Chapter 16

Computing Points and Tangents on Bézier Surface Patches

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

A rational Bézier curve in $\mathbb{R}^3$ is defined

$$\mathbf{p}(t) = \Pi(\mathbf{P}(t))$$  \hspace{1cm} (16.1)

with

$$\mathbf{P}(t) = (\mathbf{P}_x(t), \mathbf{P}_y(t), \mathbf{P}_z(t), \mathbf{P}_w(t)) = \sum_{i=0}^{n} \mathbf{P}_i B^n_i(t)$$  \hspace{1cm} (16.2)

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

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

$$\mathbf{P}(t) = (1-t)\mathbf{Q}(t) + t\mathbf{R}(t)$$  \hspace{1cm} (16.3)

with

$$\mathbf{Q}(t) = \sum_{i=0}^{n-1} \mathbf{P}_i B_i^{n-1}(t)$$  \hspace{1cm} (16.4)

and

$$\mathbf{R}(t) = \sum_{i=1}^{n} \mathbf{P}_i B_{i-1}^{n-1}(t)$$  \hspace{1cm} (16.5)

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

$$\frac{d\mathbf{p}(t)}{dt} = n \frac{\mathbf{R}_w(t) \mathbf{Q}_w(t)}{((1-t)\mathbf{Q}_w(t) + t\mathbf{R}_w(t))^2} [\mathbf{r}(t) - \mathbf{q}(t)]$$  \hspace{1cm} (16.6)
The values $Q(t)$ and $R(t)$ can be found using the modified Horner’s algorithm for Bernstein polynomials, involving a pseudo–basis conversion

$$\frac{Q(t)}{(1-t)^{n-1}} = \hat{Q}(u) = \sum_{i=0}^{n-1} \hat{Q}_i u^i$$  \hspace{1cm} (16.7)

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

$$\hat{Q}(u) = [\cdots [\hat{Q}_{n-1} u + \hat{Q}_{n-2} u + \hat{Q}_{n-3} u + \cdots \hat{Q}_1] u + \hat{Q}_0$$ \hspace{1cm} (16.8)

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

$$\Pi \left( Q(t) \right) = \Pi \left( (1-t)^{n-1} \hat{Q}(u) \right) = \Pi \left( \hat{Q}(t) \right).$$ \hspace{1cm} (16.9)

Therefore, the point $P(t)$ 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 $.5 \leq t \leq 1$ to use the form

$$\frac{Q(t)}{t^{n-1}} = \sum_{i=0}^{n-1} \hat{Q}_{n-1-i} u^i$$ \hspace{1cm} (16.10)

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

A tensor product rational Bézier surface patch is defined

$$p(s,t) = \Pi \left( P(s,t) \right)$$ \hspace{1cm} (16.11)

where

$$P(s,t) = \sum_{i=0}^{m} \sum_{j=0}^{n} P_{ij} B_i^m(s) B_j^n(t).$$ \hspace{1cm} (16.12)
We can represent the surface \( p(s, t) \) using the following construction:

\[
P(s, t) = (1 - s)(1 - t)P^{00}(s, t) + s(1 - t)P^{10}(s, t) + (1 - s)tP^{01}(s, t) + stP^{11}(s, t)
\]  

(16.13)

where

\[
P^{00}(s, t) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} P_{ij}B_i^{m-1}(s)B_j^{n-1}(t),
\]  

(16.14)

\[
P^{10}(s, t) = \sum_{i=1}^{m} \sum_{j=0}^{n-1} P_{ij}B_i^{m-1}(s)B_j^{n-1}(t),
\]  

(16.15)

\[
P^{01}(s, t) = \sum_{i=0}^{m-1} \sum_{j=1}^{n} P_{ij}B_i^{m-1}(s)B_j^{n-1}(t),
\]  

(16.16)

\[
P^{11}(s, t) = \sum_{i=1}^{m} \sum_{j=1}^{n} P_{ij}B_i^{m-1}(s)B_j^{n-1}(t).
\]  

(16.17)

The tangent vector \( p_s(s, t) \) is parallel with the line

\[
\Pi \left((1 - t)P^{00}(s, t) + tP^{01}(s, t)\right) - \Pi \left((1 - t)P^{10}(s, t) + tP^{11}(s, t)\right)
\]  

(16.18)

and the tangent vector \( p_t(s, t) \) is parallel with

\[
\Pi \left((1 - s)P^{00}(s, t) + sP^{10}(s, t)\right) - \Pi \left((1 - s)P^{01}(s, t) + sP^{11}(s, t)\right).
\]  

(16.19)

The Horner algorithm for a tensor product surface emerges by defining

\[
\frac{P^{kl}(s, t)}{(1 - s)^{m-1}(1 - t)^{n-1}} = \hat{P}^{kl}(u, v) = \sum_{i=k}^{m+k-1} \sum_{j=l}^{n+l-1} \hat{P}_{ij}^{kl} u^i v^j; \quad k, l = 0, 1
\]  

(16.20)

where \( u = \frac{s}{1-t}, v = \frac{t}{1-s} \), and \( \hat{P}_{ij}^{kl} = \binom{m-1}{i-k} \binom{n-1}{j-l} P_{ij} \). 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 \( P^{00}(s, t), P^{01}(s, t), P^{10}(s, t), \) and \( P^{11}(s, t) \) 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 \( P^{00}(s, t), P^{01}(s, t), P^{10}(s, t), \) and \( P^{11}(s, t) \) can each be evaluated even more quickly using forward differencing.