Chapter 15

TENSOR-PRODUCT SURFACES

The product of a column vector and a row vector is an example of the mathematical operation of tensor-product:

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} \otimes \begin{bmatrix} b_0 & b_1 & b_2 & b_3 \end{bmatrix} = \begin{bmatrix} a_0b_0 & a_0b_1 & a_0b_2 & a_0b_3 \\ a_1b_0 & a_1b_1 & a_1b_2 & a_1b_3 \\ a_2b_0 & a_2b_1 & a_2b_2 & a_2b_3 \end{bmatrix} \tag{15.1}$$

A *tensor-product surface* $\mathbf{P}(s,t)$ is one whose blending functions are products of pairs of univariate blending functions:

$$\mathbf{P}(s,t) = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} w_{ij}\mathbf{P}_{ij}B_i^m(s)B_j^n(t)}{\sum_{i=0}^{m}\sum_{j=0}^{n} w_{ij}B_i^m(s)B_j^n(t)} \tag{15.2}$$

where the $\mathbf{P}_{ij}$ are control points, w_{ij} are weights, and $B_i^m(s)$ and $B_j^n(t)$ are univariate blending functions. The control points in a tensor-product surface are organized topologically into a rectangular array, and the blending functions corresponding to the control points are likewise organized in an array similar to the one in (15.1).

Tensor-product surfaces can be constructed from any two types of univariate blending functions, and there is no requirement that the two sets of blending functions are of the same type or degree. If the univariate blending functions are Bernstein polynomials, the surface is a Bézier surface patch (see Section 15.1). If the blending functions are B-Splines, the surface is a NURBS surface (see Section 15.6).

15.1 Tensor-Product Bézier Surface Patches

For a Tensor-Product Bézier surface patch, m and n in (15.2) are the degrees of the respective Bernstein polynomials. If $m = n = 3$, the surface is called a "bi-cubic" patch Likewise, the case $m = n = 2$ is called "bi-quadratic," $m = n = 1$ is called "bi-linear," etc. If $m \neq n$, we normally just say that the surface is degree $m \times n$.

Figure 15.1.a shows a tensor-product Bézier surface patch of degree 2×3. Note that the $(m+1) \times (n+1)$ control points that are organized topologically into $m+1$ "rows" and $n+1$ "columns." The control points along with the blue lines connecting them in Figure 15.1.a is called the *control grid* or *control mesh* of the surface (analogous to the control polygon for a curve). The control points

for a bicubic patch are organized as follows:

$$\begin{bmatrix} \mathbf{P}_{03} & \mathbf{P}_{13} & \mathbf{P}_{23} & \mathbf{P}_{33} \\ \mathbf{P}_{02} & \mathbf{P}_{12} & \mathbf{P}_{22} & \mathbf{P}_{32} \\ \mathbf{P}_{01} & \mathbf{P}_{11} & \mathbf{P}_{21} & \mathbf{P}_{31} \\ \mathbf{P}_{00} & \mathbf{P}_{10} & \mathbf{P}_{20} & \mathbf{P}_{30} \end{bmatrix} \tag{15.3}$$

The corresponding blending functions are

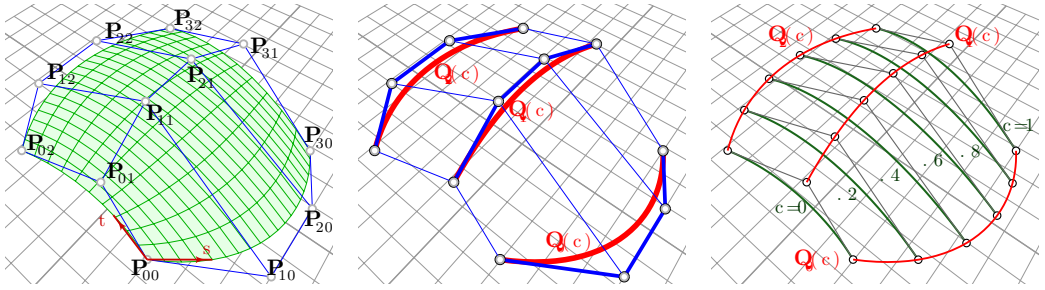
$$\begin{bmatrix} (1-s)^3 \cdot t^3 & 3s(1-s)^2 \cdot t^3 & 3s^2(1-s) \cdot t^3 & s^3 \cdot t^3 \\ (1-s)^3 \cdot 3t^2(1-t) & 3s(1-s)^2 \cdot 3t^2(1-t) & 3s^2(1-s) \cdot 3t^2(1-t) & s^3 \cdot 3t^2(1-t) \\ (1-s)^3 \cdot 3t(1-t)^2 & 3s(1-s)^2 \cdot 3t(1-t)^2 & 3s^2(1-s) \cdot 3t(1-t)^2 & s^3 \cdot 3t(1-t)^2 \\ (1-s)^3 \cdot (1-t)^3 & 3s(1-s)^2 \cdot (1-t)^3 & 3s^2(1-s) \cdot (1-t)^3 & s^3 \cdot (1-t)^3 \end{bmatrix} \tag{15.4}$$

Each row and column of control points can be interpreted as defining a curve. We will refer to the curve defined by control points $\mathbf{P}_{ij},\ i = 0, 1, \ldots m$ as the j^{th} s-control curve and the curve defined by control points $\mathbf{P}_{ij},\ j = 0, 1, \ldots n$ as the i^{th} t-control curve. Thus, there are $n+1$ s-control curves, and $m+1$ t-control curves. (The term "control curve" is not in standard use; it is a helpful concept, but is not used outside of these notes.)

If we fix one of the two parameters of the surface (for example, let $s = c$) then (15.2) reduces to

$$\mathbf{P}(c,t) = \sum_{j=0}^{n} \left[\sum_{i=0}^{m} \mathbf{P}_{ij} B_i^m(c) \right] B_j^n(t) = \sum_{j=0}^{n} \mathbf{Q}_j(c) B_j^n(t) \tag{15.5}$$

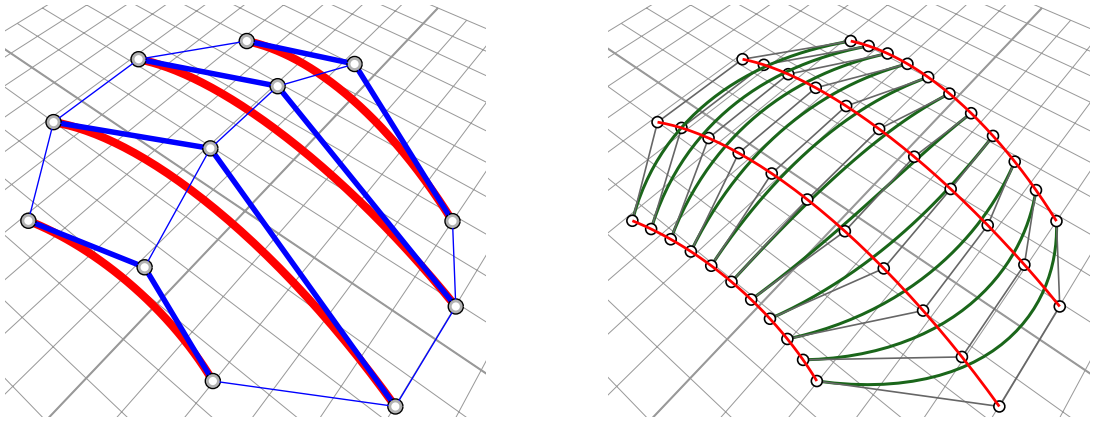
where $\mathbf{Q}_j(c) = \sum_{i=0}^{m} \mathbf{P}_{ij} B_i^m(c)$. $\mathbf{P}(c,t)$ is called an *s-iso-parameter* curve. *Iso* means *constant*, and an iso-parameter curve is a curve on a surface defined by holding one of the parameters constant.



(a) The surface and its control grid. (b) s-Control Curves. (c) Some s-iso-parameter curves $\mathbf{P}(c,t)$.

Figure 15.1: Bézier surface patch of degree 2×3.

A surface can be thought of as a family of iso-parameter curves, and the construction of those iso-parameter curves can be easily described in terms of control curves. Figure 15.1.b shows the s-control curves for the surface in Figure 15.1.a. Imagine that these control curves are wires, and imagine that a bead is free to slide along each wire. These beads serve as control points for the family of s-iso-parameter curves. That is, for a fixed value c, if we position each bead j at $\mathbf{Q}_j(c)$, they define the Bézier control point positions for s-iso-parameter curve $\mathbf{P}(c,t)$. Figure 15.1.c shows six such iso-parameter curves. Of course, $\mathbf{P}(s,t)$ can likewise be viewed as a family of t-iso-parameter curves defined by t-control curves, as illustrated in Figure 15.2.



(a) t-Control curves.

(b) Some t-iso-parameter curves $\mathbf{P}(s,c)$ for the surface in Figure 15.1.a.

Figure 15.2: Surface in Figure 15.1.a viewed as a family of t-iso-parameter curves.

In general, the only control points that lie on a Bézier surface patch are the corners:

$$\mathbf{P}(0,0) = \mathbf{P}_{00}, \quad \mathbf{P}(1,0) = \mathbf{P}_{m0}, \quad \mathbf{P}(0,1) = \mathbf{P}_{0n}, \quad \mathbf{P}(1,1) = \mathbf{P}_{mn}.$$

To evaluate the point $\mathbf{P}(\sigma,\tau)$, we could first evaluate the t-control curves at $t = \tau$, and then evaluate the t-iso-parameter curve at $s = \sigma$. Alternatively, we could first evaluate the s-control curves at $s = \sigma$, and then evaluate the s-iso-parameter curve at $t = \tau$.

Four special iso-parameter curves are the boundary curves $\mathbf{P}(0,t)$, $\mathbf{P}(1,t)$, $\mathbf{P}(s,0)$, and $\mathbf{P}(s,1)$. These boundary curves are Bézier curves defined by the corresponding boundary rows or columns of control points, and hence are control curves. For example, $\mathbf{P}(0,t)$ is the degree-three Bézier curve whose control points are $\mathbf{P}_{00}$, $\mathbf{P}_{10}$, $\mathbf{P}_{20}$, and $\mathbf{P}_{30}$. Likewise, $\mathbf{P}(s,1)$ is the degree-two Bézier curve whose control points are $\mathbf{P}_{30}$, $\mathbf{P}_{31}$, and $\mathbf{P}_{32}$.

15.2 The de Casteljau Algorithm for Bézier Surface Patches

Section 2.2 introduced the notation $\mathbf{P}_{[t_0,t_1]}(t)$ to mean a curve defined over the domain $t \in [t_0,t_1]$. In like manner, we will denote by

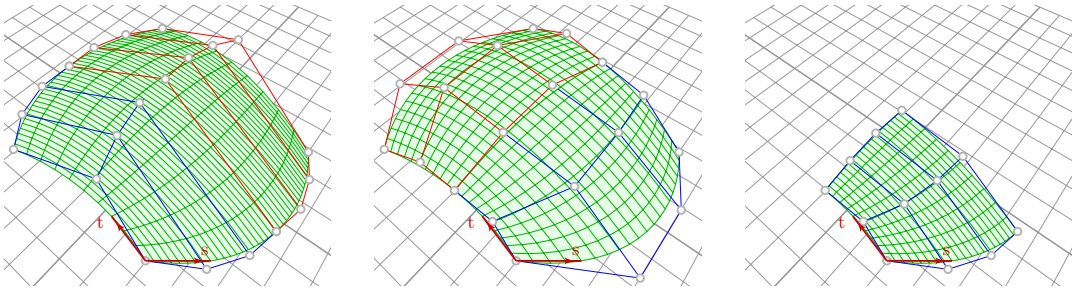
$$\mathbf{P}_{[s_0,s_1]\times[t_0,t_1]}(s,t)$$

the surface defined over the domain $s \in [s_0,s_1]$, $t \in [t_0,t_1]$.

The de Casteljau algorithm can be used to cut a Bézier surface patch $\mathbf{P}_{[s_0,s_1]\times[t_0,t_1]}(s,t)$ into two pieces, either in the s or t parameter directions. Cutting at $s = \sigma$ would produce $\mathbf{P}_{[s_0,\sigma]\times[t_0,t_1]}(s,t)$ and $\mathbf{P}_{[\sigma,s_1]\times[t_0,t_1]}(s,t)$. Cutting at $t = \tau$ would produce $\mathbf{P}_{[s_0,s_1]\times[t_0,\tau]}(s,t)$ and $\mathbf{P}_{[s_0,s_1]\times[\tau,t_1]}(s,t)$.

To subdivide a Bézier surface patch at $s = \sigma$, simply apply the de Casteljau algorithm for curves to subdivide each of the s-control curves at $s = \sigma$. To subdivide a Bézier surface patch at $t = \tau$, subdivide each of the t-control curves at $t = \tau$.

By repeated application of the de Casteljau algorithm, it is possible to obtain a patch over any sub-domain. For example, the patch $\mathbf{P}_{[0,\frac{1}{2}]\times[0,\frac{1}{2}]}$ in Figure 15.3.c can be obtained by splitting $\mathbf{P}_{[0,1]\times[0,1]}$ at $t = \frac{1}{2}$ to obtain $\mathbf{P}_{[0,1]\times[0,\frac{1}{2}]}$ and $\mathbf{P}_{[0,\frac{1}{2}]\times[\frac{1}{2},1]}$, and then splitting $\mathbf{P}_{[0,1]\times[0,\frac{1}{2}]}$ at $s = \frac{1}{2}$ to obtain $\mathbf{P}_{[0,\frac{1}{2}]\times[0,\frac{1}{2}]}$ and $\mathbf{P}_{[\frac{1}{2},1]\times[0,\frac{1}{2}]}$.



(a) Subdivision in the s direction.

(b) Subdivision in the t direction.

(c) $\mathbf{P}_{[0,\frac{1}{2}]\times[0,\frac{1}{2}]}$, obtained by subdividing at $s = \frac{1}{2}$ and then at $t = \frac{1}{2}$.

Figure 15.3: Applying the de Casteljau algorithm to the surface in Figure 15.1.a.

The idea of control curves also leads to a straightforward explanation of how to degree elevate a Bézier surface patch: simply degree elevate all of the s and/or t control curves.

15.3 Tangents and Normals

Tangent and normal vectors for a surface patch can be obtained by computing partial derivatives. As with Bézier curves, it is easiest to compute derivatives at a corner of a Bézier surface patch.

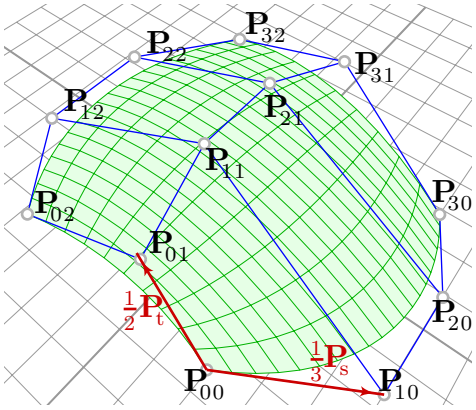


Figure 15.4: Partial derivative vectors for $\mathbf{P}_{[0,1]\times[0,1]}(s,t)$(assuming weights are unity).

For a rational Bézier surface patch $\mathbf{P}_{[s_0,s_1]\times[t_0,t_1]}(s,t)$, the partial derivatives at corner $\mathbf{P}(0,0)$ are

$$\mathbf{P}_s(s_0,t_0) = \frac{m}{s_1-s_0}\frac{w_{10}}{w_{00}}(\mathbf{P}_{10}-\mathbf{P}_{00}), \qquad \mathbf{P}_t(s_0,t_0) = \frac{n}{t_1-t_0}\frac{w_{01}}{w_{00}}(\mathbf{P}_{01}-\mathbf{P}_{00})$$

These partial derivatives are illustrated in Figure 15.4. The second partial derivatives are given by

$$\mathbf{P}_{ss}(s_0,t_0) = \frac{n(n-1)\frac{w_{20}}{w_{00}}(\mathbf{P}_{20}-\mathbf{P}_{00}) - 2n\frac{w_{10}}{w_{00}}\frac{nw_{10}-w_{00}}{w_{00}}(\mathbf{P}_{10}-\mathbf{P}_{00})}{(s_1-s_0)^2}$$

$$\mathbf{P}_{tt}(s_0,t_0)=\frac{n(n-1)\frac{w_{02}}{w_{00}}(\mathbf{P}_{02}-\mathbf{P}_{00})-2n\frac{w_{01}}{w_{00}}\frac{nw_{01}-w_{00}}{w_{00}}(\mathbf{P}_{01}-\mathbf{P}_{00})}{(t_1-t_0)^2}$$

Letting $\hat{\mathbf{P}}_{ij}=\mathbf{P}_{ij}-\mathbf{P}_{00}$,

$$\mathbf{P}_{st}(s_0,t_0)=\frac{mn}{w_{00}^2(s_1-s_0)(t_1-t_0)}(w_{00}w_{11}\hat{\mathbf{P}}_{11}-w_{10}w_{01}\hat{\mathbf{P}}_{10}-w_{10}w_{01}\hat{\mathbf{P}}_{01})$$

A vector that is perpendicular to the surface at $\mathbf{P}(0,0)$ can be obtained by taking the cross product $\mathbf{P}_s(0,0)\times\mathbf{P}_t(0,0)$. To find the normal vector at any other point on the surface, you can first perform de Casteljau subdivisions to move the desired point to a corner.

15.4 Tessellation of Bézier Curves and Surfaces

In computer graphics and geometric modeling, parametric curves and surfaces are often tessellated into piecewise linear segments for various applications, like rendering, mesh generation, and surface/surface intersection. A simple approach is to sample the curve or the surface evenly in its domain and then properly connect the sampling points to form a polyline or a triangular mesh. In this approach, the key point is to determine the sampling number. In the following we give formulae to compute the appropriate sampling number(s) such that the deviation of the approximate piecewise linear segments from the original curve or surface is within given tolerance ϵ.

15.4.1 The curve case

Section 10.6 derives the following result. Consider a degree n Bezier curve defined by

$$r(t)=\sum_{i=0}^{n}P_iB_i^n(t),\quad t\in[0,1].$$

If $L(t)$ is a straight line segment connecting $r(\alpha)$ and $r(\beta)$ where $0\le\alpha<\beta\le1$, then the deviation of $L(t)$ from $r(t)$ within domain $[\alpha,\beta]$ is bounded by (see Wang 1984, Filip et al 1986)

$$\max_{t\in[\alpha,\beta]}\|r(t)-L(t)\|\le\frac{1}{8}\max_{t\in[\alpha,\beta]}\|r''(t)\|(\beta-\alpha)^2.$$

This provides a way to compute the parameter interval (or step size) δ given a tolerance ϵ:

$$\delta\le\begin{cases}\sqrt{\dfrac{8\epsilon}{\max\limits_{t\in[\alpha,\beta]}\|r''(t)\|}}, & \text{if } \|r''(t)\|\neq0\\ 1, & \text{otherwise}\end{cases}$$

The sampling number n_t is the reciprocal of the parameter interval, i.e., $n_t=\lceil1/\delta\rceil$. That is, if the curve is evenly sampled at n_t+1 parameter values, the final polyline will not deviate from the curve greater than ϵ. In practice, the norm of the second derivative of $r(t)$ can be bounded by the Bezier control points:

$$\max_{t\in[\alpha,\beta]}\|r''(t)\|\le n(n-1)\max_{0\le i\le n-2}\|P_{i+2}-2P_{i+1}+P_i\|.$$

15.4.2 The surface case

For a surface $r(s,t)$, $(s,t) \in [0,1]\times[0,1]$, we choose a sub-domain $[\alpha,\beta]\times[\zeta,\eta]$ and construct a piecewise linear function $L(s,t)$ which consists of two triangles $r(\alpha,\beta)r(\zeta,\beta)r(\alpha,\eta)$ and $r(\zeta,\eta)r(\alpha,\eta)r(\zeta,\beta)$. It has been proved that

$$\max_{(s,t)\in[\alpha,\beta]\times[\zeta,\eta]} \|r(s,t) - L(s,t)\| \le \frac{1}{8}(M_1(\beta-\alpha)^2 + 2M_2(\beta-\alpha)(\eta-\zeta) + M_3(\eta-\zeta)^2)$$

where

$$\begin{aligned} M_1 &= \max_{(s,t)\in[\alpha,\beta]\times[\zeta,\eta]} \|r''_{ss}(s,t)\| \\ M_2 &= \max_{(s,t)\in[\alpha,\beta]\times[\zeta,\eta]} \|r''_{st}(s,t)\| \\ M_3 &= \max_{(s,t)\in[\alpha,\beta]\times[\zeta,\eta]} \|r''_{tt}(s,t)\|. \end{aligned}$$

Given tolerance ϵ, we would like to determine the parameter intervals δ_s and δ_t for $s-$ and $t-$directions such that

$$M_1\delta_s^2 + 2M_2\delta_s\delta_t + M_3\delta_t^2 \le 8\epsilon.$$

Case 1. If $M_1 = 0$, the surface is a ruled surface (linear in $s-$direction). Then we choose $\delta_s = 1$ and $\delta_t = \dfrac{\sqrt{M_2^2 + 8M_3\epsilon} - M_2}{M_3}$.

Case 2. If $M_3 = 0$, the surface is also a ruled surface (linear in $t-$direction). Then we choose $\delta_t = 1$ and $\delta_s = \dfrac{\sqrt{M_2^2 + 8M_1\epsilon} - M_2}{M_1}$.

Case 3. In general, the above inequality contains two unknowns δ_s and δ_t. We need to impose an additional constraint to solve for them. Suppose $\delta_s = k\delta_t$. Then $\delta_t = \sqrt{\dfrac{8\epsilon}{M_1k^2 + 2M_2k + M_3}}$. The default value for k is $k = \sqrt{M_3/M_1}$.

The above choice for the value of k is to minimize the number of resulting triangles. The reasoning is as follows. The sampling numbers n_s and n_t are the reciprocal of the parameter intervals δ_s and δ_t, i.e., $n_s = \lceil 1/\delta_s \rceil$ and $n_t = \lceil 1/\delta_t \rceil$. Thus the number of the triangles is $2n_sn_t$. So we have to choose k to maximize $\delta_s\delta_t$. Note that

$$\delta_s\delta_t = \frac{8\epsilon k}{M_1k^2 + 2M_2k + M_3}.$$

This requires us to minimize $M_1k + M_3/k$, which gives a unique solution $k = \sqrt{M_3/M_1}$. Now the surface is sampled evenly at $(n_s+1)(n_t+1)$ parameter values and the triangulation can be formed by properly connecting the sampling points. In this way, it is guaranteed that the final triangle mesh approximates the surface within the tolerance ϵ.

For a degree $m \times n$ Bezier patch

$$r(s,t) = \sum_{i=0}^{m}\sum_{j=0}^{n} P_{ij}B_i^m(s)B_j^n(t), \quad s,t \in [0,1]$$

bounds on the norms of its second derivatives are given by:

$$\begin{aligned} M_1 &\leq m(m-1) \max_{\substack{0\leq i\leq m-2\\ 0\leq j\leq n}} \|P_{i+2,j} - 2P_{i+1,j} + P_{ij}\| \\ M_2 &\leq mn \max_{\substack{0\leq i\leq m-1\\ 0\leq j\leq n-1}} \|P_{i+1,j+1} - P_{i+1,j} - P_{i,j+1} + P_{ij}\| \\ M_3 &\leq n(n-1) \max_{\substack{0\leq i\leq m\\ 0\leq j\leq n-2}} \|P_{i,j+2} - 2P_{i,j+1} + P_{ij}\|. \end{aligned} \tag{15.6}$$

Remark 1. When tessellating a few connecting patches, to avoid cracks, it is a good idea to compute sampling number for all four boundary curves independently of the sampling numbers for the patch interior (see Rockwood et al. 1989).

Remark 2. The above approach can also be applied to tessellating rational curves or surfaces. But the estimation of the second derivative bounds is very difficult. A simple and efficient way has been developed (see Zheng and Sederberg 2000).

Example
How many rows and columns of triangles are needed to tessellate the patch in Figure 15.5 using the fewest number of triangles such that the approximation error $\epsilon \leq .001$?

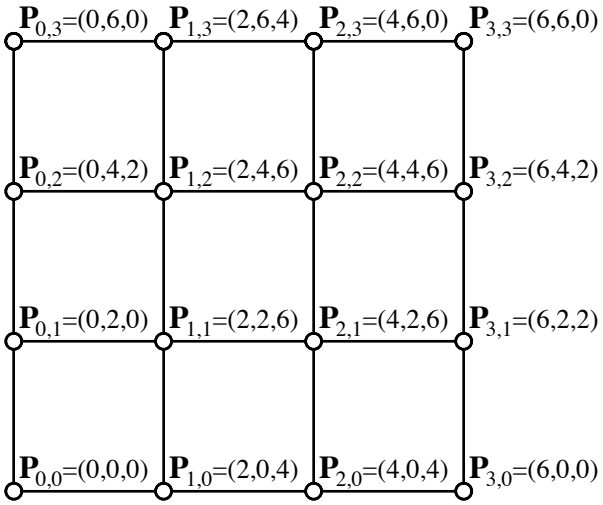


Figure 15.5: Surface Control Grid

Solution Using (15.6), compute bounds on second partial derivatives. In this example, the (x, y) coordinates lie in a rectangular lattice, which means that the the (x, y) components of all second partial derivatives are zero. The control points of the second hodographs for $\mathbf{P}_{ss}$, $\mathbf{P}_{st}$ and $\mathbf{P}_{tt}$ have

z-coordinates as follows:

$$\mathbf{P}_{ss}: \begin{bmatrix} -48 & 24 \\ -24 & -24 \\ -36 & -24 \\ -24 & -24 \end{bmatrix} \qquad \mathbf{P}_{st}: \begin{bmatrix} 0 & -36 & 36 \\ -18 & 0 & 0 \\ 18 & 0 & 0 \end{bmatrix} \qquad \mathbf{P}_{tt} = \begin{bmatrix} -24 & -12 & -36 & -12 \\ 12 & -12 & -12 & -12 \end{bmatrix}$$

from which $M_1 = 48$, $M_2 = 36$, and $M_3 = 36$. This is not a ruled surface, since $M_1, M_3 \neq 0$, so we proceed to use Case 3. Solve for $k = \sqrt{M_3/M_1} = \sqrt{3}/2$. Then

$$\delta_t = \sqrt{\frac{8\epsilon}{M_1 k^2 + 2M_2 k + M_3}} = \sqrt{\frac{8 * 0.001}{48 * (\sqrt{3}/2)^2 + 2 * 36 * \sqrt{3}/2 + 36}} = 0.00771649$$

and $\delta_s = k\delta_t = 0.00668268$. The number of rows and columns of triangles is then

$$n_s = \lceil \frac{1}{\delta_s} \rceil = 150; \qquad n_t = \lceil \frac{1}{\delta_t} \rceil = 130.$$

It should be noted that this equation for computing the number of triangles is conservative because the bound on the second derivative may not be tight, and because the second derivatives can vary dramatically across the patch. However, the equation does assure that if the specified number of triangles is used, the tolerance will be satisfied.

References

Wang, G.-Z. 1984. The subdivision method for finding the intersection between two Bezier curves or surfaces, *Zhejiang University Journal: Special Issue on Computational Geometry*, 108–119 (in Chinese).

Filip, D., Magedson,R. and Markot,R. 1986, Surface algorithms using bounds on derivatives, *Computer Aided Geometric Design* 3, 295–311.

Rockwood,A.,Heaton,K.,and Davis,T. 1989. Real-time rendering of trimmed surfaces. *SIGGRAPH Comput.Graph.* 23(3)(July 1989), 107-116.

Zheng, J. and Sederberg, T. 2000. Estimating tessellation parameter intervals for rational curves and surfaces. *ACM Transactions on Graphics* 19(1), 56–77.

15.5 C^n Surface Patches

As with Bézier curves, it is possible to piece together several individual Bézier surface patches to create a more complicated surface. Figure 15.6 shows a classic model from computer graphics, the Utah teapot. This teapot is modelled using 32 *bicubic* Bézier surface patches. For rendering, each patch is divided into a 9×9 grid of rectangles, and rectangles on the border of a patch are split into two triangles.

The conditions under which a pair of Bézier surface patches are C^n are analogous to the conditions for C^n continuity of a pair of Bézier curves. Two tensor-product surface patches are C^n if and only if every pair of adjacent control curves are C^n, as illustrated in Figure 15.7.

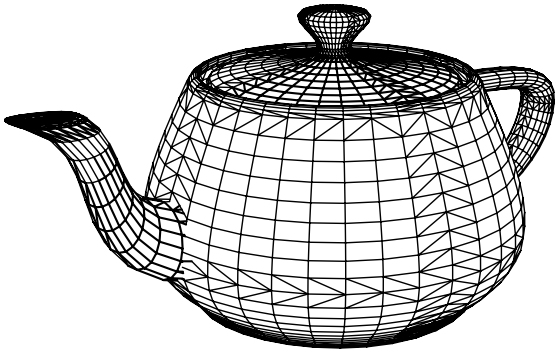
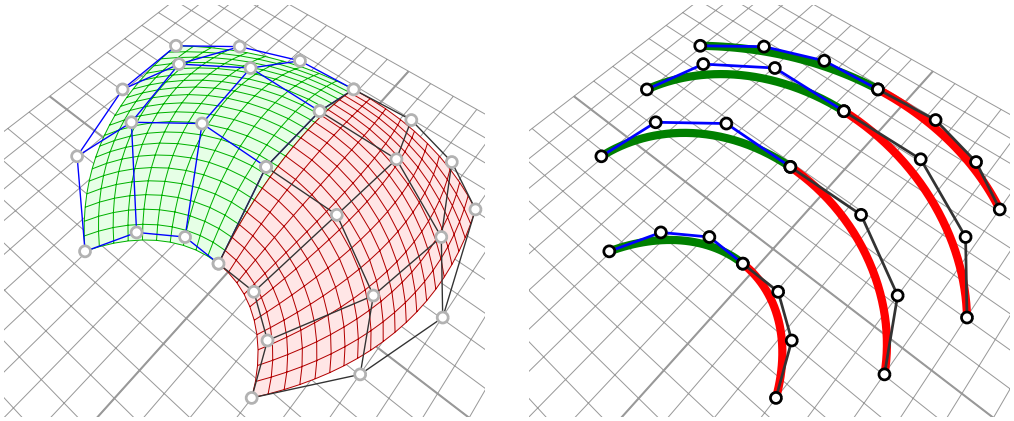


Figure 15.6: Teapot modeled using 32 bicubic Bézier surface patches



(a) Two bicubic Bézier surface patches. (b) Each matching pair of control curves is C^n.

Figure 15.7: Two C^n bicubic Bézier surface patches.

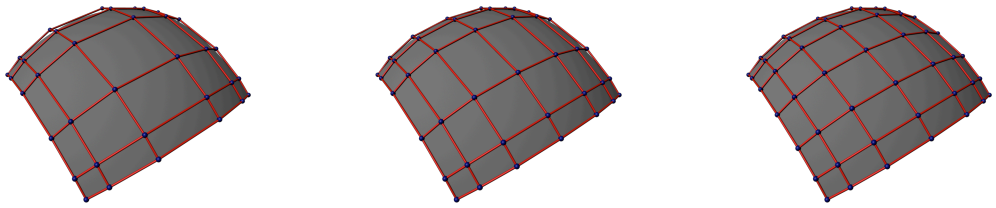
15.6 NURBS Surface

It is not too difficult to construct a Bézier patch that is C^n with a second Bézier patch. However, it is creating a network of Bézier patches that are C^n with each other is difficult for $n > 1$, because the two families of control curves (one in each parameter direction) must be C^n simultaneously. This problem does not exist with B-Spline surfaces, because the control curves in both directions are automatically C^n.

A tensor-product NURBS surface is one for which the blending functions in (15.2) are B-Spline blending functions. In order to define a NURBS surface, we must therefore specify a knot vector for the s blending functions, and a second knot vector for the t blending functions.

The operations on NURBS curves such as knot insertion, finding the Bézier curves that comprise a NURBS curve, etc., extend directly to NURBS surface: We simply operate on each of the control curves. Thus, to insert a knot into the s knot vector for a NURBS surface, we insert the knot into each s-control curve of the NURBS surface. Figure 15.9 shows a bicubic NURBS surface which first undergoes a knot insertion in the t direction, and then in the s direction.

To split a NURBS surface into Bézier surfaces, we simply split each s-control curve into Bézier



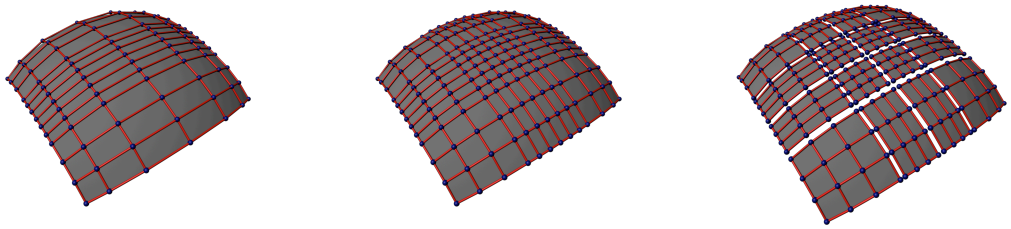
(a) A NURBS surface and its control grid.

(b) After inserting one knot in the t direction.

(c) After inserting one knot in the s direction.

Figure 15.8: Knot insertions into a NURBS surface.

curves, and then split each resulting t-control curve into Bézier curves. The most efficient way to evaluate a NURBS surface is to first split it into Bézier surfaces, then evaluate the Bézier surfaces.



(a) Inserting triple knots in the s direction.

(b) Inserting triple knots in the t direction.

(c) The Bézier surfaces, moved apart.

Figure 15.9: Splitting a NURBS surface into Bézier patches.

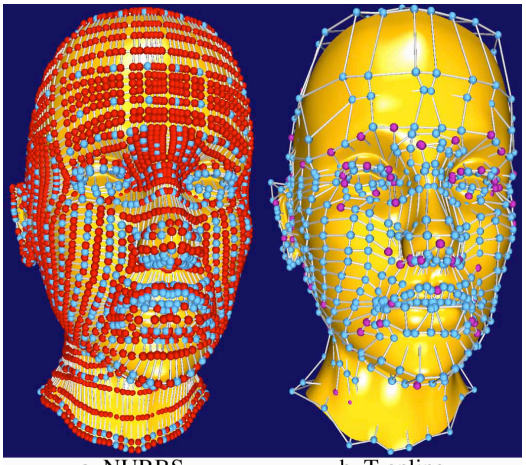
15.7 T-Splines

A serious weakness with NURBS surfaces is that NURBS control points must lie topologically in a rectangular grid. This means that typically, a large number of NURBS control points serve no purpose other than to satisfy topological constraints. They carry no significant geometric information. In Figure 15.10.a, all the red NURBS control points are, in this sense, superfluous.

T-splines are a generalization of NURBS surfaces that are capable of significantly reducing the number of superfluous control points. Figure 15.10.b shows a T-spline control grid which was obtained by eliminating the superfluous control points from the NURBS model. The main difference between a T-mesh (i.e., a T-spline control mesh) and a NURBS control mesh is that T-splines allow a row of control points to terminate. The final control point in a partial row is called a T-junction. The T-junctions are shown in purple in Figure 15.10.b.

Figure 15.11 shows another example in which the superfluous control points in a NURBS are removed to create a T-spline. The T-spline model is geometrically equivalent to the NURBS model, yet has only 1/3 as many control points.

Superfluous control points are a serious nuisance for designers, not merely because they require the designer to deal with more data, but also because they can introduce unwanted ripples in the surface as can be seen by comparing the forehead in the NURBS model in Figure 15.12.a with that



a. NURBS b. T-spline

Figure 15.10: Head modeled (a) as a NURBS with 4712 control points and (b) as a T-spline with 1109 control points. The red NURBS control points are superfluous.

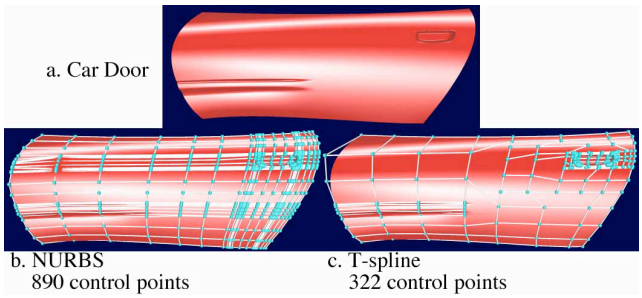
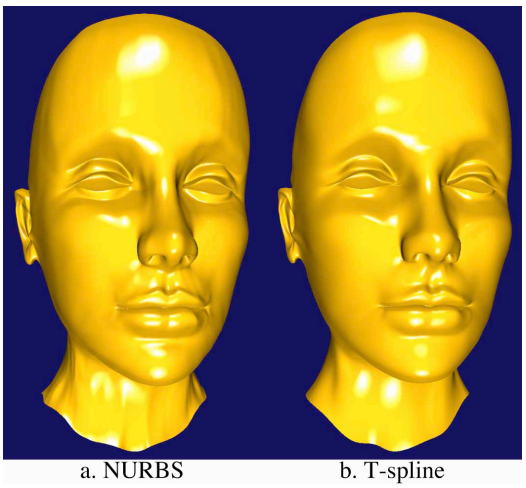


Figure 15.11: Car door modeled as a NURBS and as a T-spline.

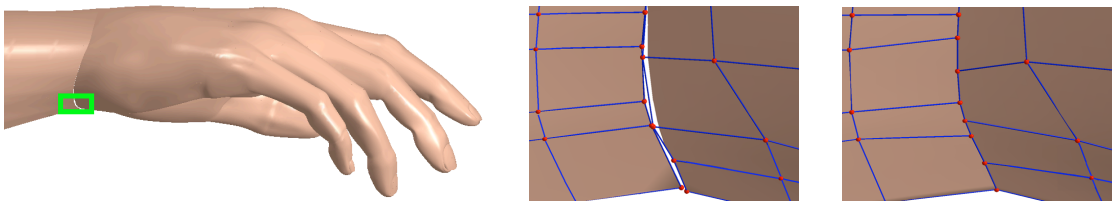


a. NURBS b. T-spline

Figure 15.12: NURBS head model, converted to a T-spline.

of the T-spline model in Figure 15.12.b. Designers can waste dozens of hours on models such as this in tweaking the NURBS control points while attempting to remove unwanted ripples. Figure 15.10.a shows a NURBS head model. Figure 15.10 shows the respective NURBS and T-spline control meshes for the surfaces in Figure 15.12. Over 3/4 of the 4712 NURBS control points are superfluous and do not appear in the T-spline control mesh.

T-splines can be used to merge non-uniform B-spline surfaces that have different knot-vectors. Figure 15.13.a shows a hand model comprised of seven B-spline surfaces. The small rectangular area is blown up in Figure 15.13.b to magnify a hole where neighboring B-spline surfaces do not match exactly. The presence of such gaps places a burden on designers, who potentially must repair a widened gap whenever the model is deformed. Figure 15.13.c shows the model after being converted into a gap-free T-spline, thereby eliminating the need for repair.



(a) Model of a hand comprised of B-spline surfaces. (b) NURBS surfaces. (c) T-Spline.

Figure 15.13: A gap between two B-spline surfaces, fixed with a T-spline.

While the notion of T-splines extends to any degree, we restrict our discussion to cubic T-splines. Cubic T-splines are C^2 in the absence of multiple knots.

15.7.1 Equation of a T-Spline

A control grid for a T-spline surface is called a T-mesh. If a T-mesh forms a rectangular grid, the T-spline degenerates to a B-spline surface.

A T-mesh is basically a rectangular grid that allows T-junctions. The pre-image of each edge in a T-mesh is a line segment of constant s (which we will call an s-edge) or of constant t (which we will call a t-edge). A T-junction is a vertex shared by one s-edge and two t-edges, or by one t-edge and two s-edges.

Knot information for T-splines is expressed using knot intervals, non-negative numbers that indicate the difference between two knots. A knot interval is assigned to each edge in the T-mesh. Figure 15.14 shows the pre-image of a portion of a T-mesh in (s,t) parameter space; the d_i and e_i denote the knot intervals. Knot intervals are constrained by the relationship that the sum of all knot intervals along one side of any face must equal the sum of the knot intervals on the opposing side. For example, in Figure 15.14 on face F_1, $e_3 + e_4 = e_6 + e_7$, and on face F_2, $d_6 + d_7 = d_9$.

The knot intervals must obey the following requirements:

1. The sum of knot intervals on opposing edges of any face must be equal. Thus, for face **F** in Figure 15.14, $d_2 + d_6 = d_7$ and $e_6 + e_7 = e_8 + e_9$.

2. If a T-junction on one edge of a face can be connected to a T-junction on an opposing edge of the face (thereby splitting the face into two faces) without violating Rule 1, that edge must be included in the T-mesh.

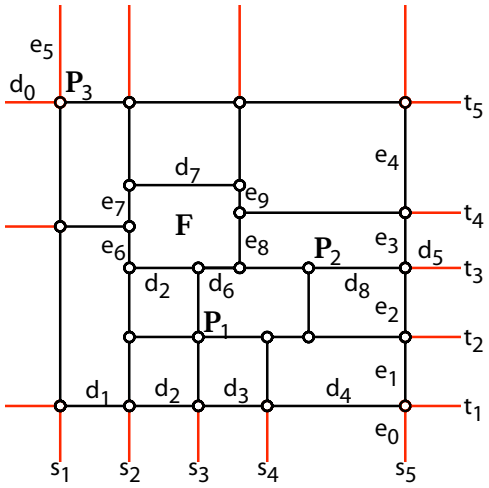


Figure 15.14: Pre-image of a T-mesh.

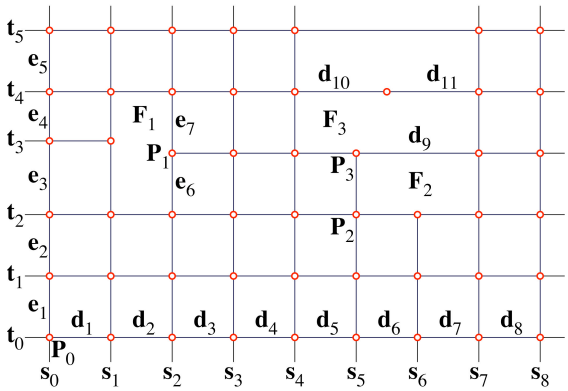


Figure 15.15: Pre-image of a T-mesh.

It is possible to infer a local knot coordinate system from the knot intervals on a T-mesh. To impose a knot coordinate system, we first choose a control point whose pre-image will serve as the origin for the parameter domain $(s,t) = (0,0)$. For the example in Figure 15.15, we designate (s_0, t_0) to be the knot origin.

Once a knot origin is chosen, we can assign an s knot value to each vertical edge in the T-mesh topology, and a t knot value to each horizontal edge in the T-mesh topology. In Figure 15.15, those knot values are labeled s_i and t_i. Based on our choice of knot origin, we have $s_0 = t_0 = 0$, $s_1 = d_1$, $s_2 = d_1 + d_2$, $s_3 = d_1 + d_2 + d_3$, $t_1 = e_1$, $t_2 = e_1 + e_2$, and so forth. Likewise, each control point has knot coordinates. For example, the knot coordinates for $\mathbf{P}_0$ are $(0,0)$, for $\mathbf{P}_1$ are $(s_2, t_2 + e_6)$, for $\mathbf{P}_2$ are (s_5, t_2), and for $\mathbf{P}_3$ are $(s_5, t_2 + e_6)$.

The knot coordinate system is used in writing an explicit formula for a T-spline surface:

$$\mathbf{P}(s,t) = \frac{\sum_{i=1}^{n} w_i \mathbf{P}_i B_i(s,t)}{\sum_{i=1}^{n} w_i B_i(s,t)}, \tag{15.7}$$

where $\mathbf{P}_i = (x_i, y_i, z_i, w_i)$ are control points in P^4 whose weights are w_i, and whose Cartesian coordinates are $\frac{1}{w_i}(x_i, y_i, z_i)$. Likewise, the Cartesian coordinates of points on the surface are given by

$$\frac{\sum_{i=1}^{n}(x_i, y_i, z_i)B_i(s,t)}{\sum_{i=1}^{n} w_i B_i(s,t)}. \tag{15.8}$$

The blending functions in (15.7) are $B_i(s,t)$ and are given by

$$B_i(s,t) = N[s_{i0}, s_{i1}, s_{i2}, s_{i3}, s_{i4}](s)N[t_{i0}, t_{i1}, t_{i2}, t_{i3}, t_{i4}](t) \tag{15.9}$$

where $N[s_{i0}, s_{i1}, s_{i2}, s_{i3}, s_{i4}](s)$ is the cubic B-spline basis function associated with the knot vector

$$\mathbf{s}_i = [s_{i0}, s_{i1}, s_{i2}, s_{i3}, s_{i4}] \tag{15.10}$$

and $N[t_{i0}, t_{i1}, t_{i2}, t_{i3}, t_{i4}](t)$ is associated with the knot vector

$$\mathbf{t}_i = [t_{i0}, t_{i1}, t_{i2}, t_{i3}, t_{i4}]. \tag{15.11}$$

as illustrated in Figure 15.16. The designer is free to adjust the weights w_i to obtain additional

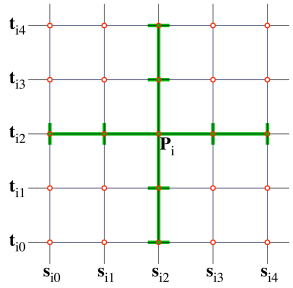


Figure 15.16: Knot lines for blending function $B_i(s,t)$.

shape control, as in rational B-splines. As we shall see in Section 15.7.2, weights also play a role in the local refinement algorithm.

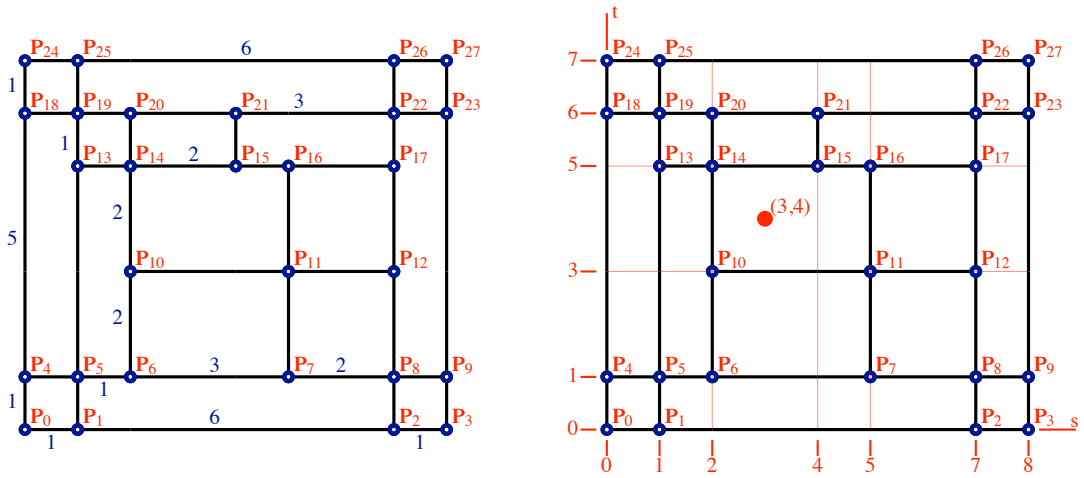
The T-spline equation is very similar to the equation for a tensor-product rational B-spline surface. The difference between the T-spline equation and a B-spline equation is in how the knot vectors $\mathbf{s}_i$ and $\mathbf{t}_i$ are determined for each blending function $B_i(s,t)$. Knot vectors $\mathbf{s}_i$ (15.10) and $\mathbf{t}_i$ (15.11) are *inferred from the T-mesh* neighborhood of $\mathbf{P}_i$. Since we will refer to the rule whereby the knot vectors are inferred, we formally state it as

Rule 1. Knot vectors $\mathbf{s}_i$ (15.10) and $\mathbf{t}_i$ (15.11) for the blending function of $\mathbf{P}_i$ are determined as follows. (s_{i2}, t_{i2}) are the knot coordinates of $\mathbf{P}_i$. Consider a ray in parameter space $\mathbf{R}(\alpha) = (s_{i2} + \alpha, t_{i2})$. Then s_{i3} and s_{i4} are the s coordinates of the first two s-edges intersected by the ray (not including the initial (s_{i2}, t_{i2})). By s-edge, we mean a vertical line segment of constant s. The other knots in $\mathbf{s}_i$ and $\mathbf{t}_i$ are found in like manner.

We illustrate Rule 1 by a few examples. The knot vectors for $\mathbf{P}_1$ in Figure 15.15 are $\mathbf{s}_1 = [s_0, s_1, s_2, s_3, s_4]$ and $\mathbf{t}_1 = [t_1, t_2, t_2 + e_6, t_4, t_5]$. For $\mathbf{P}_2$, $\mathbf{s}_2 = [s_3, s_4, s_5, s_6, s_7]$ and $\mathbf{t}_2 = [t_0, t_1, t_2, t_2 + e_6, t_4]$. For $\mathbf{P}_3$, $\mathbf{s}_3 = [s_3, s_4, s_5, s_7, s_8]$ and $\mathbf{t}_3 = [t_1, t_2, t_2 + e_6, t_4, t_5]$. Once these knot vectors are determined for each blending function, the T-spline is defined using (15.7) and (15.9).

Numerical Example

Figure 15.17.a shows a T-mesh in knot interval form. While it is possible to work entirely in knot interval form, our discussion is simplified if we convert to knot vector form. We do this by assigning the lower-left corner of the domain to have parameter coordinates $(s,t) = (0,0)$. Then, the knot lines are determined from the knot intervals, as shown in Figure 15.17.b.



(a) Model of a hand comprised of B-spline surfaces.

(b) NURBS surfaces.

Figure 15.17: Example T-Mesh.

From (15.7), the equation of the T-Spline shown in Figure 15.17.b is

$$\mathbf{P}(s,t) = \frac{\sum_{i=0}^{27} w_i \mathbf{P}_i B_i(s,t)}{\sum_{i=1}^{n} w_i B_i(s,t)}, \tag{15.12}$$

We now examine the equation $\mathbf{P}(3,4)$ for the T-Spline shown in Figure 15.17.b.

Recall from (15.9) that

$$B_i(3,4) = N[s_{i0}, s_{i1}, s_{i2}, s_{i3}, s_{i4}](3) N[t_{i0}, t_{i1}, t_{i2}, t_{i3}, t_{i4}](4) \tag{15.13}$$

For example, we have

$$B_{10}(3,4) = N[0,1,2,5,7](3) N[0,1,3,5,6](4), \quad B_{15}(3,4) = N[1,2,4,5,7](3) N[1,3,5,6,7](4),$$

$$B_0(3,4) = N[s_0,s_1,0,1,7](3) N[t_0,t_1,0,1,6](4), \quad B_{27}(3,4) = N[1,7,8,s_2,s_3](3) N[1,6,7,t_2,t_3](4).$$

We now show how to compute $B_{10}(3,4)$. The basis function $N[0,1,2,5,7](s)$ can be written as an explicit B-Spline curve with seven control points whose polar labels (with y-coordinates) are $f(s_0,s_1,0) = 0$, $f(s_1,0,1) = 0$, $f(0,1,2) = 0$, $f(1,2,5) = 1$, $f(2,5,7) = 0$, $f(5,7,s_2) = 0$, $f(7,s_2,s_3) = 0$. The knots s_0, s_1, s_2, s_3 are any real numbers that satisfy $s_0 \le s_1 \le 0$ and $7 \le s_2 \le s_3$. The value of $N[0,1,2,5,7](3)$ is the y-coordinate of the point whose polar label is $f(3,3,3)$. We can compute this as follows:

$$f(1,2,3) = \frac{2f(0,1,2) + 3f(1,2,5)}{5} = \frac{3}{5}; \quad f(2,3,5) = \frac{4f(1,2,5) + 2f(2,5,7)}{6} = \frac{2}{3}$$

$$f(3,5,7) = \frac{(s_2-3)f(2,5,7)+f(5,7,s_2)}{s_2-2} = 0; \quad f(2,3,3) = \frac{2f(1,2,3)+2f(2,3,5)}{4} = \frac{19}{30}$$

$$f(3,3,5) = \frac{4f(2,3,5)+f(3,5,7)}{5} = \frac{8}{15}; \quad f(3,3,3) = \frac{2f(2,3,3)+f(3,3,5)}{3} = \frac{3}{5} = N[0,1,2,4,5](3).$$

Likewise, we can compute $N[0,1,3,5,6](4) = \frac{23}{60}$. Thus, $B_{10}(3,4) = \frac{69}{300}$.

There are several blending functions that are zero at $s = 3$, $t = 4$. For example,

$$B_4(3,4) = N[s_0,s_1,0,1,2](3)N[t_1,0,1,3,5](4) = 0 * N[t_1,0,1,3,5](4) = 0.$$

Likewise,

$$B_2(3,4) = B_9(3,4) = B_{17}(3,4) = B_{18}(3,4) = B_{23}(3,4) = B_{25}(3,4) = B_{26}(3,4) = 0.$$

15.7.2 T-spline Local Refinement

The next three sections discuss an algorithm for local refinement of T-splines. Blending function refinement plays an important role in this algorithm, and is reviewed in Section 15.7.3. The notion of T-spline spaces is introduced in Section 15.7.4. This concept is used in the local refinement algorithm in Section 15.7.5.

15.7.3 Blending Function Refinement

If $\mathbf{s} = [s_0, s_1, s_2, s_3, s_4]$ is a knot vector and $\tilde{\mathbf{s}}$ is a knot vector with m knots with $\mathbf{s}$ a subsequence of $\tilde{\mathbf{s}}$, then $N[s_0, s_1, s_2, s_3, s_4](s)$ can be written as a linear combination of the $m-4$ B-spline basis functions defined over the substrings of length 5 in $\tilde{\mathbf{s}}$.

We now present all basis function refinement equations for the case $m = 6$. Equations for $m > 6$ can be found by repeated application of these equations.

If $\mathbf{s} = [s_0, s_1, s_2, s_3, s_4]$, $N(s) = N[s_0, s_1, s_2, s_3, s_4](s)$, and $\tilde{\mathbf{s}} = [s_0, k, s_1, s_2, s_3, s_4]$ then

$$N(s) = c_0 N[s_0, k, s_1, s_2, s_3](s) + d_0 N[k, s_1, s_2, s_3, s_4](s) \tag{15.14}$$

where $c_0 = \frac{k-s_0}{s_3-s_0}$ and $d_0 = 1$. If $\tilde{\mathbf{s}} = [s_0, s_1, k, s_2, s_3, s_4]$,

$$N(s) = c_1 N[s_0, s_1, k, s_2, s_3](s) + d_1 N[s_1, k, s_2, s_3, s_4](s) \tag{15.15}$$

where $c_1 = \frac{k-s_0}{s_3-s_0}$ and $d_1 = \frac{s_4-k}{s_4-s_1}$. If $\tilde{\mathbf{s}} = [s_0, s_1, s_2, k, s_3, s_4]$,

$$N(s) = c_2 N[s_0, s_1, s_2, k, s_3](s) + d_2 N[s_1, s_2, k, s_3, s_4](s) \tag{15.16}$$

where $c_2 = \frac{k-s_0}{s_3-s_0}$ and $d_2 = \frac{s_4-k}{s_4-s_1}$. If $\tilde{\mathbf{s}} = [s_0, s_1, s_2, s_3, k, s_4]$,

$$N(s) = c_3 N[s_0, s_1, s_2, s_3, k](s) + d_3 N[s_1, s_2, s_3, k, s_4](s) \tag{15.17}$$

where $c_3 = 1$ and $d_3 = \frac{s_4-k}{s_4-s_1}$. If $k \le s_0$ or $k \ge s_4$, $N(s)$ does not change.

A T-spline function $B(s,t)$ can undergo knot insertion in either s or t, thereby splitting it into two scaled blending functions that sum to the initial one. Further insertion into these resultant scaled blending functions yields a set of scaled blending functions that sum to the original. For example, Figure 15.18.a shows the knot vectors for a T-spline blending function B_1, and Figure 15.18.b shows a refinement of the knot vectors in Figure 15.18.a. By appropriate application of (15.14)—(15.17), we can obtain

$$B_1(s,t) = c_1^1 \tilde{B}_1(s,t) + c_1^2 \tilde{B}_2(s,t) + c_1^3 \tilde{B}_3(s,t) + c_1^4 \tilde{B}_4(s,t). \tag{15.18}$$

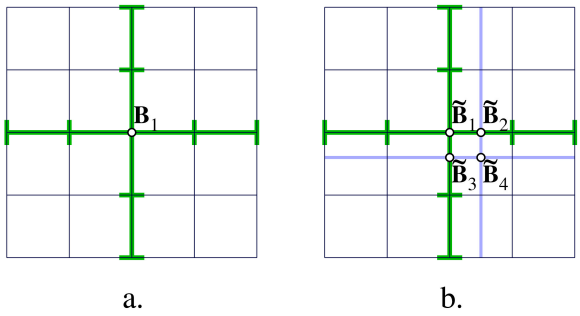


Figure 15.18: Sample Refinement of $B_1(s,t)$.

15.7.4 T-spline Spaces

We define a T-spline space to be the set of all T-splines that have the same T-mesh topology, knot intervals, and knot coordinate system. Thus, a T-spline space can be represented by the diagram of a pre-image of a T-mesh such as in Figure 15.14. Since all T-splines in a given T-spline space have the same pre-image, it is proper to speak of the pre-image of a T-spline space. A T-spline space S_1 is said to be a subspace of S_2 (denoted $S_1 \subset S_2$) if local refinement of a T-spline in S_1 will produce a T-spline in S_2 (discussed in Section 15.7.5). If T_1 is a T-spline, then $T_1 \in S_1$ means that T_1 has a control grid whose topology and knot intervals are specified by S_1.

Figure 15.19 illustrates a nested sequence of T-spline spaces, that is, $S_1 \subset S_2 \subset S_3 \subset \cdots \subset S_n$.

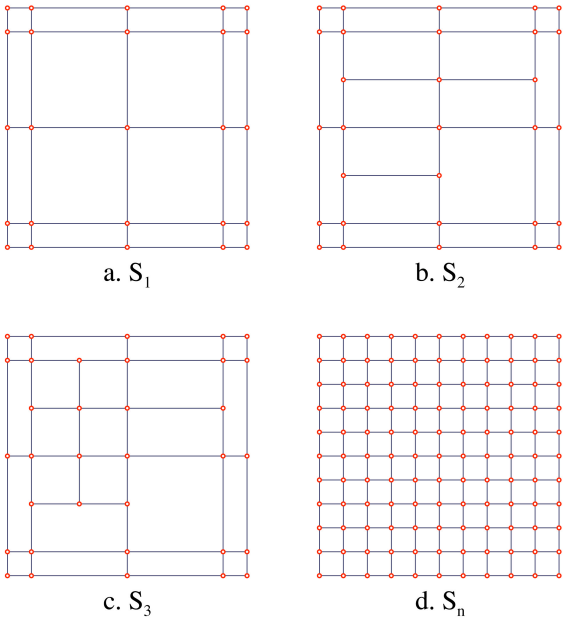


Figure 15.19: Nested sequence of T-spline spaces.

Given a T-spline $\mathbf{P}(s,t) \in S_1$, denote by $\mathbf{P}$ the column vector of control points for $\mathbf{P}(s,t)$, and

given a second T-spline $\tilde{\mathbf{P}}(s,t) \in S_2$, such that $\mathbf{P}(s,t) \equiv \tilde{\mathbf{P}}(s,t)$. Denote by $\tilde{\mathbf{P}}$ the column vector of control points for $\tilde{\mathbf{P}}(s,t)$. There exists a linear transformation that maps $\mathbf{P}$ into $\tilde{\mathbf{P}}$. We can denote the linear transformation

$$M_{1,2}\mathbf{P} = \tilde{\mathbf{P}}. \tag{15.19}$$

The matrix $M_{1,2}$ is found as follows.

$\mathbf{P}(s,t)$ is given by (15.7), and

$$\tilde{\mathbf{P}}(s,t) = \sum_{j=1}^{\tilde{n}} \tilde{\mathbf{P}}_j \tilde{B}_j(s,t) \tag{15.20}$$

Since $S_1 \subset S_2$, each $B_i(s,t)$ can be written as a linear combination of the $\tilde{B}_j(s,t)$:

$$B_i(s,t) = \sum_{j=1}^{\tilde{n}} c_i^j \tilde{B}_j(s,t). \tag{15.21}$$

We require that

$$\mathbf{P}(s,t) \equiv \tilde{\mathbf{P}}(s,t). \tag{15.22}$$

This is satisfied if

$$\tilde{\mathbf{P}}_j = \sum_{i=1}^{n} c_i^j \mathbf{P}_i. \tag{15.23}$$

Thus, the element at row j and column i of $M_{1,2}$ in (15.19) is c_i^j. In this manner, it is possible to find transformation matrices $M_{i,j}$ that maps any T-spline in S_i to an equivalent T-spline in S_j, assuming $S_i \subset S_j$.

The definition of a T-spline subspace $S_i \subset S_j$ means more than simply that the preimage of S_j has all of the control points that the preimage of S_i has.

15.7.5 Local Refinement Algorithm

T-spline local refinement means to insert one or more control points into a T-mesh without changing the shape of the T-spline surface. This procedure can also be called local knot insertion, since the addition of control points to a T-mesh must be accompanied by knots inserted into neighboring blending functions.

The refinement algorithm we now present has two phases: the topology phase, and the geometry phase. The topology phase identifies which (if any) control points must be inserted in addition to the ones requested. Once all required new control points are identified, the Cartesian coordinates and weights for the refined T-mesh are computed using the linear transformation presented in Section 15.7.4. We now explain the topology phase of the algorithm.

An important key to understanding this discussion is to keep in mind how in a T-spline, the blending functions and T-mesh are tightly coupled: To every control point there corresponds a blending function, and each blending function's knot vectors are defined by Rule 1. In our discussion, we temporarily decouple the blending functions from the T-mesh. This means that during the flow of the algorithm, we temporarily permit the existence of blending functions that violate Rule 1, and control points to which no blending functions are attached.

Our discussion distinguishes three possible violations that can occur during the course of the refinement algorithm:

- **Violation 1** A blending function is missing a knot dictated by Rule 1 for the current T-mesh.

- **Violation 2** A blending function has a knot that is not dictated by Rule 1 for the current T-mesh.

- **Violation 3** A control point has no blending function associated with it.

If no violations exist, the T-spline is valid. If violations do exist, the algorithm resolves them one by one until no further violations exist. Then a valid superspace has been found.

The topology phase of our local refinement algorithm consists of these steps:

1. Insert all desired control points into the T-mesh.

2. If any blending function is guilty of Violation 1, perform the necessary knot insertions into that blending function.

3. If any blending function is guilty of Violation 2, add an appropriate control point into the T-mesh.

4. Repeat Steps 2 and 3 until there are no more violations.

Resolving all cases of Violation 1 and 2 will automatically resolve all cases of Violation 3.

We illustrate the algorithm with an example. Figure 15.20.a shows an initial T-mesh into which we wish to insert one control point, $\mathbf{P}_2$. Because the T-mesh in Figure 15.20.a is valid, there are no violations. But if we simply insert $\mathbf{P}_2$ into the T-mesh (Figure 15.20.b) *without changing any of the blending functions*, we introduce several violations. Since $\mathbf{P}_2$ has knot coordinates (s_3, t_2), four blending functions become guilty of Violation 1: those centered at (s_1, t_2), (s_2, t_2), (s_4, t_2), and (s_5, t_2). To resolve these violations, we must insert a knot at s_3 into each of those blending functions, as discussed in Section 15.7.3. The blending function centered at (s_2, t_2) is $N[s_0, s_1, s_2, s_4, s_5](s)N[t_0, t_1, t_2, t_3, t_4](t)$. Inserting a knot $s = s_3$ into the s knot vector of this blending function splits it into two scaled blending functions: $c_2 N[s_0, s_1, s_2, s_3, s_4](s)N[t_0, t_1, t_2, t_3, t_4](t)$ (Figure 15.20.c) and $d_2 N[s_1, s_2, s_3, s_4, s_5](s)N[t_0, t_1, t_2, t_3, t_4](t)$ (Figure 15.20.d) as given in (15.16).

The blending function $c_2 N[s_0, s_1, s_2, s_3, s_4](s)N[t_0, t_1, t_2, t_3, t_4](t)$ in Figure 15.20.c satisfies Rule 1. Likewise, the refinements of the blending functions centered at (s_1, t_2), (s_4, t_2), and (s_5, t_2) all satisfy Rule 1. However, the t knot vector of blending function $d_2 N[s_1, s_2, s_3, s_4, s_5](s)N[t_0, t_1, t_2, t_3, t_4](t)$ shown in Figure 15.20.d is guilty of Violation 2 because the blending function's t knot vector is $[t_0, t_1, t_2, t_3, t_4]$, but Rule 1 does not call for a knot at t_3. This problem cannot be remedied by refining this blending function; we must add an additional control point into the T-mesh.

The needed control point is $\mathbf{P}_3$ in Figure 15.20.e. Inserting that control point fixes the case of Violation 2, but it creates a new case of Violation 1. As shown in Figure 15.20.f, the blending function centered at (s_2, t_3) has an s knot vector that does not include s_3 as required by Rule 1. Inserting s_3 into that knot vector fixes the problem, and there are no further violations of Rule 1.

This algorithm is always guaranteed to terminate, because the only blending function refinements and control point insertions must involve knot values that initially exist in the T-mesh, or that were added in Step 1. In the worst case, the algorithm would extend all partial rows of control points to cross the entire surface. In practice, the algorithm typically requires few if any additional new control points beyond the ones the user wants to insert.

15.7.6 Converting a T-spline into a B-spline surface

This refinement algorithm makes it easy to convert a T-spline $\in S_1$ into an equivalent B-spline surface $\in S_n$: simply compute the transformation matrix $M_{1,n}$ as discussed in Section 15.7.4.

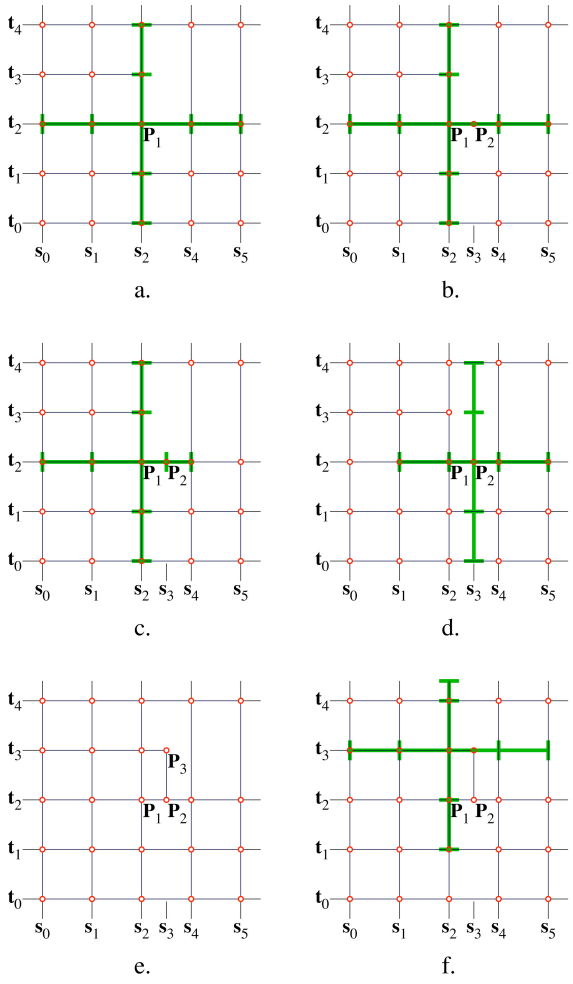


Figure 15.20: Local refinement example.

A *standard* T-spline is one for which, if all weights $w_i = 1$, then $\sum_{i=1}^{n} w_i B_i(s,t) \equiv \sum_{i=1}^{n} B_i(s,t) \equiv 1$. This means that the denominator in (15.7) is identically equal to one; hence, the blending functions provide a partition of unity and the T-spline is polynomial. Thus, an algebraic statement of necessary and sufficient conditions for a T-spline to be standard is each row of $M_{1,n}$ sums to 1.

The insertion algorithm can produce a surprising result: a T-spline for which not all $w_i = 1$ but yet $\sum_{i=1}^{n} w_i B_i(s,t) \equiv 1$. This is called a semi-standard T-spline. Figure 15.21 shows two simple examples of semi-standard T-splines. The integers (1 and 2) next to some edges are knot intervals. To verify that these T-splines are semi-standard, convert them into B-spline surfaces; the control

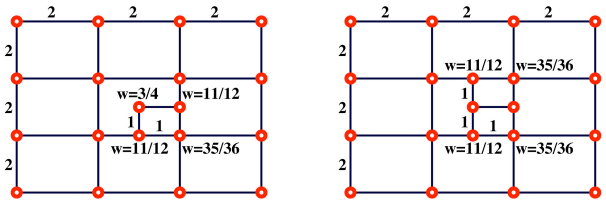


Figure 15.21: Semi-standard T-splines.

point weights will all be one.

The notion of T-spline spaces in Section 15.7.4 enables a more precise definition of semi-standard and non-standard T-splines. A semi-standard T-spline space S is one for which there exists some elements of S for which $\sum_{i=1}^{n} w_i B_i(s,t) \equiv 1$, and not all $w_i = 1$. A non-standard T-spline space is one for which no elements exist for which $\sum_{i=1}^{n} w_i B_i(s,t) \equiv 1$. These definitions are more precise because they allow for the notion of a rational T-spline (weights not all = 1) that is either standard, semi-standard, or non-standard. The distinction is made based on which type of T-spline space it belongs to.

15.8 Efficient Computation of Points and Tangents on a Bézier surface patch.

This section presents an algorithm for computing points and tangents on a tensor-product rational Bézier surface patch that has $O(n^2)$ time complexity.

Define a rational Bézier curve in $\mathbf{R}^3$ with the notation

$$\mathbf{p}(t) = \Pi\left(\mathbf{P}(t)\right) \tag{15.24}$$

with

$$\mathbf{P}(t) = (\mathbf{P}_x(t), \mathbf{P}_y(t), \mathbf{P}_z(t), \mathbf{P}_w(t)) = \sum_{i=0}^{n} \mathbf{P}_i B_i^n(t) \tag{15.25}$$

where $\mathbf{P}_i = w_i(x_i, y_i, z_i, 1)$ and the projection operator Π is defined $\Pi(x,y,z,w) = (x/w, y/w, z/w)$. In this section, we will use upper case bold-face variables to denote four-tuples (homogeneous points) and lower case bold-face for triples (points in R^3).

The point and tangent of this curve can be found using the familiar construction

$$\mathbf{P}(t) = (1-t)\mathbf{Q}(t) + t\mathbf{R}(t) \tag{15.26}$$

with

$$\mathbf{Q}(t) = \sum_{i=0}^{n-1} \mathbf{P}_i B_i^{n-1}(t) \tag{15.27}$$

and

$$\mathbf{R}(t) = \sum_{i=1}^{n} \mathbf{P}_i B_{i-1}^{n-1}(t) \tag{15.28}$$

where line $\mathbf{q}(t)$—$\mathbf{r}(t) \equiv \Pi(\mathbf{Q}(t))$—$\Pi(\mathbf{R}(t))$ is tangent to the curve, as seen in Figure 16.1. As a sidenote, the correct magnitude of the derivative of $\mathbf{p}(t)$ is given by

$$\frac{d\mathbf{p}(t)}{dt} = n\frac{\mathbf{R}_w(t)\mathbf{Q}_w(t)}{((1-t)\mathbf{Q}_w(t) + t\mathbf{R}_w(t))^2}\left[\mathbf{r}(t) - \mathbf{q}(t)\right] \tag{15.29}$$

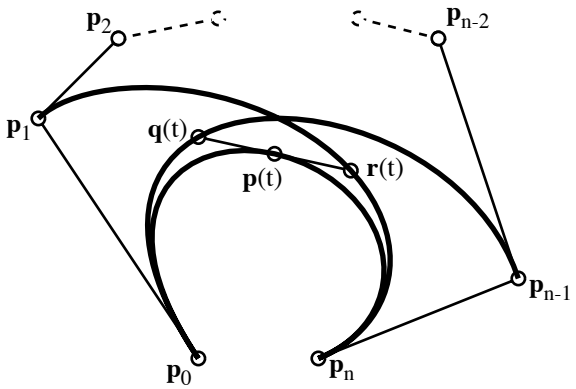


Figure 15.22: Curve example

The values $\mathbf{Q}(t)$ and $\mathbf{R}(t)$ can be found using the modified Horner's algorithm for Bernstein polynomials, involving a pseudo–basis conversion

$$\frac{\mathbf{Q}(t)}{(1-t)^{n-1}} = \hat{\mathbf{Q}}(u) = \sum_{i=0}^{n-1} \hat{\mathbf{Q}}_i u^i \tag{15.30}$$

where $u = \frac{t}{1-t}$ and $\hat{\mathbf{Q}}_i = \binom{n-1}{i}\mathbf{P}_i$, $i = 0, 1, \ldots, n-1$. Assuming the curve is to be evaluated several times, we can ignore the expense of precomputing the $\hat{\mathbf{Q}}_i$, and the nested multiplication

$$\hat{\mathbf{Q}}(u) = [\cdots[[\hat{\mathbf{Q}}_{n-1}u + \hat{\mathbf{Q}}_{n-2}]u + \hat{\mathbf{Q}}_{n-3}]u + \ldots \hat{\mathbf{Q}}_1]u + \hat{\mathbf{Q}}_0 \tag{15.31}$$

can be performed with $n-1$ multiplies and adds for each of the four x, y, z, w coordinates. It is not necessary to post-multiply by $(1-t)^{n-1}$, since

$$\Pi(\mathbf{Q}(t)) = \Pi\left((1-t)^{n-1}\hat{\mathbf{Q}}(u)\right) = \Pi\left(\hat{\mathbf{Q}}(t)\right). \tag{15.32}$$

Therefore, the point $\mathbf{P}(t)$ and its tangent direction can be computed with roughly $2n$ multiplies and adds for each of the four x, y, z, w coordinates.

This method has problems near $t = 1$, so it is best for $.5 \le t \le 1$ to use the form

$$\frac{\mathbf{Q}(t)}{t^{n-1}} = \sum_{i=0}^{n-1} \hat{\mathbf{Q}}_{n-i-1} u^i \tag{15.33}$$

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

A tensor product rational Bézier surface patch is defined

$$\mathbf{p}(s,t) = \Pi\left(\mathbf{P}(s,t)\right) \tag{15.34}$$

where

$$\mathbf{P}(s,t) = \sum_{i=0}^{m}\sum_{j=0}^{n} \mathbf{P}_{ij} B_i^m(s) B_j^n(t). \tag{15.35}$$

We can represent the surface $\mathbf{p}(s,t)$ using the following construction:

$$\mathbf{P}(s,t) = (1-s)(1-t)\mathbf{P}^{00}(s,t) + s(1-t)\mathbf{P}^{10}(s,t) + (1-s)t\mathbf{P}^{01}(s,t) + st\mathbf{P}^{11}(s,t) \tag{15.36}$$

where

$$\mathbf{P}^{00}(s,t) = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} \mathbf{P}_{ij} B_i^{m-1}(s) B_j^{n-1}(t), \tag{15.37}$$

$$\mathbf{P}^{10}(s,t) = \sum_{i=1}^{m}\sum_{j=0}^{n-1} \mathbf{P}_{ij} B_{i-1}^{m-1}(s) B_j^{n-1}(t), \tag{15.38}$$

$$\mathbf{P}^{01}(s,t) = \sum_{i=0}^{m-1}\sum_{j=1}^{n} \mathbf{P}_{ij} B_i^{m-1}(s) B_{j-1}^{n-1}(t), \tag{15.39}$$

$$\mathbf{P}^{11}(s,t) = \sum_{i=1}^{m}\sum_{j=1}^{n} \mathbf{P}_{ij} B_{i-1}^{m-1}(s) B_{j-1}^{n-1}(t). \tag{15.40}$$

The tangent vector $\mathbf{p}_s(s,t)$ is parallel with the line

$$\Pi\left((1-t)\mathbf{P}^{00}(s,t) + t\mathbf{P}^{01}(s,t)\right) \text{—} \Pi\left((1-t)\mathbf{P}^{10}(s,t) + t\mathbf{P}^{11}(s,t)\right) \tag{15.41}$$

and the tangent vector $\mathbf{p}_t(s,t)$ is parallel with

$$\Pi\left((1-s)\mathbf{P}^{00}(s,t) + s\mathbf{P}^{10}(s,t)\right) \text{—} \Pi\left((1-s)\mathbf{P}^{01}(s,t) + s\mathbf{P}^{11}(s,t)\right). \tag{15.42}$$

The Horner algorithm for a tensor product surface emerges by defining

$$\frac{\mathbf{P}^{kl}(s,t)}{(1-s)^{m-1}(1-t)^{n-1}} = \hat{\mathbf{P}}^{kl}(u,v) = \sum_{i=k}^{m+k-1}\sum_{j=l}^{n+l-1} \hat{\mathbf{P}}_{ij}^{kl} u^i v^j; \qquad k,l = 0,1 \tag{15.43}$$

where $u = \frac{s}{1-s}$, $v = \frac{t}{1-t}$, and $\hat{\mathbf{P}}_{ij}^{kl} = \binom{m-1}{i-k}\binom{n-1}{j-l}\mathbf{P}_{ij}$. The n rows of these four bivariate polynomials can each be evaluated using $m-1$ multiplies and adds per x, y, z, w component, and the final evaluation in t costs $n-1$ multiplies and adds per x, y, z, w component.

Thus, if $m = n$, the four surfaces $\mathbf{P}^{00}(s,t)$, $\mathbf{P}^{01}(s,t)$, $\mathbf{P}^{10}(s,t)$, and $\mathbf{P}^{11}(s,t)$ can each be evaluated using n^2-1 multiplies and n^2-1 adds for each of the four x, y, z, w components, a total of $16n^2-16$ multiplies and $16n^2-16$ adds.

If one wishes to compute a grid of points on this surface which are evenly spaced in parameter space, the four surfaces $\mathbf{P}^{00}(s,t)$, $\mathbf{P}^{01}(s,t)$, $\mathbf{P}^{10}(s,t)$, and $\mathbf{P}^{11}(s,t)$ can each be evaluated even more quickly using forward differencing.

15.9 Curvature at the Corner of a Bézier Surface Patch

We have seen in Section **??** how the curvature of a degree n rational Bézier curve with control points $\mathbf{P}_0, \mathbf{P}_1, \cdots, \mathbf{P}_n$ and weights $\omega_0, \omega_1, \cdots, \omega_n$, at the starting point $\mathbf{P}_0$ is given by

$$k = \frac{n-1}{n} \frac{\omega_0 \omega_2}{\omega_1^2} \frac{h}{a^2} \tag{15.44}$$

where a is the length of edge $\mathbf{P}_0\mathbf{P}_1$, and h is the distance of $\mathbf{P}_2$ to the tangent spanned by $\mathbf{P}_0$ and $\mathbf{P}_1$ (see Figure 15.23). This formula is more intuitive than the one given by classic differential geometry, and is also easier to compute, especially in the rational case.

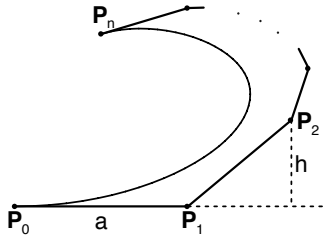


Figure 15.23: Curvature of a Bézier curve.

This section derives similar formulae for Gaussian and mean curvatures of rational Bézier patches. The derived formulae are expressed in terms of simple geometric quantities of the control mesh.

15.9.1 Curvatures of tensor-product rational Bézier surfaces

Suppose a degree $n \times m$ rational Bézier patch is defined by

$$\mathbf{r}(u,v) = \frac{\sum_{i=0}^{n} \sum_{j=0}^{m} \omega_{ij} \mathbf{P}_{ij} B_i^n(u) B_j^m(v)}{\sum_{i=0}^{n} \sum_{j=0}^{m} \omega_{ij} B_i^n(u) B_j^m(v)} \tag{15.45}$$

where $B_i^k(t) = \binom{k}{i}(1-t)^{k-i}t^i$ are Bernstein polynomials; $\mathbf{P}_{ij}$ are the control points, forming a control mesh; and w_{ij} are the weights. When all the weights are the same, the patch reduces to a polynomial surface. We derive the curvature formulae at the bottom-left corner $(u,v) = (0,0)$.

By differential geometry,Gaussian curvature K_g and mean curvature K_m can be computed by the following formulae

$$K_g = \frac{LN - M^2}{EG - F^2}, \tag{15.46}$$

$$K_m = \frac{1}{2} \frac{LG - 2MF + NE}{EG - F^2} \tag{15.47}$$

where E, F, G are the coefficients of the first fundamental form, i.e.,

$$E = \mathbf{r}_u \cdot \mathbf{r}_u, \quad F = \mathbf{r}_u \cdot \mathbf{r}_v, \quad G = \mathbf{r}_v \cdot \mathbf{r}_v, \tag{15.48}$$

L, M, N are the coefficients of the second fundamental form, i.e.,

$$L = \mathbf{r}_{uu} \cdot \mathbf{n}, \quad M = \mathbf{r}_{uv} \cdot \mathbf{n}, \quad N = \mathbf{r}_{vv} \cdot \mathbf{n}, \tag{15.49}$$

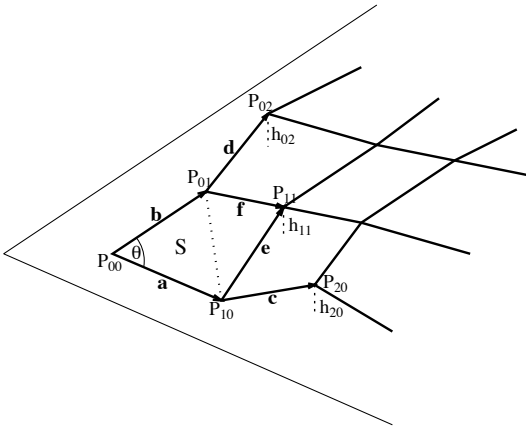


Figure 15.24: Part of a rectangular mesh.

and $\mathbf{n} = \mathbf{r}_u \times \mathbf{r}_v / \|\mathbf{r}_u \times \mathbf{r}_v\|$ is the unit normal vector.

For the rational Bézier patch (15.45), we introduce some notations as follows (see Figure 15.24):

$$\begin{array}{lll} \mathbf{a} = \mathbf{P}_{10} - \mathbf{P}_{00}, & \mathbf{b} = \mathbf{P}_{01} - \mathbf{P}_{00}, & \mathbf{c} = \mathbf{P}_{20} - \mathbf{P}_{10}, \\ \mathbf{d} = \mathbf{P}_{02} - \mathbf{P}_{01}, & \mathbf{e} = \mathbf{P}_{11} - \mathbf{P}_{10}, & \mathbf{f} = \mathbf{P}_{11} - \mathbf{P}_{01}. \end{array} \tag{15.50}$$

Furthermore, let S denote the area of the triangle $\mathbf{P}_{00}\mathbf{P}_{10}\mathbf{P}_{01}$, θ be the angle between vectors $\mathbf{P}_{00}\mathbf{P}_{10}$ and $\mathbf{P}_{00}\mathbf{P}_{01}$, and h_{ij} be the signed distance from $\mathbf{P}_{ij}$ to the plane spanned by $\mathbf{P}_{00}, \mathbf{P}_{10}$ and $\mathbf{P}_{01}$. If $\mathbf{P}_{ij}$ lies on the side of the plane $\mathbf{P}_{00}\mathbf{P}_{10}\mathbf{P}_{01}$ with the direction $\mathbf{P}_{00}\mathbf{P}_{10} \times \mathbf{P}_{00}\mathbf{P}_{01}$, h_{ij} is positive. Otherwise, h_{ij} is negative.

It is easy to check that at $(u, v) = (0, 0)$, we have

$$\mathbf{r}_u(0,0) = n\frac{\omega_{10}}{\omega_{00}}\mathbf{a}, \qquad \mathbf{r}_v(0,0) = m\frac{\omega_{01}}{\omega_{00}}\mathbf{b}. \tag{15.51}$$

For notational simplicity, in the following where there is no ambiguity, we omit the parameter values $(0, 0)$. Thus

$$E = n^2\left(\frac{\omega_{10}}{\omega_{00}}\right)^2 \mathbf{a}\cdot\mathbf{a}, \quad F = nm\left(\frac{\omega_{10}\omega_{01}}{\omega_{00}^2}\right)\mathbf{a}\cdot\mathbf{b}, \quad G = m^2\left(\frac{\omega_{01}}{\omega_{00}}\right)^2 \mathbf{b}\cdot\mathbf{b}. \tag{15.52}$$

Note that for a rational surface $\mathbf{r}(u, v) = \mathbf{R}(u, v)/\omega(u, v)$, the first partial derivative $\mathbf{r}_u(u, v) = (\mathbf{R}_u - \mathbf{r}\,\omega_u)/\omega$, and the second partial derivative $\mathbf{r}_{uu} = \dfrac{\mathbf{R}_{uu} - \mathbf{r}\omega_{uu}}{\omega} - \dfrac{\mathbf{R}_u\omega_u - \mathbf{r}\omega_u^2}{\omega^2} - \mathbf{r}_u\dfrac{\omega_u}{\omega}$. Applying these to the rational Bézier patch (15.45) and letting $(u, v) = (0, 0)$ lead to

$$\mathbf{r}_{uu} = n(n-1)\frac{\omega_{20}}{\omega_{00}}\mathbf{c} + (\cdots)\mathbf{a}$$

where $(\cdots)$ denotes a rather complicated expression that we will not need to be concerned with since subsequent dotting with $\mathbf{n}$ will cause it to vanish. Similarly, we can obtain

$$\mathbf{r}_{uv} = nm\frac{\omega_{11}}{\omega_{00}}\mathbf{f} + (\cdots)\mathbf{a} + (\cdots)\mathbf{b},$$

$$\mathbf{r}_{vv} = m(m-1)\frac{\omega_{02}}{\omega_{00}}\mathbf{d} + (\cdots)\mathbf{b}.$$

Also note that $\mathbf{n} = \mathbf{a}\times\mathbf{b}/\|\mathbf{a}\times\mathbf{b}\|$. By the definition (15.49) of L, M and N, we have

$$L = n(n-1)\frac{\omega_{20}}{\omega_{00}}h_{20}, \quad M = nm\frac{\omega_{11}}{\omega_{00}}h_{11}, \quad N = m(m-1)\frac{\omega_{02}}{\omega_{00}}h_{02}. \tag{15.53}$$

Now substituting (15.52) and (15.53) into (15.46) and (15.47), and doing some simplifications, we get the formulae for Gaussian and mean curvatures

$$K_g = \left(\frac{\omega_{00}}{\omega_{10}\omega_{01}}\right)^2 \frac{\frac{n-1}{n}\,\frac{m-1}{m}\omega_{20}\omega_{02}h_{20}h_{02} - \omega_{11}^2h_{11}^2}{\|\mathbf{a}\times\mathbf{b}\|^2}, \tag{15.54}$$

$$K_m = \frac{\omega_{00}}{\omega_{10}^2\omega_{01}^2}\,\frac{\frac{n-1}{n}\omega_{01}^2\mathbf{b}^2\omega_{20}h_{20} - 2\omega_{10}\omega_{01}(\mathbf{a}\cdot\mathbf{b})\omega_{11}h_{11} + \frac{m-1}{m}\omega_{10}^2\mathbf{a}^2\omega_{02}h_{02}}{2\|\mathbf{a}\times\mathbf{b}\|^2}. \tag{15.55}$$

If we further introduce notations

$$\begin{aligned}
&\tilde{a} = \frac{\omega_{10}}{\omega_{00}}\|\mathbf{a}\|, \qquad \tilde{b} = \frac{\omega_{01}}{\omega_{00}}\|\mathbf{b}\|, \qquad \tilde{S} = \frac{1}{2}\,\|\frac{\omega_{10}}{\omega_{00}}\mathbf{a}\times\frac{\omega_{01}}{\omega_{00}}\mathbf{b}\| = S\frac{\omega_{10}\omega_{01}}{\omega_{00}^2},\\
&\tilde{h}_{11} = \frac{\omega_{11}}{\omega_{00}}h_{11}, \quad \tilde{h}_{20} = \frac{n-1}{n}\,\frac{\omega_{20}}{\omega_{00}}h_{20}, \quad \tilde{h}_{02} = \frac{m-1}{m}\,\frac{\omega_{02}}{\omega_{00}}h_{02},
\end{aligned}$$

then the formulae can be symbolically simplified to

$$K_g = \left(\tilde{h}_{20}\,\tilde{h}_{02} - \tilde{h}_{11}^2\right)/4\tilde{S}^2, \tag{15.56}$$

$$K_m = \left(\tilde{h}_{20}\,\tilde{b}^2 - 2\tilde{h}_{11}\,\tilde{a}\,\tilde{b}\,\cos\theta + \tilde{h}_{02}\,\tilde{a}^2\right)/8\tilde{S}^2. \tag{15.57}$$

Remark 1. Obviously, formulae (15.54) and (15.55) or (15.56) and (15.57) just contain some simple geometric quantities, like (scaled) length or area, related to the control mesh of the rational Bézier patch. Compared to the formulae (15.46) and (15.47), they are more intuitive and also simpler to compute.

Remark 2. Though the equations derived above are valid at the bottom-left corner, by symmetry similar formulae are easily written out at the other three corners of the rational Bézier patch. Moreover, at any point of the surface other than the four corners, the formulae can also be used to calculate curvatures with help of subdividing the surface there.

Remark 3. According to differential geometry, from Gaussian and mean curvatures, we can compute the principal curvatures

$$k_{1,2} = K_m \pm \sqrt{K_m^2 - K_g}.$$

Meanwhile, the principal directions of curvatures can be determined by

$$\frac{du}{dv} = -\frac{M - k_{1,2}F}{L - k_{1,2}E} = -\frac{N - k_{1,2}G}{M - k_{1,2}F}.$$

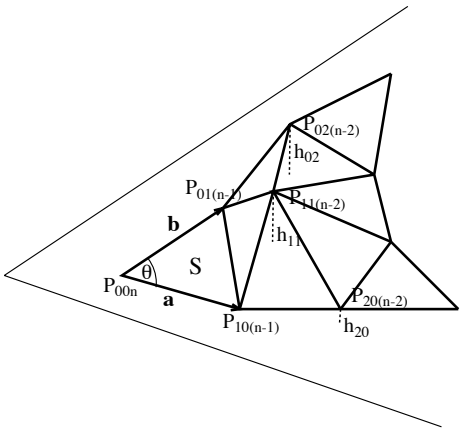


Figure 15.25: Part of a triangular mesh.

15.9.2 Curvatures of triangular rational Bézier surfaces

A triangular rational Bézier patch of degree n is defined by

$$\mathbf{r}(u,v) = \frac{\sum\limits_{i+j+k=n} \omega_{ijk}\mathbf{P}_{ijk}B^n_{ijk}(u,v)}{\sum\limits_{i+j+k+n} \omega_{ijk}B^n_{ijk}(u,v)} \tag{15.58}$$

where $B^n_{ijk}(u,v) = \dfrac{n!}{i!j!k!}u^i v^j(1-u-v)^k$ are Bernstein polynomials; $\mathbf{P}_{ijk}$ are the control points, forming a triangular control mesh (see Figure 15.25); and ω_{ijk} are the weights.

As in the tensor-product case, we only consider the curvatures at corner $(u,v) = (0,0)$. Although we can follow the straightforward approach of Section 15.9.1, i.e., computing the partial derivatives of the triangular Bézier patch, here we take another approach. Since the triangular patch $\mathbf{r}(u,v)$ can be considered as a degree $n \times n$ tensor-product rational Bézier patch, we can utilize the established formulae (15.54) and (15.55).

Let $\mathbf{a} = \mathbf{P}_{10(n-1)} - \mathbf{P}_{00n}$, $\mathbf{b} = \mathbf{P}_{01(n-1)} - \mathbf{P}_{00n}$, S denote the area of the triangle $\mathbf{P}_{00n}\mathbf{P}_{10(n-1)}\mathbf{P}_{01(n-1)}$, θ be the angle between vectors $\mathbf{P}_{00n}\mathbf{P}_{10(n-1)}$ and $\mathbf{P}_{00n}\mathbf{P}_{01(n-1)}$, and h_{ij} be the signed distance from $\mathbf{P}_{ij(n-i-j)}$ to the plane spanned by $\mathbf{P}_{00n}, \mathbf{P}_{10(n-1)}$ and $\mathbf{P}_{01(n-1)}$. Suppose the tensor-product representation of the patch (15.58) is

$$\mathbf{r}(u,v) = \frac{\sum\limits_{i=0}^{n}\sum\limits_{j=0}^{n} \bar{\omega}_{ij}\bar{\mathbf{P}}_{ij}B^n_i(u)B^n_j(v)}{\sum\limits_{i=0}^{n}\sum\limits_{j=0}^{n} \bar{\omega}_{ij}B^n_i(u)B^n_j(v)}.$$

It is easy to show that $\bar{\mathbf{P}}_{ij} = \mathbf{P}_{ij(n-i-j)}$ and $\bar{\omega}_{ij} = \omega_{ij(n-i-j)}$ for $(i,j) = (0,0), (1,0)$, and $(0,1)$. Thus the geometric quantities, like $\mathbf{a}$, $\mathbf{b}$, h_{ij}, and etc., are the same as their counterparts in the tensor-product representation except for h_{11}. Therefore we only need to check $\bar{\mathbf{P}}_{11}$ and $\bar{\omega}_{11}$. Considering the mixed derivative of the denominators of the above two representations at $(u,v) = (0,0)$, we get

$$\bar{\omega}_{11} = \frac{n-1}{n}\omega_{11(n-2)} + \frac{\omega_{10(n-1)} + \omega_{01(n-1)} - \omega_{00n}}{n}$$

Similarly, by considering the numerators, we have

$$\bar{\mathbf{P}}_{11} = \frac{n-1}{n}\frac{\omega_{11(n-2)}}{\bar{\omega}_{11}}\mathbf{P}_{11(n-2)} + \frac{\omega_{10(n-1)}\mathbf{P}_{10(n-1)} + \omega_{01(n-1)}\mathbf{P}_{01(n-1)} - \omega_{00n}\mathbf{P}_{00n}}{n\bar{\omega}_{11}}$$

If we denote the signed distance of $\bar{\mathbf{P}}_{11}$ to the plane $\bar{\mathbf{P}}_{00}\bar{\mathbf{P}}_{10}\bar{\mathbf{P}}_{01}$ by $\bar{h}_{11}$, then $\bar{\omega}_{11}\bar{h}_{11} = \dfrac{n-1}{n}\omega_{11(n-2)}h_{11}$. We substitute all the above relations into (15.54) and (15.55), and the formulae of Gaussian and mean curvatures for the triangular rational Bézier patch (15.58) turn out to be

$$K_g = \left(\frac{n-1}{n}\right)^2 \left(\frac{\omega_{00n}}{\omega_{10(n-1)}\omega_{01(n-1)}}\right)^2 \frac{\omega_{20(n-2)}\omega_{02(n-2)}h_{20}h_{02} - \omega_{11(n-2)}^2 h_{11}^2}{\|\mathbf{a}\times\mathbf{b}\|^2}, \tag{15.59}$$

$$K_m = \frac{n-1}{n}\,\frac{\omega_{00n}}{\omega_{10(n-1)}^2\omega_{01(n-1)}^2}\cdot \frac{\omega_{01(n-1)}^2\mathbf{b}^2\omega_{20(n-2)}h_{20} - 2\omega_{10(n-1)}\omega_{01(n-1)}(\mathbf{a}\cdot\mathbf{b})\omega_{11(n-2)}h_{11} + \omega_{10(n-1)}^2\mathbf{a}^2\omega_{02(n-2)}h_{02}}{2\|\mathbf{a}\times\mathbf{b}\|^2}, \tag{15.60}$$

or just (15.56) and (15.57) if the following notations are adopted

$$\tilde{a} = \frac{\omega_{10(n-1)}}{\omega_{00n}}\|\mathbf{a}\|, \qquad \tilde{b} = \frac{\omega_{01(n-1)}}{\omega_{00n}}\|\mathbf{b}\|,$$
$$\tilde{S} = S\frac{\omega_{10(n-1)}\omega_{01(n-1)}}{\omega_{00n}^2}, \qquad \tilde{h}_{ij} = \frac{n-1}{n}\,\frac{\omega_{ij(n-i-j)}}{\omega_{00n}}h_{ij}.$$

15.9.3 Curvature of an Implicit Surface

We complete our discussion of curvature by providing the formulae for the Gaussian and mean curvatures at a point on an implicit surface $f(x,y,z)=0$:

$$K_m = \frac{f_{zz}(f_y^2+f_x^2) - 2f_xf_yf_{xy} + f_{xx}(f_y^2+f_z^2) - 2f_yf_zf_{yz} + f_{yy}(f_x^2+f_z^2) - 2f_zf_xf_{xz}}{2\lambda^3} \tag{15.61}$$

$$K_g = \frac{1}{\lambda^4}[-2f_xf_yf_{xy}f_{zz} + f_{yy}f_{xx}f_z^2 - 2f_yf_zf_{yz}f_{xx} + f_{zz}f_{yy}f_x^2 - 2f_zf_xf_{xz}f_{yy} + f_{xx}f_{zz}f_y^2$$
$$-f_x^2f_{yz}^2 - f_y^2f_{xz}^2 - f_z^2f_{xy}^2 + 2f_xf_yf_{xz}f_{yz} + 2f_yf_zf_{xy}f_{xz} + 2f_xf_zf_{xy}f_{yz}] \tag{15.62}$$

where

$$\lambda = \sqrt{f_x^2+f_y^2+f_z^2}$$