An Interpolation Method with Weights and Relaxation Parameters

Chiew-Lan Tai, Brian A. Barsky, and Kia-Fock Loe

Abstract. This paper presents a new interpolation method that is based on blending a nonuniform rational B-spline (NURB) curve with a singularly reparametrized (SR) linear spline. The resulting curve is called the α-spline. It has weights and relaxation parameters. Given the data points to be interpolated, a NURB curve is obtained by using these data points as its control points. The SR linear spline is then determined by imposing constraints on the α-spline to interpolate the data points. The α-spline is parametrically continuous. It involves only simple computation, and does not require solving linear systems. This approach is extended to produce interpolatory surfaces, and can be used as a modeling tool for deforming polygonal shapes into smooth spline surface models.

§1. Introduction

Cubic interpolating splines are known to exhibit undesirable oscillations due to "extraneous" inflection points. This undesirable property motivated the introduction of tension applied to interpolating splines. Barsky analyzed two well-known approaches for applying tension to interpolating curves [1]. The first approach is the exponential-based spline in tension [12]; the second is the more efficient polynomial alternative, the ν-spline [11]. The tension parameters are associated with data points in these splines.

Tension has also been introduced as a shape parameter to non-interpolating splines. The Beta-spline [2] incorporates bias and tension parameters, based on the theory of geometric continuity. Another spline technique that has bias and tension parameters (and also a "continuity" parameter) was proposed by Kochanek and Bartels [8]. Unlike those in the Beta-spline, these parameters control only the first derivatives, and the resulting curve is C^1 and interpolatory. The weighted splines by Foley [6] provide tension control on curve segments between interpolation points, rather than at the interpolation points.

Copyright ©2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1357-3.
All rights of reproduction in any form reserved.

In addition to spline methods that allow the user to modify the shape parameters, a related research area is the automatic determination of shape parameters to produce curves and surfaces that satisfy certain shape-preserving criteria [5,7]. Another more subjective criterion used is fairness, or "visual appeal".

NURBS are also often used to solve interpolation problems. However, using the NURB as a surface interpolant requires performing $O(mn)$ operations to solve the $O(m+n)$ linear systems for a dataset of $m \times n$ points. Although these systems are tridiagonal, they still incur substantial computations. Moreover, the interpolation method is global; thus, changes to any data point will require solving again all the linear systems.

The idea of using singular blending to solve the interpolation problem, with tension control, was proposed by Loe and Tai [10]. In this paper, we use this idea to propose another interpolation method that has weights and relaxation parameters. A high relaxation value has the effect of reducing tautness. The method is based on blending a NURB curve with a sequence of line segments, or a linear spline, that is reparametrized such that each of the line segments has zero derivatives at both ends.

This work is closely related to some earlier work of Coons [4]. He used this blending technique to modify any given piecewise curve by letting the line segments be those connecting the joints of the given curve, and proved that if the given curve is C^k, then the blending function only has to be C^{k-1} for the resulting curve to retain the continuity of the original curve. More precisely, Coons' aim was not to solve the interpolation problem: for the new curve to be interpolatory, the original curve has to be interpolatory.

§2. Singular Blending

Given a smooth piecewise curve $\boldsymbol{C}(u)$, a tension control can be introduced by blending the smooth curve with a linear spline that approximates the curve [4,9]. This linear spline can simply be obtained by connecting the joints of the smooth curve, but in general it need not interpolate the smooth curve. Let $\boldsymbol{L}(u)$ denotes a linear spline. The blending then gives

$$\boldsymbol{Q}(u) = (1-\alpha)\boldsymbol{C}(u) + \alpha\boldsymbol{L}(u), \qquad u \in [u_0, u_n], \qquad 0 \le \alpha \le 1. \tag{1}$$

It is easy to see that as α increases, the contribution of $\boldsymbol{L}(u)$ to $\boldsymbol{Q}(u)$ increases; thus, the resulting curve is more taut, simulating a higher tension.

Assuming that the smooth curve $\boldsymbol{C}(u)$ is C^2 (generalization to higher order curves is straightforward), then for $\boldsymbol{Q}(u)$ to retain this continuity, in general $\boldsymbol{L}(u)$ must be at least C^2. We define $\boldsymbol{L}(u)$ as

$$\boldsymbol{L}(u) = (1-s(t))\boldsymbol{V}_j + s(t)\boldsymbol{V}_{j+1}, \qquad u \in [u_j, u_{j+1}), \tag{2}$$

where $t = \frac{u-u_j}{u_{j+1}-u_j}$, $\boldsymbol{V}_j$, $j = 0, \ldots, n$, are the vertices of the linear spline, and $s(t)$ is a monotonically increasing function yet to be defined. Since $\boldsymbol{L}(u)$ must be C^2, it must satisfy the following conditions:

$$\boldsymbol{L}'(u_j) = \boldsymbol{L}'(u_{j+1}) = 0 \qquad \text{and} \qquad \boldsymbol{L}''(u_j) = \boldsymbol{L}''(u_{j+1}) = 0.$$

These conditions can be satisfied by letting $s(t)$ be the quintic Hermite polynomial $10t^3 - 15t^4 + 6t^5$ [3]. We call a linear spline that satisfies these conditions a singularly reparametrized (SR) linear spline.

For the technique to be useful, it should have local tension control; that is, the α in (1) must be a function of u,

$$\boldsymbol{Q}(u) = (1 - \alpha(u))\boldsymbol{C}(u) + \alpha(u)\boldsymbol{L}(u), \qquad u \in [u_0, u_n]. \tag{3}$$

The $\alpha(u)$ function must satisfy three criteria: (1) interpolate the local parameters α_j^*, $j = 0, ..., n$; (2) employ a local interpolation method so that modifying a particular α_j^* will affect only the neighboring curve segments; (3) be at least C^2 so that the blended curve is also C^2 (Coons [4] showed that if the SR line segments are restricted to those connecting the joints of the given curve, then $\alpha(u)$ only need to be C^1). We observe that $s(t)$ can be used to define $\alpha(u)$, with all three criteria satisfied; hence, we define

$$\alpha(u) = (1 - s(t))\alpha_j^* + s(t)\alpha_{j+1}^*, \qquad u \in [u_j, u_{j+1}), \tag{4}$$

where $t = \frac{u-u_j}{u_{j+1}-u_j}$. We then have $\alpha'(u_j) = \alpha'(u_{j+1}) = \alpha''(u_j) = \alpha''(u_{j+1}) = 0$. A drawback of this definition of $\alpha(u)$ is that drastic changes in adjacent α_j^* values can cause the curve to undulate. An alternate method is to estimate the first and second derivatives at the joints by some approximation method, and obtain $\alpha(u)$ by Hermite interpolation.

This idea of singular blending can be applied to many combinations of curves and SR linear splines. Since NURBS are prevalent in industry, we let the smooth curve $\boldsymbol{C}(u)$ be a NURB curve and call the resulting curve the α-spline. If the SR linear spline is the control polygon of the NURBS curve, then the resulting α-spline is non-interpolatory. The non-interpolatory α-spline includes the NURB as a special case; when all tension parameters are zero, the α-spline reduces to a NURB. Some other geometric properties inherited from NURB include convex hull, affine and projective invariance, and local control.

§3. The Interpolating α-spline Curve

The α-spline is non-interpolatory when the SR linear spline is the control polygon of the NURB curve. However, there is no reason to restrict the SR linear spline to be the control polygon. In this paper, given a NURB curve, we will determine a new SR linear spline such that when they are blended, the resulting α-spline interpolates a given set of data points.

Let the data points be $\boldsymbol{P}_j, j = 0, ..., n$. The cubic NURB curve must somehow approximate the data points. A simple way to achieve this is by letting the given data points serve as the control points. Next, we must introduce two new control points so that the number of NURB curve segments is n, where each curve segment corresponds to an interval between two data points. Since we want the NURB curve to interpolate the endpoints $\boldsymbol{P}_0$ and

$\boldsymbol{P}_n$, we let the new points be coincident with the two endpoints, and set the first four and last four end knot values to be equal; that is,

$$\boldsymbol{P}_{-1} = \boldsymbol{P}_0 \qquad \text{and} \qquad \boldsymbol{P}_{n+1} = \boldsymbol{P}_n,$$

and $u_{-3}=u_{-2}=u_{-1}=u_0$, and $u_n=u_{n+1}=u_{n+2}=u_{n+3}$ in $\boldsymbol{U}=\{u_{-3},...,u_{n+3}\}$. The interior knot values $u_0,...,u_n$ are computed using methods such as the chord-length or centripetal method. All the examples in this paper use the centripetal parametrization. The cubic NURB curve sequence is given by

$$\boldsymbol{C}(u) = \sum_{j=-1}^{n+1} R_{j,4}(u)\boldsymbol{P}_j, \qquad u \in [u_0, u_n],$$

where $R_{j,4}(u)$ are the cubic rational B-spline basis functions defined over the knot vector $\boldsymbol{U}$.

To determine the unknown vertices $\boldsymbol{V}_j$, $j = 0,...,n$, of the SR linear spline, we impose constraints on the resulting α-spline $\boldsymbol{Q}(u)$ to interpolate all the data points: $\boldsymbol{P}_j = \boldsymbol{Q}(u_j), j = 0,...,n$. Substituting $\boldsymbol{Q}(u_j)$ from (3),

$$\boldsymbol{P}_j = (1-\alpha(u_j))\boldsymbol{C}(u_j) + \alpha(u_j)\boldsymbol{L}(u_j) = (1-\alpha_j^*)\boldsymbol{C}(u_j) + \alpha_j^*\boldsymbol{V}_j.$$

Solving for $\boldsymbol{V}_j$, then adding $\boldsymbol{P}_j$ and subtracting $\frac{\alpha_j^*\boldsymbol{P}_j}{\alpha_j^*}$ yields equations that require only simple computations:

$$\boldsymbol{V}_j = \frac{\boldsymbol{P}_j - (1-\alpha_j^*)\boldsymbol{C}(u_j)}{\alpha_j^*} = \boldsymbol{P}_j + \frac{1-\alpha_j^*}{\alpha_j^*}(\boldsymbol{P}_j - \boldsymbol{C}(u_j)).$$

Defining $\rho_j = \frac{1-\alpha_j^*}{\alpha_j^*}$ yields

$$\boldsymbol{V}_j = \boldsymbol{P}_j + \rho_j\ (\boldsymbol{P}_j - \boldsymbol{C}(u_j)), \qquad j = 0,...,n. \tag{5}$$

We call ρ_j the relaxation parameters. The geometric interpretation of (5) is that $\boldsymbol{C}(u_j)$, $\boldsymbol{P}_j$, and $\boldsymbol{V}_j$ are collinear, and the distance between $\boldsymbol{C}(u_j)$ and $\boldsymbol{P}_j$, and between $\boldsymbol{P}_j$ and $\boldsymbol{V}_j$ are in the ratio $1 : \rho_j$. We know that $\rho_j \geq 0$, because $\alpha_j^* \leq 1$. From empirical study, we have found $0 \leq \rho_j \leq 4$ to be a useful range. The midpoint value $\rho_j = 2$ yields visually appealing shapes for most datasets; thus, we use that value as the default relaxation value. The effect of ρ_j can be interpreted from noting that $\alpha_j^* = \frac{1}{\rho_j+1}$ and observing the role of α in (3). The value $\rho_j = 0$ corresponds to the maximum tension $\alpha_j^* = 1$; when all $\rho_j = 0$, only the SR linear spline contributes to the α-spline and we have a linear interpolant. When the relaxation value ρ_j increases, α_j^* decreases, and the contribution of the SR linear spline to the α-spline decreases; thus, the resulting curve is being relaxed locally. By using different relaxation values, we can easily obtain rounder or sharper corners without specifying multiple knots or multiple control points. Fig. 1 shows the effect of

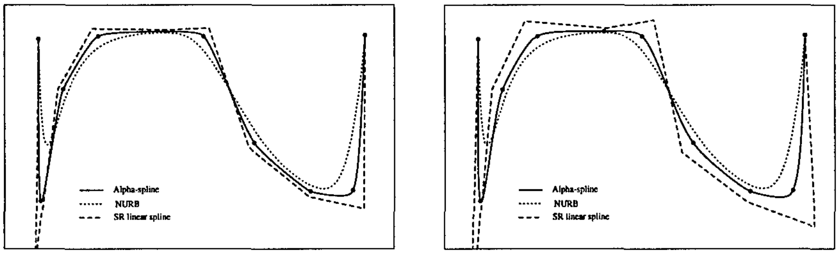

Fig. 1. Effect of global relaxation: (left) 1, (right) 2.

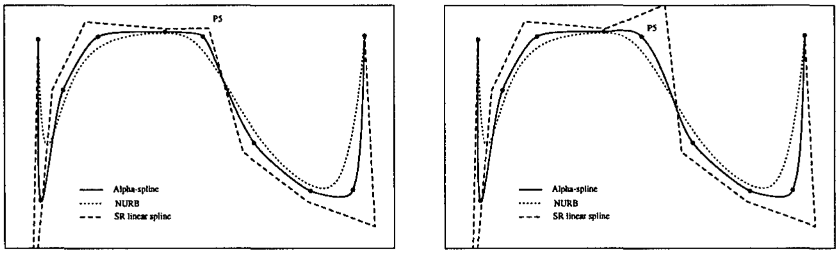

Fig. 2. Effect of varying ρ_5: 1, 4; the other ρ_j are 2.

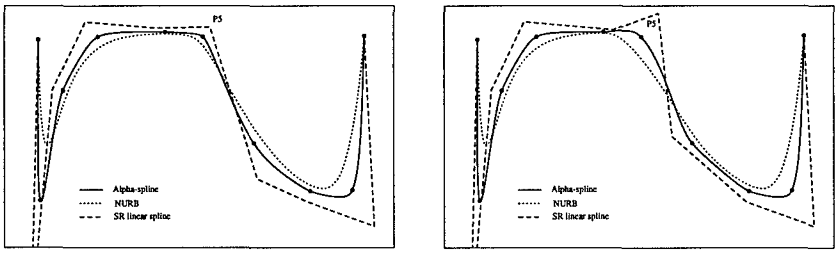

Fig. 3. Effect of varying w_5: 2, 0.5; the other w_j are unity. All ρ_j are 2.

applying the relaxation globally. It can be observed that while the SR linear spline found is dependent on the global relaxation value specified, the cubic NURB curve remains fixed. Fig. 2 shows the effect of varying the relaxation locally (while the other part of the curve has the default relaxation value). Note that the effect of each ρ_j is very localized; only the nearest two curve segments are affected.

The effect of the weight is less obvious. Each w_j affects four neighboring segments, i.e., $[u_{j-2}, u_{j+2}]$. When w_j decreases, $\boldsymbol{C}(u_j)$ moves further from $\boldsymbol{P}_j$ (assuming ρ_j is fixed); thus, from (5), $\boldsymbol{V}_j$ found is further from $\boldsymbol{P}_j$. Since the weights are relative in nature, decreasing w_j also causes $\boldsymbol{C}(u_{j-1})$ and $\boldsymbol{C}(u_{j+1})$ to move closer to the edge $\boldsymbol{P}_{j-2}\boldsymbol{P}_{j-1}$ and $\boldsymbol{P}_{j+1}\boldsymbol{P}_{j+2}$, respectively, and to be on that edge when w_j=0. Hence, when $w_j = 0$, from (5) again, $\boldsymbol{P}_{j-2}$, $\boldsymbol{C}(u_{j-1})$, $\boldsymbol{P}_{j-1}$, and $\boldsymbol{V}_{j-1}$ are all collinear, and so are $\boldsymbol{P}_{j+2}$, $\boldsymbol{C}(u_{j+1})$,

$\boldsymbol{P}_{j+1}$, and $\boldsymbol{V}_{j+1}$. This effect on the SR linear spline is depicted in Fig. 3; only w_5 is varied here. To summarize, decreasing w_j causes the interpolating α-spline to be rounder near $\boldsymbol{P}_j$, but causes its segments between $\boldsymbol{P}_{j-2}\boldsymbol{P}_{j-1}$ and $\boldsymbol{P}_{j+1}\boldsymbol{P}_{j+2}$ to be more taut near the points $\boldsymbol{P}_{j-1}$ and $\boldsymbol{P}_{j+1}$.

The α-spline $\boldsymbol{Q}(u)$ is clearly C^2 continuous since $\alpha(u)$, $\boldsymbol{L}(u)$, and $\boldsymbol{C}(u)$ are all C^2. The first and second derivatives at the knots are as follows:

$$\boldsymbol{Q}'(u_j) = (1-\alpha_j^*)\boldsymbol{C}'(u_j) + \alpha'(u_j)(\boldsymbol{L}(u_j) - \boldsymbol{C}(u_j)),$$
$$\boldsymbol{Q}''(u_j) = (1-\alpha_j^*)\boldsymbol{C}''(u_j) - 2\alpha'(u_j)\boldsymbol{C}'(u_j) + \alpha''(u_j)(\boldsymbol{L}(u_j) - \boldsymbol{C}(u_j)).$$

If $\alpha(u)$ is defined as in (4), then $\boldsymbol{Q}'(u_j) = (1-\alpha_j^*)\boldsymbol{C}'(u_j)$ and $\boldsymbol{Q}''(u_j) = (1-\alpha_j^*)\boldsymbol{C}''(u_j)$. That is, the derivatives at the joints of the blended curve $\boldsymbol{Q}(u)$ are in the same directions as their counterparts of $\boldsymbol{C}(u)$.

§4. The Interpolating α-spline Surface

Analogous to the blending of a NURB curve with an SR linear spline, we can blend a NURB surface with a network of singularly reparametrized (SR) bilinear patches. An SR bilinear patch is defined as follows:

$$\begin{aligned}\boldsymbol{L}(u,v) =& (1-s(r))(1-s(t))\boldsymbol{V}_{i,j} + (1-s(r))s(t)\boldsymbol{V}_{i,j+1} \\ &+ s(r)(1-s(t))\boldsymbol{V}_{i+1,j} + s(r)s(t)\boldsymbol{V}_{i+1,j+1},\end{aligned} \tag{6}$$

where $u \in [u_i, u_{i+1}), v \in [v_j, v_{j+1}), r = \frac{u-u_i}{u_{i+1}-u_i}, t = \frac{v-v_j}{v_{j+1}-v_j}$, and the functions $s(\cdot)$ are the Hermite polynomials given earlier. That is, it is parametrized such that its first and second order partial derivatives go to zero at the boundaries:

$$\partial_u \boldsymbol{L}(u_i,v) = \partial_u \boldsymbol{L}(u_{i+1},v) = \partial_v \boldsymbol{L}(u,v_j) = \partial_v \boldsymbol{L}(u,v_{j+1}) = 0,$$
$$\partial_u^2 \boldsymbol{L}(u_i,v) = \partial_u^2 \boldsymbol{L}(u_{i+1},v) = \partial_v^2 \boldsymbol{L}(u,v_j) = \partial_v^2 \boldsymbol{L}(u,v_{j+1}) = 0,$$
$$\partial_{uv}^2 \boldsymbol{L}(u_i,v) = \partial_{uv}^2 \boldsymbol{L}(u_{i+1},v) = \partial_{uv}^2 \boldsymbol{L}(u,v_j) = \partial_{uv}^2 \boldsymbol{L}(u,v_{j+1}) = 0.$$

The α-spline surface is then given by

$$\boldsymbol{Q}(u,v) = (1-\alpha(u,v))\boldsymbol{S}(u,v) + \alpha(u,v)\boldsymbol{L}(u,v), \qquad u \in [u_0,u_m];\ v \in [v_0,v_n],$$

where $\alpha(u,v)$ is the blending function that interpolates the local tension parameters $\alpha_{i,j}^*$, defined by an equation similar to (6).

To find an α-spline surface interpolating a given network of data points $\boldsymbol{P}_{i,j}, i = 0,...,m,\ j = 0,...,n$, we must first find a NURB surface, then determine a network of SR bilinear patches to be blended with the NURB surface. The NURB surface is defined by simply letting the data points be the control points. As in the case of curves, we repeat the boundary vertices and let the first and last four knot values be equal. Repeating the boundary vertices along the j-index, then repeating those along the i-index, we obtain

$$\begin{aligned} &\boldsymbol{P}_{-1,j} = \boldsymbol{P}_{0,j} \quad \text{and} \quad \boldsymbol{P}_{m+1,j} = \boldsymbol{P}_{m,j}, \qquad j = 0,...,n, \\ &\boldsymbol{P}_{i,-1} = \boldsymbol{P}_{i,0} \quad \text{and} \quad \boldsymbol{P}_{i,n+1} = \boldsymbol{P}_{i,n}, \qquad i = -1,...,m+1.\end{aligned}$$

Setting the end knot values to be equal in the knot vectors $U=\{u_{-3},...,u_{m+3}\}$ and $V=\{v_{-3},...,v_{n+3}\}$,

$$u_{-3}=u_{-2}=u_{-1}=u_0, \qquad u_m=u_{m+1}=u_{m+2}=u_{m+3},$$
$$v_{-3}=v_{-2}=v_{-1}=v_0, \qquad v_n=v_{n+1}=v_{n+2}=v_{n+3}.$$

The interior knot values can be determined using any good parametrization method. The NURB surface is then defined by

$$S(u,v)=\sum_{i=-1}^{m+1}\sum_{j=-1}^{n+1} R_{i,4}(u)R_{j,4}(v)P_{i,j}, \qquad u\in[u_0,u_m];\ v\in[v_0,v_n],$$

where $R_{i,4}(u)$ and $R_{j,4}(v)$ are cubic rational B-spline basis functions defined over U and V.

To determine the vertices $V_{i,j}$'s that define the SR bilinear patches, we impose constraints on the α-spline surface to interpolate the data points:

$$\begin{aligned} P_{i,j} &= Q(u_i,v_j) \\ &= (1-\alpha^*_{i,j})S(u_i,v_j)+\alpha^*_{i,j}V_{i,j}, \qquad i=0,...,m;\ j=0,...,n. \end{aligned}$$

Solving for $V_{i,j}$, we obtain

$$V_{i,j}=P_{i,j}+\frac{1-\alpha^*_{i,j}}{\alpha^*_{i,j}}(P_{i,j}-S(u_i,v_j)).$$

Defining $\rho_{i,j}=\frac{1-\alpha^*_{i,j}}{\alpha^*_{i,j}}$ yields

$$V_{i,j}=P_{i,j}+\rho_{i,j}(P_{i,j}-S(u_i,v_j)), \qquad i=0,...,m;\ j=0,...,n.$$

This enables the direct evaluation of $V_{i,j}$, and avoids the necessity of solving linear systems required by interpolation with NURB surface.

§5. Smoothing Polygonal Shapes

In addition to fitting a smooth surface over a dataset, the proposed interpolation method can also be viewed as an interactive modeling tool for deforming polygonal shapes (with an underlying rectangular topology) into smooth objects. The vertices of the polygonal shape are the data points to be interpolated by a smooth surface. With this modeling tool, the user only has to specify the polygonal vertices, which are fewer than the number of control points of most spline schemes. Manipulating polygonal shapes is also simple and easy for novice designers.

Figures 4 and 5 show some modeling examples, all of which are obtained from the same input polygonal shape, shown in the top left corner of Fig. 4. In Fig. 4, the relaxation parameters are varied: globally in the top row, and locally in the bottom row (the third row of vertices have their relaxation parameters varied while the other vertices have fixed $\rho_{i,j}=1$). In Fig. 5, the weights of the third row of vertices are varied, while the other weights are fixed at unity, and the global relaxation is set at $\rho_{i,j}=1$ for all i,j.

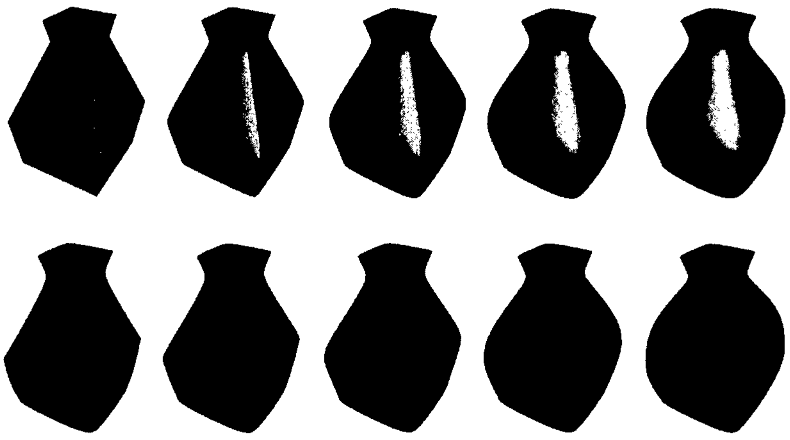
Fig. 4. Effect of global (top) and local (bottom) relaxation: 0, 0.25, 0.67, 1.5, 2.3.

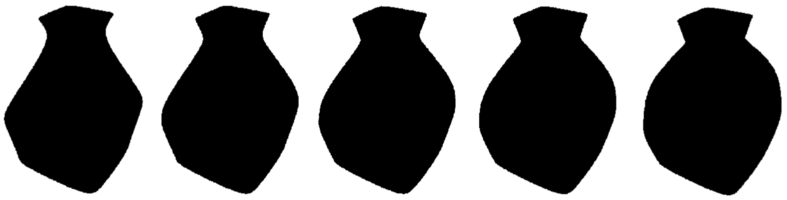
Fig. 5. Effect of local weight (third-row vertices): 2.5, 1, 0.5, 0.25, 0.

§6. Homogeneous Representation of α-spline

It is well known that the rational B-spline can be viewed as the projection of a polynomial B-spline in homogeneous space. This property is important because it implies that all the algorithms for the polynomial B-spline can also be applied to the rational B-spline. The rational B-spline is given by

$$\boldsymbol{C}(u) = \sum R_{j,4}(u)\boldsymbol{P}_j, \tag{7}$$

where $\boldsymbol{P}_j = (x_j, y_j, z_j)$ are the control points and $R_{j,4}(u) = \frac{N_{j,4}(u)w_j}{\sum N_{i,4}(u)w_i}$, $w_j \geq 0$, are the weights, and $N_{j,4}(u)$ denotes the cubic B-spline basis functions. The polynomial B-spline curve in the homogeneous space, $\boldsymbol{C}^h(u)$, whose projection yields the rational B-spline $\boldsymbol{C}(u)$ is given by

$$\boldsymbol{C}^h(u) = \sum N_{j,4}(u)\boldsymbol{P}_j^h, \tag{8}$$

where $\boldsymbol{P}_j^h = (w_j x_j, w_j y_j, w_j z_j, w_j)$, since $\boldsymbol{C}(u) = (\frac{C_x^h(u)}{C_w^h(u)}, \frac{C_y^h(u)}{C_w^h(u)}, \frac{C_z^h(u)}{C_w^h(u)})$, and the x, y, z and w subscripts denote the respective components of $\boldsymbol{C}^h(u)$.

We can show that this property is also true for the α-spline. Substituting $\boldsymbol{C}(u)$ from (7) into the alpha-spline equation in (3), and rewriting it as a rational function, yields

$$\boldsymbol{Q}(u) = \frac{(1-\alpha(u))\sum N_{j,4}(u)w_j\boldsymbol{P}_j + \alpha(u)C_w^h(u)\boldsymbol{L}(u)}{C_w^h(u)}.$$

The denominator can be expressed as $(1-\alpha(u))C_w^h(u) + \alpha(u)C_w^h(u)$; hence, the polynomial form of the α-spline in homogeneous space is

$$\boldsymbol{Q}^h(u) = (1-\alpha(u))\boldsymbol{C}^h(u) + \alpha(u)\boldsymbol{L}^h(u), \qquad u \in [u_0, u_n],$$

where $\boldsymbol{C}^h(u)$ is given in (8), and $\boldsymbol{L}^h(u)$ is given by

$$\boldsymbol{L}^h(u) = C_w^h(u)((1-s(t))\boldsymbol{V}_j^h + s(t)\boldsymbol{V}_{j+1}^h), \qquad u \in [u_j, u_{j+1}),$$

where $\boldsymbol{V}_j^h = (x_j', y_j', z_j', 1)$ is obtained from $\boldsymbol{V}_j = (x_j', y_j', z_j')$.

§7. Conclusion

We have proposed a new interpolation scheme based on blending a non-interpolatory NURB curve (surface) with an SR linear spline (SR bilinear patches). The resulting interpolating α-spline inherits the continuity, and the affine and projective invariant properties of the NURB. The method provides weight and relaxation control, involves only simple computations, and supports the modeling paradigm of deforming polygonal shapes into smooth spline surfaces.

Several issues have yet to be investigated for the α-spline. One example is knot insertion which is useful for shape refinement and rendering. Another issue is the automatic determination of the parameters to satisfy certain shape-preserving conditions.

Acknowledgments. Gratitude is extended to Yim-Hung Chan for the implementation. This work is partially supported by the grant HKUST6215/99 awarded by the Hong Kong Research Grant Council.

References

1. Barsky, B. A., Exponential and polynomial methods for applying tension to an interpolating spline curve, Computer Vision, Graphics, and Image Processing **27** (1984), 1–18.
2. Barsky, B. A., *Computer Graphics and Geometric Modeling Using Beta-splines,* Springer-Verlag, Heidelberg, 1988.
3. Bartels, R. H., J. C. Beatty, and B. A. Barsky, *An Introduction to Splines for Use in Computer Graphics & Computer Modeling,* Morgan Kaufmann Publ., San Francisco, CA, 1987.

4. Coons, S. A., Modification of the shape of piecewise curves, Computer-Aided Design **9** (1977), 178–180.
5. Costantini, P., T. N. T. Goodman, and C. Manni, Constructing C^3 shape preserving interpolating space curves, manuscript.
6. Foley, T. F., Interpolation with interval and point tension controls using cubic weighted ν-splines, ACM Trans. Math. Software **13** (1987), 68–96.
7. Kaklis, P. D. and N. S. Sapidis, Convexity-preserving interpolatory parametric splines of non-uniform polynomial degree, Comput. Aided Geom. Design **11** (1995), 1–26.
8. Kochanek, D. H. U. and R. H. Bartels, Interpolating splines with local tension, continuity, and bias control, ACM SIGGRAPH '84 **18** (1984), 33–41.
9. Loe, K. F., αB-spline: a linear singular blending B-spline, Visual Comp. **12** (1996), 18–25.
10. Loe, K. F. and C. L. Tai, Interpolation with tension control using singular blending, submitted for publication.
11. Nielson, G. M., Some piecewise polynomial alternatives to splines in tension, in *Computer Aided Geometric Design*, R. E. Barnhill and R. F. Riesenfeld (eds), Academic Press, New York, 1974, 209–235.
12. Schweikert, D. G., An interpolation curve using a spline in tension, J. Math. Phys. **45** (1966), 312–317.

Chiew-Lan Tai
Department of Computer Science
The Hong Kong University of Science & Technology
Clear Water Bay, Kowloon, Hong Kong
`taicl@cs.ust.hk`

Brian A. Barsky
Computer Science Division
University of California
Berkeley, CA 94720-1776, U.S.A.
`barsky@cs.berkeley.edu`

Kia-Fock Loe
Department of Computer Science
School of Computing
National University of Singapore
Lower Kent Ridge, Singapore 119260
`loekf@comp.nus.edu.sg`