|} \in N_{E_B} & (3) \end{cases}$$

Proof : (3) is equivalent to an outward normal direction of B at p to be the same as one of the outward normal directions of $-A$ at q. □

One can obtain the implicit algebraic equation of the *Convolution*(F_{-A}, E_B) in a similar way as in Theorem 3.22. The polynomial $\nabla g \cdot (\nabla f \times \nabla \hat{f})$ can be expanded into $O(d^6)$ terms of the form $c_{ijklmn} \cdot (\bar{x}-\alpha)^{d_i} \cdot (\bar{y}-\beta)^{d_j} \cdot (\bar{z}-\gamma)^{d_k} \cdot \alpha^{d_l} \cdot \beta^{d_m} \cdot \gamma^{d_n}$ ($d_i, d_j, d_k \leq 2d$ and $d_l, d_m, d_n \leq d$) within $O(d^6)$ time. Each of these terms and the terms of f and $\hat{f}$ will be expanded into $O(d^3)$ terms of $c'_{ijklmn} \cdot \bar{x}^{d'_i} \cdot \bar{y}^{d'_j} \cdot \bar{z}^{d'_k} \cdot \alpha^{d'_l} \cdot \beta^{d'_m} \cdot \gamma^{d'_n}$ ($d'_i, d'_j, d'_k \leq 2d$ and $d'_l, d'_m, d'_n \leq 3d$). Hence, the expansion and simplification takes $O(d^9)$ time, the multivariate resultant construction takes $O(d^6)$ time and the convolution face degree is at most $O(d^4)$, which are the same as Theorem 3.22.

When the face F_{-A} is a parametric surface patch $G(u,v) = (\alpha(u,v), \beta(u,v), \gamma(u,v))$ with gradients $G_u \times G_v$, the corresponding Corollary follows by changing every ∇g into $G_u \times G_v$ and the statement "$g(\alpha,\beta,\gamma) = 0$ *and* $q = (\alpha, \beta, \gamma) \in G(E_B, F_{-A})$" into "$q = (\alpha(u,v), \beta(u,v), \gamma(u,v)) \in G(E_B, F_{-A})$" in the above Theorem. Further, the corresponding time and degree complexities are the same as Corollary 3.23. Similar changes yield corresponding Corollaries for the case of F_B and/or $\hat{F}_B$ being parametric surface patches. The time and degree complexity bounds are the same as Corollary 3.23 if at least one of $F_B, \hat{F}_B, F_{-A}$ is an implicit and at least one is a parametric surface patch. When all three $F_B, \hat{F}_B, F_{-A}$ are parametric, the complexity bounds are the same as Corollary 3.24.

When two faces F_B and $\hat{F}_B$ are tangent to each other along E_B, *Convolution*(F_{-A}, E_B) is a degenerate curve on the *C-space* obstacle boundary. Actually, it is a common edge of two convolution faces generated in §4.1.

Boundary Edges of *Convolution*(F_{-A}, E_B) For a *sub-compatible* edge–face pair E_B and F_{-A} where F_{-A} is relatively open with respect to $Bdr(-A)$ and E_B is relatively open with respect to the intersection curve of two algebraic surfaces $f = 0$ and $\hat{f} = 0$ defining faces F_B and $\hat{F}_B$, each boundary edge E_N of $N(E_B, F_{-A})$ ($= N(B, E_B) \cap N(-A, F_{-A})$) is either a segment of a boundary edge of $N(B, E_B)$ or a segment of a boundary edge of $N(-A, F_{-A})$. Further E_N is either (a) a segment of the common edge of $N(B, E_B)$ and $N(B, F_B)$ for some face F_B adjacent to E_B, (b) a segment of the common edge of $N(-A, F_{-A})$ and $N(-A, E_{-A})$ for some edge E_{-A} of F_{-A}, or (c) a segment of the common edge of $N(B, E_B)$ and $N(B, p_B)$ for a vertex p_B of E_B. Similarly, each boundary edge $E_{CO(A,B)}$ of the surface patch *Convolution*(F_{-A}, E_B) is either (a) a segment of the common edge of *Convolution*(F_{-A}, E_B) and *sub-Convolution*$_{Cl(N(B,F_B))}(Cl(F_{-A}), Cl(F_B))$,

(b) a segment of the common edge of *Convolution*(F_{-A}, E_B) and *Convolution*$(E_{-A}, Cl(E_B))$, (c) a segment of the common edge of *Convolution*(F_{-A}, E_B) and *Convolution*$(Cl(F_{-A}), p_B)$ where $Cl(F_{-A})$ is the closure of F_{-A} with respect to $Bdr(-A)$ and $Cl(E_B)$ is the closure of E_B with respect to the intersection curve of two algebraic surfaces $f = 0$ and $\hat{f} = 0$ defining faces F_B and $\hat{F}_B$. Edges of type (a) have been described in Theorem 3.25, edges of type (b) are described in Theorem 3.27, and edges of type (c) are described in Theorem 3.28. The proofs of Theorems 3.27–3.28 are similar to that of Theorem 3.25.

Theorem 3.27. *Let E_B and F_{-A} be a* sub-compatible *edge–face pair, E_{-A} be an edge of F_{-A} and E_N be an edge of $N(E_B, F_{-A})$ such that E_N is a segment of the common edge $N(-A, E_{-A}) \cap Cl(N(-A, F_{-A}))$. Suppose E_{-A} is the common edge of two surface patches F_{-A} and $\hat{F}_{-A}$, where F_{-A} is a patch of an algebraic surface $g = 0$ with gradients ∇g, and $\hat{F}_{-A}$ is a patch of an algebraic surface $\hat{g} = 0$ with gradients $\nabla \hat{g}$. Then*
(A) the convolution edge $E_{CO(A,B)}$ = sub-Convolution$_{E_N}(Cl(F_{-A}), Cl(E_B))$ due to the normal directions E_N is the set of points $\bar{p} = (\bar{x}, \bar{y}, \bar{z}) = p + q = (x + \alpha, y + \beta, z + \gamma)$ such that

$$\begin{cases} g(\alpha, \beta, \gamma) = 0 \text{ and } q = (\alpha, \beta, \gamma) \in Cl(G(E_B, F_{-A})) & (1) \\ \hat{g}(\alpha, \beta, \gamma) = 0 \text{ and } q = (\alpha, \beta, \gamma) \in Cl(\hat{F}_{-A}) & (2) \\ f(x, y, z) = \hat{f}(x, y, z) = 0 \text{ and } p = (x, y, z) \in Cl(K(E_B, F_{-A})) & (3) \\ \nabla g \cdot (\nabla f \times \nabla \hat{f}) = 0 \text{ and } \frac{\nabla g}{\|\nabla g\|} \in N(E_B, F_{-A}) & (4) \end{cases}$$

(B) The surface patch defined by (1) and (3)–(4) and the surface patch defined by (2)–(4) intersect along the convolution edge $E_{CO(A,B)}$.

When the surface patches of (B) intersect tangentially, one may use different auxiliary surface patch $\hat{F}_{-A}$. One may also select an auxiliary surface patch intersecting transversally to the surface patches of (B) from the ideal of the curve C defining the edge $E_{CO(A,B)}$.

Theorem 3.28. *Let E_B and F_{-A} be a* sub-compatible *edge–face pair, $p_B = (x, y, z)$ be a vertex of E_B and E_N be a boundary edge of $N(E_B, F_{-A})$ such that E_N is a segment of the common edge $Cl(N(B, E_B)) \cap N(B, p_B)$. Suppose E_B is the common edge of two transversally intersecting surface patches F_B and $\hat{F}_B$, where F_B is a patch of an algebraic surface $f = 0$ with gradients ∇f, and $\hat{F}_B$ is a patch of an algebraic surface $\hat{f} = 0$ with gradients $\nabla \hat{f}$. Further let $n = \nabla f(p_B)$ and $\hat{n} = \nabla \hat{f}(p_B)$. Then*
(A) the convolution edge $E_{CO(A,B)}$ = sub-Convolution$_{E_N}(Cl(F_{-A}), Cl(E_B))$ due to the normal directions E_N is the set of all the points $\bar{p} = (\bar{x}, \bar{y}, \bar{z}) = \{p_B\} + q = (x + \alpha, y + \beta, z + \gamma)$ such that

$$\begin{cases} g(\alpha, \beta, \gamma) = 0 \text{ and } q = (\alpha, \beta, \gamma) \in Cl(G(p_B, F_{-A})) & (1) \\ \nabla g \cdot (n \times \hat{n}) = 0 & (2) \\ \nabla g \cdot (n - (n \cdot \hat{n})\ \hat{n}) \geq 0 & (3) \\ \nabla g \cdot (\hat{n} - (n \cdot \hat{n})\ n) \geq 0 & (4) \end{cases}$$

(B) The surface patch defined by (1) and the surface patch defined by (2)–(4) intersect along the convolution edge $E_{CO(A,B)}$

When the surface patches of (B) intersect tangentially, one may select an auxiliary surface patch intersecting transversally to the surface patches of (B) from the ideal of the curve C defining the edge $E_{CO(A,B)}$.

Boundary Vertices of Convolution(F_{-A}, E_B) Each vertex of Convolution(F_{-A}, E_B) is either (a) a vertex of Convolution(F_{-A}, F_B) for some adjacent face F_B of E_B or (b) a vertex of Convolution(E_{-A}, E_B) for some adjacent edge E_{-A} of F_{-A}. The case (a) has been considered in §4.1, and the case (b) will be considered in §4.3.

Generating Convolution(E_{-A}, E_B) In this section, we consider how to generate the algebraic surface equation, edges and vertices of a convolution surface patch CONVOLUTION(E_{-A}, E_B). We can use Theorem 3.29 for the case of both E_{-A} and E_B being defined by two implicit algebraic surfaces. The other combinations of implicit and parametric surfaces defining E_{-A} and E_B have similar results as easy Corollaries of Theorem 3.29.

Theorem 3.29. *Let $E_B \subset Bdr(B)$ be a segment of the common edge of two faces F_B and $\hat{F}_B$, where $F_B \subset Bdr(B)$ is a patch of an algebraic surface $f = 0$ with gradients ∇f and $\hat{F}_B \subset Bdr(B)$ is a patch of an algebraic surface $\hat{f} = 0$ with gradients $\nabla \hat{f}$. Further let $E_{-A} \subset Bdr(-A)$ be a segment of the common edge of two faces F_{-A} and $\hat{F}_{-A}$, where $F_{-A} \subset Bdr(-A)$ is a patch of an algebraic surface $g = 0$ with gradients ∇g and $\hat{F}_{-A} \subset Bdr(-A)$ is a patch of an algebraic surface $\hat{g} = 0$ with gradients $\nabla \hat{g}$. Suppose that E_B and E_{-A} are* sub-compatible. *Then* CONVOLUTION(E_{-A}, E_B) = CONVOLUTION$(G(E_B, E_{-A}), K(E_B, E_{-A}))$ *is the set of points $\bar{p} = (\bar{x}, \bar{y}, \bar{z}) = p + q = (x + \alpha, y + \beta, z + \gamma)$ such that*

$$\begin{cases} f(x,y,z) = \hat{f}(x,y,z) = 0 \text{ and } p = (x,y,z) \in K(E_B, E_{-A}) & (1) \\ g(\alpha,\beta,\gamma) = \hat{g}(\alpha,\beta,\gamma) = 0 \text{ and } q = (\alpha,\beta,\gamma) \in G(E_B, E_{-A}) & (2) \\ \frac{\lambda \cdot \nabla f + (1-\lambda) \cdot \nabla \hat{f}}{\|\lambda \cdot \nabla f + (1-\lambda) \cdot \nabla \hat{f}\|} \in N(E_B, E_{-A}) \text{ and} \\ \frac{\mu \cdot \nabla g + (1-\mu) \cdot \nabla \hat{g}}{\|\mu \cdot \nabla g + (1-\mu) \cdot \nabla \hat{g}\|} \in N(E_B, E_{-A}) \text{for some } 0 \le \lambda, \mu \le 1 & (3) \end{cases}$$

Proof : (3) is equivalent to an outward normal direction of B at p to be the same as an outward normal direction of $-A$ at q. □

One can obtain the implicit algebraic equation of the CONVOLUTION(E_{-A}, E_B) in a similar way as in Theorem 3.22. When the face F_B is a parametric surface patch $F(s,t) = (x(s,t), y(s,t), z(s,t))$ with gradients $F_s \times F_t$, the corresponding Corollary follows by changing every ∇f into $F_s \times F_t$ and the statement "$f(x,y,z) = 0$ *and* $p = (x,y,z) \in K(E_B, E_{-A})$" into "$p = (x(s,t), y(s,t), z(s,t)) \in K(E_B, E_{-A})$" in the above Theorem. Similar changes yield corresponding Corollaries for the case of F_B, $\hat{F}_B$, F_{-A} and/or $\hat{F}_{-A}$ being parametric surface patches. The time and degree complexities are the same as (1) Theorem 3.22 when all four surface patches $F_B, \hat{F}_B, F_{-A}, \hat{F}_{-A}$ are implicit, (2) Corollary 3.23 when at least one is implicit and at least one is parametric, and (3) Corollary 3.24 when all four are parametric.

When F_B and $\hat{F}_B$ are tangent to each other along E_B, or F_{-A} and $\hat{F}_{-A}$ are tangent to each other along E_{-A}, CONVOLUTION(E_{-A}, E_B) is a degenerate curve on the *C-space* obstacle boundary and is a common edge of two convolution faces generated in §4.2. In the special case of F_B and $\hat{F}_B$ being tangent along E_B, and also F_{-A} and $\hat{F}_{-A}$ being tangent along E_{-A}, CONVOLUTION(E_{-A}, E_B) is either a degenerate curve or a degenerate point.

Let $N_{E_T}(p) = N_{E_T} \cap N(T,p)$ for an edge E_T and $p \in E_T$, then $N_{E_T}(p)$ is a geodesic arc on S^2. When two line segments in a plane intersects, either there is a unique intersection point or they overlap entirely on the same line. One can show a similar fact for minimal geodesic arcs on S^2 as follows.

Fact 3.30. *If $N_{E_B}(p) \cap N_{E_{-A}}(q) \neq \emptyset$, either (1) $N_{E_B}(p) \cap N_{E_{-A}}(q)$ is a point or (2) $N_{E_B}(p) \cap N(p,q) = N_{E_{-A}}(q) \cap N(p,q)$ where $N(p,q) = N(B,p) \cap N(-A,q)$.*

By subdividing E_B and E_{-A} if necessary, we may assume only one of the conditions (1) or (2) holds for the whole edges E_B and E_{-A}. We call E_B and E_{-A} to be *parallel* if the condition (2) holds on the whole edges E_B and E_{-A}. If E_B and E_{-A} is a *parallel* edge pair, the CONVOLUTION(E_{-A}, E_B) generated in Theorem 3.28 is a degenerate curve on the *C-space* obstacle. Otherwise it is a surface patch.

Boundary Edges of Convolution(E_{-A}, E_B) For a *sub-compatible* edge pair E_B and E_{-A} where E_B (resp. E_{-A}) is relatively open with respect to the intersection curve of two algebraic surfaces $f = 0$ and $\hat{f} = 0$ defining faces F_B and $\hat{F}_B$ (resp. $g = 0$ and $\hat{g} = 0$ defining faces F_{-A} and $\hat{F}_{-A}$), each edge E_N of $N(E_B, E_{-A})$ is either a segment of an edge of $N(B, E_B)$ or a segment of an edge of $N(-A, E_{-A})$. Further E_N is either (a) a segment of the common edge of $N(B, E_B)$ and $N(B, F_B)$ for some face F_B adjacent to E_B, (b) a segment of the common edge of $N(-A, E_{-A})$ and $N(-A, F_{-A})$ for some face F_{-A} adjacent to E_{-A}, (c) a segment of the common edge of $N(B, E_B)$ and $N(B, p_B)$ for some vertex p_B of E_B, or (d) a segment of the common edge of $N(-A, E_{-A})$ and $N(-A, p_{-A})$ for some vertex p_{-A} of E_{-A}. Similarly, each boundary edge $E_{CO(A,B)}$ of the surface patch CONVOLUTION(E_{-A}, E_B) is either (a) a segment of the common edge of CONVOLUTION(E_{-A}, E_B) and *sub-Convolution*$_{Cl(N(B,F_B))}(Cl(E_{-A}), Cl(F_B))$, (b) a segment of the common edge of CONVOLUTION(E_{-A}, E_B) and *sub-Convolution*$_{Cl(N(-A,F_{-A}))}(Cl(F_{-A}), Cl(E_B))$, (c) a segment of the common edge of CONVOLUTION(E_{-A}, E_B) and CONVOLUTION($Cl(E_{-A}), p_B$), or (d) a segment of the common edge of CONVOLUTION(E_{-A}, E_B) and CONVOLUTION($p_{-A}, Cl(E_B)$). Edges of type (a)–(b) have been described in Theorem 3.27. In the case of (c), CONVOLUTION($Cl(E_{-A}), p_B$) is a degenerate curve segment which is non–smooth on *Bdr(CO(A,B))* and also equals to the edge $E_{CO(A,B)}$. Hence, $E_{CO(A,B)}$ is the common edge of the face CONVOLUTION(E_{-A}, E_B) with the face CONVOLUTION($E_{-A}, \hat{E}_B$) for some edge $\hat{E}_B$ adjacent to p_B (or with the face CONVOLUTION(F_{-A}, p_B) for some face F_{-A} adjacent to E_{-A}). Since *Bdr(CO(A,B))* is non–smooth on $E_{CO(A,B)}$, $E_{CO(A,B)}$ can be represented as a common edge of two transversally intersecting convolution surface patches. The case of (d) is similar to the case of (c).

Boundary Vertices of Convolution(E_{-A}, E_B) For *sub-compatible* edge pairs E_B and E_{-A}, each vertex e_N of $N(E_B, E_{-A})$ $(= N(B, E_B) \cap N(-A,\ E_{-A}))$ is either (a) a vertex of $N(B, E_B)$, (b) a vertex of $N(-A, E_{-A})$, or (c) the intersection of one edge of $N(B, E_B)$ with another edge of $N(-A, E_{-A})$. In the case of (a), suppose p is a vertex of E_B and q is a point of E_{-A} such that $p \in N(B, e_N)$ and $q \in N(-A, e_N)$, then the point $p + q$ is the vertex of CONVOLUTION(E_{-A}, E_B) such that $p + q \in N^{-1}(CO(A, B), e_N)$. Further suppose that E_{-A} is the common edge of two faces F_{-A} and $\hat{F}_{-A}$ defined by $g = 0$ and $\hat{g} = 0$ respectively, then the point $q = (\alpha, \beta, \gamma) \in E_{-A}$ can be computed by solving $g = \hat{g} = 0$ and $(\nabla g \times \nabla \hat{g}) \cdot e_N = 0$. The case of (b) is similar to the case of (a). In the case (c), this intersection is also the intersection of one edge of $N(B, F_B)$ with another edge of $N(-A, F_{-A})$ where F_B is a face adjacent to E_B and F_{-A} is a face adjacent to F_{-A}. This case has been considered in §4.1.

3.4.3 Constructing the Gaussian Model of C-space Obstacles

We now show how to construct the Gaussian Model of $CO(A, B)$. Let S^2_B and S^2_{-A} be the Gaussian Models of B and $-A$ respectively. These define face adjacency graphs on S^2 with degeneracies tagged appropriately. Let a new graph $S^2_{CO(A,B)}$ on S^2 be defined as the *overlay* of S^2_B and S^2_{-A}. Then $S^2_{CO(A,B)}$ is the Gaussian Model of $CO(A, B)$ and determines all *sub-compatible* face, edge and vertex pairs between $Bdr(B)$ and $Bdr(-A)$. Further the topology of the faces, edges and vertices of $Bdr(CO(A, B))$ is given by the topology of the faces, edges and vertices of $S^2_{CO(A,B)}$. Construction of $S^2_{CO(A,B)}$ requires computing the intersections of edges of S^2_B with edges of S^2_{-A}. These intersections can be computed by using Theorems 3.31–3.33 below. Edges of S^2_B and S^2_{-A} are either minimal geodesic arcs on the unit sphere or curve segments of the form $\frac{\nabla f(p)}{\|\nabla f(p)\|}$ for $p \in E$ where $f = 0$ is a face equation and E is an edge of this face. Note this curve segment is well defined since we are assuming the nonsingularity of each face on its boundary. By the regularity and convexity of the object we may assume that the end points of each minimal geodesic arc are not antipodal points of each other. Hence, for two end points n_1 and n_2 of a minimal geodesic arc one has $\lambda \cdot n_1 + (1 - \lambda) \cdot n_2 \neq 0$ and $\frac{(\lambda \cdot n_1 + (1-\lambda) \cdot n_2)}{\|\lambda \cdot n_1 + (1-\lambda) \cdot n_2\|}$ is well defined. The intersection of two minimal

geodesic arcs can be computed by Theorem 3.31. The intersection of one general curve segment and one minimal geodesic arc can be computed by Theorem 3.32. The intersection of two general curve segments can be computed by Theorem 3.33.

Theorem 3.31. *Let γ be a minimal geodesic arc connecting n_1 and n_2 on S^2_B and γ' be a minimal geodesic arc connecting n'_1 and n'_2 on S^2_{-A}. Then γ and γ' intersect at $\frac{\lambda\cdot n_1+(1-\lambda)\cdot n_2}{\|\lambda\cdot n_1+(1-\lambda)\cdot n_2\|}$ if and only if*

$$\begin{cases} (\lambda\cdot n_1+(1-\lambda)\cdot n_2)\times(\mu\cdot n'_1+(1-\mu)\cdot n'_2)=0 & (1)\\ (\lambda\cdot n_1+(1-\lambda)\cdot n_2)\cdot(\mu\cdot n'_1+(1-\mu)\cdot n'_2)>0 & (2)\end{cases}$$

for some $0\le\lambda,\mu\le 1$.

Proof : (1)–(2) are equivalent to that $\lambda\cdot n_1+(1-\lambda)\cdot n_2$ is in the same direction as $\mu\cdot n'_1+(1-\mu)\cdot n'_2$ for some $0\le\lambda,\mu\le 1$. □

Since the vector equation (1) gives two independent polynomial equations in two variables λ,μ, one can solve this system of equations via the u-resultant, see [38].

Theorem 3.32. *Let γ be a curve segment on S^2_B given by the set of points $\frac{\nabla f(p)}{\|\nabla f(p)\|}$ for $p\in E_B$, where $E_B\subset Bdr(B)$ is the common edge of two faces F_B and $\hat{F}_B$, F_B is a patch of an algebraic surface $f=0$ with gradients ∇f and $\hat{F}_B$ is a patch of an algebraic surface $\hat{f}=0$ with gradients $\nabla\hat{f}$. And, let γ' be a minimal geodesic arc connecting n_1 and n_2 on S^2_{-A}. Then γ and γ' intersect at $\frac{\nabla f(p)}{\|\nabla f(p)\|}$ if and only if*

$$\begin{cases} f(x,y,z)=\hat{f}(x,y,z)=0 \text{ and } p=(x,y,z)\in E_B & (1)\\ \nabla f\cdot(n_1\times n_2)=0 & (2)\\ \nabla f\cdot(n_1-(n_1\cdot n_2)n_2)\ge 0 & (3)\\ \nabla f\cdot(n_2-(n_1\cdot n_2)n_1)\ge 0 & (4)\end{cases}$$

Proof: (2)–(4) are equivalent to that ∇f is in the same direction as $\lambda\cdot n_1+(1-\lambda)\cdot n_2$ for some $0\le\lambda\le 1$. (1) restricts the solution for p to the edge E_B. □

Since (1)–(2) give three polynomial equations in three variables x,y,z, one can solve this system of equations again via the u-resultant, [38]. The case of γ being a minimal geodesic arc on S^2_B and γ' being a general curve segment on S^2_{-A} is similar to Theorem 3.32.

Theorem 3.33. *Let γ be a curve segment on S^2_B given by the set of points $\frac{\nabla f(p)}{\|\nabla f(p)\|}$ for $p\in E_B$, where $E_B\subset Bdr(B)$ is the common edge of two faces F_B and $\hat{F}_B$, F_B is a patch of an algebraic surface $f=0$ with gradients ∇f and $\hat{F}_B$ is a patch of an algebraic surface $\hat{f}=0$ with gradients $\nabla\hat{f}$. And, let γ' be a curve segment on S^2_{-A} given by the set of points $\frac{\nabla g(q)}{\|\nabla g(q)\|}$ for $q\in E_{-A}$, where $E_{-A}\subset Bdr(-A)$ is the common edge of two faces G_{-A} and $\hat{G}_{-A}$, G_{-A} is a patch of an algebraic surface $g=0$ with gradients ∇g and $\hat{G}_{-A}$ is a patch of an algebraic surface $\hat{g}=0$ with gradients $\nabla\hat{g}$. Then γ and γ' intersect at $\frac{\nabla f(p)}{\|\nabla f(p)\|}$ if and only if*

$$\begin{cases} f(x,y,z)=\hat{f}(x,y,z)=0 \text{ and } p=(x,y,z)\in E_B & (1)\\ g(\alpha,\beta,\gamma)=\hat{g}(\alpha,\beta,\gamma)=0 \text{ and } q=(\alpha,\beta,\gamma)\in E_{-A} & (2)\\ \nabla f\times\nabla g=0 & (3)\\ \nabla f\cdot\nabla g>0 & (4)\end{cases}$$

Proof : (3)–(4) are equivalent to that ∇f is in the same direction as ∇g. (1) restricts the solution for p to the edge E_B and (2) restricts the solution for q to the edge E_{-A}. □

Since the vector equation (3) gives two independent polynomial equations, one has six polynomial equations in six variables from (1)–(3).

Each face of the overlay graph $S^2_{CO(A,B)}$ corresponds to a compatible pair $((K_B, N_{K_B}), (G_{-A}, N_{G_{-A}}))$ of faces, edges and vertices of $Bdr(B)$ and $Bdr(-A)$. Note that we consider the degenerate curves and degenerate points as generic faces of $S^2_{CO(A,B)}$. Using the formula defining K_B and G_{-A} one can compute the equation for $\textsc{Convolution}(G_{-A}, K_B)$. The edges and vertices of each face $\textsc{Convolution}(G_{-A}, K_B)$ can be computed by using the boundary informations of K_B and G_{-A}.

3.4.4 Compliant Motion in C-space

Having presented ways of constructing the curved surface boundary of the *C-space* obstacles our next step is to generate piecewise continuous curves on the *C-space* obstacle boundary connecting various specified start and end points. These correspond to compliant paths for the original object and obstacles.

In the past several authors have posed solutions to finding boundary restricted paths between points on *C-space* obstacles bounded by planar faces.

Finding shortest paths on curved *C-space* obstacles with algebraic boundary surfaces as generated in the earlier section, is correspondingly more difficult. On single surfaces such paths are known as *geodesics*, and arise in cutter tool paths in machining, efficient terrain navigation and winding rotor coils, etc. There are closed solutions of geodesics for various quadrics and surfaces of revolution. For given two points p and q on a closed surface there exists a minimal geodesic joining p and q, [48]. However computing exact analytical solutions for geodesics in general is quite difficult since these are given as solutions of nonlinear ordinary differential equations. Even for certain simple algebraic surfaces the geodesic curves are non-unique and non-algebraic in nature, e.g., a non-algebraic helical curve is geodesic on a circular cylinder (degree two algebraic surface).

Arbitrary Compliant Paths Before considering shortest paths on curved convex *C-space* obstacle boundaries, we first consider the generation of certain algebraic curves which lie on the boundary and connect specified start points (s_i) and end points (e_j) on the boundary. Such curves provide compliant paths for a certain object and obstacle pair. Perhaps the simplest method is to choose a point p in the interior of the convex *C-space* obstacle and consider the unique plane Π containing a specified pair s_i and e_j, and the interior point p. The intersection of Π with the convex *C-space* obstacle boundary yields an algebraic curve on the boundary with breakpoints on vertices and edges. The correctness of this procedure is justified by the following Fact which stems from the theorem that the convexity is closed under intersection.

Fact 3.34. *A plane passing through an interior point of a convex object intersects the convex object boundary in a planar convex curve which is a Jordan curve (i.e. a curve homeomorphic to a circle).*

The disadvantage of this simplistic approach is that the arbitrary choice of the interior point p may yield undesirable algebraic curves which pass through edges and vertices of the *C-space* obstacle boundary. Such paths then correspond to vertices and edges of the object riding on vertices and edges of the obstacle during compliant motion.

To circumvent this undesirable prospect an alternate method for choosing compliant paths may be adopted as follows. Consider the *Gaussian* Model of the *C-space* obstacle as constructed in §5. The start and end points of the *C-space* obstacle surface are easily mapped to this spherical model. Next piecewise geodesic curves on the sphere can be constructed connecting these start and end points which avoid degeneracies on the *C-space* obstacle boundary. Various interior points to surface patches on the sphere can be chosen and for each triple of points (two surface points and the center of sphere) the corresponding plane intersection with the sphere provide geodesic curve segments on the sphere. Discrete points $p_1, p_2, \ldots, p_k$ on these piecewise curves are generated and

then mapped to the points $q_1, q_2, \ldots, q_k$ on the *C-space* obstacle boundary. A plane containing two adjacent points q_i and q_{i+1} $(i = 1, \ldots, k-1)$ and a predefined interior point q_o determines a curve segment connecting q_i and q_{i+1} on the C-space obstacle boundary. By taking more dense points if necessary, we can make this boundary curve segment contained in a convolution surface patch. In this way compliant paths which are piecewise algebraic can be determined which traverse a specific desired sequence of surface patches or desired sequence of surface and edge contacts between object and obstacle.

Geodesic Approximation We now describe a method of obtaining piecewise algebraic approximate geodesic paths on convex *C-space* obstacles with algebraic surfaces. One first computes a "good" approximate convex polyhedron of the curved convex *C-space* obstacle as we now describe. Next the start and end points on the *C-space* obstacle boundary are mapped onto the convex polyhedron and the shortest path between these points is computed on this polyhedron. Finally the approximate shortest path of the *C-space* obstacle boundary $Bdr(CO(A, B))$ can be obtained by projecting the shortest path of the polyhedron onto $Bdr(CO(A, B))$.

Hierarchical Convex Polyhedral Approximation Suppose S is strictly convex, i.e. S is convex and for each point $p \in Bdr(S)$ the supporting plane L_p has only one common point (i.e. p) with S. Strictly convex objects exclude surfaces like ruled surface patches, planar patches, etc. We consider the strictly convex case first and then consider the general convex case by adding special features to the strictly convex case. The following Theorems hold for the boundaries of strictly convex objects.

Theorem 3.35. *Let $F \subset Bdr(S)$ be a patch of an algebraic surface $f = 0$ with gradients ∇f. Suppose $\{e\} = N(S, p)$ for some $p \in F$, then $p = (x, y, z)$ is the solution of the following equations.*

$$\begin{cases} f(x,y,z) = 0 \text{ and } p = (x,y,z) \in F & (1) \\ \nabla f \times e = 0 & (2) \\ \nabla f \cdot e > 0 & (3) \end{cases}$$

Proof : (2)–(3) are equivalent to that ∇f is in the same direction as e. □

Theorem 3.36. *Let (E, N_E) be a pair such that $E \subset Bdr(S)$ be a segment of the common edge of two faces F and G, where F and $G \subset Bdr(S)$ are patches of algebraic surfaces $f = 0$ with gradients ∇f and $g = 0$ with gradients ∇g, and $N_E \subset N(S, E)$ with $N_E \cap N(S, p) \neq \emptyset$ for all $p \in E$. Suppose $e \in N_E \cap N(S, p)$ for some $p \in E$, then $p = (x, y, z)$ is the solution of the following equations.*

$$\begin{cases} f(x,y,z) = g(x,y,z) = 0 \text{ and } p = (x,y,z) \in E & (1) \\ e \cdot (\nabla f \times \nabla g) = 0 \text{ and } e \in N_E & (2) \\ e \cdot ((\nabla g \cdot \nabla g)\nabla f - (\nabla f \cdot \nabla g)\nabla g) > 0 & (3) \\ e \cdot ((\nabla f \cdot \nabla f)\nabla g - (\nabla f \cdot \nabla g)\nabla f) > 0 & (4) \end{cases}$$

Proof : (2)–(4) are equivalent to that $e \in N_E \cap N(S, p)$. □

When F is a parametric surface patch $F(s, t) = (x(s, t), y(s, t), z(s, t))$ with gradients $F_s \times F_t$, one can obtain the corresponding Corollaries by changing ∇f into $F_s \times F_t$ and the statement "$f(x, y, z) = 0$ *and* $p = (x, y, z) \in F$" into "$p = (x(s, t), y(s, t), z(s, t)) \in F$" in the above Theorems. Similarly in the case of G being a parametric surface patch.

Using the above Theorems we can approximate a strictly convex object S by convex polyhedra as follows. Our approximation scheme is hierarchical and curvature dependent. At the coarsest level we inscribe a regular polyhedron (say, icosahedron) inside a unit sphere and project it onto

the surface of the sphere. This projection defines a regular subdivision or tesselation on S^2. We can further triangulate each polyhedral face if it is not triangular. Let $S^2_\triangle$ be this triangular subdivision of S^2, and S^2_T be the subdivision given by the Gaussian Model of T. Further let $S^2_{\triangle,T}$ be the overlay of $S^2_\triangle$ and S^2_T. The overlay $S^2_{\triangle,T}$ can be computed by using Theorems 3.31–3.32. To compute an approximating polyhedron of T corresponding to the triangulation $S^2_\triangle$ one computes the boundary points of T at which $Bdr(T)$ have gradients corresponding to vertices of $S^2_\triangle$. For each vertex e of $S^2_\triangle$, by looking at the vertex e in $S^2_{\triangle,T}$ one can tell which face, edge or vertex of $Bdr(T)$ has e as one of its outward normal direction(s). Suppose a face F with a surface equation $f = 0$ with gradients ∇f has e as one of its outward normal directions at p, then the point p can be computed by using Theorem 3.35. When an edge E whose adjacent faces are given by $f = 0$ and $g = 0$ with gradients ∇f and ∇g, has e as one of its outward normal directions at p, then the point p can be computed by using Theorem 3.36. If e is in the Gaussian Image of a vertex p, the point p is directly obtained by the coordinates of p. When any of the faces are parametric surfaces, one can use a modified version of Theorem 3.35 or 3.36 as discussed before.

Further we can approximate T by two related polyhedra V and W (an inner and an outer) determined by the points p corresponding to the vertices e of $S^2_\triangle$. Let P be the set of all these points p. Since T is convex, $V = \textit{Convex-Hull}(P) \subset T$. Let L_p be the supporting plane of T at p and $H_p =$ the half-space defined by L_p such that $T \subset H_p$, then $T \subset W = \cap_{p \in P} H_p$. For a strictly convex object T each face of V is a triangle F_{ijk} determined by three boundary points p_i, p_j and p_k such that the corresponding Gaussian Images e_i, e_j and e_k makes a triangular face N_{ijk} on the tessellation of the Gaussian Sphere. F_{ijk} is a planar approximation of the boundary surface patch T_{ijk} on which T has normal directions corresponding to angular range N_{ijk}. Each vertex of W is a common intersection point H_{ijk} of three supporting planes L_{p_i}, L_{p_j} and L_{p_k}. The distance between the vertex H_{ijk} and the face F_{ijk} gives an estimation of how closely V and W approximates T over T_{ijk}. If the difference is bigger than a suitable normalized bound $\epsilon > 0$, we can further triangulate N_{ijk} into a finer resolution and modify V and W locally by repeating the above procedure. We can continue this refinement over all coarsely approximated faces and recursively to all the sub faces thus obtained.

When a convex object T is not strictly convex, $Bdr(T)$ can have some ruled surface pathes and planar patches which have degenerate curves and points as Gaussian Images. The degenerate point for the Gaussian Image of a planar patch can be on the interior of a single triangular patch, on the common edge of two triangular patches, or at a common vertex of several triangular patches. Polygonal approximation can be obtained by connecting the vertices of the planar patch and the vertices of the triangular patches where the degenerate point lies. The degenerate curve for the Gaussian Image of a ruled surface patch passes through a sequence of triangular patches of the Gaussian Sphere. By subdividing this curve into finite subsegments we can approximate the ruled surface into a finite sequence of planar patches. Each corner of this planar patches can be connected to the vertices of the triangular patches where the segments of the degenerate curve lie.

Approximating Shortest Path By continuing the above finer refinements upto an arbitrarily small $\epsilon > 0$, we can construct a sequence $\{V_n\}$ of inscribed convex polyhedra converging to T. It can be shown that the areas of $Bdr(V_n)$ converges to those of $Bdr(T)$. Hence, this fact justifies our strategy of using the shortest paths of $Bdr(V_n)$ as approximate shortest paths of $Bdr(T)$ for sufficiently large n.

Convex Polyhedral Approximation of C-space Obstacle One may compute a convex polyhedral approximation of $CO(A, B)$ by applying the above procedure to the boundary representation and the Gaussian Model of $CO(A, B)$. But, since the degrees of *C-space* obstacle boundary surfaces are extremely high, it is more efficient to deal with the boundary surfaces of $-A$ and B to approximate $CO(A, B)$. One can start with a triangular subdivision $S^2_\triangle$ as before. But, at this

time, instead of computing the point $p \in Bdr(CO(A,B))$ which has an outward normal direction e on the *C-space* obstacle boundary, one can compute $p_B \in Bdr(B)$ and $p_{-A} \in Bdr(-A)$ which correspond to e on the object and obstacle boundary surface, and compute $p_B + p_{-A}$. One can continue the refinement steps as above.

In this way we do not need to even construct the entire Gaussian Model of $CO(A,B)$ either. Computation of p_B and p_{-A} requires computing the overlays of $S^2_{\triangle}$ with S^2_B and S^2_{-A}. Only a partial boundary of $S^2_{CO(A,B)}$ needs to be constructed to obtain the topology of the corresponding partial boundary of $Bdr(CO(A,B))$ where the approximate shortest path lies.

Shortest Path on Convex Polyhedron and Projection onto C-space Obstacle Once we have a convex polyhedron V approximating $CO(A,B)$ we can compute a shortest path γ on $Bdr(V)$ as mentioned earlier by using the algorithm of [76]. By projecting γ from an interior point onto $Bdr(CO(A,B))$ one can get an approximate shortest path on the *C-space* obstacle boundary. One can choose the center of mass c of V as the projection point and consider the intersections of the plane (containing each line segment of the shortest path on V and c) with the corresponding surface patch of $Bdr(CO(A,B))$. This surface patch corresponds to the triangular facet of V which contains the line segment which we are projecting. For certain applications with complicated *C-space* obstacles, a single projection point may not give nice properties for the projected curve. We may have this difficulty for instance when the *C-space* obstacle has many degenerate boundary surfaces. In this case, we may choose a multiple projection points or use other projection methods as appropriate for the application.

Conclusion We have described algebraic algorithms for computing *C-space* obstacles using boundary representations and Gaussian Image geometric models. The numerical information defining the faces, edges and vertices of the *C-space* obstacle boundary were obtained by solving systems of multivariate polynomial equations. The symbolic solution by means of multivariate resultants, though computationally extensive, yields the implicit algebraic equations of the curves and surfaces on the *C-space* obstacle boundary. The topological information defining the adjacency relationships of faces, edges and vertices of the *C-space* obstacle boundary were obtained by constructing and overlaying (merging) the Gaussian Image models of the individual moving objects and obstacles.

In comparison with the algorithms for obtaining the *C-space* obstacle boundary for planar case, [22], one notes a large increase in the complexity in obtaining numerical and topological information for *C-space* obstacle generation in space. A significant problem that also arises in the *C-space* generation for curved objects is the analysis of singularities. While all types of point singularities that arise in planar curves can be completely analyzed by the affine quadratic transformations of [1], the singularities in algebraic surfaces are considerably harder to deal with. The complete analysis of singularities in plane curves allows one to also deal with the topological constructions of *C-space* obstacles for non–convex planar curved moving objects and obstacles, see [22]. A similar analysis of the possible point and curve singularities that arise in *C-space* obstacle surfaces may be achieved by a sequence of monoidal and quadratic transformations as recently given by [2]. This may be the starting point for future research, leading to the construction of *C-space* obstacles for non–convex curved solid moving objects and obstacles – one of the currently immediate open problems.

4 Piecewise Representations of Surfaces

4.1 Modeling Surfaces with Patches

While it is possible to model a general closed surface of arbitrary genus as a single implicit surface patch, the geometry of such a global surface is difficult to specify, interactively control, and polygonize. The main difficulties stem from the fact that implicit representations are iso-contours

which generally have multiple real sheets, self–intersections and several other undesirable singularities. We will present details of several implicit algebraic (polynomial) surface splines (one is termed the A-patch) which overcome the above difficulties, and show how these are used in C^1 and C^2 interpolation/approximation and interactive free-form modeling schemes. The potential of implicit polynomial (and other analytic function) patch splines remains largely latent and virtually all commercial and many research modeling systems are based on parametric polynomial spline representations. An exception are the toolkits of the project SHASTRA, which allows modeling with both implicit and parametric polynomial splines [12].

The important issues in free-form polynomial patch modeling of shapes with arbitrary topology are:

1. the patch representation (implicit or parametric, and over power, Bézier, B-spline, or other piecewise polynomial bases)
2. the polynomial degree of the patches
3. the number of patches per face of some input or benchmarking polyhedron
4. functional connectivity and nonsingularity of the patches
5. conditions for the desired continuity between adjacent patches (patches "stitched" together to form a smooth surface)
6. curvature and higher derivative variation of the patches, especially around the "stitches"

Chui [44], Dahmen and Michelli [45], Hollig [64] and DeBoor, Hollig and Riemenscheider [46] summarize much of the history of multivariate splines. A significant amount of recent research has focussed on these questions with varying emphasis on non-tensor product patches, multivariate generalizations of B-splines, geometric continuity, approximation order, and the fairness of fit. Common free-form patch modeling schemes include convex combinations of blending functions, local interpolation of a mesh of curves, simplex and box based schemes, and stationary / non-stationary recursive subdivision. In this tutorial we address some of these issues with different implicit algebraic surface patches (A-patches) in a variety of algorithms. There are two broad categories of data fitting algorithms : *interpolatory A-patch* based, and *non-interpolatory A-patch* based.

We also consider patches of algebraic surfaces, i.e. closed and compact subsets of two dimensional zeroes of polynomial equations. A real algebraic surface S in $\mathbb{R}^3$ is implicitly defined by a single polynomial equation $\mathcal{F} : f(x, y, z) = 0$, where coefficients of f are over the real numbers $\mathbb{R}$. While **all** real algebraic surfaces have an implicit definition $\mathcal{F}$ only a small subset of these real surfaces can also be defined parametrically by the triple $\mathcal{G}(s, t) : (x = G_1(s, t), y = G_2(s, t), z = G_3(s, t))$ where each G_i, $i = 1, 2, 3$, is a rational function (ratio of polynomials) in s and t over $\mathbb{R}$. The choice of algebraic surfaces was primarily motivated from the fact that manipulating polynomials, as opposed to arbitrary analytic functions, is computationally more efficient [12]. Furthermore, algebraic surfaces provide enough generality to accurately model most rigid objects. The primary advantage of the implicit definition $\mathcal{F}$ is its closure properties under modeling operations such as intersection, convolution, offset, blending, etc. The smaller class of parametrically defined algebraic surfaces $\mathcal{G}(s, t)$ is not closed under any of these operations. Closure under modeling operations allow cascading repetitions[3] without any need of approximation. Furthermore, designing with the complete class of algebraic surfaces leads sometimes to better possibilities (as we attempt to show here) of being able to satisfy the same geometric design constraints with much lower degree algebraic surfaces. The implicit representation of smooth algebraic surfaces also naturally yields half-spaces

[3]The output of one operation acts as the input to another operation

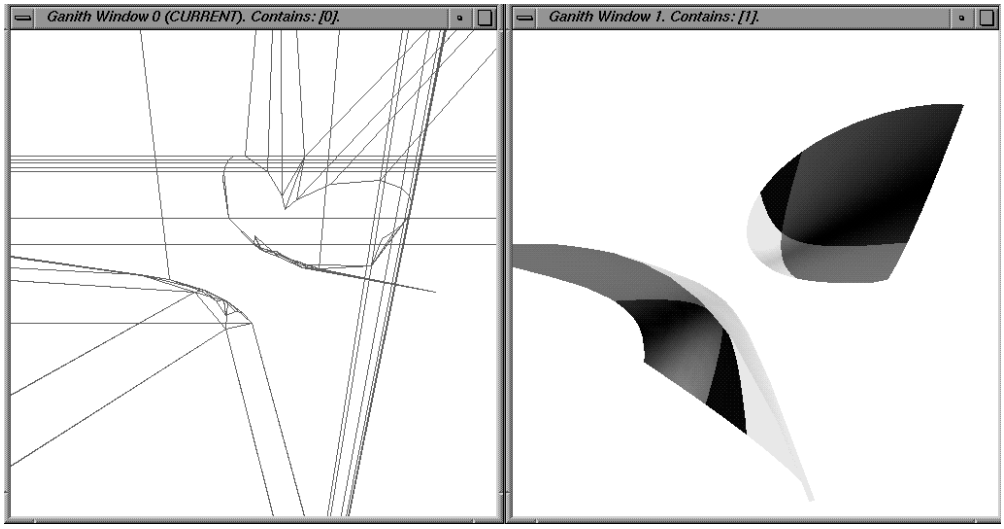

Figure 16: A quadratic parametric surface with domain poles.

$\mathcal{F}^+ : f(x,y,z) \geq 0$ and $\mathcal{F}^- : f(x,y,z) \leq 0$, a fact quite useful for intersection and offset modeling operations.

4.2 Triangulating Algebraic Surfaces

4.2.1 Approximation of Rational Surfaces

We consider the problem of computing piecewise linear or polygonal approximations of real algebraic surfaces. Modern day computer graphics hardware accept such polygonal approximations and accurately render the complicated surfaces with sophisticated lighting and shading models. Similar, more structured, linear approximations of surfaces are required for finite element approaches to solving system of partial differential equations.

A well-known strength of the parametric representation (its mapping from $\mathbb{R}^2$ to $\mathbb{R}^3$) is the ease by which real points can be generated on the parametric curve or surface. To compute real points on implicit algebraic surfaces requires the solution of polynomial equations. Furthermore, the problem of constructing a polygonal approximation, especially for finite element meshes, is complicated by the need for a correct topology of the mesh even in the presence of singularities and multiple sheets of the real algebraic surface. Direct schemes which work for arbitrary implicit algebraic surfaces are based on either the regular subdivision of the cube [35] or a finite subdivision of an enclosing tetrahedron [103]. However, such sampling methods fail in the presence of point and curve singularities of the algebraic surface, or yield ambiguous topologies in neighborhoods where multiple sheets of the surface come close together. Symbolic methods are necessary to disambiguate or calculate the correct topology for general algebraic curves and surfaces[4, 92].

While the issue of surface singularities are not as critical for rational parametric surfaces, the problems of constructing polygonal approximations with consistent topology is still highly non-trivial. Rational parametric surfaces have *pole curves* in their domain, where the denominators of the parameter functions vanish, domain *base points* for which all four numerator and denominator polynomials vanish simultaneously, and other features that cause naiive polygonal approximation algorithms to fail. These are ubiquitous problems occurring even among the natural quadrics. We illustrate the problems in more detail.

1. [Finite Parameter Range] To fully cover the parametric curve or surface, one must allow the parameters to somehow range over the entire parametric domain, which is infinite. For

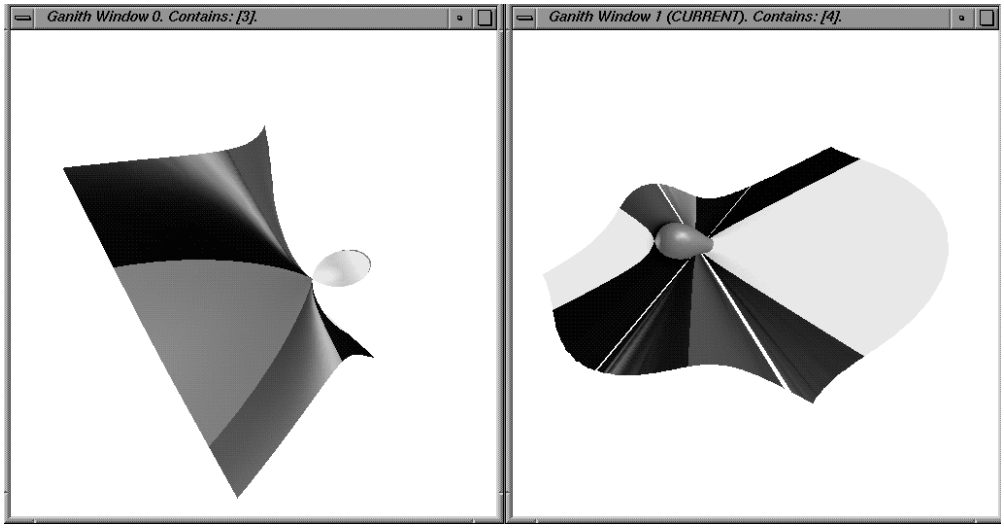

Figure 17: A cubic parametric surface with seam curves due to base points.

example, the unit sphere $f(x,y,z) = x^2+y^2+z^2-1 = 0$ has the standard rational parametric representation $(x = \frac{2s}{1+s^2+t^2}, y = \frac{2t}{1+s^2+t^2}, z = \frac{1-s^2-t^2}{1+s^2+t^2})$ In this parameterization the point $(0,0,-1)$ can only be reached by the parameter values $s = t = \infty$.

2. [Complex Parameter Range] It is possible for real points of a curve or surface to be generated only by complex parameter values. For instance, the rational algebraic curve $f(x,y) = x^3+x^2+y^2 = 0$ has an isolated real point at the origin. A rational parametric representation of this curve is $(x(s), y(s)) = (-(s^2+1), -s(s^2+1))$. In this parameterization the origin can only be reached by the complex parameter value $s = \sqrt{-1} = i$.

3. [Poles] Even when restricting the surface to a bounded real part of the parametric domain, the rational functions describing the surface may have poles over that domain. A hyperboloid of two sheets, with implicit equation $z^2+yz+xz-y^2-xy-x^2-1 = 0$, has the parametric representation

$$\begin{aligned} x(s,t) &= \frac{4s}{5t^2+6st+5s^2-1} \\ y(s,t) &= \frac{4t}{5t^2+6st+5s^2-1} \\ z(s,t) &= \frac{5t^2+6st-2t+5s^2-2s+1}{5t^2+6st+5s^2-1} \end{aligned}$$

then problems arise because of the pole curve described by $5t^2+6st+5s^2-1 = 0$ in the parameter domain. See Figure 16.

4. [Base Points] The rational parameter functions describing curves and surfaces are generally assumed to be reduced to lowest common denominators, i.e., the numerator and denominator of each rational function are relatively prime. Thus for a curve, there is no parameter value that can cause both numerator and denominator of a rational parameter function to vanish.

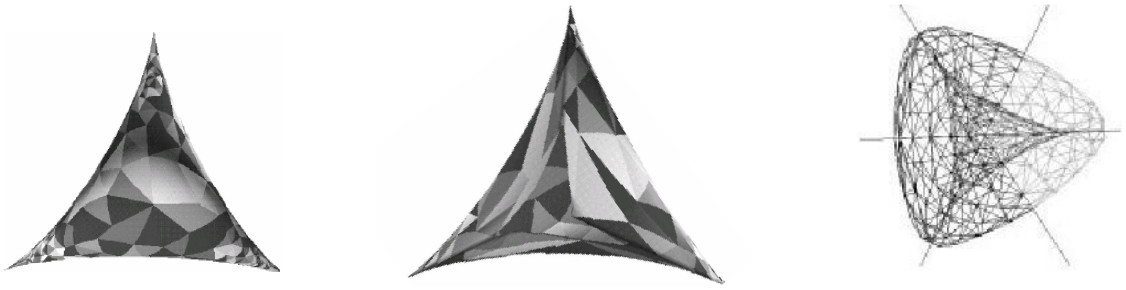

Figure 18: Triangulation of the Steiner Rational Surface $f = x^2y^2 + y^2z^2 + z^2x^2 - 4xyz = 0$ along line singularities

For surfaces, the situation is different. A surface is defined by three bivariate rational functions

$$\begin{aligned} x(s,t) &= \frac{f_1(s,t)}{f_4(s,t)} \\ y(s,t) &= \frac{f_2(s,t)}{f_4(s,t)} \\ z(s,t) &= \frac{f_3(s,t)}{f_4(s,t)} \end{aligned}$$

Even if F_1, F_2, F_3, F_4 are relatively prime polynomials, it is still possible that there are a finite number of points (a, b) such that $F_1(a, b) = F_2(a, b) = F_3(a, b) = F_4(a, b) = 0$. Each such point is called a *base point* of the parametric surface. There may also be base points at infinity in the parameter domain, and the base points can be complex as well as real-valued. Information about base points can be found in books on algebraic geometry such as [95, 107]. Base points are problematic since there is no one surface point for the corresponding domain point. To each base point there actually corresponds a curve on the surface [95], and since there is no parameter value for surface points on such a curve, the entire curve will be missing from the parametric surface. Such a curve is called a *seam curve*. See the right side of Figure 17 which corresponds the cubic parametric surface $x = \frac{t^3-t+s^3-s^2+1}{t^3+s^3+1}, y = \frac{2t^3-t^2-s^2t+2s^3+2}{t^3+s^3+1}, z = \frac{-st-s^3}{t^3+s^3+1}$. Thus for a truly accurate display of a parametric surface, one should also display the seam curves, alongside the parametric surface. See the left side of Figure 17 where the seam curves are bridged. Details are given in [84].

In [26] we give solutions to the above problems for the C^0 meshing of rational parametric curves, surfaces and hypersurfaces of any dimension. The technique is based on homogeneous linear (projective) reparameterizations and yields a complete and accurate C^0 planar mesh of free-form, discontinuous rational parametric domains. For the Cartan surface $(x = s, y = \frac{s^2}{t^2}, z = t)$, a single reparameterization $(x = st, y = s^2, z = t)$ removes the pole $t = 0$ of the original parameterization.

For the Steiner surface $(x = \frac{2st}{1+s^2+t^2}, y = \frac{2s}{1+s^2+t^2}, z = \frac{2t}{1+s^2+t^2})$, and the cubic elbow surface $(x = \frac{4t^2+(s^2+6s+4)t-4s-8}{2t^2-4t+s^2+4s+8}, y = \frac{4t^2+(-s^2-6s-20)t+2s^2+8s+16}{2t^2-4t+s^2+4s+8}, z = \frac{(2s+6)t^2+(-4s-12)t-s^2-4s}{2t^2-4t+s^2+4s+8})$, four different projective reparameterizations yield a complete covering of the rational parametric surface.

Figure 18 shows a triangular mesh approximating the a Steiner quartic surface. The mesh was constructed using the surface display algorithm. The surface crosses itself along the x,y and z axes. In this case, the mesh is actually a surface triangulation. This is a coincidence, and happens because the four quadrants of the parameter domain happen to map onto four pieces of the surface that meet exactly along the singular lines. Since the surface display algorithm maps each of the four domain quadrants separately, the resulting triangles on the mesh also meet along the singular lines.

Suppose we apply a random linear reparameterization to get another map for the same Steiner surface, and apply the display algorithm to generate a mesh for the new map. In general, the new mesh will not be a surface triangulation.

In [26] for surfaces, four reparameterizations always suffice. In general 2^d projective reparameterizations suffice for a d dimensional parametric hypersurface, see [27]. The algorithms which computes these reparameterizations as well as generates the C^0 planar meshes have been implemented in C in our GANITH toolkit[25].

4.3 Topologically Correct Approximations of Arbitrary Rational Parametric Surfaces

Points on a parametric surface patch can be generated by sampling the parametric functions over some region of the parameter domain. Because of this, the display of patches of parametric surfaces is well-understood [39, 88, 69, 55, 83, 96]. Some methods address in detail the problem of generating a polygonal mesh on a surface that is sensitive to variations in surface curvature: view-dependent methods [101] as well as view-independent [68, 24].

The parametric functions that define a surface can be viewed as a map from $\mathbb{R}^2$ into $\mathbb{R}^3$. "Domain sampling" methods such as the above assume that the parametric functions are defined and continuous in the region of the parameter domain that is being mapped. If the parametric functions are *rational*, however, they could be undefined at some points in $\mathbb{R}^2$. Many surfaces (including simple ones such as some quadrics) are given by rational maps which are undefined at some points.

We investigate how to correctly approximate a part of an *arbitrary* parametric surface, given a rational map that defines the surface. Our techniques are applicable whether this part of the surface is described by a bounded portion of the parameter domain, or by a bounding box in $\mathbb{R}^3$. If a bounding box is specified the algorithm will use the entire (infinite) parameter domain to compute parts of the surface that lie inside the box.

In this formulation the problem is of interest to CAD designers as well as mathematicians interested in surface visualization. The former usually express the rational functions defining the surface in terms of the rational Bézier or B-spline bases [36] with non-negative weights, restricting the rational functions to a standard part of the domain. However, researchers are considering generalizations to rational patches in which the rational functions are not defined everywhere [54, 104], making our techniques relevant.

In addition to topologically correct approximations, we consider the problem of constructing *triangulations* on arbitrary rational parametric surfaces, especially surfaces that self-intersect. Constructing triangulations on surfaces is useful for mesh generation in finite-element analysis. It turns out that our surface approximation technique can be extended in a straightforward way to handle this useful companion problem.

Thus our surface approximation techniques find application in the mathematical visualization of surfaces (our original motivation), rendering of arbitrary NURBS, and in finite-element meshing.

Let a parametric surface be given by a rational map of three rational functions:

$$\begin{aligned} x(s,t) &= \frac{X(s,t)}{W(s,t)}, \\ y(s,t) &= \frac{Y(s,t)}{W(s,t)}, \\ z(s,t) &= \frac{Z(s,t)}{W(s,t)} \end{aligned} \tag{77}$$

where X, Y, Z, W are polynomials with real coefficients and no common factor. Then we formulate two problems as follows:

• Given a portion of the domain, compute a topologically correct piecewise-linear approximation to the corresponding part of the surface defined by (77). Or, given a bounding box in $\mathbb{R}^3$, compute a topologically correct piecewise-linear approximation to the parts of the surface lying inside the box. We assume the latter case of the problem since the techniques to be discussed apply to the former case also.

• As above, except we further require the piecewise-linear approximation to be a surface *triangulation*, i.e. it must be a triangular mesh whose edges meet only at vertices and along edges.

Our basic approach to constructing a surface triangulation is to map a domain triangulation onto the surface using the parametric map, as in common. However, the domain triangulation is constructed carefully so that the surface triangulation is topologically correct.

We describe various subproblems that arise in trying to solve the above problems when we don't place any restrictions on the rational functions $x(s,t)$,$y(s,t)$,$z(s,t)$. We then give a solution for each subproblem, and combine the solutions in an algorithm for generating topologically correct triangulations on arbitrary rational parametric surfaces.

The subproblems are explained in detail in section 3. They are: domain poles, domain base points, surface self-intersections, complex parameter values, and infinite parameter values. We describe them briefly here.

1. **Domain poles**. The map is undefined at points satisfying $W(s,t) = 0$. There is a one-dimensional family of such domain points. The parametric functions can't be evaluated at such points; even if they never are, we might construct a surface approximation that does not represent its shape correctly.

2. **Domain base points**. The map is undefined at points satisfying $X(s,t) = Y(s,t) = Z(s,t) = W(s,t) = 0$. There are finitely many such points, called domain *base points*. It is known that an entire curve on the parametric surface corresponds to each base point; the points of this curve can't be directly computed using the rational map. Ignoring base points can lead to a topologically incorrect surface approximation.

3. **Surface self-intersections**. The surface intersects itself. Even if the rational map has no poles or base points, mapping an arbitrary domain triangulation onto a parametric surface may not yield a surface triangulation because surface triangles cross each other.

4. **Complex parameter values**. Some real points of the surface are generated only by complex parameter values.

5. **Infinite parameter values**. Some finite points of the surface are generated only by infinite parameter values.

For graphics display and NURBS rendering, subproblem (3) is not necessary (although z-buffering still causes wavy lines along polygon intersections due to aliasing). If finite-element meshing is the application, subproblem (3) is of interest.

The problems can be extended to include rational parametric surfaces in higher dimensions, but we don't discuss this here. The general flavor of the methods discussed will still apply, although implementing higher-dimensional methods would require more tools.

In a preliminary paper [26] we discussed subproblems (1) and (5). In the current paper we give solutions for (2), (3), and (4) as well. Because of this, the current paper has a much broader scope and more applications than [26].

The rest of this paper is organized as follows. First, we discuss two approaches: either directly approximating the surface in the range space of the parametric functions, or computing those portions of the domain that map onto the desired parts of the surface. We argue that the domain-space approach is preferable in this context. After explaining the above subproblems in detail, we

present techniques for dealing with each subproblem. We then use these techniques in an algorithm for generating topologically accurate surface triangulations. After explaining the algorithm detail we discuss situations in which it can fail and where it could be improved, based on extensive experimentation.

4.3.1 Domain space vs. range space approaches

One way to construct a piecewise-linear approximation to a parametric surface is to evaluate the parametric functions at various points on the parameter domain, and link together the resulting surface points to form an approximating mesh. When considering arbitrary rational parametric surfaces, the parametric functions may not be defined at some points, since rational functions are not defined at points where the denominator vanishes. Such points are called *poles*, and usually correspond to surface points at infinity. The exception occurs when all the polynomials X, Y, Z, W vanish there (an event that can happen only finitely many times since they have no common divisor, by assumption). In this case the parameter point is a domain *base point*.

We shall later explore poles and base points in detail, showing examples of how they can cause domain sampling techniques to fail.

Another way to approach the problem is to work directly in the range space of the rational function map. Since we are only interested in portions of a surface inside a bounding box, and poles correspond to surface points at infinity, a range-space method can avoid explicitly evaluating the rational functions at poles (base points still cause problems).

The following system of equations is equivalent to (77):

$$\begin{aligned} W(s,t)x - X(s,t) &= 0 \\ W(s,t)y - Y(s,t) &= 0 \\ W(s,t)z - Z(s,t) &= 0 \end{aligned}$$

One can theoretically implicitize the parametric surface by eliminating s, t from this system [71] using several available methods [37, 42, 41, 56, 73] and then approximate the resulting implicit surface directly. Note that a parametric surface of degree n could have an implicit equation of degree n^2.

However, implicit surface approximation techniques [18, 35, 42, 60, 79, 98] don't handle surface self-intersections very well, although research is being done to overcome this [8, 9, 32]. Since we would like to display surfaces with complicated singularities and several real sheets, we avoid the range-space approach. We show instead that a careful evaluation of the domain is sufficient to generate an accurate piecewise-linear approximation of the parametric surface.

Difficulties in domain sampling In this section we explain why domain base points and poles sometimes cause sampling techniques to fail, and give simple examples that are representative of the kinds of failures that occur. The main problem is that domain sampling techniques which don't take poles and base points into account can generate surface approximations which do not accurately represent the topology of the surface.

Domain poles Inability to evaluate a rational function at a pole (i.e., generating a divide by zero exception in a numerical program) is not the main reason that domain sampling methods fail when poles are present. Even if a domain sampling method avoids evaluating a rational map at a pole, it may construct an approximation that does not reflect the actual shape of the surface. This happens when a part of the domain that contains a pole is mapped onto the surface.

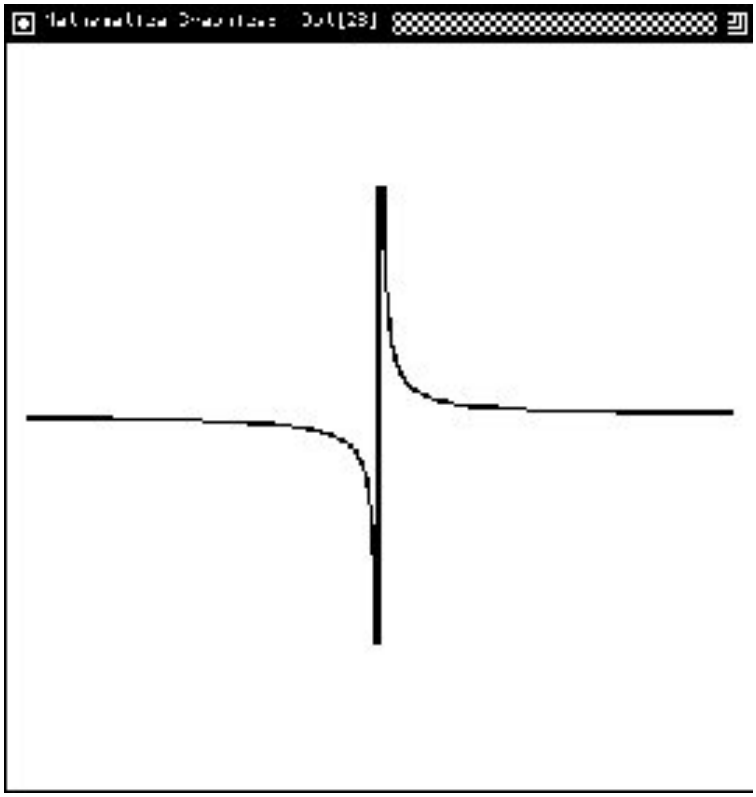

Figure 19: Disjoint branches being wrongly connected (*Mathematica*)

When a parameterization contains poles, the surface may have multiple branches or sheets. We show a simple example using a parametric curve. The hyperbola given by

$$x(s) = s, \quad y(s) = \frac{1}{s}$$

has a pole at $s = 0$. The real part of the curve consists of two branches.

A simple domain sampling algorithm for approximating this hyperbola might select a closed interval $[a, b]$ in the parameter domain, generate n equally-spaced parameter values $s_i = a + i(\frac{b-a}{n-1}), i = 0, \ldots, n$, and then connect the points $(x(s_{i-1}), y(s_{i-1}))$, $(x(s_i), y(s_i))$ with a straight-line segment. In this example, a line segment could be drawn between points whose parameter values lie on opposite sides of a pole. As a result, the approximation does not accurately represent the shape of the curve.

Figure 19 shows the output of the program *Mathematica* for plotting the hyperbola over the domain interval $s \in [-\frac{1}{2}, \frac{1}{2}]$.

With surfaces the problem is acute, and poles can cause problems even when the surface has a single real sheet. For instance, a hyperboloid of one sheet with implicit equation $x^2+y^2-z^2-1 = 0$ is a surface whose real part is single-sheeted (i.e. connected). However, if we work from the equivalent parametric representation

$$x(s,t) = \frac{t^2 - s^2 + 1}{s^2 + t^2 - 1}, \quad y(s,t) = \frac{2st}{s^2 + t^2 - 1}, \quad z(s,t) = \frac{2t}{s^2 + t^2 - 1} \tag{78}$$

then problems arise because of the pole curve described by $s^2+t^2-1 = 0$ in the parameter domain. The right picture in

Figure 20 shows the output produced by *Maple V* for this surface with $(s,t) \in [-2,2] \times [-2,2]$ (a domain region containing the pole curve).

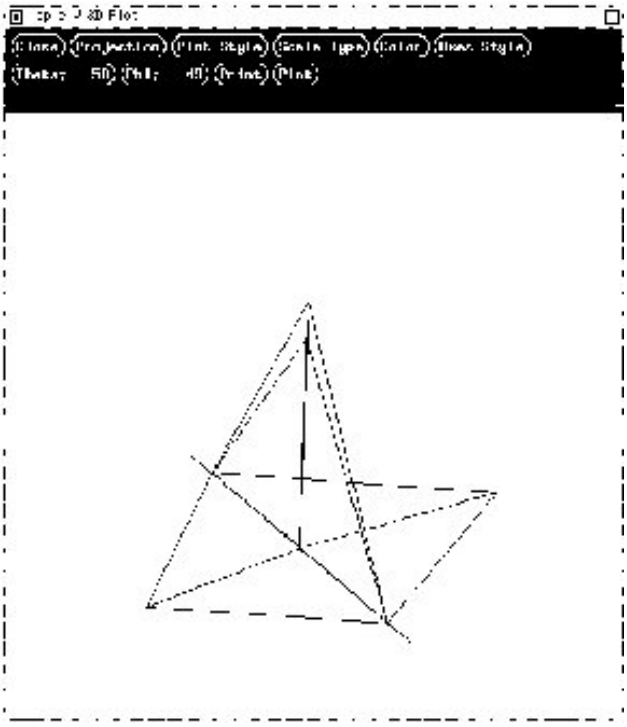

Figure 20: Single-sheeted surface with domain poles (*Maple*)

A small digression is in order about the programs *Mathematica* and *MapleV*. Both programs use sophisticated strategies for graphing curves and surfaces that are generally effective. However, they use domain sampling techniques which are not equipped to handle parametric functions that are not defined everywhere, and hence fail for simple examples such as the above.

Domain base points We assumed that the numerators and common denominator of the rational map (77) have no common factor. It is still possible that there are a finite number of points (a, b) such that $X(a, b) = Y(a, b) = Z(a, b) = W(a, b) = 0$. Each such point is called a *base point* of the parametric surface. Information about base points can be found in books on algebraic geometry such as [61, 94, 106]. Interesting material on base points in the context of CAGD appears in [41, 74, 91, 104]. In particular, [104] shows how to represent patches with up to six sides in the triangular rational Bézier patch form, by a clever use of domain base points.

Base points are problematic since there is no one surface point for the corresponding domain point. To each base point there actually corresponds a rational curve on the surface [94]. Approaching the base point along different directions leads to different points on the surface; the points corresponding to all directions form a space curve that lies on the surface. Since there is no parameter value for points on this curve (at which the surface map is defined), the entire curve will be missing from the parametric surface. Such a curve is called a *seam curve*. Even if poles are taken care of in some way, the seam curves can show up as *gaps* on the surface. This figure shows the hyperboloid of one sheet given by (78). This parameterization has the two base points $(s, t) = (\pm 1, 0)$. The corresponding seam curves can be parameterized in parameters u, v, giving the lines $(x(u), y(u), z(u)) = (-1, u, u)$ and $(x(v), y(v), z(v)) = (-1, v, -v)$ on the surface.

If base points are not taken into account, the domain sampling density may need to be unnecessarily dense (with respect to surface curvature) in order for the gaps to be narrow. Furthermore, even if the gap is narrow enough to suffice for display, the surface approximation will not correctly represent the surface's topology because of the gap.

Figure 21: Hyperboloid of 1 sheet with seam curve gaps

Surface self-intersections A triangular mesh on a parametric surface is derived by constructing a planar triangulation in the domain and mapping it onto the surface. However, when a planar domain triangulation is mapped onto a curved surface, the resulting triangles in space may no longer form a triangulation.

There are two reasons for this. First, if the domain sampling density is not fine enough with respect to the surface curvature, two surface triangles may overlap each other. Second, if the surface actually crosses itself, some surface triangles near the crossing may cross each other. For finite-element mesh generation, surface triangulations are preferred. Even for display, a surface triangulation is preferable. This is because scanline-rendering algorithms suffer from aliasing effects along triangle intersections; this causes what should appear as a sharp edge on the screen to appear wavy.

Complex parameter Values While the parameterization (77) defines a map from $\mathbb{R}^2$ into $\mathbb{R}^3$, it also defines a unique algebraic surface in $\mathbb{C}^3$ which can be given by a single equation in three variables, with real coefficients. This algebraic surface may contain real points which are not mapped by any real parameter values. If we want to view the entire real part of the algebraic surface defined by the map, and not just the image of $\mathbb{R}^2$, additional computations are needed.

For instance, consider a Steiner surface given implicitly by $F(x,y,z) = x^2y^2 + y^2z^2 + x^2z^2 - 2xyz = 0$, or parametrically by

$$x(s,t) = \frac{2s}{s^2+t^2+1}, \quad y(s,t) = \frac{2t}{s^2+t^2+1}, \quad z(s,t) = \frac{2st}{s^2+t^2+1}$$

(it is a quartic algebraic surface defined by a quadratic rational map).

Note that the x, y and z axes lie entirely on the algebraic surface $F(x,y,z) = 0$. Let us consider the parametric map to see which parameter values give rise to the x axis, which is described by $y = z = 0$. Setting $y(s,t) = z(s,t) = 0$ and solving for s,t yields $t = 0$. Thus $(x(s,0),0,0) = (2s/(s^2+1),0,0)$, $s \in \mathbb{R}$, are the points on the x axis that are given by the map. This shows that any parameter value $s \in \mathbb{R}$ yields a surface point $(x,0,0)$ with $|x| \leq 1$.

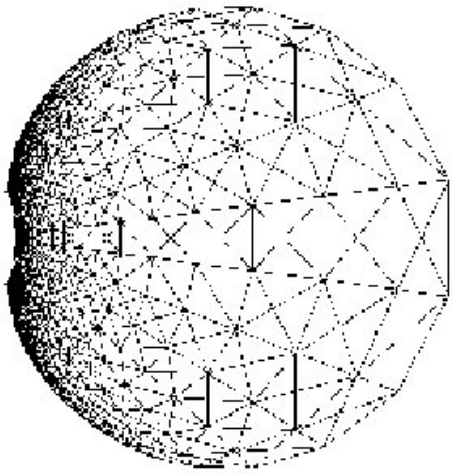

Figure 22: Infinite parameter values mapping to finite point

To find parameter values giving rise to the remaining surface points on the x-axis we must extend the parameter domain to $\mathbb{C}^2$.

Infinite parameter values Consider the following map for the unit sphere in $\mathbb{R}^3$:

$$\begin{aligned} x &= \frac{1-s^2-t^2}{s^2+t^2+1} \\ y &= \frac{2s}{s^2+t^2+1} \\ z &= \frac{2t}{s^2+t^2+1} \end{aligned}$$

The (finite) point $(-1,0,0)$ on the sphere is the image of the entire line at infinity in $\mathbb{R}^2$. Simply using large parameter values to represent infinity is not enough to construct a topologically correct polygonal approximation; the polygons will approach the "missing point" ever closer but never fill the gap.

To compute certain finite points on the surface we may need to extend the parameter domain to include parameter values at infinity, i.e. extend the parameter domain to be the projective plane.

We have described the main problems that occur in constructing topologically accurate polygonal meshes on rational parametric surfaces, when no restriction is placed on the rational map defining the surface.

These problems generally occur because a particular rational map for the surface can be locally "bad" near some domain points. However, from any single map for the parametric surface we can extract all information necessary to compute all parts of the surface inside the bounding box.

Techniques for overcoming difficulties In this section we outline the basic idea for solving each of the problems addressed above. Additional details are given in the next section, when the complete algorithm is shown.

Partition of domain by pole curves Rational functions are undefined at points in the domain where their denominator vanishes, and continuous everywhere else. Hence, the pole curve partitions the parameter domain into regions, such that inside each (open) region the functions of the parametric map are defined and continuous.

Therefore, our approach to handling pole curves is simple: we partition the domain by the pole curve. In particular, we construct a special triangulation of the domain that respects this partition. In this triangulation, a domain triangle contains pole points only on its boundary and not in its interior. Since pole curves may not be lines, in practice we shall construct a piecewise-linear approximation of the pole curve and then identify linear curve approximants with edges of the triangulation.

Once such a triangulation is constructed, we know that each domain triangle maps onto a single-sheeted patch, since there are no pole points in the interior (pole points at a vertex correspond to points at infinity, and therefore the patch may be semi-infinite). A conventional domain sampling technique is used in the interior of the triangle to mesh the patch to any desired precision. The patch can then be clipped against a bounding box, if necessary.

If base points are not present, domain partitioning combined with the handling of infinite parameter values (discussed below) suffices to generate a topologically correct mesh of the parametric surface, even if it is multi-sheeted.

Base points and seam curve parameterizations When base points are present, it is not sufficient to just handle pole curves as gaps may still be present. The surface approximation will then not be topologically correct, since the surface approximation will be "torn" along the seam curves.

To handle base points, we must "stitch" the surface up along seam curves. This can be done in the framework of domain partitioning, as follows. We compute all base points and insert them into the domain triangulation as additional vertices – thus base points will occur explicitly at the vertices of a domain triangle.

In general, approaching a base point along different directions in the domain leads to a different surface point (in the limit). Thus a base point "blows up" onto an entire "seam" curve on the surface [94] – each point of this curve corresponds to a different limit direction at the base point. A consequence of this fact is that a domain triangle with a base point vertex maps onto a four-sided patch on the surface. In general, a triangle with b base point vertices maps onto a $(b+3)$-sided patch – a fact exploited in [104] to represent multi-sided patches over triangular domains.

Once we have a *parameterization* of the seam curves, it is easy to generate the patch corresponding to a domain triangle with base point vertices, however many sides it has. Each of the two edges adjacent to a base point vertex corresponds to a particular direction, and therefore to a particular parameter value. The two parameter values then define a segment of the seam curve. This curve segment is the side on the patch that corresponds to the domain base point.

We now discuss the computation of seam curve parameterizations. Points on a rational parametric surface are given as follows (temporarily using projective coordinates for notational convenience):

$$\rho X = X(s,t), \quad \rho Y = Y(s,t), \quad \rho Z = Z(s,t), \quad \rho W = W(s,t)$$

where ρ is a non-zero constant of proportionality (we still use an affine domain, which is sufficient as we later show).

Then, let O be a common solution of the curves $X = 0, \ldots, W = 0$. Furthermore, let us suppose that O is a point of multiplicity q on each of the curves $X = 0, \ldots, W = 0$, and that the curves have no common tangent at O. Then the image of the base point O is a rational curve of degree q on the surface [94].

In [41], a method is given to find the parametric equations of this curve. The basic idea is to pass a pencil of lines through the base point and then use the slope of these lines as a parameter, since approaching the base point from each direction leads to a different point on the seam curve. The seam curve equations are not given explicitly, but as quotients of certain polynomials. The algorithm fails when the curves $X = 0, \ldots, W = 0$ have common tangents at O; in this case the parametric equations given by this algorithm generate only a single point of the seam curve.

In [74] a method is given for parameterizing seam curves that works for all cases (i.e., even when the tangents are equal). However, it is much more expensive than the previous method and not currently practical: multivariate resultants are used to compute a projection onto a plane of all the seam curves simultaneously, yielding a bivariate equation. Along with the projection, a rational map R is computed between the projection and the curves on the surface. A bivariate factorization algorithm (over the complexes) such as [13, 67] must first be applied to separate out the the projections of the individual curves. Each projected seam curve is then parameterized using a general curve parameterization technique [4], and finally mapped onto the surface using the rational map M.

The method of [41] is much simpler than that of [74], and could be implemented as part of the surface display algorithm. However, we present a further simplification of [41] based on the the same idea, which is found in algebraic geometry textbooks such as [94] (and hence it also fails when the tangents at O are all equal). This simplification makes the method easier to implement numerically, since we find an explicit formula for the parametric equations of the seam curve. Furthermore, the formula clearly shows how the number of common tangents of $X = 0, \ldots, W = 0$ at the base point affects the seam curve, explaining why this method breaks down when the tangents at the base point are all equal.

Theorem 4.1. *Let (a, b) be a base point of multiplicity q. Then for any $m \in \mathbb{R}$, the image of a domain point approaching (a, b) along a line of slope m is given by* $(X(m), Y(m), Z(m), W(m)) =$

$$\left(\sum_{i=0}^{q} \left(\frac{\partial^q X}{\partial s^{q-i} \partial t^i}(a,b) \right) \binom{q}{i} m^i, \ldots, \sum_{i=0}^{q} \left(\frac{\partial^q W}{\partial s^{q-i} \partial t^i}(a,b) \right) \binom{q}{i} m^i \right) \tag{79}$$

PROOF. Consider the image of a point (s, t) as it approaches (a, b) along the line of slope m through (a, b). Expressing the line as $t = m(s - a) + b$, this yields the point

$$\lim_{s \to a} (X(s, m(s-a)+b), Y(s, m(s-a)+b), Z(s, m(s-a)+b), W(s, m(s-a)+b)) \tag{80}$$

Expanding $X(s, t)$ in a Taylor series at (a, b) yields

$$X(s,t) = \sum_{k=0}^{p} \sum_{i=0}^{k} \frac{(s-a)^i (t-b)^{k-i} \binom{k}{i}}{k!} \frac{\partial^k X}{\partial s^i \partial t^{k-i}}(a,b) \tag{81}$$

Substituting $t = m(s - a) + b$ in (81) yields

$$\begin{aligned} X(s) &= \sum_{k=0}^{p} \sum_{i=0}^{k} \frac{(s-a)^k \binom{k}{i} m^{k-i}}{k!} \frac{\partial^k X}{\partial s^i \partial t^{k-i}}(a,b) \\ &= (s-a)^q \sum_{k=q}^{p} \sum_{i=0}^{k} \frac{(s-a)^{k-q} \binom{k}{i} m^{k-i}}{k!} \frac{\partial^k X}{\partial s^i \partial t^{k-i}}(a,b) \end{aligned}$$

where q is the multiplicity of the base point (a, b), which implies that all derivatives of $X(s, t)$ up to order $q - 1$ vanish at (a, b).

Substituting $t = m(s - a) + b$ into the Taylor expansions of $Y(s, t), Z(s, t), W(s, t)$ yields $(X(s), \ldots, W(s)) =$

$$(s-a)^q \left(\sum_{k=q}^{p} \sum_{i=0}^{k} \frac{(s-a)^{k-q} \binom{k}{i} m^{k-i}}{k!} \frac{\partial^k X}{\partial s^i \partial t^{k-i}}(a,b), \ldots, \sum_{k=q}^{p} \sum_{i=0}^{k} \frac{(s-a)^{k-q} \binom{k}{i} m^{k-i}}{k!} \frac{\partial^k W}{\partial s^i \partial t^{k-i}}(a,b) \right)$$

We drop the factor of proportionality $(s-a)^q$ and compute the limit (80):

$$\begin{aligned}\lim_{s\to a}(X(s),Y(s),Z(s),W(s)) &= \frac{1}{q!}\left(\sum_{i=0}^{q}\binom{q}{i}m^{q-i}\frac{\partial^q X}{\partial s^i\partial t^{q-i}}(a,b),\ldots,\sum_{i=0}^{q}\binom{q}{i}m^{q-i}\frac{\partial^q W}{\partial s^i\partial t^{q-i}}(a,b)\right)\\ &= \left(\sum_{i=0}^{q}\left(\frac{\partial^q X}{\partial s^{q-i}\partial t^i}(a,b)\right)\binom{q}{i}m^i,\ldots,\sum_{i=0}^{q}\left(\frac{\partial^q W}{\partial s^{q-i}\partial t^i}(a,b)\right)\binom{q}{i}m^i\right)\end{aligned}$$

Thus for each $m \in R$ there is a corresponding point (79) on the parametric surface. These points collectively form a one-dimensional family or curve on the surface. □

Corollary 4.2. *If the curves $X(s,t)=0,\ldots,W(s,t)=0$ share t tangent lines at (a,b), then the seam curve $(X(m),Y(m),Z(m),W(m))$ has degree $q-t$. In particular, if $X(s,t)=0,\ldots,W(s,t)=0$ have identical tangents at (a,b), then for all $m\in R$ the coordinates $(X(m),\ldots,W(m))$ represent a single point.*

PROOF. The equations of the tangent lines to the curve $X(s,t)=0$ at (a,b) are given by equating to zero the factors of the following curve, which are all linear (since it is homogeneous):

$$\sum_{i=0}^{q}\left(\frac{\partial^q X}{\partial s^{q-i}\partial t^i}(a,b)\right)\binom{q}{i}s^{q-i}t^i = 0 \tag{82}$$

and similarly for $Y(s,t)=0$ etc. Moreover, there is a 1-1 correspondence between the linear factors of this curve and the roots of the polynomial $X(m)$ in (79). Thus each common tangent of $X(s,t)=0,\ldots,W(s,t)=0$ at (a,b) leads to a common root, and hence a common factor, among $X(m),\ldots,W(m)$. If there are t common tangents there will be a common factor of degree t, which can be divided out of the seam curve parameterization $(X(m),\ldots,W(m))$ since proportional homogeneous coordinates represent the same point. Thus the seam curve is of degree $q-t$. □

Partitioning along surface self-intersections Earlier, we mentioned two reasons why a domain triangulation might not stay a triangulation when it is mapped onto a parametric surface. The first reason was because the domain sampling density was not high enough, and the second reason was because the surface might self-intersect.

The first case can be handled by increasing the domain sampling density (either locally or globally, although local curvature-sensitive sampling is much preferred since it generates fewer polygons). Several domain sampling techniques already adjust the sampling density due to curvature, so we focus on the second case.

The domain-partitioning technique lends itself to generating triangulations on surfaces that self-intersect. The key idea is to compute those points and curves in the parametric domain that map onto surface self-intersections, and then partion the domain by these points and curves (as well as by the pole curves). If this is done, no domain triangle will contain in its interior a point that map onto a surface singularity. Hence, triangles on the surface will meet only along their edges or at their vertices, even if the surface is singular.

Domain curves (and points) mapping onto surface singularities can be computed by solving systems of polynomial equations. For instance, cuspidal singularities correspond to domain points where the Jacobian matrix of the rational map does not have full rank. We can compute the symbolic Jacobian matrix and equate its minors to zero, yielding a set of polynomial equations whose common solution are domain points that map onto surface cusps. Nodal singularities can also be computed by solving a system of polynomial equations.

The system of equations has a one-dimensional solution set in general. Multivariate resultants [71, 72, 16, 73] can be used to project the solutions onto the parameter plane, after which a

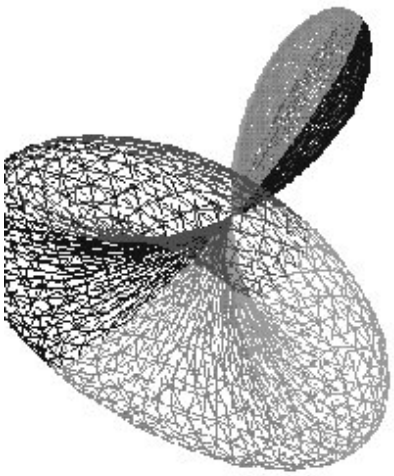

Figure 23: Triangulation of parametric surface with point singularity

curve-tracer can be used to compute an approximation. For tracing the curve one can use either subdivision methods, e.g. [57], or a marching method such as [17].

For example, consider the surface given by the the following equations, taking $x(s,t) = X(s,t)/W(s,t)$, etc.

$$\begin{aligned} X(s,t) &= s^3 + st^2 - 3s \\ Y(s,t) &= (s^2+t^2)^2 - 3(s^2+t^2) \\ Z(s,t) &= s^2t + t^3 - 3t \\ W(s,t) &= (s^2+t^2)^2 + 2(s^2+t^2) + 1 \end{aligned}$$

By substitution, one can verify that its implicit equation is

$$F(x,y,z) = z^4 + 2y^2z^2 + 3yz^2 + 2x^2z^2 + y^4 - y^3 + 2x^2y^2 + 3x^2y + x^4 = 0$$

This is a surface of revolution; it has a point singularity at the origin.

It can be shown that the domain points mapping onto the surface singularity satisfy $(t^2 + s^2 - 3)(t^2 + s^2) = 0$. Thus the circle of radius $\sqrt{3}$ centered at the origin, and the origin itself both map onto the surface (nodal) self-intersection at $(0,0,0)$. This circle and the origin partition the parameter domain into regions that meet at the surface self-intersection. By partitioning the parameter domain by the curve $t^2 + s^2 - 3 = 0$ and the point $(0,0)$, as by pole curves, we can construct a triangulation on this surface.

Computing complex parameter values We now show one way to compute the complex parameter values that map onto these points. Let the parameters s, t denote complex numbers given as $s = a + bi$, $t = c + di$, where $a, b, c, d \in \mathbb{R}$ and $i = \sqrt{-1}$.

Then the parametric map from $\mathbb{C}^2 \to \mathbb{R}^3$ can be expressed as

$$\begin{aligned} x(s,t) = x(a+bi, c+di) &= X_R(a,b,c,d) + X_I(a,b,c,d) \cdot i \\ y(s,t) = y(a+bi, c+di) &= Y_R(a,b,c,d) + Y_I(a,b,c,d) \cdot i \\ z(s,t) = z(a+bi, c+di) &= Z_R(a,b,c,d) + Z_I(a,b,c,d) \cdot i \end{aligned}$$

where X_R denotes the real part of $x(a+bi, c+di)$ and X_I denotes its imaginary part, etc.

Then $X_I(a,b,c,d) = 0$, $Y_I(a,b,c,d) = 0$, $Z_I(a,b,c,d) = 0$ form a system of three equations in four unknowns whose solutions give parameter values that map to real surface points. In general, such a system has a one-dimensional solution set.

Note that this particular system has the trivial two-dimensional solution $b = d = 0$ which must be excluded. Thus the marching method [17] cannot be used directly; rather, as for surface self-intersections, we must use resultants to first compute a projection of the space curve. After deleting the extraneous component due to the trivial solution, we can trace the projected plane curve and finally map it onto the space curve using the inverse of the projection.

The points (a,b,c,d) of the space curve give complex parameter values $s = a+bi$ and $t = c+di$ that map onto real points of the surface.

Mapping infinity using projective reparameterization To handle infinite parameter values, we use projective reparameterizations. In [47], a technique called "homogeneous sampling" is used to sample finite and infinite points of a surface equally. We use similar idea based on projective reparameterizations, so that only affine parameter values are needed. Specializing theorem 1 of [27], we use four reparameterizations of the original rational map, given by

$$\begin{aligned} s &= \pm\frac{u}{1-u-v} \\ t &= \pm\frac{v}{1-u-v} \end{aligned}$$

Each reparameterized map needs to be sampled only over the unit triangle of its domain ($u \geq 0$, $v \geq 0$, $u+v \leq 1$), yielding a triangular patch. The patches meet along their boundaries and together cover the entire surface (including finite points that were generated by infinite parameter values in the original surface).

4.4 Spline Approximations of Real (Implicit) Algebraic Surfaces

Real algebraic surfaces are often used to cope with the problem of modeling complicated shapes. Implicitly defined algebraic surfaces have both advantages, and disadvantages over functional and parametric surfaces. The class of implicit algebraic surfaces is closed under several geometric operations (intersections, union, offset, etc.), often desired in a solid modeling system. On the other hand, free-form geometric modeling (display and shape control) is much easier with parametric curve and surface spline representations (and evidenced by available software systems). This largely motivates the need for constructing parametric spline approximations of real algebraic surfaces.

As the role of implicit algebraic surfaces are increasing in importance in geometric modeling, rendering and approximating implicit algebraic surfaces becomes crucial in surface design. In computer graphics, most rendering algorithms for implicit surfaces rely on piecewise linear approximations (polygons) based on space subdivision or polyhedron continuation. Bloomenthal [35] used octrees by spatial partition to reach a polygonal approximation. Micchelli and Prautzsch [75] used uniform refinement algorithms for surface generation. Allgower and Gnutzmann, et al, [7], used simplicial continuation or pivoting algorithms to generate a triangular or quadrilateral polygonal approximations. For the similar purpose, Chuang [43] used cubic continuation technique.

We use neither space subdivision nor polyhedron continuation. Instead, we use triangular surface patch expansion by choosing first three points on the surface and then constructing a wire frame (with normal functions for G^1 continuity) and finally finding a patch to cover the wire frame. Comparing with the approaches above mentioned, the patch expansion uses fully the property of the function to be approximated locally, it therefore is adaptive.

Once a wire frame is established, we are lead to an interpolation problem. Relating to this, there are many prior works. Bajaj and Ihm [19] use implicitly defined surface to accomplish the

task but it can not serve us for our purpose. A lot of works deal with C^k continuity problem under various assumptions. For example, Nasri [77] consider surface interpolation on irregular networks with normal conditions at vertices but without specifying space wire frame. Herron [62] uses parametric surfaces interpolating function values and tangential derivatives at the vertices of a triangle. His method was generalized [63] to cover G^1 continuity case for closed surfaces. The interpolation problems are solved by two approaches. One is using functional patch by subdividing each triangular patch into three smaller patches. The other approach is using parametric patches to cover a wire frame. Compared with the method given in [63], the second approach leads to a lower degree of approximation.

4.4.1 The Main Steps of the Algorithm

1. **Singularity Computation**. The algorithm itself is an expansion approach of triangular patch from a given seed point on the surface. This expansion approach works in the smooth part of the surface. Hence the singularity of the surface must be treated separately. The *Singularity Computation* includes the computation of singular points and singular curves in the given bounding box. For the singular curves, we provide a list of points on each curve.

2. **Triangulation**. This step will provide a piecewise linear(triangular) approximation of the surface with correct topology. The approximation of the smooth part is based on the power series expansion and from which triangles are produced. When the vertices of the triangles approach to the singular points, the vertices will be stitched to the singular points by adding edges. This will guarantee the triangulation have correct topology.

3. **Wire Construction**. This step will construct a wire frame by providing normals at each vertex of the triangulation and then building a space curve and a normal function for each edge such that the curve passes the two vertices of the edge and the normal function has the given normal at the vertices and orthogonal to the tangent of the curve. The normal at a point that is the smooth point of the original surface is provided by the surface. At the singular points, the surface normals are not well defined, the wire frame do not take account the normal condition.

4. **Patch fitting**. For each three wires with normal functions over a triangle, construct a surface patch that contains the wires and has the normal function on the boundary. Hence the composite surface is G^1 continuous.

4.4.2 Adaptive Triangulation

Edge Expansion Approach We begin with a few notational definitions

Expansible edge. During the process of expansion of the triangular polygon, an edge is called expansible if we can go further outward from this edge to get a new triangle. That is

(a) this edge is on the boundary of the present constructed polygon,

(b) this edge is inside the given boundary box.

P-plane, The P-expression, expansion point.

Let $p_0 = (x_0, y_0, z_0)$ be a point on the surface $f(x, y, z) = 0$. Then the orthogonal transform

$$T: \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} c_2 & s_2 & 0 \\ -c_1 s_2 & c_1 c_2 & s_1 \\ s_1 s_2 & -s_1 c_2 & c_1 \end{bmatrix} \begin{bmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{bmatrix}$$

with

$$c_1 = f_z(p_0)/\|\nabla f(p_0)\|, \quad c_2 = -f_y(p_0)/\sqrt{f_x(p_0)^2 + f_y(p_0)^2},$$

$$s_1 = \sqrt{f_x(p_0)^2 + f_y(p_0)^2}/\|\nabla f(p_0)\|, \quad s_2 = f_x(p_0)/\sqrt{f_x(p_0)^2 + f_y(p_0)^2}$$

establishes an one-to-one map between (x, y, z) space and (X, Y, Z) space. It is easy to see that the XY-plane is the tangent plane of the surface $f(x, y, z) = 0$ at the point p_0. We call this plane as projection plane of $f = 0$ at p_0, denoted as *P-plane*. The projection of a space point p on *P-plane* is denoted by $P(p)$ that is consists of the first two components of $T(p)$. On the *P-plane*, $f(x, y, z) = 0$ can be expressed locally as a power series $Z = \phi_{p_0}(X, Y)$. We call its trunction up to degree k a *P-expression*, denoted by $\phi_{p_0,k}(X, Y)$. The point p_0 is referred to as *expansion point*.

(p, k, ϵ)*–circle*, (p, k, ϵ)*–sphere*, (p, k, ϵ)*–radius*

If the maximal $r = r(p, k, \epsilon)$ for which

$$\|\phi_{p,k}(X, Y) - \phi_p(X, Y)\| < \epsilon \quad \text{for} \quad X^2 + Y^2 \le r^2$$

Then we say the circle $X^2 + Y^2 = r^2$ is a (p, k, ϵ)*–circle* $X^2 + Y^2 + Z^2 = r^2$ is (p, k, ϵ)*–sphere* of f at p and r is (p, k, ϵ)*–radius*.

It is easy to see that $r(p, k, \epsilon)$ converge to the convergence radius of $\phi_p(X, Y)$ at p as $k \to \infty$.

Now we can state the steps of expansion of polygons. Let S be the collection of singular points and point lists on the singular curves.

Algorithm 1

1. *Initial Step*. For a given smooth point p_0 on one component of the surface $f(x, y, z) = 0$ and in the given bounding box, we first compute the P-expression $Z = \phi_{p_0,k}(X, Y)$. On the P-plane, then find the (p_0, k, ϵ)*–radius*. Take three points on the (p_0, k, ϵ)*–circle* uniformly, say q_0, q_1, q_2, and refine the points $(q_i, \phi_{p_0,k}(q_i))$ by Newton method such that the resulted points V_i are on the surface. If Newton method fails, reduce the radius of the circle and then try again. If the point V_i is outside the bounding box, then adjust it to the boundary. The triangle $[V_0, V_1, V_2]$ is the first one we wanted. And then the angle at each vertex of the triangle that is defined by the adjacent edges counted outward are computed. In this initial case, each edge is expansible except the one that is on the boundary.

2. *General Step*. Suppose we have constructed several space triangles that forms one or more than one connected mesh consisting of triangles. For each mesh, we keep the boundary information such as edges with related expansion point, vertices with angles. Assume now that at least one of the edges is expansible. Then the general step is to construct one more triangle that joins the original one and enlarge the mesh.

   (a) Find a vertex on the present boundary such that the angle at this point is minimal and the related two boundary edges are expansible. Start from one of the two expansible edges that has longer length, say $[V_1, V_2]$ which is also the edge of triangle $[V_0, V_1, V_2]$ with expansion point p_0 and P-expression $Z = \phi_{p_0,k}(X, Y)$. Choose one point q on P-plane outward the present triangle and within the (p_0, k, ϵ)*–circle* such that q is on the middle-perpendicular line of $[P(V_1), P(V_2)]$ and as far as possible from $P(p_0)$.

   (b) Refine the point $(q, \phi_{p_0,k}(q))$ by Newton method to get a new expansion point p_1. As before, if Newton method fails, a nearer point q to the circle center is used.

   (c) Compute the new P-expression $Z = \phi_{p_1,k}(X, Y)$ and new (p_1, k, ϵ)*–circle* $X^2 + Y^2 = r_1^2$.

(d) In the new P-plane at p_1, choose a point q_1 on the intersection of the the middle-perpendicular line of $[P(V_1), P(V_2)]$ and the new (p_1, k, ϵ)–*circle*. Then form a new triangle according to the following cases:

- If the line segment $[(P(V_1) + P(V_2))/2, q_1]$ intersect a previous edge's projection on the P-plane and the point $T^{-1}(q_1, \phi_{p_1}(q_1))$ lies on a previous surface patch, we take the intersection point to be q_1 and a new triangle $[V_1, V_2, T^{-1}(q_1, \phi_{p_1}(q_1))]$ is formed, or alternatively, if the vertices is near(within ϵ) to a singular point in S, then a new edge is added by connecting the vertices to the nearest singular point. In practice, this ϵ would be chosen interactively. Otherwise,
- Let $[V_2, V_4]$ be the other edge connecting to the present vertex. Then if the angle $< P(V_4)P(V_2)P(V_1) \leq \frac{\pi}{2}$, the new triangle is formed by the three points V_4, V_2, V_1.Otherwise,
- If angle $< P(V_4)P(V_2)P(V_1) > \frac{\pi}{2}$, the new triangle is $(V_1, V_2, P^{-1}(q_1, \phi_{p_1}(q_1))$.

3. *Final Step.* Use general step iteratively, until every edge is non-expansible. Then we finish the generation of the triangle polygon for one component of the surface.

Vertex Expansion Approach In this subsection, we describe another way for constructing the triangular polygon by expansion from vertex. The approach start from one initial vertex on the surface and then expand outward by using degree k power series expansions as a tool and a ϵ as a controller. We refer to it as (k, ϵ)–*Triangulation*. For easy of description of the algorithm, we first introduce some terminologies.

Expansible vertex. During the process of expansion of the triangular polygon, a vertex is called expansible if it is a smooth point of the surface $f = 0$, it is in the interior of the given boundary box and it is on the boundary of the present constructed polygon.

Binary partition process.

For a given point p on the surface, its (p, k, ϵ)–circle and two points q_1 and q_2 on the circle, let $p_i = (q_i, \phi_p(q_i))$. The *binary partition process* is to produce a series triangles by the following process.

a. If $\angle p_1 p p_2 \geq \pi$, then choose a point q in the middle of q_1 and q_2 and on the circle and define $p_i = (q, \phi_p(q))$. If the point p_i is outside the given bounding box, then adjust it to the boundary. Repeat this step till the angle is less than π.

b. If $\angle p_1 p p_2 \leq \pi$, then if p_2 is in (p_1, k, ϵ)–*sphere* and p_1 is in (p_2, k, ϵ)–*sphere*, then $[p_1, p_2]$ is a new edge and a new triangle $[p_1, p_2, p]$ is formed. Otherwise, a new point q' on the (p, k, ϵ)–*circle* and in the middle of q_1 and q_2 is taken and a new vertex $p' = (q', \phi_p(q'))$ is defined. If p' is outside the given bounding box, then adjust it to the boundary. This step is repeated untill (p, k, ϵ)–*sphere* of every vertex p contains its neighbor vertices. The connection of all the points whose projection on the edges with p forms the triangles

Now we can state the steps of expansion of polygons.
Algorithm 2

1. *Initial Step.* For a given smooth point p_0 on one component of the surface $f(x, y, z) = 0$, we first compute the P-expression $Z = \phi_{p_0,k}(X, Y)$. On the P-plane, then find the (p_0, k, ϵ)–*circle* with center $P(p_0) = (0, 0)$. Take three equally distributed points $q_i,\ i = 1, \cdots, 3$ on the circle and let $p_i = (q_i, \phi_{p_0,k}(q_i))$. For each pair of p_i and p_{i+1}, the *binary partition process* is conducted.

In this initial case, all the boundary vertices are expansible except the one that is not inside in the given box. Then compute the angle at each boundary vertex of the triangle that is defined by the adjacent edges counted outward.

2. *General Step.* Suppose we have constructed several space triangles that form one or more than one connected mesh. For each mesh, we keep the boundary information such as edges, vertices with angles. Assume now that at least one of the boundary vertices is expansible. Then the general step is to construct more triangles that join the original ones and enlarge the mesh. We shall try to keep mesh as convex as possible, so we always expand at the vertex that has sharpest angle.

   Find an expansible vertex, say p_0, on the present boundary such that the angle at p_0 is minimal. Let the two related boundary edges be $[p_1, p_0]$ and $[p_0, p_2]$, Compute P-expression $Z = \phi_{p_0,k}(X, Y)$ and (p_0, k, ϵ)–radius $r(p_0, k, \epsilon)$. Let $[q_1', q_0']$ and $[q_0', q_2']$ be the projection of $[p_1, p_0]$ and $[p_0, p_2]$ on the P-plane Let q_1 and q_2 be the intersection of rays $[q_0', q_1'\rangle$ and $[q_0', q_2'\rangle$ with (p_0, k, ϵ)–circle. Then perform the *binary partition process* for q_1 and q_2.

   - If the line segment $[q_i, q_0]$ intersect a previous edge's projection on the P-plane and the point $T^{-1}(q_i, \phi(q_i))$ lies on a previous surface patch, then the new vertex become no-expansible. Then local re-triangulation is needed.

3. *Final Step.* Use general step iteratively, until every vertex is non-expansible. Then we finish the generation of the triangle polygon for one component of the surface.

5 Operations on Spline Surfaces

5.1 Piecewise Parameterization of Surface Patches and their Trimming Curves

5.1.1 Tetrahedral patches

An A-patch of degree n over the tetrahedron $[\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3\mathbf{p}_4]$ is defined by

$$G_n(x, y, z) := F_n(\boldsymbol{\alpha}) = F_n(\alpha_1, \alpha_2, \alpha_3, \alpha_4) = 0, \tag{83}$$

where

$$F_n(\alpha_1, \alpha_2, \alpha_3, \alpha_4) = \sum_{i+j+k+l=n} a_{ijkl} B^n_{ijkl}(\alpha_1, \alpha_2, \alpha_3, \alpha_4), \tag{84}$$

$$B^n_{ijkl}(\alpha_1, \alpha_2, \alpha_3, \alpha_4) = \frac{n!}{i!j!k!l!}\alpha_1^i \alpha_2^j \alpha_3^k \alpha_4^l,$$

and $(x, y, z)^T$ and $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)^T$ are related by

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_3 & \mathbf{p}_4 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix}. \tag{85}$$

The construction details of cubic A-patches can be found in [15]. With all these computational formulas, there are still several degrees of freedom. Specifically, the weights a^m_{1110}, a^m_{1002}, a^m_{0102}, a^m_{0012}, a^m_{0003}, and b^m_{2001} may be chosen freely. We wish to use these degrees of freedom to make the cubic A-patch single-sheeted and have boundary curves that are rational parametric.

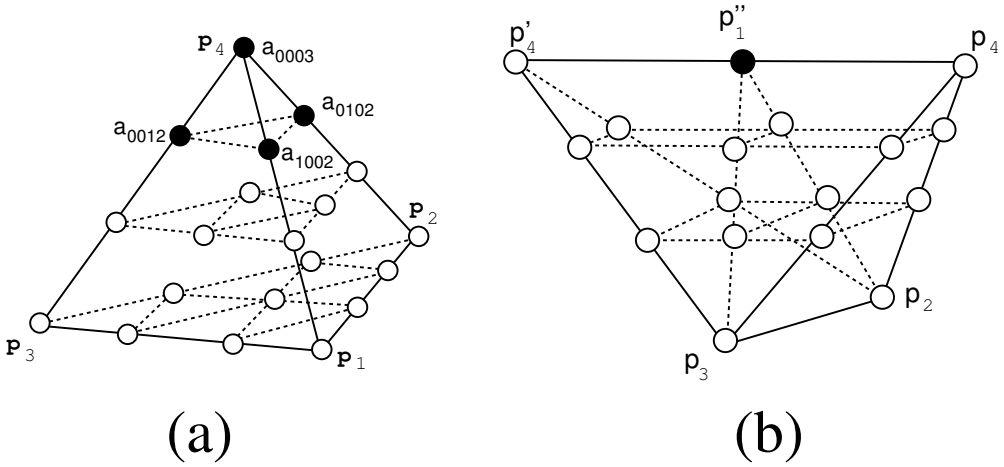

Figure 24: (a) Four free weights of a cubic A-patch for a face tetrahedron. (b) One free weight of a cubic A-patch for an edge tetrahedron.

5.1.2 Rational parametric boundary curves

For the face A-patch, we have four weights free [15] (see Figure 24(a)). These weights will be used to make the three boundary curves rationally parameterizable. Since forcing a C^1-continuous cubic A-spline to be rationally parameterizable requires the imposition of a single constraint (46), the splines on the three faces $[\mathbf{p}_1\mathbf{p}_2\mathbf{p}_4]$, $[\mathbf{p}_2\mathbf{p}_3\mathbf{p}_4]$, and $[\mathbf{p}_3\mathbf{p}_1\mathbf{p}_4]$ lead to three equations:

$$\begin{aligned} G_{[\mathbf{p}_1\mathbf{p}_2\mathbf{p}_4]}(a_{2100}, a_{1200}, a_{1101}, a_{1002}, a_{0102}, a_{0003}) &= 0 \\ G_{[\mathbf{p}_1\mathbf{p}_3\mathbf{p}_4]}(a_{1020}, a_{2010}, a_{1011}, a_{0012}, a_{1002}, a_{0003}) &= 0 \\ G_{[\mathbf{p}_2\mathbf{p}_3\mathbf{p}_4]}(a_{0210}, a_{0120}, a_{0111}, a_{0102}, a_{0012}, a_{0003}) &= 0 \ , \end{aligned} \tag{86}$$

where $G_{[\mathbf{p}_i\mathbf{p}_j\mathbf{p}_k]}(\cdot,\cdot,\cdot,\cdot,\cdot,\cdot)$ is defined by (46), and four unknowns $(a_{1002}, a_{0102}, a_{0012}, a_{0003})$.

For the edge patch, we have one weight free on the interface $[\mathbf{p}_2\mathbf{p}_3\mathbf{p}_1'']$ (see Figure 24(b)). If we let b_{ijkl} denote the weights for tetrahedron $[\mathbf{p}_1''\mathbf{p}_2\mathbf{p}_3\mathbf{p}_4]$, then the free weight is b_{3000}. Solving the equation $G_{[\mathbf{p}_2\mathbf{p}_3\mathbf{p}_1'']}(b_{0210}, b_{0120}, b_{1110}, b_{2100}, b_{2010}, b_{3000}) = 0$, provides the required coefficient.

If we are given two rationally parameterized curves on a cubic surface, we can obtain a rational parameterization for the surface in a manner similar to that in [33]. The idea is that a line that passes through two nonsingular real points on a cubic surface must intersect the surface in a third real point. Let the two curves on the surface $f(x, y, z) = 0$ be

$$\mathbf{c}_1(u) = [x_1(u) \ \ y_1(u) \ \ z_1(u)]^T \text{ and } \mathbf{c}_2(u) = [x_2(u) \ \ y_2(u) \ \ z_2(u)]^T \ .$$

Then the cubic parameterization formula for a point $\mathbf{p}(u, v)$ on the surface is

$$\mathbf{p}(u,v) = \begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{bmatrix} = \frac{a\mathbf{c}_1 + b\mathbf{c}_2}{a+b} = \frac{a(u,v)\mathbf{c}_1(u) + b(u,v)\mathbf{c}_2(v)}{a(u,v) + b(u,v)} \tag{87}$$

where

$$\begin{aligned} a &= a(u,v) = \nabla f(\mathbf{c}_2(v)) \cdot [\mathbf{c}_1(u) - \mathbf{c}_2(v)] \\ b &= b(u,v) = \nabla f(\mathbf{c}_1(u)) \cdot [\mathbf{c}_1(u) - \mathbf{c}_2(v)] \ . \end{aligned}$$

A simpler, lower degree parameterization can be obtained if we know and can use two skew lines on the cubic surface rather than cubic curves. This was the approach in [33], and results in a 1-to-1 covering of the cubic surface, while using cubic curves as $\mathbf{c}_1$ and $\mathbf{c}_2$ can result in a 9-to-1 covering. Nonsingular cubic surfaces can be put into five categories based on the number of real lines upon them, and rational parameterizations are possible in four of them.

5.1.3 Addition of a singular point

In this section we determine the free coefficients (dropping the superscript ^{1}) a_{1002}, a_{0102}, a_{0012}, and a_{0003} of tetrahedron $[\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3\mathbf{p}_4]$ by forcing the cubic surface to have a singular point at a specific location outside the tetrahedron, say at $\mathbf{p}_0 : (\alpha_1, \alpha_2, \alpha_3, \alpha_4) = (-k, -k, -k, 3k+1)$ for some $k \geq 0$. A singular point on the surface $S(\boldsymbol{\alpha}) = 0$ is one where the gradient vanishes, so that $\nabla S(\mathbf{p}_0) = 0$. Here S is considered to be a function of the three independent variables $\{\alpha_1, \alpha_2, \alpha_3\}$. The conditions that $S(-k, -k, -k) = 0$ and $(\partial S/\partial \alpha_i)_{\boldsymbol{\alpha}=\mathbf{p}_0} = 0$, $i = 1, 2, 3$, are equivalent to

$$\begin{aligned}
0 &= -3k^3(a_{2100} + a_{2010} + a_{1200} + 2a_{1110} + a_{1020} + a_{0210} + a_{0120}) \\
&\quad + 3k^2(3k+1)(a_{2001} + 2a_{1101} + 2a_{1011} + a_{0201} + 2a_{0111} + a_{0021}) \\
&\quad - 3k(3k+1)^2(a_{1002} + a_{0102} + a_{0012}) + (3k+1)^3 a_{0003}, \\
0 &= k^2(2a_{2100} + 2a_{2010} + a_{1200} + 2a_{1110} + a_{1020} - a_{0201} - 2a_{0111} - a_{0021}) \\
&\quad - k(7k+2)a_{2001} - 2k(4k+1)(a_{1101} + a_{1011}) + 2k(3k+1)(a_{0102} + a_{0012}) \\
&\quad + (5k+1)(3k+1)a_{1002} - (3k+1)^2 a_{0003}, \\
0 &= k^2(a_{2100} - a_{2001} + 2a_{1200} + 2a_{1110} - 2a_{1011} + 2a_{0210} + a_{0120} - a_{0021}) \\
&\quad - k(7k+2)a_{0201} - 2k(4k+1)(a_{1101} - a_{0111}) + 2k(3k+1)(a_{1002} + 8a_{0012}) \\
&\quad + (5k+1)(3k+1)a_{0102} - (3k+1)^2 a_{0003}, \\
0 &= k^2(a_{2010} - a_{2001} + 2a_{1110} - 2a_{1101} + 2a_{1020} + a_{0210} - a_{0201} + 2a_{0120}) \\
&\quad - k(7k+2)a_{0021} - 2k(4k+1)(a_{1011} + a_{0111}) + 2k(3k+1)(a_{1002} + 8a_{0102}) \\
&\quad + (5k+1)(3k+1) + a_{0012} - (3k+1)^2 a_{0003},
\end{aligned}$$

and this system has the solution

$$\begin{aligned}
a_{1002} &= k[-(2a_{2100} + 2a_{2010} + a_{1200} + 2a_{1110} + a_{1020})k \\
&\quad + 2(3k+1)(a_{2001} + a_{1101} + a_{1011})]/(3k+1)^2, \\
a_{0102} &= k[-(a_{2100} + 2a_{1200} + 2a_{1110} + 2a_{0210} + a_{0120})k \\
&\quad + 2(3k+1)(a_{1101} + a_{0201} + a_{0111})]/(3k+1)^2, \\
a_{0012} &= k[-(a_{2010} + 2a_{1110} + 2a_{1020} + a_{0210} + 2a_{0120})k \\
&\quad + 2(3k+1)(a_{1011} + a_{0111} + a_{0021})]/(3k+1)^2, \\
a_{0003} &= 3k^2[-2(a_{2100} + a_{2010} + a_{1200} + 2a_{1110} + a_{1020} + a_{0210} + a_{0120})k \\
&\quad + (3k+1)(a_{2001} + 2a_{1101} + 2a_{1011} + a_{0201} + 2a_{0111} + a_{0021})]/(3k+1)^3 \,. \qquad (88)
\end{aligned}$$

According to the inequality constraints in [15], a_{2100}, a_{2010}, a_{1200}, a_{1110}, a_{1020}, a_{0210} and a_{0120} are all negative, while a_{2001}, a_{0201}, and a_{0021} are all positive. The conditions that the cubic A-patch is single-sheeted are that a_{1002}, a_{0102}, a_{0012}, and a_{0003} must all be positive. This will be the case for $k > 0$ when $a_{1101} + a_{1011} > -a_{2001}$, $a_{1101} + a_{0111} > -a_{0201}$, $a_{1011} + a_{0111} > -a_{0021}$. These three conditions guarantee that a_{1002}, a_{0102}, and a_{0012} are positive, while combined they are equivalent to $a_{1101} + a_{1011} + a_{0111} > -(a_{2001} + a_{0102} + a_{0021})/2$, which guarantees that a_{0003} is positive. Even if these conditions is not satisfied, there may be values of k for which the solution for a_{0003} as given by (88) is positive. These conditions are more easily satisfied the more negative the quantities a_{2100}, a_{2010}, a_{1200}, a_{1110}, a_{1020}, a_{0210} and a_{0120} are.

Next, points on the cubic A-patch are parameterized by lines passing through the singular point and the plane determined by $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$. Lines passing through a singular point, or double point, intersect the cubic surface in exactly one more point. These lines have the form

$$L(t) = t(u\mathbf{p}_1 + v\mathbf{p}_2 + w\mathbf{p}_3) + (1-t)\mathbf{p}_0,$$

where $u + v + w = 1$. Thus we make the substitutions

$$\begin{aligned} \alpha_1 &= tu - (1-t)k, \quad \alpha_2 = tv - (1-t)k, \\ \alpha_3 &= tw - (1-t)k, \quad \alpha_4 = (1-t)(3k+1), \end{aligned} \tag{89}$$

and (88) into the cubic A-patch (84). This produces an equation which is linear in t. The region in the uv-plane over which the parameterization takes place can be described by $0 \le u \le 1$, $0 \le 1 - u \le v$. Let

$$\begin{aligned} P_2 = {} & [k(a_{2100} + a_{2010}) - (3k+1)a_{2001}]u^2 + 2[k(a_{2100} + a_{1200} + a_{1110}) - (3k+1)a_{1101}]uv \\ & + [k(a_{1200} + a_{0210}) - (3k+1)a_{0201}]v^2 + 2[k(a_{2010} + a_{1110} + a_{1020}) - (3k+1)a_{1011}]uw \\ & + [k(a_{1020} + a_{0120}) - (3k+1)a_{0021}]w^2 + 2[k(a_{1110} + a_{0210} + a_{0120}) - (3k+1)a_{0111}]vw \end{aligned}$$

and

$$\begin{aligned} P_3 = {} & a_{3000}u^3 + a_{2100}u^2v + a_{2010}u^2w + a_{1200}uv^2 + 2a_{1110}uvw \\ & + a_{1020}uw^2 + a_{0300}v^3 + a_{0210}v^2w + a_{0120}vw^2 + a_{0030}w^3, \end{aligned} \tag{90}$$

so that P_2 and P_3 consist of quadratic and cubic terms in $\{u, v, w\}$, respectively. Then t satisfies

$$t = \frac{P_2}{P_2 + P_3}, \quad \text{and} \quad 1 - t = \frac{P_3}{P_2 + P_3}. \tag{91}$$

Now considering (89), each of α_1, α_2, α_3, and α_4 is seen to be a quotient of cubic polynomials in u, v, and w. Writing $w = 1 - u - v$, each of the α is seen to be a function of two independent variables.

Of particular interest is the situation when $k = 0$, for in that case the cubic splines which are the intersections of the cubic A-patch with the side faces of the tetrahedron are immediately parameterizable. Equations (89) with $w = 0$, $v = 0$, and $u = 0$ will parameterize the faces where $\alpha_3 = 0$, $\alpha_2 = 0$, and $\alpha_1 = 0$, respectively. In order for this to work, $\mathbf{p}_4$ must be chosen sufficiently far from $[\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3]$. In this case, we have

$$P_2 = -(a_{2001}u^2 + 2a_{1101}uv + 2a_{1011}uw + a_{0201}v^2 + 2a_{0111}vw + a_{0021}w^2). \tag{92}$$

A sufficient condition that the A-patch is single-sheeted in this case is for the denominator in (91) to always have the same sign, say negative, and this can be guaranteed if the coefficients $\{a_{2001}, a_{1101}, a_{1011}, a_{0201}, a_{0111}, a_{0021}\}$ are all positive while $\{a_{2100}, a_{2010}, a_{1200}, a_{1110}, a_{1020}, a_{0210}, a_{0120}\}$ are all negative.

5.1.4 Parameterizing the base triangle in the non-convex case

If the triangle $[\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3]$ is non-convex, or is convex but not all of its neighbors are convex with the same sign, we are in the non-convex case, and the cubic A-patch intersects $[\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3]$ in a cubic curve $C(\alpha_1, \alpha_2, \alpha_3) = \sum_{i+j+k=3}[3!/(i!j!k!)]a_{ijk0}\alpha_1^i\alpha_2^j\alpha_3^k = 0$, where $a_{3000} = a_{0300} = a_{0030} = 0$. Let

$$(d, e, f, g, h, i, j) = (3a_{2100}, 3a_{2010}, 3a_{0210}, 3a_{1200}, 3a_{1020}, 3a_{0120}, 6a_{1110}).$$

Then each of $\{d, e, f, g, h, i\}$ is determined, but we still have one degree of freedom left in the coefficients j. This degree of freedom can sometimes be used to make $C(\alpha_1, \alpha_2, \alpha_3)$ rationally parameterizable.

As triangle $[\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3]$ lies on the plane $\alpha_4 = 0$, it can be regarded as a function of two variables, say x and y, where $(\alpha_1, \alpha_2, \alpha_3) = (x, y, 1 - x - y)$. An irreducible plane cubic curve $F(x, y) = 0$ is singular if it has a double point, that is, a point (x_0, y_0) where $F(x_0, y_0) = F_x(x_0, y_0) =$

$F_y(x_0, y_0) = 0$. By taking resultants of these polynomials and eliminating x_0 and y_0, we obtain this polynomial whose vanishing guarantees either the existence of a double point on $C(\alpha_1, \alpha_2, \alpha_3)$ or that $C(\alpha_1, \alpha_2, \alpha_3)$ is reducible:

$$\begin{aligned} H(j) = {} & t_1\, j^6 - t_2 t_3\, j^5 + [t_3^2 + t_4(t_2^2 - 12t_1)]j^4 + t_2(8t_3t_4 - t_2^2 + 36t_1)j^3 \\ & + [8(6t_1 - t_2^2)t_4^2 - 8t_3^2 t_4 - 6(12t_1 + 5t_2^2)t_3]j^2 \\ & - 4t_2[4t_3t_4^2 + 9(4t_1 - t_2^2)t_4 - 24t_3^2]j \\ & + 8(2t_4^2 - 9t_3)t_4(t_2^2 - 4t_1) + 16t_3^2t_4^2 - 64t_3^3 - 27(t_2^2 - 4t_1)^2, \end{aligned} \tag{93}$$

where

$$\begin{aligned} t_1 &= defghi, & t_2 &= dfh + egi, \\ t_3 &= defi + dghi + efgh, & t_4 &= di + ef + gh. \end{aligned}$$

If this $H(j) = 0$ has real solutions, then $C(\alpha_1, \alpha_2, \alpha_3)$ is singular and can be rationally parameterized. If all the solutions of $H(j) = 0$ are complex, then we use the approximate (within any given approximation error) parameterization method given in [31].

If $H(j) = 0$ is satisfied, then the following is the (3/3) rational parameterization, which is obtained by intersecting the curve with lines $(1-u)(y-y_0) = u(x-x_0)$ through the double point (x_0, y_0):

$$\begin{aligned} x &= \{[-2(e-h)x_0 - t_5y_0 - (e-2h)](1-u)^3 + [-t_5x_0 - 2t_6y_0 - (2h+2i-j)]u(1-u)^2 \\ &\quad + [-3(f-i)y_0 + (f-2i)]u^2(1-u) + (f-i)x_0u^3\}/D \\ y &= \{[-t_6x_0 - 2(f-i)y_0 - (f-2i)]u^3 + [-2t_5x_0 - t_6y_0 - (2h+2i-j)]u^2(1-u) \\ &\quad + [-3(e-h)x_0 + (e-2h)]u(1-u)^2 + (e-h)y_0(1-u)^3\}/D \end{aligned} \tag{94}$$

where

$$\begin{aligned} D &= (e-h)(1-u)^3 + t_5u(1-u)^2 + t_6u^2(1-u) + (f-i)u^3 \\ t_5 &= d - e + 2h + i - j \\ t_6 &= -f + g + h + 2i - j \ . \end{aligned}$$

The existence of the condition (93) also provides a method for finding the "best singular approximation" to a nonsingular cubic curve. Given a set $\{d_0, e_0, f_0, g_0, h_0, i_0, j_0\}$, one seeks the value of j, say j_1, nearest j_0 for which (93) is satisfied for the set $\{d_0, e_0, f_0, g_0, h_0, i_0, j_1\}$. All these curves intersect the lines $x = 0$, $y = 0$, and $x + y = 1$ in the same points, namely $(1, 0)$, $(0, 1)$, $(0, 0)$, $(-h/(e-h), 0)$, $(0, -i/(f-i))$, $(-g/(d-g), d/(d-g))$. As j changes continuously from j_0 to j_1, the topology of the cubic curve within the triangle can change only at the endpoint $j = j_1$, a value of j for which the cubic curve is singular. In particular, the same points of intersection with the sides will be connected by non-crossing arcs for all j strictly between j_0 and j_1.

5.1.5 Prism patches

Here we outline the construction steps of prism A-patches (we refer the interested reader to [105]).

Step 1. For each triangle $[\mathbf{p}_i\mathbf{p}_j\mathbf{p}_k]$, construct a prism D_{ijk} as (see Figure 25)

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

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

$$\mathbf{p}_{ijk}(b_1, b_2, b_3, \lambda) = b_1\mathbf{v}_i(\lambda) + b_2\mathbf{v}_j(\lambda) + b_3\mathbf{v}_k(\lambda), \quad b_1 + b_2 + b_3 = 1\ ,$$

and $\mathbf{v}_l(\lambda) = \mathbf{v}_l + \lambda\mathbf{n}_l$, $\mathbf{n}_l = \mathbf{N}_l/\|\mathbf{N}_l\|$, $l = i, j, k$; and I_{ijk} is a maximal open interval such that $0 \in I_{ijk}$ and for any $\lambda \in I_{ijk}$, the points $\mathbf{v}_i(\lambda)$, $\mathbf{v}_j(\lambda)$ and $\mathbf{v}_k(\lambda)$ are not collinear. For varying

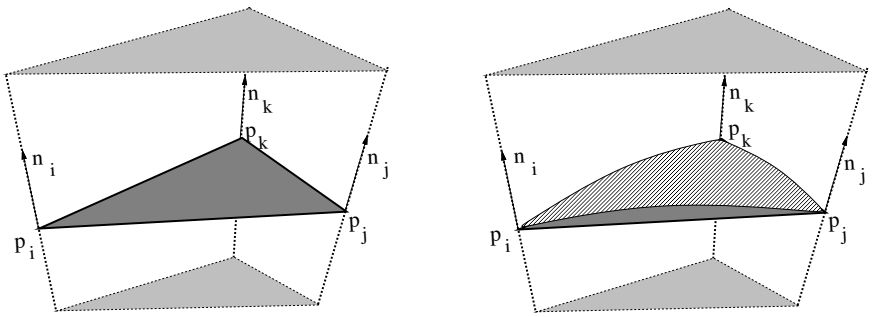

Figure 25: Each prism is defined over a triangle. Each patch is defined within a prism.

$\lambda \in I_{ijk}$, $\mathbf{v}_i(\lambda)$ $\mathbf{v}_j(\lambda)$ and $\mathbf{v}_k(\lambda)$ are the edges of the prism. A face of the prism is a ruled surface defined by two of its edges.

Step 2. Construct 3D function values and gradients on each of the three edges of the prism using the normal information. For instance, on the edge $\mathbf{v}_i(\lambda)$ of prism D_{ijk}, $F(\mathbf{v}_i(\lambda)) = \|\mathbf{N}_i\|\lambda$, $\nabla F(\mathbf{v}_i(\lambda)) = \mathbf{N}_i$ for $\lambda \in I_{ijk}$.

Step 3. Construct 3D function values and gradients on each of the three faces of the prism using the information on the edges that have been constructed in Step 2. Given two edges on the face being considered, the function on the face is defined by Hermite interpolation of the data on these edges. The gradient is similarly defined by linear interpolation of the gradients on the edges.

Step 4. Construct 3D function values within the prism using the information on the faces that have been constructed in Step 3. The approach used is transfinite interpolation of the data on the faces.

These are the basic steps of constructing F. Additional degrees of freedom may be introduced to improve the surface quality (see [105]) for more detail).

After these steps, function F is piecewise constructed, and the composite function is C^1 over the collection of the prisms (prism hull) and interpolates the C^1 vertex data (vertex positions and normals). The zero contour of F is a smooth surface that passes through the vertices and has the given normal at each vertex.

In each prism, $F := F(b_1, b_2, b_3, \lambda)$ depends upon the local barycentric coordinate (b_1, b_2, b_3) and λ. Hence, the zero contour is defined as follows: for each (b_1, b_2, b_3), find $\lambda := \lambda(b_1, b_2, b_3)$, such that $F(b_1, b_2, b_3, \lambda) = 0$. Then the surface point is defined by

$$P_{ijk}(b_1, b_2, b_3) = b_1\mathbf{v}_i(\lambda(b_1, b_2, b_3)) + b_2\mathbf{v}_j(\lambda(b_1, b_2, b_3)) + b_3\mathbf{v}_k(\lambda(b_1, b_2, b_3)) \ .$$

5.1.6 Approximation by Triangular Rational Bézier

From the construction of the function F, we know that the surface patch in each volume can be expressed in parametric form with $\Delta = \{(b_1, b_2, b_3) : \quad b_i \geq 0; \quad b_1 + b_2 + b_3 = 1\}$ as the parametric domain. However, this parametric form has no closed form representation. Hence, we appeal to rational Bézier form approximation instead of exact conversion. With the increase of the degree of rational Bézier, the error of the approximation will decrease. Let d be the degree of the rational Bézier patch, the approximant is obtained as follows:

1. Generate a degree $d-1$ functional Bézier form approximation $\tilde{\lambda}(b_1, b_2, b_3)$ of $\lambda(b_1, b_2, b_3)$. This is a classical polynomial approximation problem on a triangle. To obtain a global C^0 approximation, we generate this approximant in the following steps:

a. Compute Bézier coefficients of $\tilde{\lambda}(b_1, b_2, b_3)$ on the three boundaries by interpolating $\lambda(b_1, b_2, b_3)$.

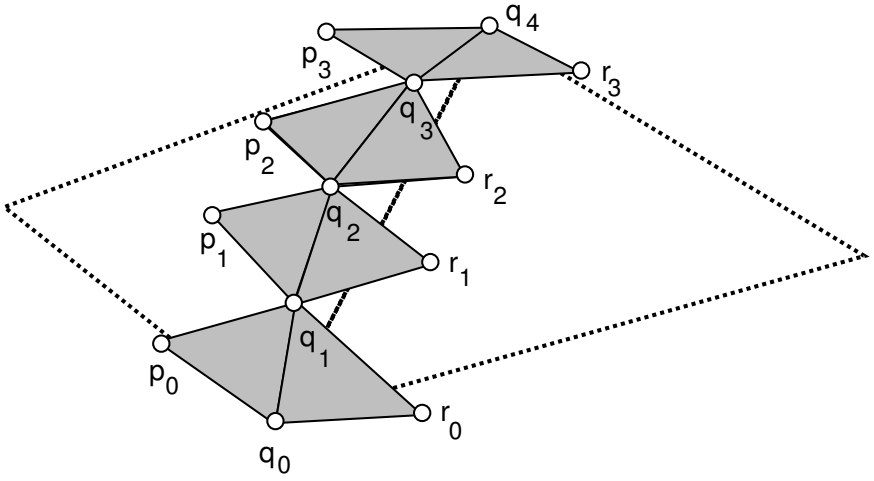

Figure 26: Related Bézier coefficients by G^1 continuity for $d = 4$.

b. The inner coefficients are defined by the least square fitting:

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

The integration above is computed on regularly subdivided triangles and on each sub-triangle, a 6-points numerical quadrature rule (see [34], page 35) is employed.

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

$$P^{(0)}_{ijk}(b_1, b_2, b_3) = b_1\mathbf{v}_i(\tilde{\lambda}(b_1, b_2, b_3)) + b_2\mathbf{v}_j(\tilde{\lambda}(b_1, b_2, b_3)) + b_3\mathbf{v}_k(\tilde{\lambda}(b_1, b_2, b_3))$$

The collection of $P^{(0)}_{ijk}$ define a continuous (not smooth) parametric surface.

3. Generate G^1 degree d parametric form Bézier coefficients. Let

$$P^{(0)}_{ijk}(b_1, b_2, b_3) = \sum_{l+m+n=d} \mathbf{b}^{(0)}_{lmn} B^d_{lmn}(b_1, b_2, b_3). \tag{95}$$

Then we adjust the coefficients $\mathbf{b}^{(0)}_{lmn}$ for $l \le 1$, $m \le 1$ and $n \le 1$ so that the G^1 continuous condition

$$\begin{aligned} &\frac{i}{d}\big[[\alpha_1\mathbf{p}_{i-1} + (1-\alpha_1)\mathbf{r}_{i-1}] - [\beta_1\mathbf{q}_{i-1} + (1-\beta_1)\mathbf{q}_i]\big] = \\ &-(1-\frac{i}{d})\big[[\alpha_0\mathbf{p}_i + (1-\alpha_0)\mathbf{r}_i] - [\beta_0\mathbf{q}_i + (1-\beta_0)\mathbf{q}_{i+1}]\big],\ i = 0, 1, \cdots, d\,, \end{aligned} \tag{96}$$

given by Farin in [53] (see pages 334–339), is satisfied. Using Farin's notation $\mathbf{q}_i$ represents the Bézier coefficients on the boundary (see Figure 26), and $\mathbf{r}_i$ and $\mathbf{p}_i$ represent the Bézier coefficients near the boundary coefficients on two adjacent triangles.

The condition (96) is applied as follows: For fixed $\mathbf{q}_0, \mathbf{q}_1, \mathbf{p}_0$, and $\mathbf{r}_0$, we solve α_0 and β_0 from the equation (96) for $i = 0$. Similarly, α_1 and β_1 are solved from (96) for $i = d$ and for fixed $\mathbf{q}_{d-1}, \mathbf{q}_d, \mathbf{p}_{d-1}$, and $\mathbf{r}_{d-1}$. After $\alpha_0, \alpha_1, \beta_0$ and β_1 are determined, the other equations ($i = 1, \cdots, d-1$) are used to solve for the other coefficients. For our problem, $\mathbf{q}_0$ and $\mathbf{q}_d$ are known, as they are the vertices. To ensure that equation (96) for $i = 0$ has a unique solution, we need to adjust the coefficients $\mathbf{q}_1, \mathbf{p}_0$ and $\mathbf{r}_0$ so that the four points $\mathbf{q}_0, \mathbf{q}_1, \mathbf{p}_0$ and $\mathbf{r}_0$ are coplanar. We

adjust these coefficients so that they lie on the boundaries of the prisms considered and the tangent plane defined by the normal $\mathbf{n}_0$ at $\mathbf{q}_0$. For example, $\mathbf{q}_1$ is adjusted as

$$\mathbf{q}_1 + \frac{t[(d-1)\mathbf{n}_0 + \mathbf{n}_d]}{d}, \qquad t = \frac{d\mathbf{n}_0^T(\mathbf{q}_0 - \mathbf{q}_1)}{d - 1 + \mathbf{n}_0^T\mathbf{n}_d},$$

where $\mathbf{n}_d$ is the given normal at $\mathbf{q}_d$. These adjustments need to be made to the coefficients $\mathbf{q}_{d-1}$, $\mathbf{p}_{d-1}$, and $\mathbf{r}_{d-1}$ as well.

For $d \geq 4$, the system (96) for $i = 1, \cdots, d-1$ is underdetermined. 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 strategy leads to a restricted least square problem

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

where X is a vector consisting of unknowns. B is the left-hand side. C consists of the corresponding known coefficients of $P_{ijk}^{(0)}$. We solve this problem by singular value decomposition of the matrix M. The details are omitted.

4. Produce the rational Bézier representation

$$P_{ijk}^{(1)}(b_1, b_2, b_3) = \frac{\sum\limits_{l+m+n=d+2} \mathbf{b}_{lmn} B_{lmn}^d(b_1, b_2, b_3)}{\sum\limits_{l+m+n=d+2} w_{lmn} B_{lmn}^d(b_1, b_2, b_3)}, \tag{97}$$

where $\mathbf{b}_{lmn} \in \mathbb{R}^3$, $w_{lmn} \in \mathbb{R}$. It should be noted that when solving (96) for each edge, the coefficients at the corner, that is, $\mathbf{p}_1$, $\mathbf{r}_1$, $\mathbf{p}_{d-2}$ and $\mathbf{r}_{d-2}$ (they are called twist term), are doubly determined. We take their average as the required value. However, such defined twist terms will destroy the G^1 continuity. To satisfy the G^1 condition (96), a rational function $R_d(b_1, b_2, b_3)$, that is given as follows, is added to the Bézier function (95):

$$\begin{aligned} R_d = \frac{1}{b_2b_3 + b_1b_3 + b_1b_2} \big[& (b_1b_3\mathbf{b}_{d-2,1,1}^{(1)} + b_1b_2\mathbf{b}_{d-2,1,1}^{(2)}) B_{d-2,1,1}^d(b_1, b_2, b_3) \\ & + (b_1b_2\mathbf{b}_{1,d-2,1}^{(2)} + b_2b_3\mathbf{b}_{1,d-2,1}^{(0)}) B_{1,d-2,1}^d(b_1, b_2, b_3) \\ & + (b_2b_3\mathbf{b}_{1,1,d-2}^{(0)} + b_1b_3\mathbf{b}_{1,1,d-2}^{(1)}) B_{1,1,d-2}^d(b_1, b_2, b_3) \big] \end{aligned} \tag{98}$$

where $\mathbf{b}_{d-2,1,1}^{(1)}$ is defined by the G^1 condition on edge 1 less the corresponding average value, and similarly for the other coefficients. The degree d Bézier form polynomial (95) plus the rational function (98) can be written in the rational form (97).

It should be noted that the vertices of the triangle are base points of the surface (97). To eliminate these base points, we perturb the denominator of (98) into $b_2b_3+b_1b_3+b_1b_2+\epsilon(b_1^2+b_2^2+b_3^2)$ and the numerator of (98) into

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

where $\epsilon > 0$ is a small number which is chosen so that the function value is not affected by this ϵ for fixed word-length computation. Hence, the perturbation has no influence on the patch in practice. Let $\tilde{R}_d(b_1, b_2, b_3)$ be the perturbed function of R_d. Let $\tilde{R}_d(b_1, b_2, b_3)$ be the perturbed function of R_d. Then it was found experimentally that

$$\|\tilde{R}_d(b_1, b_2, b_3) - R_d(b_1, b_2, b_3)\| \leq 0.0013\ \varepsilon\ \max|\mathbf{b}_{ijk}|$$

From this bound, we could determine an ε so that the right-hand sided of the above inequality is less than the chosen word-length accuracy.

Trimmed NURBs Patch Representation NURBs (Non Uniform Rational B-Splines) are bounded with trimming curves defined over the surface. We can define the trimming curves over the domain of the mapping to obtain the bounding curves in the new dimension for the NURBs surface.

Let the surface S be defined with parameters s, t. Then we can define rational parametric univariate functions to represent the trimming curves as $s = f_1/f_2$ and $t = f_3/f_4$

References

[1] S. Abhyankar. Desingularization of plane curves. *Singularities*, 40(Part 1):1, 1983.

[2] S. Abhyankar. Good points of a hypersurface. *Advances in Mathematics*, 68(2):87–256, 1988.

[3] 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.

[4] S. S. Abhyankar and C. L. Bajaj. Automatic parameterization of rational curves and surfaces III: algebraic plane curves. *Comput. Aided Geom. Des.*, 5(4):309–321, 1988.

[5] S. S. Abhyankar and C. L. Bajaj. Automatic parameterization of rational curves and surfaces IV: algebraic space curves. *ACM Trans. Graph.*, 8(4):325–334, 1989.

[6] A. Aho and J. Hopcroft. *Design & Analysis of Computer Algorithms*. Pearson Education India, 1974.

[7] E. Allgower and S. Gnutzmann. Simplicial pivoting for mesh generation of implicitly defined surfaces. *Computer Aided Geometric Design*, 8:305–325, 1991.

[8] D. Arnon. Topologically reliable display of algebraic curves. *ACM SIGGRAPH Computer Graphics*, 17(3):219–227, 1983.

[9] D. Arnon and J. Rauen. On the display of cell decompositions of algebraic surfaces. Technical report, XEROX Technical Report, 1988, 1988.

[10] C. Bajaj. Algorithmic implicitization of algebraic curves and surfaces. *CAPO Research Report CER-88-11, Department of Computer Science, Purdue University*, 1988.

[11] C. Bajaj. Geometric Modeling with Algebraic Surfaces. In D. Handscomb, editor, *The Mathematics of Surfaces III*, pages 3–48. Oxford Univ. Press, 1988.

[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, J. Canny, R. Garrity, and J. Warren. Factoring rational polynomials over the complexes. In *Proceedings of the ACM-SIGSAM 1989 international symposium on Symbolic and algebraic computation*, page 90. ACM, 1989.

[14] C. Bajaj, J. Chen, and G. Xu. Free form Surface Design with A-Patches. In *Proceedings of Graphics Interface'94*, pages 174–181, Banff, Canada., 1994.

[15] C. Bajaj, J. Chen, and G. Xu. *Modeling with Cubic A-Patches. ACM Transactions on Graphics*, 14, 2:103–133, 1995.

[16] C. Bajaj, T. Garrity, and J. Warren. On the applications of multi-equational resultants. *COMP TR88-83, Department of Computer Science, Rice University*, 1988.

[17] C. Bajaj, C. Hoffmann, R. Lynch, and J. Hopcroft. Tracing surface intersections. *Computer aided geometric design*, 5(4):285–307, 1988.

[18] C. Bajaj and I. Ihm. Algebraic surface design with hermite interpolation. *ACM Transactions on Graphics*, 11(1):61–91, 1992.

[19] C. Bajaj and I. Ihm. Algebraic surface design with Hermite interpolation. *ACM Transactions on Graphics*, 11(1):61–91, January 1992.

[20] C. Bajaj and I. Ihm. Lower Degree Approximation of Surfaces for Revolved Objects. Technical Report 92-084, Computer Sci., Purdue Univ., November 1992.

[21] C. Bajaj and M. Kim. Algorithms for planar geometric models. *Automata, Languages and Programming*, pages 67–81, 1988.

[22] C. Bajaj and M. Kim. Generation of configuration space obstacles: Moving algebraic surfaces. *The International Journal of Robotics Research*, 9(1):92, 1990.

[23] C. Bajaj and A. Royappa. A note on an efficient implementation of the Sylvester resultant for multivariate polynomials. *Computer Science Technical Report CSD-TR-718, Purdue University*, 1988.

[24] C. Bajaj and A. Royappa. The ganith algebraic geometry toolkit. In *Lecture Notes in Computer Science*, volume 429, pages 268–26, 1990.

[25] C. Bajaj and A. Royappa. *The GANITH Algebraic Geometry Toolkit.* In Proceedings of the First International Symposium on the Design and Implementation of Symbolic Computation Systems, pages 268–269. Springer-Verlag, *Lecture Notes in Computer Science No. 429*, 1990.

[26] C. Bajaj and A. Royappa. The Robust Display of Arbitrary Rational Parametric Surfaces. In *Curves and Surfaces in Computer Vision and Graphics III*, pages 70 – 80, Boston, MA, 1992.

[27] C. Bajaj and A. Royappa. Finite Representation of Parametric Curves and Surfaces. In *Proc. of IFIP TC 5/WG 5.10 II Conference on Geometric Modeling in Computer Graphics*, pages x–y, Genova, Italy, 1993.

[28] C. Bajaj and A. Royappa. Finite Representations of Real Parametric Curves and Surfaces. *International Journal of Computational Geometry and Applications*, 5(3):313–326, 1995.

[29] 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.

[30] C. Bajaj and G. Xu. Data fitting with cubic A-splines. In M. Gigante and T. Kunii, editors, *Proceedings of Computer Graphics International, CGI'94*, pages 329–340, Melbourne, Australia, 1996. World Scientific Publishing.

[31] C. Bajaj and G. Xu. Piecewise rational approximations of real algebraic curves. *J. Comp. Math.*, 15:1:55–71, 1997.

[32] C. Bajaj and G. Xu. Piecewise rational approximations of real algebraic curves. *J. Comp. Math.*, 15:1:55–71, 1997.

[33] C. L. Bajaj, R. J. Holt, and A. N. Netravali. Rational parametrizations of nonsingular real cubic surfaces. *ACM Transactions on Graphics*, 17(1):1–31, 1998.

[34] M. Bernadou and J. M. Boisserie. *The Finite Element Method in Thin Shell Theory: Application to Arch Dam Simulations*. Birkhäuser, 1982.

[35] J. Bloomenthal. Polygonization of implicit surfaces. *Computer Aided Geometric Design*, 5(4):341–355, 1988.

[36] W. Böhm, G. Farin, and J. Kahmann. A survey of curve and surface methods in CAGD. *Computer Aided Geometric Design*, 1:1–60, 1984.

[37] B. Buchberger. Gröbner-bases: An algorithmic method in polynomial ideal theory, 1985.

[38] J. Canny. The Complexity of Robot Motion Planning, 1988.

[39] E. Catmull. A subdivision algorithm for computer display of curved surfaces., 1974.

[40] B. Chazelle. Convex partitions of polyhedra: a lower bound and worst-case optimal algorithm. *SIAM Journal on Computing*, 13:488, 1984.

[41] E. Chionh. *Base points, resultants, and the implicit representation of rational surfaces*. PhD thesis, University of Waterloo Waterloo, Ont., Canada, Canada, 1990.

[42] J. Chuang and C. Hoffmann. On local implicit approximation and its applications. *ACM Transactions on Graphics (TOG)*, 8(4):324, 1989.

[43] J. H. Chuang. *Surface approximations in geometric modeling*. PhD thesis, Computer Science Department, Purdue University, 1990.

[44] C. Chui. *Multivariate Splines*. CBMS-NSF 54, SIAM Publications, 1988.

[45] W. Dahmen and C. Michelli. Recent progress in multivariate splines. In L. Schumaker, C. Chui and J. Ward, editor, *Approximation Theory IV*, pages 27–121. Academic Press, 1983.

[46] C. deBoor, K. Hollig, and S. Riemenscheider. *Box Splines*. Springer Verlag, 1993.

[47] T. DeRose. Rational Bézier curves and surfaces on projective domains. *NURBS for Curve and Surface Design*, pages 1–14, 1991.

[48] M. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, 1976.

[49] D. Dobkin and D. Souvaine. Computational geometry in a curved world. *Algorithmica*, 5(1):421–457, 1990.

[50] D. Dobkin, D. Souvaine, and C. Van Wyk. Decomposition and intersection of simple splinegons. *Algorithmica*, 3(1):473–485, 1988.

[51] J. Fairfield. Contoured shape generation: Forms that people see in dot patterns. In *Proc. IEEE Conf. on Cybernetics and Society*, pages 60–64, 1979.

[52] G. Farin. Triangular Bernstein-Bézier Patches. *Computer Aided Geometric Design*, 3:83–127, 1986.

[53] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide, Second Edition*. Academic Press Inc., 1990.

[54] G. Farin. *NURBS for curve and surface design*. Society for Industrial and Applied Mathematics Philadelphia, PA, USA, 1991.

[55] J. Foley and A. Van Dam. *Fundamentals of interactive computer graphics*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1982.

[56] X. Gao and S. Chou. Implicitization of rational parametric equations* 1. *Journal of Symbolic Computation*, 14(5):459–470, 1992.

[57] A. Geisow. *Surface interrogations*. PhD thesis, University of East Anglia, 1983.

[58] R. Graham and F. Frances Yao. Finding the convex hull of a simple polygon. *Journal of Algorithms*, 4(4):324–331, 1983.

[59] B. Guo. *Modeling Arbitrary Smooth Objects with Algebraic Surfaces*. PhD thesis, Computer Science, Cornell University, 1991.

[60] M. Hall and J. Warren. Adaptive polygonization of implicitly de ned surfaces. *IEEE Computer Graphics and Applications*, 10(5):33–42, 1990.

[61] R. Hartshorne. *Algebraic geometry*. springer Verlag, 1977.

[62] G. Herron. A characterization of certain C^1 discrete triangular interpolation. *SIAM J. Numer. Anal.*, 22(4):811–819, 1985.

[63] G. Herron. Smooth closed surfaces with discrete triangular interpolants. *Computer Aided Geometric Design*, 2:297–306, 1985.

[64] K. Hollig. Multivariate splines. *SIAM Journal on Numerical Analysis*, 19:1013–1031, 1982.

[65] B. Horn. *Robot vision*. McGraw-Hill Higher Education, 1986.

[66] J. Johnstone. *The Sorting of points along an algebraic curve*. PhD thesis, Cornell University, 1987.

[67] E. Kaltofen. Factorization of polynomials. *Computing supplementum*, 4:95–113, 1982.

[68] M. Kosters. Curvature-dependent parameterization of curves and surfaces. *Computer-Aided Design*, 23(8):569–578, 1991.

[69] J. Lane and L. Carpenter. A generalized scan line algorithm for the computer display of parametrically defined surfaces. *Computer Graphics and Image Processing*, 11(3):290–297, 1979.

[70] D. Lee and F. Preparata. Computational Geometry – A Survey. *IEEE Transactions on Computers*, pages 1072–1101, 1984.

[71] F. Macaulay. Some formulae in elimination. *Proceedings of the London Mathematical Society*, 1(1):3, 1902.

[72] F. Macaulay and P. Roberts. *The algebraic theory of modular systems*. Cambridge Univ Pr, 1994.

[73] D. Manocha and J. Canny. Efficient techniques for multipolynomial resultant algorithms. In *Proceedings of the 1991 international symposium on Symbolic and algebraic computation*, pages 86–95. ACM, 1991.

[74] D. Manocha and A. In. Algebraic and numeric techniques in modeling and robotics. *University of California at Berkeley, Berkeley, CA*, 1992.

[75] C. Micchelli and H. Prautzsch. Computing surfaces invariant under subdivision. *Computer Aided Geometric Design*, 4:321–328, 1987.

[76] D. Mount. On finding shortest paths on convex polyhedra, 1985.

[77] A. Nasri. Surface interpolation on irregular networks with normal conditions. *Computer Aided Geometric Design*, 8:89–96, 1991.

[78] M. Paluszny and R. Patterson. Tangent Continuous Cubic Algebraic Splines. To appear in ACM Transactions on Graphics, 1992.

[79] C. Petersen. Adaptive contouring of three-dimensional surfaces. *Computer Aided Geometric Design*, 1(1):61–74, 1984.

[80] V. Pratt. Techniques for conic splines. *sig85*, 19(3):151–159, 1985.

[81] F. Preparata and M. Shamos. *Computational geometry: an introduction*. Springer, 1985.

[82] A. Requicha. Representations of rigid solid objects. *Computer Aided Design Modelling, Systems Engineering, CAD-Systems*, pages 1–78, 1980.

[83] A. Rockwood, K. Heaton, and T. Davis. Real-time rendering of trimmed surfaces. *ACM SIGGRAPH Computer Graphics*, 23(3):116, 1989.

[84] A. Royappa. *Symbolic Methods in Computer Graphics and Geometric Modelling*. PhD thesis, Purdue University, 1992.

[85] G. Salmon. Modern higher algebra. *GE Stechert and Co., New York*, 1885.

[86] A. Sch
"affer and C. Van Wyk. Convex hulls of piecewise-smooth Jordan curves. *Journal of Algorithms*, 8(1):66–94, 1987.

[87] J. Schwartz and M. Sharir. On the piano movers problem: II. General techniques for computing topological properties of real algebraic manifolds. *Advances in applied Mathematics*, 4(1):298–351, 1983.

[88] D. Schweitzer and E. Cobb. Scanline rendering of parametric surfaces. *ACM SIGGRAPH Computer Graphics*, 16(3):265–271, 1982.

[89] T. Sederberg. Planar piecewise algebraic curves. *Computer Aided Geometric Design*, 1(3):241–255, 1984.

[90] T. Sederberg. Piecewise algebraic surface patches. *CAGD*, 2(1-3):53–59, 1985.

[91] T. Sederberg. Surfaces: Techniques for cubic algebraic surfaces. *IEEE Computer Graphics and Applications*, pages 14–25, 1990.

[92] T. Sederberg and J. Snively. Parameterization of Cubic Algebraic Surfaces. In R. Martin, editor, *The Mathematics of Surfaces II*, pages 299 – 319, 1987.

[93] Seidel, H-P. Polar Forms for Geometrically Continuous Spline Curves of Arbitrary Degree. *ACM Trans. on Graphics*, 12(1):1–34, 1993.

[94] J. Semple and L. Roth. *Introduction to algebraic geometry.* Clarendon Press, 1949.

[95] J. Semple and L. Roth. *Introduction to Algebraic Geometry.* Oxford University Press, Oxford, U.K., 1949.

[96] M. Shantz and S. Chang. Rendering trimmed NURBS with adaptive forward differencing. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, page 198. ACM, 1988.

[97] R. Tarjan and C. Van Wyk. An $O(n \log \log n)$-Time Algorithm for Triangulating a Simple Polygon. *SIAM Journal on Computing*, 17:143, 1988.

[98] G. Tindle. Tetrahedral triangulation. In *Proceedings on Mathematics of surfaces II*, page 394. Clarendon Press, 1987.

[99] G. T. Toussiant. Pattern recognition and geometrical complexity. In *5th International Conference on Patteren Recognition*, pages 1324–1347, Miami Beach, FL, 1980.

[100] B. Van der Waerden. *Modern algebra, vol. 2.* Ungar Publishing, 1950.

[101] B. Von Herzen. *Applications of surface networks to sampling problems in computer graphics.* PhD thesis, California Institute of Technology, 1989.

[102] R. Walker. *Algebraic Curves.* Springer Verlag, New York, 1978.

[103] J. Warren. Blending algebraic surfaces. *ACM Transactions on Graphics*, 8(4):263–278, 1989.

[104] J. Warren. Creating multisided rational Bézier surfaces using base points. *ACM Transactions on Graphics (TOG)*, 11(2):139, 1992.

[105] G. Xu and C. Bajaj. Adaptive model reconstruction by triangular A-patches, 2000. Unpublished manuscript.

[106] O. Zariski. *Algebraic surfaces.* Springer Verlag, 1995.

[107] Zariski, O. *Algebraic Surfaces.* Ergebnisse der Mathematik und ihre Grenzgebiete 4, 1935.