ELSEVIER

Computer Aided Geometric Design 12 (1995) 103–106

COMPUTER AIDED GEOMETRIC DESIGN

Short Communication

Point and tangent computation of tensor product rational Bézier surfaces

Thomas W. Sederberg *

Department of Civil Engineering, Brigham Young University, Provo, UT 84602, USA

Received March 1994; revised July 1994

Abstract

This note presents an $O(n^2)$ algorithm for evaluating point and tangents of a rational tensor product Bézier surface patch.

This note is motivated by a procedure suggested in (Mann and DeRose, 1995) with which point and tangents of a rational tensor product Bézier surface patch can be computed in $O(n^3)$ time complexity, but with a smaller leading coefficient than is obtained using the conventional repeated bilinear interpolation (which is also $O(n^3)$). The algorithm in this paper has $O(n^2)$ time complexity, but holds less geometric appeal than does that in (Mann and DeRose, 1995).

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

$$\boldsymbol{p}(t) = \Pi(\boldsymbol{P}(t)) \tag{1}$$

with

$$\boldsymbol{P}(t) = (\boldsymbol{P}_x(t), \boldsymbol{P}_y(t), \boldsymbol{P}_z(t), \boldsymbol{P}_w(t)) = \sum_{i=0}^{n} \boldsymbol{P}_i B_i^n(t) \tag{2}$$

where $\boldsymbol{P}_i = w_i(x_i, y_i, z_i, 1)$ and the projection operator Π is defined $\Pi(x, y, z, w) = (x/w, y/w, z/w)$. 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

$$\boldsymbol{P}(t) = (1-t)\boldsymbol{Q}(t) + t\boldsymbol{R}(t) \tag{3}$$

* Email: tom@byu.edu.

SSDI 0167-8396(94)00029-8

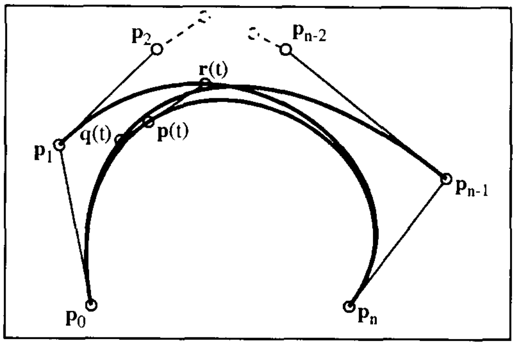

Fig. 1. Curve example.

with

$$Q(t) = \sum_{i=0}^{n-1} P_i B_i^{n-1}(t) \tag{4}$$

and

$$R(t) = \sum_{i=1}^{n} P_i B_{i-1}^{n-1}(t) \tag{5}$$

where line $q(t)$—$r(t) \equiv \Pi(Q(t))$—$\Pi(R(t))$ is tangent to the curve, as seen in Fig. 1. As a sidenote, the correct magnitude of the derivative of $p(t)$ is given by

$$\frac{dp(t)}{dt} = n\frac{R_w(t)Q_w(t)}{((1-t)Q_w(t)+tR_w(t))^2}[r(t)-q(t)]. \tag{6}$$

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

$$\frac{Q(t)}{(1-t)^{n-1}} = \hat{Q}(u) = \sum_{i=0}^{n-1} \hat{Q}_i u^i \tag{7}$$

where $u = t/(1-t)$ 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

$$\hat{Q}(u) = [\cdots[[\hat{Q}_{n-1}u + \hat{Q}_{n-2}]u + \hat{Q}_{n-3}]u + \cdots \hat{Q}_1]u + \hat{Q}_0 \tag{8}$$

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

$$\Pi(Q(t)) = \Pi((1-t)^{n-1}\hat{Q}(u)) = \Pi(\hat{Q}(t)). \tag{9}$$

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 $0.5 \leqslant t \leqslant 1$ to use the form

$$\frac{Q(t)}{t^{n-1}} = \sum_{i=0}^{n-1} \hat{Q}_{n-i-1} u^i \tag{10}$$

with $u = (1-t)/t$.

The origin of this Horner-like algorithm traces to (Pavlitis, 1982). A modified version appears in (Farin, 1990), which has the advantages that it avoids precomputing the $\hat{Q}_i$ and it handles the entire domain $[0,1]$, and the disadvantage that it costs $6n$ multiplies and $3n$ adds for each of the x, y, z, w coordinates. A variation of this modified Horner's algorithm applied to multivariate Bernstein polynomials is found in (Schumaker and Volk, 1986).

A tensor product rational Bézier surface patch is defined

$$\boldsymbol{p}(s,t) = \Pi(\boldsymbol{P}(s,t)) \tag{11}$$

where

$$\boldsymbol{P}(s,t) = \sum_{i=0}^{m} \sum_{j=0}^{n} \boldsymbol{P}_{ij} B_i^m(s) B_j^n(t). \tag{12}$$

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

$$\begin{aligned} \boldsymbol{P}(s,t) = {} & (1-s)(1-t)\boldsymbol{P}^{00}(s,t) + s(1-t)\boldsymbol{P}^{10}(s,t) \\ & + (1-s)t\boldsymbol{P}^{01}(s,t) + st\boldsymbol{P}^{11}(s,t) \end{aligned} \tag{13}$$

where

$$\boldsymbol{P}^{00}(s,t) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \boldsymbol{P}_{ij} B_i^{m-1}(s) B_j^{n-1}(t), \tag{14}$$

$$\boldsymbol{P}^{10}(s,t) = \sum_{i=1}^{m} \sum_{j=0}^{n-1} \boldsymbol{P}_{ij} B_{i-1}^{m-1}(s) B_j^{n-1}(t), \tag{15}$$

$$\boldsymbol{P}^{01}(s,t) = \sum_{i=0}^{m-1} \sum_{j=1}^{n} \boldsymbol{P}_{ij} B_i^{m-1}(s) B_{j-1}^{n-1}(t), \tag{16}$$

$$\boldsymbol{P}^{11}(s,t) = \sum_{i=1}^{m} \sum_{j=1}^{n} \boldsymbol{P}_{ij} B_{i-1}^{m-1}(s) B_{j-1}^{n-1}(t). \tag{17}$$

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

$$\Pi((1-t)\boldsymbol{P}^{00}(s,t) + t\boldsymbol{P}^{01}(s,t)) - \Pi((1-t)\boldsymbol{P}^{10}(s,t) + t\boldsymbol{P}^{11}(s,t)) \tag{18}$$

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

$$\Pi((1-s)\boldsymbol{P}^{00}(s,t) + s\boldsymbol{P}^{10}(s,t)) - \Pi((1-s)\boldsymbol{P}^{01}(s,t) + s\boldsymbol{P}^{11}(s,t)). \tag{19}$$

The Horner algorithm for a tensor product surface emerges by defining

$$\frac{P^{kl}(s,t)}{(1-s)^{m-1}(1-t)^{n-1}} = \hat{P}^{kl}(u,v) = \sum_{i=k}^{m+k-1} \sum_{j=l}^{n+l-1} \hat{P}_{ij}^{kl} u^i v^j, \quad k,l = 0,1 \tag{20}$$

where $u = s/(1-s)$, $v = t/(1-t)$, 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. Computing those same four points using the method in (Mann and DeRose, 1995) requires $4n^3 + 16n^2 + 12n$ multiplies and half that number 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 as discussed in (Lien, Shantz and Pratt, 1987; Lien, Shantz and Roccetti, 1989), though floating point error propagation must be dealt with. The method discussed here provides floating point robustness similar to the de Casteljau-based method in (Mann and DeRose, 1995) (see (Farouki and Rajan, 1987)).

Acknowledgements

The author gratefully acknowledges the careful assistance of Wang Guojin and Chen Falai. The author is currently supported by ONR.

References

Farin, G. (1990), *Curves and Surfaces for Computer Aided Geometric Design*, Academic Press, San Diego, CA.

Farouki, R.T. and Rajan, V.T. (1987), On the numerical condition of polynomials in Bernstein form, Computer Aided Geometric Design 4, 191–216.

Lien, S.-L., Shantz, M. and Pratt, V. (1987), Adaptive forward differencing for rendering curves and surfaces, Computer Graph. 21, 111–118.

Lien, S.-L., Shantz, M., and Rocchetti, R. (1989) Rendering cubic curves and surface with integer adaptive forward differencing, Comput. Graph. 23, 157–166.

Mann, S. and DeRose, T. (1995), Computing values and derivatives of Bézier and B-spline tensor products, Computer Aided Geometric Design 12, 107–110, this issue.

Pavlitis, T. (1982), *Algorithms for graphics and image processing*, Computer Science Press, Rockville, MD.

Schumaker, L.L. and Volk, W. (1986), Efficient evaluation of multivariate polynomials, Computer Aided Geometric Design 3, 149–154.