Projective Reparameterization of Rational Bézier Simplices

Michael McCool

Dynamic Graphics Project, CSRI, University of Toronto
6 King's College Road, Toronto, Ontario, M5S 1A1
Internet: mccool@dgp.utoronto.ca

ABSTRACT

A group-theoretic analysis is applied to find the transformation of the homogeneous control points of k-dimensional Bézier simplices (such as two dimensional triangles and three dimensional tetrahedra) under a k-dimensional projective reparameterization. This transformation has applications in the perspective projection of textures represented as triangular spline intensity surfaces, in arbitrary Bézier simplicial subdivision, and in weight normalization.

The theoretical results contained in this exposition are generalizations of similar results for 1-dimensional Bézier curves, reported by Richard R. Patterson in 1985.

RÉSUMÉ

Une analyse en théorie des groupes est faite pour trouver la transformation des points de contrôle homogènes pour un simplexe Bézier de dimension k, en faisant une réparaméterisation projectif de dimension k. On peut utiliser cette transformation pour la projection en perspective des textures faites avec les surfaces splines d'intensité, pour la sous-division des simplexes Béziers, et pour régulariser les poids pour la normalisation.

Les résultats théoretiques qu'on présente dans cet exposé sont des généralisations des résultats que Richard R. Patterson a présenté en 1985 pour des courbes Béziers ayant une seule dimension.

KEYWORDS: rational Bézier spline curves, triangles, and surfaces; perspective; projective transformation and reparameterization; rational reparameterization; texture mapping.

1 INTRODUCTION

It is a well known result that the perspective projection of a spline space curve or surface can be represented as a rational spline curve or surface [7]. It is less well understood how an intensity function, or texture, represented as a spline is transformed by the same operation. In the space spline case, the positions and weights of the control points are changed. In the case of an intensity distribution, the perspective transformation results instead in a projective reparameterization. It will be shown in this paper that this can also be represented as a change in the values and weights of the control points. The required transformation operates by blending the same coordinate of different control points, rather than blending different coordinates of each control point as in the space spline.

A projective transformation, of which the perspective transformation is one case, can be represented as a fractional linear transformation. In one dimension, this transformation is given by

$$u = \frac{a + bx}{c + dx}.$$

If this transformation is substituted into a polynomial of order n, then the result is a ratio of polynomials of order n. We will call such ratios of polynomials *rational functions*. Substitution of this same transformation into a rational function will yield another rational function of the same order. Rational functions of a given order are therefore closed under projective reparameterization. In the same way, m-variate rational functions are closed under fractional linear m-variate transforms of the form

$$u_k = \frac{a + \sum_{i=1}^{m} b_i x_i}{c + \sum_{j=1}^{m} d_j x_j}.$$

Results for the projective reparameterization of nonuniform rational B-spline curves were presented by Lee [5]. The tensor product B-spline basis function typically used for B-spline surfaces does not remain a tensor product under projection, so it is difficult to use that result directly. Instead, we will work with the simpler Bézier triangular surfaces first with the hope that results can later be extended to triangular B-spline surfaces.

A projective transformation can be represented as a change from one homogeneous coordinate system to another. We would like to represent a change of parametric coordinate system as a transformation of control

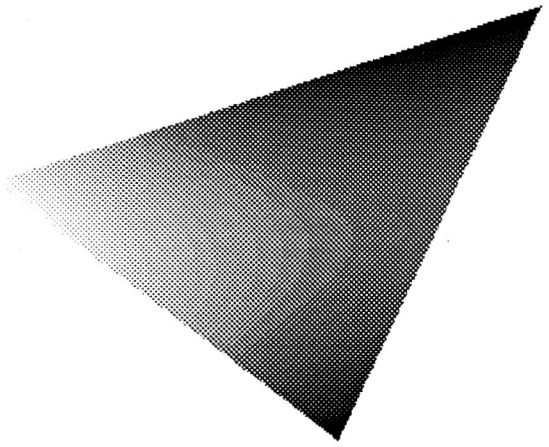

Figure 1: Example of a quadratic Bézier patch used to define the intensity of a surface.

points and weights. For Bézier spline curves, a general solution was found by Patterson under a univariate reparameterization [6]. This paper is a generalization of that result to Bézier splines over triangular, tetrahedral, and higher-dimensional simplicial parameter domains.

Our generalization hinges on the use of suitable notation; the symmetrical index notation used here is based upon that used in [8]. This notation will be introduced as needed, but a full reference is available in Appendix A.

This result has many potential applications besides texture mapping, and several are given in this paper. Texture mapping, however, *requires* multidimensional projective reparameterization and so has not been covered previously.

Suppose the colour variation in a texture is represented as a functional spline, i.e. a representation composed of patches such as the one shown in Figure 1. Such a representation could be reconstructed from a discretized texture via an interpolation algorithm, or it could be the result of a radiosity algorithm using spline interpolation [1, 2]. If the texture is mapped via a projective transformation onto a flat surface and then reprojected via perspective, the overall transformation is a single two-dimensional projective reparameterization. The control points and weights of the texture can then be transformed directly from texture space to screen space, resulting in an analytic representation of the texture, in screen space. Antialiasing filters and rasterization algorithms can then take advantage of this explicit representation. This particular application is covered in more detail in Section 4.2.

We first derive the reparameterization transformation and then show its application to several sample problems including texture mapping.

2 BÉZIER SIMPLICES

The k-dimensional Bézier simplices are defined in terms of a k-dimensional barycentric coordinate system [3] with $k+1$ redundant but symmetrical parameters. Like the Bézier curves, k-dimensional Bézier simplices are built upon recursive blends. Each blend computes an affine combination of $k+1$ other points. Consider the two dimensional triangular case: a convex blend of three points $\mathbf{P}, \mathbf{Q}$, and $\mathbf{R}$ is given by $p\mathbf{P} + q\mathbf{Q} + r\mathbf{R}$, where $p, q, r \in \mathbb{R}$ and $p+q+r = 1$. Bézier tetrahedral simplices are similarily built from blends of four points, using four coordinates with three degrees of freedom.

Bézier simplices are still multivariate polynomials, however. Define the basis of nth-order, k-dimensional multinomials as a row vector:

$$\begin{aligned}\mathbf{F}_{n,k}(\mathbf{t}_k) &:= \left(\left(\begin{array}{c} n \\ \mathbf{i}_k \end{array}\right) \mathbf{t}_k^{\mathbf{i}_k}\right) \\ &= \left(\left(\begin{array}{c} n \\ i_0\ i_1\ \ldots\ i_k \end{array}\right) t_0^{i_0} t_1^{i_1} \ldots t_k^{i_k}\right) \\ &= \left(\mathrm{F}_{n,k,\mathbf{i}}(\mathbf{t}_k)\right).\end{aligned}$$

Any multinomial can be defined as the dot product of this vector with a coefficient vector, although to represent constant terms we must set $t_k = 1$. The number of basis functions required is given by

$$N_{n,k} = \left(\begin{array}{c} n+k \\ k \end{array}\right).$$

In a homogeneous coordinate system, we consider any non-zero multiple of the parameter vector to be equivalent to any other. If we represent this arbitrary multiplicative constant by s, then the k-dimensional homogeneous multinomial basis becomes

$$\begin{aligned}\mathbf{F}^H_{n,k}(\mathbf{t}_k) &:= \left[\left(\begin{array}{c} n \\ \mathbf{i}_k \end{array}\right) (s\mathbf{t}_k)^{\mathbf{i}_k}\right] \\ &= \left[s^n \mathrm{F}_{n,k,\mathbf{i}}(\mathbf{t}_k)\right].\end{aligned}$$

We use square brackets to distinguish homogeneous vectors from ordinary vectors. The non-homogeneous form can be recovered by dividing by s^n.

Bézier simplices are defined in barycentric coordinate systems, so we must have $|\mathbf{t}_k| = 1$. We define the Bernstein basis functions, upon which Bézier simplices are built, by removing the extra degree of freedom from the multinomial form:

$$\begin{aligned}&\mathbf{B}_{n,k}(\mathbf{t}_{k-1}) \\ &\quad := \mathbf{F}_{n,k}\left(t_0, t_1, \ldots, t_{k-1}, \left(1 - \sum_{j=0}^{k-1} t_j\right)\right)\end{aligned}$$

Note that this transformation of parameters may be represented by a matrix, and in fact we do so in Section 3.3. We have assumed that $t_k = 1$ before the transformation.

With these definitions, we can finally define the nth order, k-dimensional homogeneously parameterized Bernstein-Bézier basis functions as

$$\begin{aligned}
&\mathbf{B}^H_{n,k}(\mathbf{t}_k) \\
&:= \mathbf{F}^H_{n,k}\left(t_0, t_1, \ldots, t_{k-1}, \left(t_k - \sum_{j=0}^{k-1} t_j\right)\right) \\
&= t_k^n \mathbf{F}_{n,k}\left(\frac{t_0}{t_k}, \frac{t_1}{t_k}, \ldots, \frac{t_{k-1}}{t_k}, \left(1 - \sum_{j=0}^{k-1} \frac{t_j}{t_k}\right)\right) \\
&= \left[\mathrm{B}^H_{n,k,\mathbf{i}}(\mathbf{t}_k)\right].
\end{aligned}$$

The nth order, k-dimensional homogeneous Bézier simplices $\mathbf{P}^H_{n,k}$ are defined by

$$\mathbf{P}^H_{n,k}(\mathbf{t}_k) := \mathbf{B}^H_{n,k}(\mathbf{t}_k)\mathbf{P}^H,$$

where $\mathbf{P}^H$ is a $N_{n,k} \times (m+1)$ matrix; a column vector of row vectors, each row being a m-channel homogeneous *control vertex* of the form:

$$\mathbf{P}^H_{\mathbf{i}} = [w_{\mathbf{i}} p^1_{\mathbf{i}}, w_{\mathbf{i}} p^2_{\mathbf{i}}, \ldots, w_{\mathbf{i}} p^m_{\mathbf{i}}, w_{\mathbf{i}} \beta_{\mathbf{i}}].$$

The superscript is a label, not an exponent. The switch value $\beta_{\mathbf{i}}$ is 1 for finite control points and 0 for points at infinity. The value $w_{\mathbf{i}}$ is called the *weight* of control vertex $\mathbf{P}^H_{\mathbf{i}}$.

The matrix of control vertices can be factored into the product of a diagonal weight matrix W and a normalized version of the control points $\mathbf{Q}^H$:

$$\begin{aligned}
\mathbf{P}^H &= W\mathbf{Q}^H \\
&= \mathrm{diag}(w_{\mathbf{i}}) \left([p^1_{\mathbf{i}}, p^2_{\mathbf{i}}, \ldots p^m_{\mathbf{i}}, \beta_{\mathbf{i}}]\right).
\end{aligned}$$

This defines a Bézier simplex over the projective parameter space $\mathbb{R}P^k$. Therefore, $\mathbf{P}^H_{n,k}$ is an injective map from $\mathbb{R}P^k$ into $\mathbb{R}P^m$, where $\mathbb{R}P^\ell$ is the real projective space of dimension ℓ.

The form of the nth order rational Bézier simplex, after normalization, is given by

$$\mathbf{R}_{n,k}(\mathbf{t}_{k-1}) = \left(\frac{\sum_{|\mathbf{i}|=n} \mathrm{B}_{n,k,\mathbf{i}}(\mathbf{t}_{k-1}) w_{\mathbf{i}} p^q_{\mathbf{i}}}{\sum_{|\mathbf{i}|=n} \mathrm{B}_{n,k,\mathbf{i}}(\mathbf{t}_{k-1}) w_{\mathbf{i}} \beta_{\mathbf{i}}}\right),$$

which is a m-dimensional row vector indexed by q, the channel index. Because of the fact that the Bernstein basis forms a partition of unity, this reduces to a non-rational form if all weights are equal and all control points are finite:

$$\mathbf{R}_{n,k}(\mathbf{t}_{k-1}) = \left(\sum_{|\mathbf{i}|=n} \mathrm{B}_{n,k,\mathbf{i}}(\mathbf{t}_{k-1}) p^q_{\mathbf{i}}\right)$$

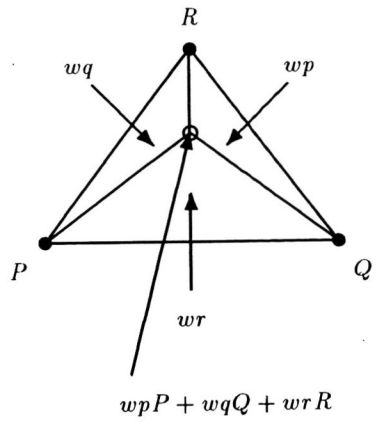

Figure 2: Homogeneous barycentric coordinates have $wp + wr + wq = w$, where w is an arbitrary non-zero constant. In this figure, if wp, wr, and wq are given by the areas shown, and w is the area of triangle PRQ, then the homogeneous barycentric coordinates of the central point are $[wp, wr, wq]$.

3 REPARAMETERIZATION

In this section we will show that the homogeneous reparameterization of a linearly parameterized functional Bézier patch is given by a matrix multiplication combining the same coordinates in all control points. This characterization admits several applications that will be examined in Section 4.

3.1 Homogeneous Multinomials

Define an isomorphism ϕ_n between a vector space composed of tensors $\mathbf{s}_{n,k}$ and the vector space of all multinomials with k variables and total degree n as

$$\phi_n(\mathbf{s}_{n,k}) = \sum_{|\mathbf{i}_k|=n} s_{\mathbf{i}_k} \mathbf{x}_k^{\mathbf{i}_k}.$$

The power on each variable in the multinomial identifies that term's coefficient as the corresponding element of the tensor.

The homogeneous parameter space of a linearly parameterized k-dimensional object is given by $\mathbf{s}_{1,k}$. By convention we will assume that the last element in lexicographic order is the "homogeneous" parameter, set to 1 (or 0, for infinite points) when normalized. With all the above definitions in place, we can express the basic single-term homogeneous multinomials (defined in Section 2) as

$$\mathbf{F}^H_{n,k}(\mathbf{s}_{1,k}) := \phi_n^{-1}\left(\phi_1\left(\mathbf{s}_{1,k}\right)^n\right)$$

For $n = 1, k = 2$ we have

$$\mathbf{F}^H_{1,2}(\mathbf{s}_{1,2}) = \phi_1^{-1}\left(s_{001}x_0 + s_{010}x_1 + s_{100}x_2\right)$$

$$= \begin{bmatrix} F_{001} & = & s_{001} \\ F_{010} & = & s_{010} \\ F_{100} & = & s_{100} \end{bmatrix}^T,$$

and for $n = 2, k = 2$ we have

$$\begin{aligned} \mathbf{F}^H_{2,2}(\mathrm{s}_{1,2}) &= \phi_2^{-1}\left((s_{001}x_0 + s_{010}x_1 + s_{100}x_2)^2\right) \\ &= \phi_2^{-1}\begin{pmatrix} s_{001}^2x_0^2 + s_{010}^2x_1^2 + s_{100}^2x_2^2 \\ +\,2s_{001}s_{010}x_0x_1 \\ +\,2s_{001}s_{100}x_0x_2 \\ +\,2s_{010}s_{100}x_1x_2 \end{pmatrix} \\ &= \begin{bmatrix} F_{002} & = & s_{001}^2 \\ F_{011} & = & 2s_{001}s_{010} \\ F_{020} & = & s_{010}^2 \\ F_{101} & = & 2s_{001}s_{100} \\ F_{110} & = & 2s_{010}s_{100} \\ F_{200} & = & s_{100}^2 \end{bmatrix}^T \end{aligned}$$

3.2 Projective Reparameterization

A projective reparameterization of a k-dimensional homogeneous multinomial can be represented by a $(k+1)\times(k+1)$ matrix postmultiplying the parameter vector. For any given $(k+1)\times(k+1)$ matrix $M = [m_{ij}]$, the reparameterized multinomials are given by

$$\begin{aligned} \mathbf{F}^H_{n,k}(\mathrm{s}_{1,k}M) &:= \phi_n^{-1}\left(\phi_1\left(\mathrm{s}_{1,k}M\right)^n\right) \\ &= \phi_n^{-1}\left(\phi_1\left(\mathrm{s}_{1,k}\right)^n\right)\delta_{n,k}(M) \end{aligned}$$

We can find the $N_{n,k}\times N_{n,k}$ matrix $\delta_{n,k}(M)$ by direct symbolic computation[1] or by transforming the generating function into a combinatoric expression. See Appendix C for specific examples, and Appendix B for the combinatoric form of this operator.

The combinatoric operator $\delta_{n,k}$ is a homomorphism of the group of $(k+1)\times(k+1)$ matrices; a proof is provided in Appendix B. Therefore inverses map to inverses, and matrix multiplications inside the isomorphism can be moved outside by mapping the component matrices through the $\delta_{n,k}$ operator.

3.3 Reparameterization of Bézier Simplices

The $\delta_{n,k}(M)$ matrices only allow reparameterization of homogeneous multinomials. To reparameterize Bézier simplices, we have to preceed the reparameterization matrix by a matrix that transforms the multinomials into Bézier basis functions. Such a matrix for triangles is given by

$$M_1 = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix}.$$

The inverse of this matrix, which we shall need later, is given by

$$M_1^{-1} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}.$$

[1] Maple code is available via anonymous ftp from dgp.utoronto.ca (128.100.1.129).

We will also define $M_n = \delta_{n,k}(M_1)$; since $\delta_{n,k}$ is a homomorphism, $M_n^{-1} = \delta_{n,k}(M_1^{-1})$. The generalization of these matrices to higher dimensions should be obvious. Now

$$\begin{aligned} \mathbf{B}^H_{n,k} &= \mathbf{F}^H_{n,k}(\mathrm{s}_{1,k}M_1) \\ &= \mathbf{F}^H_{n,k}(\mathrm{s}_{1,k})M_n. \end{aligned}$$

Define the matrix operator $\rho_{n,k}$ such that

$$\mathbf{B}^H_{n,k}(\mathrm{s}_{1,k}A) = \mathbf{B}^H_{n,k}(\mathrm{s}_{1,k})\rho_{n,k}(A).$$

We have

$$\begin{aligned} &\mathbf{F}^H_{n,k}(\mathrm{s}_{1,k})\delta_{n,k}(A)M_n \\ &= \mathbf{F}^H_{n,k}(\mathrm{s}_{1,k}A)M_n \\ &= \mathbf{B}^H_{n,k}(\mathrm{s}_{1,k}A) \\ &= \mathbf{B}^H_{n,k}(\mathrm{s}_{1,k})\rho_{n,k}(A) \\ &= \mathbf{F}^H_{n,k}(\mathrm{s}_{1,k})M_n\rho_{n,k}(A). \end{aligned}$$

Therefore, $\rho_{n,k}$ is a conjugate representation to $\delta_{n,k}$:

$$\begin{aligned} \rho_{n,k}(A) &= M_n^{-1}\delta_{n,k}(A)M_n \\ &= \delta_{n,k}(M_1^{-1}AM_1) \end{aligned}$$

Recall our definition of a patch expressed as a matrix product:

$$\mathbf{P}^H_{n,k}(\mathrm{s}_{1,k}) = \mathbf{B}^H_{n,k}(\mathrm{s}_{1,k})\mathbf{P}^H.$$

If we substitute our expression for an arbitrary projective reparameterization, we get

$$\begin{aligned} \mathbf{P}^H_{n,k}(\mathrm{s}_{1,k}A) &= \mathbf{B}^H_{n,k}(\mathrm{s}_{1,k}A)\mathbf{P}^H \\ &= \mathbf{B}^H_{n,k}(\mathrm{s}_{1,k})\rho_{n,k}(A)\mathbf{P}^H \\ &= \mathbf{B}^H_{n,k}(\mathrm{s}_{1,k})\mathbf{V}^H, \end{aligned}$$

where

$$\mathbf{V}^H = \rho_{n,k}(A)\mathbf{P}^H$$

is the new set of control points. This transformation can also be expressed as

$$\mathbf{V}^H = \rho_{n,k}(A)W\mathbf{Q}^H$$

for normalized control points. If $\rho_{n,k}(A)$ is diagonal, then only the weights need to be changed to carry out the reparameterization.

3.4 Complexity

A projective reparameterization of a Bézier simplex corresponds to a projective transformation of each coordinate of its control points, treated together as a vector in $N_{n,k}$-dimensional space. The transformation of these control points is combinatorically related to the reparameterization matrix A, and is given by the $N_{n,k}\times N_{n,k}$ matrix $\rho_{n,k}(A)$. Performing the matrix multiplication $\rho_{n,k}(A)\mathbf{P}^H$ requires $O((m+1)N_{n,k}^2)$ operations for m channels, with no assumptions about weights. It should be noted that $N_{n,k} = O(n^k)$, so the complexity may be

written $O(mn^{2k})$. Compare this to a spatial projective transformation given by a $(k+1) \times (k+1)$ matrix B: $\mathbf{V}^H = \mathbf{P}^H B$. This multiplication requires only $O((k+1)^2 N_{n,k}) = O(k^2 n^k)$ operations.

Summary: the upper bound on complexity of general reparameterization has an extra factor of $O(mn^k/k^2)$ over spatial transformation. This result does not preclude special cases from being more efficient, however. For some matrices A, $\rho_{n,k}(A)$ may be sparse, diagonal, or afford a recursive factorization into sparse matrices.

4 APPLICATIONS

In this section we cover some specific applications of the relationship we have derived.

4.1 Weight Normalization

It is known in the case of rational Bézier curves that varying patterns of weights can lead to the same space curve, albeit with different parameterizations. A curve is considered to have a normalized weight pattern when $w_{0n} = 1$ and $w_{n0} = 1$.

Correspondingly, we define the weight pattern of a surface to be normalized when the weights at all "corners" have been set to 1. Corners have one index set to n and all others set to 0. This normalization must be accomplished without changing the shape of the surface, and preferably by only changing the weights.

Consider first the problem of normalizing a surface that uses the homogeneous multinomials $\mathbf{F}^H_{n,k}$ as a basis. The surface is given by

$$\mathbf{P}^H(\mathbf{s}_{1,k}) = \mathbf{F}^H_{n,k}(\mathbf{s}_{1,k}) W \mathbf{Q}^H,$$

for some matrix of normalized homogeneous control points $\mathbf{Q}^H$ and matrix of weights $W = \mathrm{diag}(w_{\mathbf{i}})$. Define the reparameterizing transformation matrix

$$A_{\mathbf{r}_k} = \mathrm{diag}(r_0, r_1, \ldots, r_k).$$

This transformation fixes the corner parameter values in $\mathbb{R}P^k$, since constant factors can be ignored in projective space.

$$\begin{aligned}
r_0(1,0,0,\ldots,0) &= (1,0,0,\ldots,0) A_{\mathbf{r}_k} \\
r_1(0,1,0,\ldots,0) &= (0,1,0,\ldots,0) A_{\mathbf{r}_k} \\
r_2(0,0,1,\ldots,0) &= (0,0,1,\ldots,0) A_{\mathbf{r}_k} \\
&\vdots \\
r_k(0,0,0\ldots,1) &= (0,0,0,\ldots,1) A_{\mathbf{r}_k}.
\end{aligned}$$

If the parametric corner points are fixed in projective space, then the shape of the resulting curve will be unchanged.

Consider $\delta_{n,k}(A_{\mathbf{r}_k})$. Each entry in this matrix is composed of sums of products of entries of $A_{\mathbf{r}_k}$. Only diagonal elements contain terms that are composed entirely of diagonal elements of A. Therefore, if $A_{\mathbf{r}_k}$ is diagonal then so is $\delta_{n,k}(A_{\mathbf{r}_k})$ and $\delta_{n,k}(A_{\mathbf{r}_k})W$. The weights will change but the Euclidean positions of the control vertices will not.

Create the vector $\mathbf{v}_{n,k}$ to contain the diagonal entries of $\delta_{n,k}(A_{\mathbf{r}_k})$. It can be seen that the ith element of $\mathbf{v}$ will be equal to $\mathbf{r}^{\mathbf{i}}_k$. Therefore, to normalize the corner weights, we simply set the elements of $\mathbf{r}_k$ as follows:

$$\begin{aligned}
r_0 &= \sqrt[n]{\frac{1}{w_{000\ldots00n}}} \\
r_1 &= \sqrt[n]{\frac{1}{w_{000\ldots0n0}}} \\
&\vdots \\
r_{k-1} &= \sqrt[n]{\frac{1}{w_{0n0\ldots000}}} \\
r_k &= \sqrt[n]{\frac{1}{w_{n00\ldots000}}}
\end{aligned}$$

Furthermore, let

$$\begin{aligned}
B_{\mathbf{r}_k} &= M_1 A_{\mathbf{r}_k} M_1^{-1} \\
&= \begin{pmatrix}
r_0 & 0 & 0 & \ldots & 0 & r_0 - r_k \\
0 & r_1 & 0 & \ldots & 0 & r_1 - r_k \\
0 & 0 & r_2 & \ldots & 0 & r_2 - r_k \\
\vdots & & & & & \\
0 & 0 & 0 & \ldots & r_{k-1} & r_{k-1} - r_k \\
0 & 0 & 0 & \ldots & 0 & r_k
\end{pmatrix}
\end{aligned}$$

This reparameterization fixes the projective parameters

$$\begin{aligned}
r_0(1,0,0,\ldots,0,1) &= (1,0,0,\ldots,0,1) B_{\mathbf{r}_k} \\
r_1(0,1,0,\ldots,0,1) &= (0,1,0,\ldots,0,1) B_{\mathbf{r}_k} \\
r_2(0,0,1,\ldots,0,1) &= (0,0,1,\ldots,0,1) B_{\mathbf{r}_k} \\
&\vdots \\
r_k(0,0,0,\ldots,0,1) &= (0,0,0,\ldots,0,1) B_{\mathbf{r}_k}.
\end{aligned}$$

Since the parametric corners of the Bézier patch are fixed, the shape will remain unchanged.

Note that $\rho_{n,k}(B_{\mathbf{r}_k}) = \delta_{n,k}(M_1^{-1} B_{\mathbf{r}_k} M_1) = \delta_{n,k}(A_{\mathbf{r}_k})$, so we can state the Bézier reparameterization using $B_{\mathbf{r}_k}$:

$$\begin{aligned}
\mathbf{P}^H(\mathbf{s}_{1,k} B_{\mathbf{r}_k}) &= \mathbf{B}^H_{n,k}(\mathbf{s}_{1,k} B_{\mathbf{r}_k}) W \mathbf{Q}^H \\
\hat{\mathbf{P}}^H(\mathbf{s}_{1,k}) &= \mathbf{B}^H_{n,k}(\mathbf{s}_{1,k}) \rho_{n,k}(B_{\mathbf{r}_k}) W \mathbf{Q}^H \\
&= \mathbf{B}^H_{n,k}(\mathbf{s}_{1,k}) \delta_{n,k}(A_{\mathbf{r}_k}) W \mathbf{Q}^H \\
&= \mathbf{B}^H_{n,k}(\mathbf{s}_{1,k}) W' \mathbf{Q}^H.
\end{aligned}$$

The matrix $B_{\mathbf{r}_k}$ gives the same transformation as $A_{\mathbf{r}_k}$, but with respect to the Bernstein-Bézier basis.

4.2 Perspective Reparameterization

Perspective projection is the most obvious and inescapable non-affine projective transformation in computer graphics. The perspective projection of a flat texture mapped polygon results in a projective reparameterization of the colour function. Hence, if the texture is

represented as a two-dimensional functional Bézier surface, then the rational Bézier representation in screen coordinates may be found.

We first review the parameterization, projection, and transformation composition process that takes place during texture mapping.

4.2.1 Texture Transformations

When a texture is mapped onto a flat polygon, some sort of parameterization is used. For the moment, assume that the map between texture coordinates $\mathbf{u}_{1,2}$ and polygon coordinates $\mathbf{v}_{1,2}$ is in the form

$$\begin{aligned} v_{001} &= \frac{m_{00}u_{001} + m_{10}u_{010} + m_{20}u_{100}}{m_{02}u_{001} + m_{12}u_{010} + m_{22}u_{100}} \\ v_{010} &= \frac{m_{01}u_{001} + m_{11}u_{010} + m_{21}u_{100}}{m_{02}u_{001} + m_{12}u_{010} + m_{22}u_{100}} \\ v_{100} &= 1 \end{aligned}$$

This projective transformation includes as a special case the affine transforms, including the identity. It also allows any four points to be mapped to any other four points, while preserving straight lines.

An equivalent transformation can be expressed more succinctly in homogeneous coordinates:

$$\begin{aligned} &[v_{001}, v_{010}, v_{100}] \\ &\quad = [u_{001}, u_{010}, u_{100}] \begin{pmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ m_{20} & m_{21} & m_{22} \end{pmatrix}, \end{aligned}$$

which can be written as

$$\mathbf{v}_{1,2} = \mathbf{u}_{1,2}M.$$

The variables u_{100} and v_{100} are dummy, non-zero variables that will cancel when we eventually normalize. The matrix M also has an extra degree of freedom.

The polygon coordinate space is now transformed into homogeneous world space coordinates, $\mathbf{x}_{1,3}$, using the object modelling transformation

$$\begin{aligned} &[x_{0001}, x_{0010}, x_{0100}, x_{1000}] \\ &\quad = [v_{001}, v_{010}, v_{100}] \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \end{pmatrix}. \end{aligned}$$

We can also write this

$$\mathbf{x}_{1,3} = \mathbf{v}_{1,2}A$$

Note that the world coordinate system has an extra dimension. Viewing transformations follow, composed with a projection matrix. The projection matrix discards a dimension, so that we have a final transformation to the screen coordinate system $\mathbf{s}_{1,2}$:

$$\begin{aligned} &[s_{001}, s_{010}, s_{100}] \\ &\quad = [x_{0001}, x_{0010}, x_{0100}, x_{1000}] \begin{pmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \\ b_{30} & b_{31} & b_{32} \end{pmatrix}. \end{aligned}$$

Again, we can write this as

$$\mathbf{s}_{1,2} = \mathbf{x}_{1,3}B.$$

The composed transform can be found by matrix multiplication, and will be a strictly two-dimensional projective transformation. The intermediate three-dimensional world space may be ignored.

$$\mathbf{s}_{1,2} = \mathbf{u}_{1,2}MAB = \mathbf{u}_{1,2}C.$$

4.2.2 Bézier Texture Projection

Assume that our texture is represented not as an array of pixels, but as a continuous bivariate polynomial intensity function defined over a triangle. Bézier triangles form a basis for bivariate polynomials within a triangle, so assume that we have homogeneous control vertices $\mathbf{g}_{n,2} = [w_{\mathbf{i}}g_{\mathbf{i}}, w_{\mathbf{i}}]$. Note that this is a functional form, so the control vertices are not associated with a specific position in space. They are only associated with a specific basis function. We also assume here a monochrome texture map. A generalization to a colour texture map would simply use higher-dimensional control vertices $\mathbf{g}_{n,m}$.

The projected texture is a rational function $RG : \mathbb{R}^2 \rightarrow \mathbb{R}$, which is the normalized version of a homogeneously parameterized function $G^H_{n,2} : \mathbb{R}P^2 \rightarrow \mathbb{R}P^1$.

Described using homogeneous texture coordinates,

$$\begin{aligned} &G^H_{n,2}(u_{001}, u_{010}, u_{100}) \\ &\quad = \sum_{|\mathbf{i}|=n} \mathrm{B}^H_{n,1,\mathbf{i}}(u_{001}, u_{010}, u_{100}) \cdot [w_{\mathbf{i}}g_{\mathbf{i}}, w_{\mathbf{i}}]. \end{aligned}$$

If we stack the control vertices into a matrix

$$\mathbf{P}^H := \left([w_{\mathbf{i}}g_{\mathbf{i}},\ w_{\mathbf{i}}]\right)$$

we have the form

$$G^H_{n,2}(\mathbf{u}_{1,2}) = \mathbf{B}^H_{n,2}(\mathbf{u}_{1,2})\mathbf{P}^H.$$

Texture mapping, as we have seen, results in a projective map $\mathbf{s} = \mathbf{u}C$ from our texture coordinates to screen space. The inverse of this map is $\mathbf{u} = \mathbf{s}C^{-1}$. Using the results of Section 3, we can express the reparameterization due to the transformation $\mathbf{u} = \mathbf{s}C^{-1}$ as

$$\begin{aligned} G^H_{n,2}(\mathbf{s}_{1,2}C^{-1}) &= \mathbf{B}^H_{n,2}(\mathbf{s}_{1,2}C^{-1})\mathbf{P}^H \\ \hat{G}^H_{n,2}(\mathbf{s}_{1,2}) &= \mathbf{B}^H_{n,2}(\mathbf{s}_{1,2})\left(\rho_{n,2}(C^{-1})\mathbf{P}^H\right). \end{aligned}$$

We can set the extra homogeneous screen coordinate to 1, hence removing the redundant degree of freedom. We will still be able to address all points on the screen. If we define

$$\mathbf{V}^H = \rho_{n,k}(C^{-1})\mathbf{P}^H = ([w'_{\mathbf{i}}g'_{\mathbf{i}}, w'_{\mathbf{i}}]),$$

then

$$\begin{aligned} \hat{G}^H_{n,2}(s_{001}, s_{010}) &= \mathbf{B}^{H'}_{n,2}(s_{001}, s_{010}, 1)\mathbf{V}^H \\ RG(s_{001}, s_{010}) &= \frac{\sum_{|\mathbf{i}|=n} \mathrm{B}_{n,2,\mathbf{i}}(s_{001}, s_{010})w'_{\mathbf{i}}g'_{\mathbf{i}}}{\sum_{|\mathbf{i}|=n} \mathrm{B}_{n,2,\mathbf{i}}(s_{001}, s_{010})w'_{\mathbf{i}}}, \end{aligned}$$

4.2.3 Evaluation

To evaluate the texture map in screen space, we need to evaluate $RG(s_{001}, s_{010})$ which is a ratio of linear combinations of Bézier basis functions *in screen space*. Unfortunately, every new output control value depends, in general, on every input control value. This non-locality limits the practical use of this technique, unless the texture is broken up into many segments of low order. For cubic triangles, $N_{3,2} = 10$, and so a brute-force transformation of a single segment requires 200 multiplications and 180 additions, not to mention the computation of $\rho_{n,k}(C^{-1})$. Since $N_{2,2} = 6$, quadratic triangles require 72 multiplications and 60 additions. Linear triangles require 18 multiplications and 12 additions.

In the special case in which all the control points have been normalized so all $w_{\mathbf{i}} = 1$, then half the multiplications may be eliminated. This is the case where the source is always known to be a polynomial, which can be arranged in a texture mapping application.

4.3 Arbitrary Subdivision

Suppose we were given a triangular rational Bézier patch and needed to extract some arbitrary triangular subpatch, with the subpatch represented as a triangular Bézier patch (i.e. in terms of control points).

Define the parametric corners of the subpatch by the homogeneous parameters

$$\begin{aligned} \mathbf{a} &= (a_{001}, a_{010}, a_{100}), \\ \mathbf{b} &= (b_{001}, b_{010}, b_{100}), \\ \mathbf{c} &= (c_{001}, c_{010}, c_{100}). \end{aligned}$$

To create the subpatch, we have to describe a transformation that maps

$$\begin{aligned} \mathbf{a}M &\rightarrow \mathbf{s}_a = (1,0,0), \\ \mathbf{b}M &\rightarrow \mathbf{s}_b = (0,1,0), \\ \mathbf{c}M &\rightarrow \mathbf{s}_c = (0,0,1). \end{aligned}$$

Such a transformation is given by

$$M^{-1} = M_1^{-1} \begin{pmatrix} a_{001} & a_{010} & a_{100} \\ b_{001} & b_{010} & b_{100} \\ c_{001} & c_{010} & c_{100} \end{pmatrix} M_1.$$

We define this matrix by its inverse because the inverse is exactly what will be needed.

Recall that a rational, triangular Bézier patch is given in homogeneous coordinates by

$$\mathbf{P}^H_{n,2}(\mathbf{u}_{1,2}) = \mathbf{B}^H_{n,2}(\mathbf{u}_{1,2})\mathbf{P}^H$$

We can reparameterize this patch so that the desired triangle lies in the standard parameter range using the parameter transformation $\mathbf{u}_{1,2} = \mathbf{v}_{1,2}M^{-1}$:

$$\begin{aligned} \mathbf{P}^H_{n,2}(\mathbf{v}_{1,2}M^{-1}) &= \mathbf{B}^H_{n,2}(\mathbf{v}_{1,2}M^{-1})\mathbf{P}^H, \\ \hat{\mathbf{P}}^H_{n,2}(\mathbf{v}_{1,2}) &= \mathbf{B}^H_{n,2}(\mathbf{v}_{1,2})\left(\rho_{n,2}(M^{-1})\mathbf{P}^H\right), \\ &= \mathbf{B}^H_{n,2}(\mathbf{v}_{1,2})\mathbf{V}^H. \end{aligned}$$

The control points of the subdivided patch are therefore $\mathbf{V}^H = \rho_{n,2}(M^{-1})\mathbf{P}^H$. Once again, this is a relatively expensive operation; however, the fact that it is totally general can be useful in some cases. There are, for example, no restrictions on the orientation of the subpatch with respect to the larger patch, and no special cases.

5 CONCLUSIONS

A multidimensional generalization of a 1-D result has been presented. The general theoretical result has many applications, although specialization is needed to derive efficient algorithms. Application of the techniques presented in this paper results in algorithms for a variety of applications of rational reparameterization of polynomial spline surfaces, including weight normalization, texture mapping, and subdivision.

6 ACKNOWLEDGEMENTS

The long-term financial support of the Natural Sciences and Engineering Research Council of Canada and the Information Technology Research Centre of Ontario is gratefully acknowledged. Our graphics lab has also benefited from the generous financial support of Alias Research, Apple Computer, DEC and Xerox Corp.

The referees provided useful suggestions that led to an improvement in the notation from horrific to merely baroque; the readers should be as thankful as the author is! Eugene Fiume provided additional assistance in this regard.

Timely and appreciated assistance in the translation of the abstract into a *résumé français* was provided by Michiel van de Panne, Kim H. Veltman, and Marc Ouellette.

A NOTATION

The objects used to construct Bézier simplices are most properly understood as tensors. A symmetrical multiindex notation is used to refer to elements of these tensors, and they are implicitly stacked in lexicographical order. This is formalized in the following sections.

A.1 Tensors and Multiindices

Define the simplicial tensor $\mathbf{s}_{n,k} := \{s_{\mathbf{i}_k}\}$ such that $\mathbf{i}_k = \{i_0, i_1, \ldots, i_k\}$ with $i_j \in \mathbb{N}$ and $|\mathbf{i}_k| := \sum_{j=0}^{k} i_j = n$. Note that the multiindex $\mathbf{i}_k$ has $k+1$ elements, although it is redundant and should be considered as only having dimension k.

We will call k the *order* of the tensor $\mathbf{s}_{n,k}$, and n the *dimension*. The short form $\mathbf{s}_k := \mathbf{s}_{1,k}$ may be used for one-dimensional vectors, hence $\mathbf{i}_k = \mathbf{i}_{1,k}$. When possible and unambiguous, we will leave out commas in subscripts of the elements of tensors.

Examples:

$$\begin{aligned}
\mathbf{s}_{1,2} &= (s_{001}, s_{010}, s_{100}) \\
&\approx \left\{ \begin{array}{c} s_{100}, \\ s_{001},\ s_{010} \end{array} \right\} \\
\mathbf{s}_{2,2} &= (s_{002}, s_{011}, s_{020}, s_{101}, s_{110}, s_{200}) \\
&\approx \left\{ \begin{array}{c} s_{200}, \\ s_{101},\ s_{110}, \\ s_{002},\ s_{101},\ s_{020} \end{array} \right\}
\end{aligned}$$

For a one-dimensional tensor $\mathbf{x}_k$, define exponentiation by a multiindex of the same dimension as

$$\mathbf{x}_k^{\mathbf{i}_k} := \prod_{j=0}^{k} x_j^{i_j} = x_0^{i_0} x_1^{i_1} \ldots x_k^{i_k}$$

Define the multiindex combination, with $|\mathbf{i}_k| = n$, as

$$\begin{aligned}
\left(\begin{array}{c} n \\ \mathbf{i}_k \end{array} \right) &:= \frac{n}{\prod_{j=0}^{k} i_j!} \\
&= \frac{n}{i_0!\, i_1! \ldots i_k!}.
\end{aligned}$$

A.2 Stacking

Tensors are somewhat awkward to work with, especially tensors of the form used here, where the entire multiindex has to be considered to determine a valid (or equivalently, non-zero) element of the tensor. While keeping the tensor structure in mind, we would like to map the variables into a familiar matrix-vector form. This we can accomplish with a *stacking operator*. The stacking operator simply arranges the elements of a tensor in a specific order within a row vector. A matrix can then specify arbitrary linear combinations of tensor elements to create a new tensor.

The stacking operator is defined by any bijective mapping $\theta_{n,k} : \mathbb{N}^{k+1} \rightarrow \mathbb{N}$ of the multiindex onto a single index: $i = \theta_{n,k}(\mathbf{i}_k)$. This mapping is invertible by definition.

There are many possible stacking maps, but to make the mathematics cleaner we can choose maps that satisfy some simple conditions. The stacking map should map $(0, 0, \ldots, n)$ to 1, should not have any gaps, and should be lexicographically ordered.

A mapping that satisfies these conditions in two dimensions is

$$\theta_{n,2}(\mathbf{i}_2) := \frac{(1 + i_2 + i_1)(i_2 + i_1)}{2} + i_2 + 1.$$

An example for $n = 5$ is given in Table 1.

The maximum value in this mapping, which is also the number of tensor elements, is given by

$$N_{n,2} := \theta_{n,2}(\mathbf{i}_2 = n00) = \frac{(n+1)(n+2)}{2}.$$

Note that we typically write multiindices in reverse order in subscripts, so that the most significant digit comes first. In other words, the index 102 indicates $i_0 = 2, i_1 = 0, i_2 = 1$.

	5	16				$\theta_{5,2}(i_2, i_1, i_0)$	
	4	11	17				
	3	7	12	18			
	2	4	8	13	19		
↑	1	2	5	9	14	20	
i_1	0	1	3	6	10	15	21
		0	1	2	3	4	5
		$i_2 \rightarrow$					

Table 1: Example mapping between tensor and linear indices for n=5,k=2 (three-variable multiindices). Note that i_0 is redundant.

In general, these mappings are generated by [4] (pp. 297–298,303):

$$\theta_{n,k}(\mathbf{i}_k) := \sum_{r=0}^{k} \left(\begin{array}{c} k - r - 1 + \sum_{\ell=r+1}^{k} i_\ell \\ k - r \end{array} \right).$$

For example, the mapping in three dimensions would be

$$\begin{aligned}
&\theta_{n,3}(\mathbf{i}_3) = \\
&\quad \left(\begin{array}{c} 2 + i_3 + i_2 + i_1 \\ 3 \end{array} \right) \\
&\quad + \left(\begin{array}{c} 1 + i_3 + i_2 \\ 2 \end{array} \right) + \left(\begin{array}{c} i_3 \\ 1 \end{array} \right) + 1 \\
&= \frac{(2 + i_3 + i_2 + i_1)(1 + i_3 + i_2 + i_1)(i_3 + i_2 + i_1)}{6} \\
&\quad + \frac{(1 + i_3 + i_2)(i_3 + i_2)}{2} + i_3 + 1.
\end{aligned}$$

Inverses of these maps may be computed via a lookup table.

B PROOF OF HOMOMORPHISM

If M is a $(k+1)\times(k+1)$ invertible matrix, let $\Phi_n(M)$ be the column vector of multinomials obtained by applying ϕ_n to every row of M. Let $\Phi_n(M) = (p_j(\mathbf{x}_k))^T = \mathbf{p}_k(\mathbf{x}_k)^T$, where $p_j(\mathbf{x}_k)$ is a multinomial in $\mathbf{x}_k$ and j is the index of the corresponding row in M.

Before continuing with the proof of homomorphism, we need the following:

LEMMA: *Given* $\mathbf{v}_k \in \mathbb{R}^{k+1}$ *and* M *an invertible* $(k+1) \times (k+1)$ *matrix, then* $\phi_n(\mathbf{v}_k M) = \mathbf{v}_k \Phi_n(M)$.

PROOF OF LEMMA: Let $\alpha_{n,k} = (\mathbf{x}_k^{\mathbf{i}_k})^T$ be an $N_{n,k}$ element column vector containing the power basis of multinomials in $\mathbf{x}_k$ of total degree n. Then $\phi_n(\mathbf{v}_k) = \mathbf{v}_k \alpha_{n,k}$, and

$$\begin{aligned}
\phi_n(\mathbf{v}_k M) &= (\mathbf{v}_k M)\alpha_{n,k} = \mathbf{v}_k (M \alpha_{n,k}) \\
&= \mathbf{v}_k \Phi_n(M). \blacksquare
\end{aligned}$$

Now we can prove the following theorem, which shows that the reparameterization can be externalized, and also gives a more explicit form for evaluating the combinatorical matrix:

THEOREM 1: *If $\delta_{n,k}(M)$ is defined by*

$$F_{n,k}(\mathbf{s}_k M) = F_{n,k}(\mathbf{s}_k)\delta_{n,k}(M),$$

then

$$\begin{aligned}\delta_{n,k}(M) &= \\ &= \Phi_n^{-1}\begin{pmatrix} p_0^0(\mathbf{x}_k)p_1^0(\mathbf{x}_k)\dots p_{k-1}^0(\mathbf{x}_k)p_k^n(\mathbf{x}_k) \\ p_0^0(\mathbf{x}_k)p_1^0(\mathbf{x}_k)\dots p_{k-1}^1(\mathbf{x}_k)p_k^{n-1}(\mathbf{x}_k) \\ p_0^0(\mathbf{x}_k)p_1^0(\mathbf{x}_k)\dots p_{k-1}^2(\mathbf{x}_k)p_k^{n-2}(\mathbf{x}_k) \\ \vdots \\ p_0^{n-1}(\mathbf{x}_k)p_1^1(\mathbf{x}_k)\dots p_{k-1}^0(\mathbf{x}_k)p_k^0(\mathbf{x}_k) \\ p_0^n(\mathbf{x}_k)p_1^0(\mathbf{x}_k)\dots p_{k-1}^0(\mathbf{x}_k)p_k^0(\mathbf{x}_k) \end{pmatrix} \\ &= \Phi_n^{-1}\left(\left(\mathbf{p}_k^{\mathbf{i}_k}(\mathbf{x}_k)\right)^T\right).\end{aligned}$$

PROOF OF THEOREM 1: Combine the Lemma and the multinomial theorem:

$$\begin{aligned} F_{n,k}(\mathbf{s}_k M) &= \phi_n^{-1}\left(\phi_1(\mathbf{s}_k M)^n\right) \\ &= \phi_n^{-1}\left(\{\mathbf{s}_k \Phi_1(M)\}^n\right) \\ &= \phi_n^{-1}\left(\left\{\sum_{j=0}^{k} s_j p_j(\mathbf{x}_k)\right\}^n\right) \\ &= \phi_n^{-1}\left(\sum_{|\mathbf{i}_k|=n}\binom{n}{\mathbf{i}_k}\mathbf{s}_k^{\mathbf{i}_k}\mathbf{p}_k^{\mathbf{i}_k}(\mathbf{x}_k)\right) \\ &= \phi_n^{-1}\left(\left(\binom{n}{\mathbf{i}_k}\mathbf{s}_k^{\mathbf{i}_k}\right)\left(\mathbf{p}_k^{\mathbf{i}_k}(\mathbf{x}_k)\right)^T\right) \\ &= \left(\binom{n}{\mathbf{i}_k}\mathbf{s}_k^{\mathbf{i}_k}\right)\Phi_n^{-1}\left(\left(\mathbf{p}_k^{\mathbf{i}_k}(\mathbf{x}_k)\right)^T\right) \\ &= F_{n,k}(\mathbf{s}_k)\delta_{n,k}(M). \blacksquare \end{aligned}$$

THEOREM 2: *The operator $\delta_{n,k}$ is a homomorphism of the group of invertible $(k+1)\times(k+1)$ matricies.*

PROOF OF THEOREM 2: Let A, B be two invertible $(k+1)\times(k+1)$ matricies. Note that

$$\begin{aligned} F_{n,k}(\mathbf{s}_k(AB)) &= F_{n,k}(\mathbf{s}_k)\delta_{n,k}(AB), \\ F_{n,k}((\mathbf{s}_k A)B) &= F_{n,k}(\mathbf{s}_k)\delta_{n,k}(A)\delta_{n,k}(B). \end{aligned}$$

By the associativity of matrix multiplication, we must have $\delta_{n,k}(AB) = \delta_{n,k}(A)\delta_{n,k}(B)$ on the linear span of $F_{n,k}(\mathbf{s}_k)$. Since $F_{n,k}(\mathbf{s}_k)$ spans all of $\mathbb{R}^{k+1}$, $\delta_{n,k}$ is a homomorphism. ■

C COMBINATORIAL MATRICES

Let

$$M = \begin{pmatrix} a & b & c \\ d & e & f \\ G & H & 1 \end{pmatrix}$$

For $n = 1$ and $k = 2$, corresponding to linear triangles, the combinatorial matrix $\delta_{1,2}(M)$ is:

$$\delta_{1,2}(M) = M = \begin{pmatrix} a & b & c \\ d & e & f \\ G & H & 1 \end{pmatrix}.$$

For $n = 2$ and $k = 2$, corresponding to quadratic triangles, the combinatorial matrix $\delta_{2,2}(M)$ is:

$$\delta_{2,2}(M) = \begin{pmatrix} a^2 & 2ab & 2ac & b^2 & 2bc & c^2 \\ ad & db+ae & af+dc & be & bf+ec & cf \\ aG & aH+Gb & Gc+a & bH & Hc+b & c \\ d^2 & 2de & 2df & e^2 & 2ef & f^2 \\ dG & Ge+dH & Gf+d & eH & Hf+e & f \\ G^2 & 2GH & 2G & H^2 & 2H & 1 \end{pmatrix}.$$

REFERENCES

[1] Zoltan J. Cendes and Steven H. Wong. "C1 quadratic interpolation over arbitrary point sets". *IEEE Computer Graphics and Applications*, Vol. 7, No. 11, pp. 8–16, November 1987.

[2] G. Farin. "A modified Clough-Tocher interpolant". *Computer Aided Geometric Design*, Vol. 2, No. 1-3, pp. 19–27, 1985.

[3] Gerald Farin. *Curves and Surfaces for Computer Aided Geometric Design.* Academic Press, 1990.

[4] Donald E. Knuth. *The Art of Computer Programming: Volume 1, Fundamental Algorithms.* Addison-Wesley Publishing Company, second edition, 1973.

[5] E. T. Y. Lee and Miriam L. Lucian. "Möbius reparameterizations of rational B-splines". *Computer Aided Geometric Design*, Vol. 8, No. 3, pp. 213–215, August 1991.

[6] R.R. Patterson. "Projective Transformations of the Parameter of a Bernstein-Bezier Curve". *ACM Transactions on Graphics*, Vol. 4, No. 4, pp. 276–290, October 1985.

[7] L. Piegl and W. Tiller. "Curve and surface constructions using rational B-splines". *Computer-Aided Design*, Vol. 19, No. 9, pp. 485–98, 1987.

[8] Thomas W. Sederberg. "Techniques for Cubic Algebraic Surfaces". *IEEE Computer Graphics and Applications*, Vol. 10, No. 5, pp. 12–21, September 1990.