A Symbolic Approach to Freeform Parametric Surface Blends *

Kwansik Kim† and Gershon Elber‡
Department of Computer Science
University of Utah
Salt Lake City, UT 84112 USA

May 25, 1999

Abstract

This paper presents a symbolic approach to the computation of blend surfaces across piecewise polynomial and rational surfaces. Curves in the parameter space of the primary surfaces can be specified as the rail curves, also known as trimlines, of the blend. Several techniques which require various levels of user interaction are presented for defining the cross boundary tangent curves along the rail curves. The resulting blend is represented as a polynomial surface having tangent plane continuity with the primary surfaces to an accuracy bounded only by the machine precision. Also presented is a normalization method that approximates a *unit* vector field, an approach that might benefit other applications such as offset approximation and animation curve construction.

Key Words: Symbolic computation, Hermite interpolation, Fillets and Rounds, Blends, Reparameterization.

1 Introduction

Blend surfaces are ones which smoothly blend two adjacent surface boundaries. In addition to being a modeling tool, blend surfaces are used to emulate manufacturing procedures using ball end tools and are employed to alleviate stress. In most cases blend surface are used to smoothly connect two other surfaces, so the exact shape of the blend may be relatively unimportant as long as continuity is satisfied. Tangent plane continuity is frequently considered sufficient although higher order continuity is sometimes desired as well.

As geometric surface modelers have matured, the ability to represent blend surfaces has become increasingly important. One reason for this is that models are desired that more

*This work was supported in part by DARPA (N00014-91-J-4123). All opinions, findings, conclusions or recommendations expressed in this document are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

†Current address: Computer and Information Science Department, 225 Applied Sciences, University of California, Santa Cruz, CA 95060

‡Current address: Computer Science Department, Technion, Haifa 32000, Israel

accurately represent finished manufactured parts. In many cases, the use of Boolean operations in modeling systems results in models with undesirable sharp seams between surfaces.

The nomenclature used in discussing these smoothing surfaces differs somewhat from author to author, with the surfaces in question being known variously as rounding, fillet, or blend surfaces. We will adopt the term "blend" to denote both rounding and filleting operations when referring to these surfaces. Blend surfaces are usually constructed along the seam or intersection between two other already specified surfaces, called the *primary surfaces*. The boundary curves of the blend surface, which lie on the primary surfaces, are referred to as *rail curves* [10] or *trimlines* [18]. In [18], a nice survey of the different existing blending techniques can be found.

Constant radius blends, also known as rolling ball blends, are a common means of specifying a blend surface. A standard approach to generating these blends involves offsetting both primary surfaces [4, 13, 17]. Constant radius blend surfaces are important if the geometry of the blend needs to be precisely prescribed. Frequently, when the aesthetic shape of the model is a major concern, or when sharp corners should simply be chamfered or rounded, the exact geometry constraint can be removed. In [3, 10, 12], approaches are developed which do not necessarily construct constant radius blends. These approaches are appealing for their simplicity and the elimination of the surface offset operation requirements. Blending or joining algorithms for algebraic surfaces including [2, 19] have been developed separately.

This paper presents new symbolic techniques for the specification and representation of blend surfaces from within the context of a NURBs based geometric modeling system called Alpha_1, developed at the University of Utah. Section 2 presents techniques for representing blend surfaces exactly in polynomial (Bézier) or piecewise polynomial and rational (NURBs) domains from given rail curves and cross tangent curve specifications. Section 3 details some blending examples while we conclude in section 4.

2 Symbolic Blending

In this section we use symbolic computation to derive blend surfaces. Towards this end we use the cubic Hermite interpolation scheme with two rail curves and two cross boundary tangent curves along these rail curves.

In Section 2.1, symbolic computation is reviewed. In Section 2.2, several methods to define the cross boundary tangent information along the rail curves are discussed. Finally, in Section 2.3, an iterative approach is suggested for the normalization of the cross boundary tangent curves. This normalization process can benefit other applications such as unit normal approximation for offset computation or unit length vector fields for constant speed motion planning in animation. Herein, these normalized cross boundary tangent definitions and the cubic Hermite form one can symbolically compute a blend surface that provides plane continuity, in exact form up to machine accuracy.

2.1 Background

2.1.1 Symbolic Operations

To be able to symbolically compute blend surfaces, one must be able to symbolically represent the derivative, sum, difference, product, and composition of scalar Bézier and NURBs curves and surfaces. Any manipulation of Bézier or NURBs curves or surfaces using these tools should result in a scalar or vector field represented as a Bézier or NURBs curve or surface. The resulting curve or surface is exact to within the accuracy of the numerical computation, since these operations have closed forms and are, in fact, symbol manipulators. Therefore, we refer to the use of these tools as *symbolic* computation. These operations in the polynomial (Bézier) and piecewise polynomial and rational (NURBs) domains are discussed in several places [6, 8, 9, 12, 16]. Here, we briefly discuss only the symbolic computation of a curve-surface composition in the Bézier domain as an example of this approach.

A curve-surface composition in the polynomial Bézier domain can be computed as,

$$\begin{aligned} S(c(t)) = S(u(t), v(t)) &= \sum_{i=0}^{n}\sum_{j=0}^{m} P_{ij} B_j^m(v(t)) B_i^n(u(t)) \\ &= \sum_{i=0}^{n}\left(\sum_{j=0}^{m} P_{ij} B_j^m(v(t))\right) B_i^n(u(t)), \end{aligned} \tag{1}$$

where P_{ij} are the control points of S, $B_i^n(u)$ and $B_j^m(v)$ are the basis functions of S and $c(t) = (u(t), v(t))$ is a curve in the parametric domain of S.

Assuming one can compute and represent $B_k^l(w(t))$, where $w(t) \in [0, 1]$ is a scalar Bézier curve, as a Bézier curve, the curve $S(u(t), v(t))$ is also representable as a Bézier curve since it is the result of sums and products of $B_k^l(w(t))$ terms only. From the definition of the Bernstein polynomials,

$$B_k^l(w(t)) = \binom{l}{k}(1.0 - w(t))^{l-k}(w(t))^k. \tag{2}$$

Let $w(t) = \sum_{p=0}^{d} w_i B_p^d(t)$. Then,

$$1 - w(t) = \sum_{p=0}^{d} B_p^d(t) - \sum_{p=0}^{d} w_i B_p^d(t) = \sum_{p=0}^{d}(1 - w_i) B_p^d(t).$$

Moreover (see [6, 9]),

$$w(t)w(t) = \sum_{p=0}^{2d} \mathcal{W}_p B_p^{2d}(t), \qquad \mathcal{W}_p = \sum_{q=max(0,p-d)}^{min(p,d)} \frac{\binom{d}{q}\binom{d}{p-q}}{\binom{2d}{p}} w_q w_{p-q}.$$

Therefore, the composition $S(u(t), v(t))$ can be symbolically computed and represented in the polynomial Bézier domain. The computation and representation of the composition operation in the rational domain is a simple extension to the above (see [6]). For the piecewise polynomial and rational domains (NURBs), the problem is more difficult. Either the composition is posed as an interpolation problem or the curve and the surface are subdivided into polynomial or rational regions. In both cases the intersection locations of the curve with the

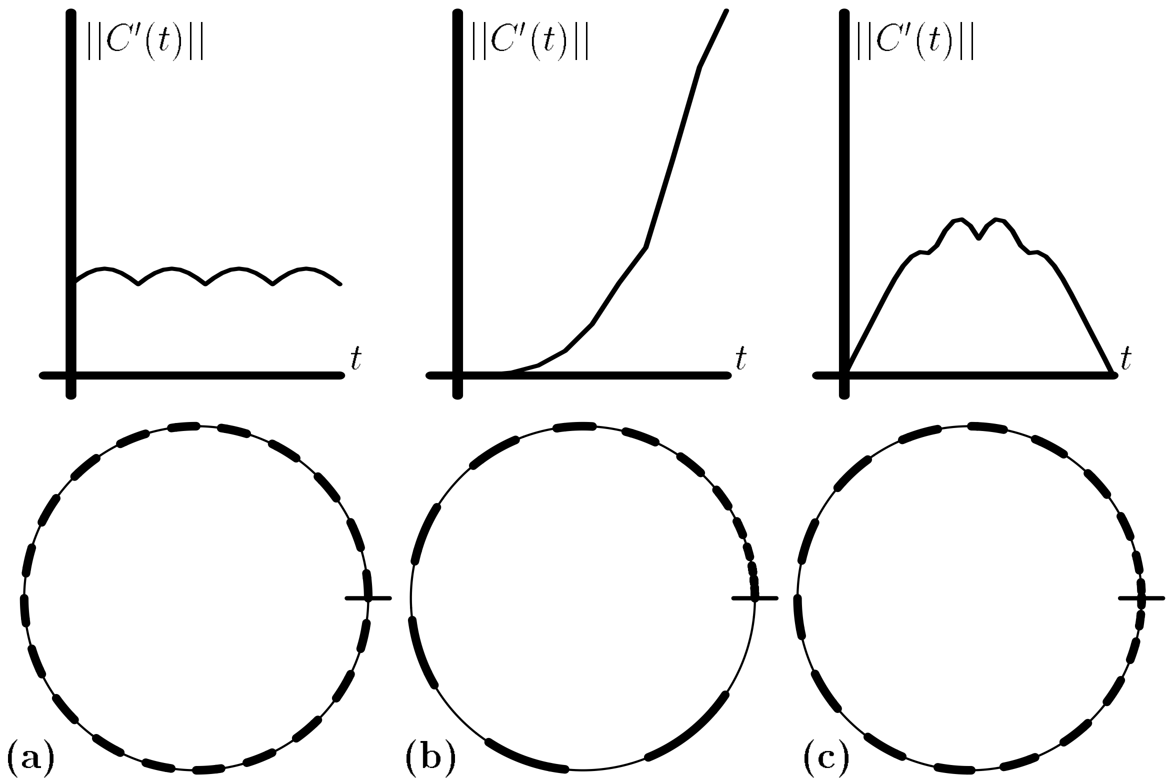


Figure 1: A circle with its speed profile curve in (a) is reparametrized using composition to start slowly (b) or to both start and terminate slowly (c). The markers shows equal domains in the parametric space of the circle.

knot lines of the surface, in the surface's parametric space, should be computed. See [12] for more.

This composition can be applied to various other applications, for example for the purpose of reparameterization of a given (animation) curve to a prescribed speed [1]. Given a parametric curve $C(t)$, and an allowable change of parameter $t(r)$, the velocity of C with respect to r equals,

$$\left\|\frac{dC(t(r))}{dr}\right\| = \left\|\frac{dC(t(r))}{dt}\right\| |t'(r)|.$$

Hence, one can construct an allowable change of parameter $t(r)$ so that $t(0) = 0,\ t(1) = 1$ and

$$|t'(0)| = \frac{1}{\left\|\frac{dC(0)}{dt}\right\|}, \qquad |t'(1)| = \frac{1}{\left\|\frac{dC(1)}{dt}\right\|},$$

resulting in a curve $C(r)$ with unit velocity at its end points, a type of curve that is useful for animation, controlling the speed of the motion. Clearly, this reparameterization can be employed to arbitrarily vary the speed of the curve. Figure 1 (a) shows a NURBs circle curve reparametrized to start slowly (Figure 1 (b)) or to both start and terminate slowly (Figure 1 (c)). Both Figure 1 (b) and Figure 1 (c) were computed using a reparameterization curve, $t(r)$.

$h_{00}(t)$	$=$	$t^2(2t-3)+1$
$h_{01}(t)$	$=$	$t^2(3-2t)$
$h_{10}(t)$	$=$	$t(t-1)^2$
$h_{11}(t)$	$=$	$t^2(t-1)$

Table 1: The cubic Hermite basis functions.

2.1.2 Hermite Interpolant

The cubic Hermite interpolant is used to define the blend surface that is symbolically computed. Table 1 defines the four basis functions of the cubic Hermite interpolant. Two positional constraints, C_1 and C_2, and two tangential constraints, T_1 and T_2, are needed to define the cubic Hermite curve (See [11] for more details on the Hermite interpolant),

$$H(t) = h_{00}(t)C_1 + h_{01}(t)C_2 + h_{10}(t)T_1 + h_{11}(t)T_2. \tag{3}$$

It is worth noting that an immediate consequence of using the cubic Hermite basis functions is the following: $H(0) = C_1$, $H(1) = C_2$, $H'(0) = T_1$, and $H'(1) = T_2$.

2.2 Cross Boundary Tangent Definition

To use a cubic Hermite interpolation scheme (equation (3)), one must have cross boundary tangent constraints, $T_i(t)$, $i = 1, 2$, as well as the two rail curves, $C_1(t)$ and $C_2(t)$.

One way to define the cross boundary tangential constraints is to specify cross boundary tangent curves, $T_1(t)$ and $T_2(t)$. The blend surface can then be defined using the cubic Hermite basis functions (see equation 3 and Table 1) as

$$H(u, v) = h_{00}(v)C_1(u) + h_{01}(v)C_2(u) + h_{10}(v)T_1(u) + h_{11}(v)T_2(u). \tag{4}$$

Defining the $T_i(u)$ curves is a difficult problem. Either these curves must be computed automatically or the user should be provided with geometrically intuitive tools to derive them. We explore several possibilities with different levels of user interactions that are required.

Let $c(t) = (u(t), v(t))$ be a curve in the parametric space of surface $S(u, v)$, and let curve $C(t)$ be the composition $C(t) = S(c(t)) = S(u(t), v(t))$. Let $S^u(u, v) = \frac{\partial S(u,v)}{\partial u}$ and $S^v(u, v) = \frac{\partial S(u,v)}{\partial v}$, be the two partial derivatives of S. $N(t) = \pm(\frac{dv(t)}{dt}, -\frac{du(t)}{dt})$ is a vector field [5] curve representing the unnormalized normal of $c(t)$ in S's parametric space. For a regular surface, when $S^u(u(t), v(t)) \times S^v(u(t), v(t)) \neq 0$, $S^u(u(t), v(t))$ and $S^v(u(t), v(t))$ form a basis for P, the tangent plane of S. In [14], it is suggested that the two cross boundary tangents, T_i, $i = 1, 2$, could be selected using the partial derivatives of S and $N(t)$, yielding,

$$\begin{aligned} T_i^1(t) &= \pm\left\langle \left(\frac{dv_i(t)}{dt}, -\frac{du_i(t)}{dt}\right), \left(S_i^u(u_i(t), v_i(t)), S_i^v(u_i(t), v_i(t))\right) \right\rangle \\ &= \pm\left(\frac{dv_i(t)}{dt} S_i^u(u_i(t), v_i(t)) - \frac{du_i(t)}{dt} S_i^v(u_i(t), v_i(t))\right), \end{aligned}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product.

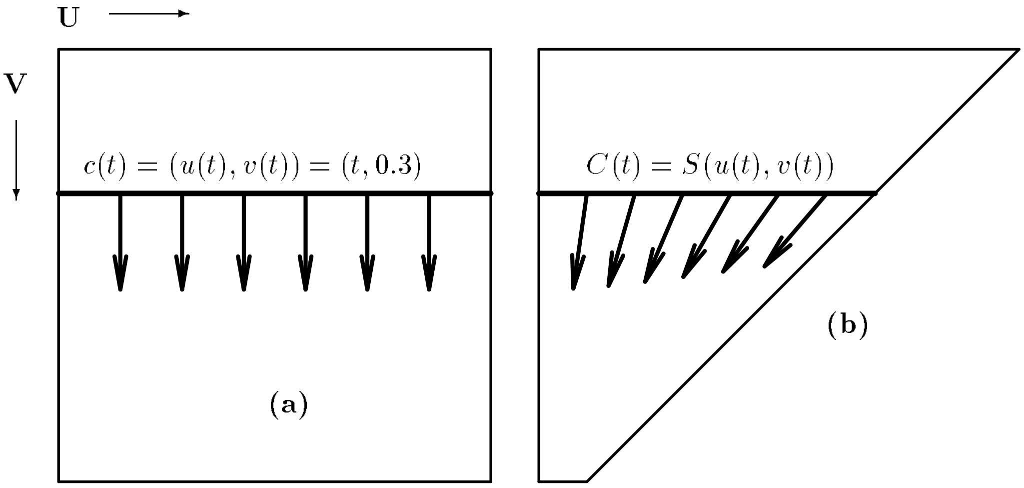


Figure 2: A normal to $c(t)$ in parametric space (a) is not, in general, normal to $C(t)$ in Euclidean space (b).

This method uses the unnormalized normal to $c(t)$ in the parametric space of S, $N(t)$. However, $T_i^1(t)$ is not, in general, normal to $C(t)$ in Euclidean space. Figure 2 shows one such example.

Denote by $n(u,v) = (S^u(u,v) \times S^v(u,v))$ the vector field of the unnormalized normal of surface S. Using $n(u,v)$, an alternative definition to $T_i^1(t)$ exploits $n(u,v)$:

$$T_i^2(t) = \pm\left(\frac{dC_i(t)}{dt} \times n_i(u_i(t), v_i(t))\right). \tag{5}$$

$T_i^2(t)$ is guaranteed to be orthogonal to $C(t)$ [10]. The vector field curve $n(u(t), v(t))$ provides a normal to surface S for each point along the Euclidean curve $C(t) = S(u(t), v(t))$, while $\frac{dC(t)}{dt}$ is the tangent curve of $C(t)$. Equation(5) provides the directions in P, the tangent plane of S, that are also normal to $C(t)$.

In [10], a different approach is proposed for the cross boundary tangent selection. Let $K_1(t) = C_2(t) - C_1(t)$ and $K_2(t) = C_1(t) - C_2(t) = -K_1(t)$ be vector field curves. $K_i(t)$ are called *guiding curves* [14]. The projection of $K_i(t)$ on the tangent plane of $S_i(u,v)$ can provide a cross boundary tangent direction in surface $S_i(u,v)$,

$$T_i^3(t) = K_i(t) - \frac{\langle K_i(t), n_i(u_i(t), v_i(t))\rangle}{\langle n_i(u_i(t), v_i(t)), n_i(u_i(t), v_i(t))\rangle} n_i(u_i(t), v_i(t)). \tag{6}$$

One can consider defining the two guiding curves, $K_i(t)$, independently for each of the blend rail curves, resulting in cross boundary tangent curve definitions $T_i^4(t)$. $T_i^3(t)$ is then a special case of $T_i^4(t)$, in which $K_1(t) = -K_2(t)$.

Although $n_i(u(t), v(t))$ are not the unit length normal, equation (6) is implicitly normalized by constructing a rational expression in which the denominator is equal to the magnitude of $n(u(t), v(t))$ squared, $\langle n(u(t), v(t)), n(u(t), v(t))\rangle = \|n(u(t), v(t))\|^2$. In other words, a division by a scalar curve can be represented as a rational curve without explicitly computing the division, hence $T_i^3(t)$ and $T_i^4(t)$ are rational expressions, even if S is a polynomial surface.

One can extend the specification method for $T_i^1(t)$ and allow an arbitrary linear combination of S^u and S^v for another type of cross boundary derivatives, as

$$\begin{aligned} T_i^5(u) &= \pm\left\langle \Big(S^u(u(t),v(t)), S^v(u(t),v(t))\Big), \Big(r(t),s(t)\Big)\right\rangle \\ &= \pm\Big(S^u(u(t),v(t))r(t) + S^v(u(t),v(t))s(t)\Big), \end{aligned} \tag{7}$$

where $d(t) = (r(t), s(t))$ is an arbitrary two dimensional curve that is independent of $c(t) = (u(t), v(t))$.

$T_i^1(t)$ to $T_i^5(t)$ are five different methods to define the cross boundary tangents along the rail curves, $C_i(t)$. These five methods are all symbolically computable and exactly representable in the (piecewise) polynomial or rational domains since in these cross tangent definitions only the derivative, sum, difference, product, and composition operators are used.

We conclude this section with a remark on the degree of the expected tangent fields, T_1 to T_5. Assume S is a surface of degree n by n. Assume $c(t) = (u(t), v(t))$ is of degree m. Then, the composition of $S(u(t), v(t))$ is of degree $2mn$. The composition of $c(t)$ into the unnormalized normal vector field, $n(u(t), v(t))$ yields a vector field with an even higher degree of $2m(2n-1)$. For a biquadratic surface S and a quadratic curve $c(t)$, $S(u(t), v(t))$ becomes degree 8 and $n(u(t), v(t))$ degree 12. For $m = n = 3$, the cubic cases, $S(u(t), v(t))$ becomes degree 18 and $n(u(t), v(t))$ degree 30. These high degree vector fields immediately affects the degree of the resulting Hermite blend surface (Equation (4)) and should be taken into account in cases it might impose difficulties. Finally, one should recall that low degree $c(t)$ curves can greatly alleviate the high degrees in the composition. For an arbitrary straight line in the parametric space of S, $m = 1$.

2.3 Cross Boundary Tangent Normalization

The resulting cross boundary tangent curves $T_i(t)$, discussed in Section 2.2, are not normalized to a unit length. Computed symbolically, the magnitude of these vector field curves can significantly vary, on the order of several magnitudes. Since this has an immediate effect on the shape of the computed blend it is desirable to normalize them close to a fixed magnitude. Some variation on the magnitude of $T_i(t)$ can probably be allowed if this variation is within a user specified tolerance. Once normalized to a unit length within this bound, scaling $T_i(t)$ with a constant k, can be symbolically computed as $kT_i(t)$, whereas variable scaling is achievable by a scalar function of t, $k(t)$, as the product $k(t)T_i(t)$.

Let $m(t) = \langle T(t), T(t)\rangle = \|T(t)\|^2$ be the scalar curve equal to the magnitude squared of the cross boundary tangent curve $T(t)$. Let $M(t)$ be a scalar curve defined using $m(t)$, with the same order and continuity (knot vector), by substituting each coefficient k in the control polygon of $m(t)$ as $\frac{1}{\sqrt{k}}$ in the control polygon of $M(t)$.

The error between a C^2 continuous function and its Schoenberg variation diminishing spline approximation [15] over a knot vector $\{t_i\}$ is $O(|\{t_i\}|^2)$, where $|\{t_i\}| = max_i\{t_{i+1} - t_i\}$. By using a sequence $\{T^i(t)\}$, of refined representations of $T(t)$, based on the same sequence of knot vectors, and a sequence $\{m^i(t)\}$, where $m^i(t) = \langle T^i(t), T^i(t)\rangle$, using the Schoenberg variation diminishing spline approximations to $T(t)$, the sequence $\{M^i(t)\}$ converges to $M(t) = \frac{1}{\|T(t)\|}$. Let $\hat{T}^i(t) = T^i(t)M^i(t)$. By suitable refinement, $\hat{T}^i(t)$ can converge as closely as needed to the

Curve (Figure 3)	Maximum error
Original $T(t)$ (a)	2.025
3 Inserted knots (b)	0.25
6 Inserted knots (c)	0.08333
12 Inserted knots (d)	0.02571

Table 2: Convergence of $\|T(t)\|$ to unit length using refinement on $T(t)$.

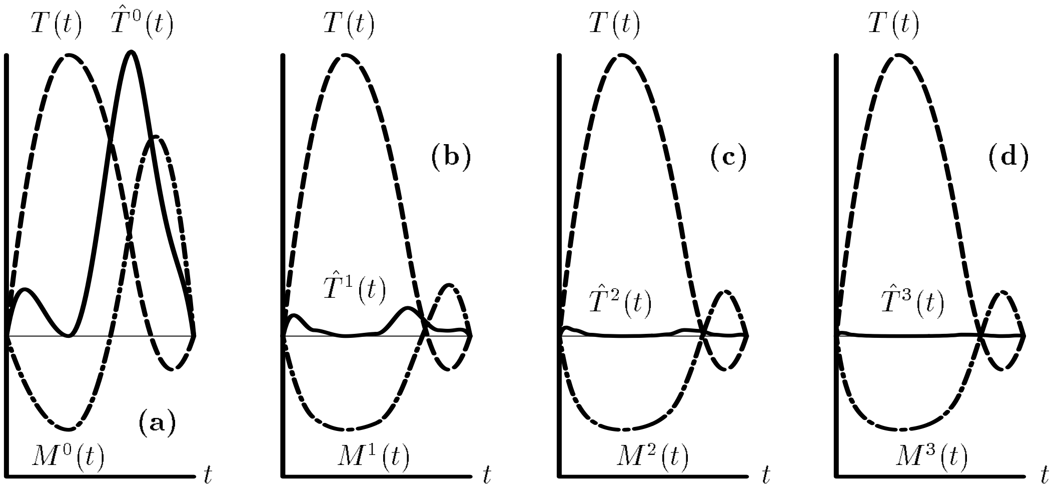


Figure 3: $\|\hat{T}(t)\|$ converges to unit length by scaling $T^i(t)$ with $M^i(t)$, an approximation to $1/\|T^i(t)\|$. Three refinement steps are shown with $T(t)$ dashed, $M^i(t)$ dotted-dashed, and $\hat{T}^i(t)$ solid. In (a) the original curve, $T(t)$, is shown, while (b) to (d) show $T(t)$ refined with 3, 6, and 12 equally spaced interior knots, respectively.

exact unit length cross boundary tangent. Figure 3 shows an example of this approach and demonstrates the effect of the refinement on the convergence. $M^i(t)$ was computed from the original quadratic B-spline curve, $T(t)$, which has two interior knots in Figure 3(a). In Figure 3(b) to (d), 3, 6, and 12 knots were inserted, respectively, equally spaced in the parametric domain of $T(t)$. The convergence rate of $\|\hat{T}^i(t)\|$ to the unit length is shown in Table 2.

Adaptive refinement control could apply refinement only to those portions of the domain of $T(t)$ that have errors in $\|T(t)\|$ larger than an allowed tolerance. An iterative algorithm that converges as closely as needed to the exact unit length of the cross boundary tangent curve, can easily be derived in a similar way to the one used in [7] for computation of offset approximations of freeform curves and surfaces with error bounds.

The operation of vector field normalization approximation can be applied to various other applications. The normalization of a normal field of a curve, can open the way for a simple offset approximation of a curve. Normalized vector fields can also be employed in animation for motion control.

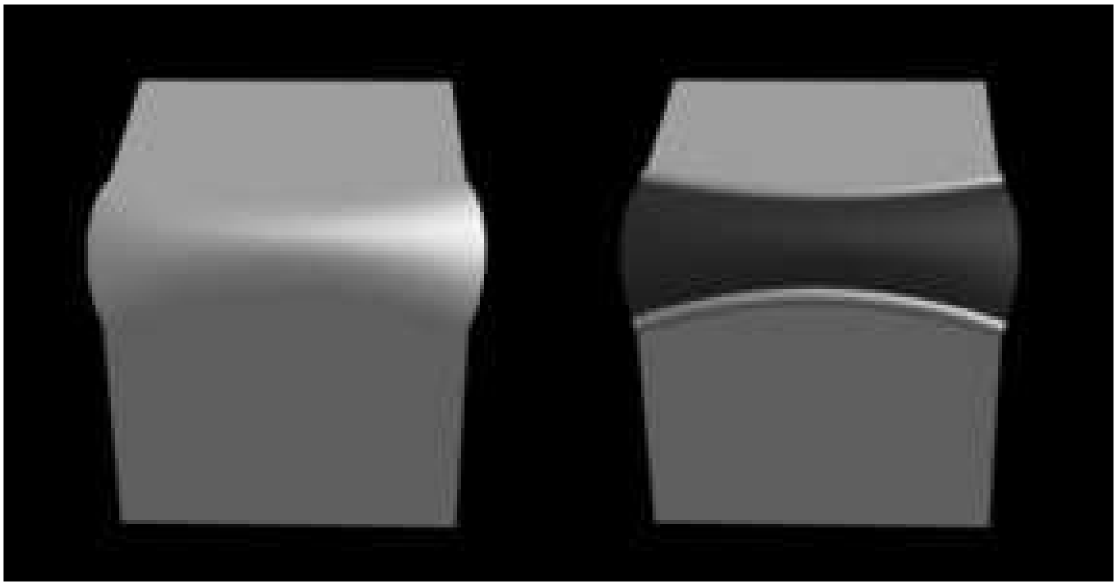


Figure 4: Blending using $T_i^2(t) = \pm \left(\frac{dC(t)}{dt} \times n(u(t), v(t)) \right)$.

3 Examples

Figure 4 provides a simple example for $T_i^2(t)$ cross boundary tangent curve selection. Figure 5 is another, more complex, example of the use of $T_i^2(t)$. The user, once selecting the $T_i^2(t)$ method, can control the *magnitude* of the cross boundary tangent curves, by specifying a scaling curve $k(t)$ as discussed Section 2.3. In Figure 5, the cross boundary tangent magnitude varies along the blend surface, larger at the wide blend surface region and smaller at the narrow blend surface region.

By specifying two guiding curves, $K_i(t)$, and projecting then onto the tangent planes of the surfaces along the two rail curves, the vector field curves $T_i^3(t)$ are defined. By defining two independent guiding curves, $K_i(t)$, one for each rail curve, the user is provided with a large degree of freedom which may be helpful when warped blending is required (see Figure 6).

In some cases, the approach using $T_i^2(t)$ may have undesired effects because it provides the user with no control on the cross boundary tangent directions. In Figure 4, the blend surface bends out of the primary surfaces since its interpolated boundaries are perpendicular to its rail curves where they all meet. Using the $T_i^5(t)$ method, one can specify the cross boundary tangent curves as linear combinations of $S^u(u, v)$ and $S^v(u, v)$. Although not automatic, this method does provide a large degree of freedom to control the cross boundary tangent curves, $T_i(t)$. Figure 7 shows an example of the use of $T_i^5(t)$, which overcomes the rounding effects created by using $T_i^2(t)$. The two scalar curves, $r(t)$ and $t(t)$, of $d(t) = (r(t), s(t))$, are used to provide the coefficients of the linear combination of the partial derivative surfaces (see equation (7)). Each is specified as the difference of two curves, $d_1(t)$ and $d_2(t)$, in the parametric space of the surface. That is $d(t) = (r(t), s(t)) = d_1(t) - d_2(t) = (r_1(t) - r_2(t), s_1(t) - s_2(t))$. Figure 8 shows the effect of scaling the cross boundary tangent magnitudes when using the $T_i^2(t)$ method to compute a blend surface between two identical spheres.

Finally, the spout connection to the body of the Utah teapot was rounded using the $T_i^2(t)$

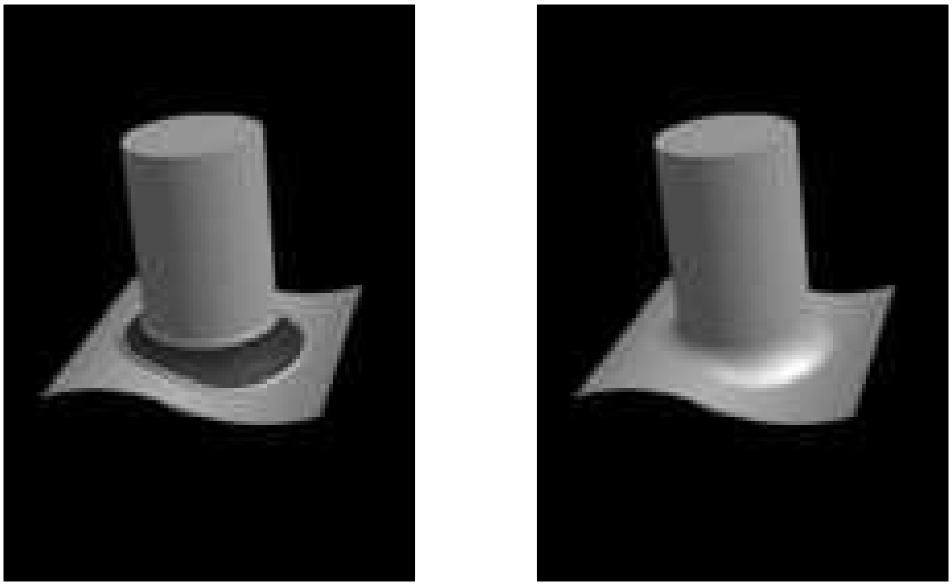


Figure 5: Blending using $T_i^2(t) = \pm \left(\frac{dC(t)}{dt} \times n(u(t), v(t)) \right)$.

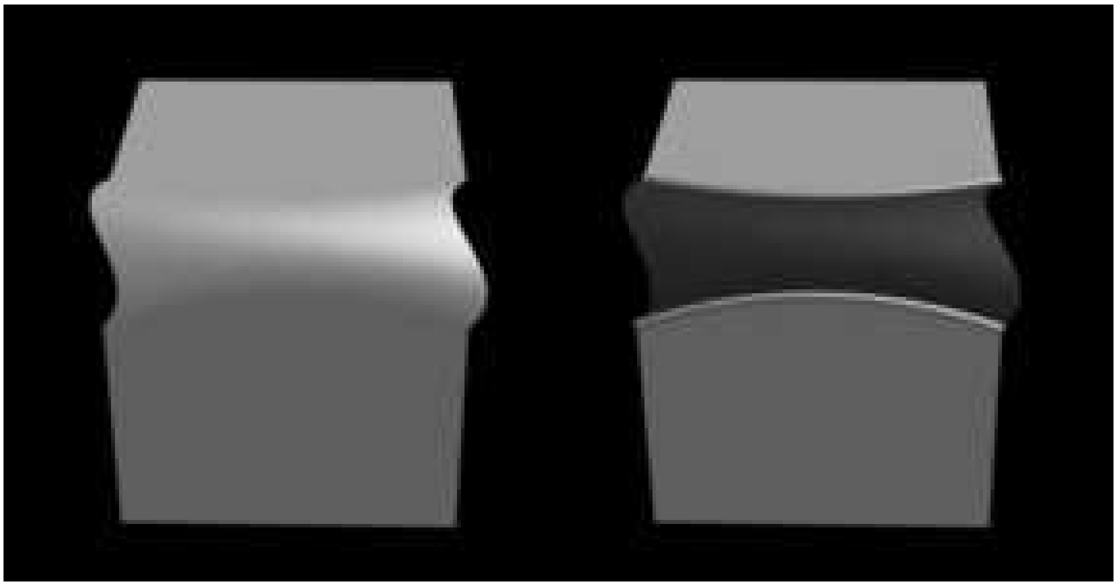


Figure 6: Blending using $T_i^4(t) = K_i(t) - \frac{\langle K_i(t), n_i(u(t), v(t)) \rangle}{\langle n_i(u(t), v(t)), n_i(u(t), v(t)) \rangle} n_i(u(t), v(t))$.

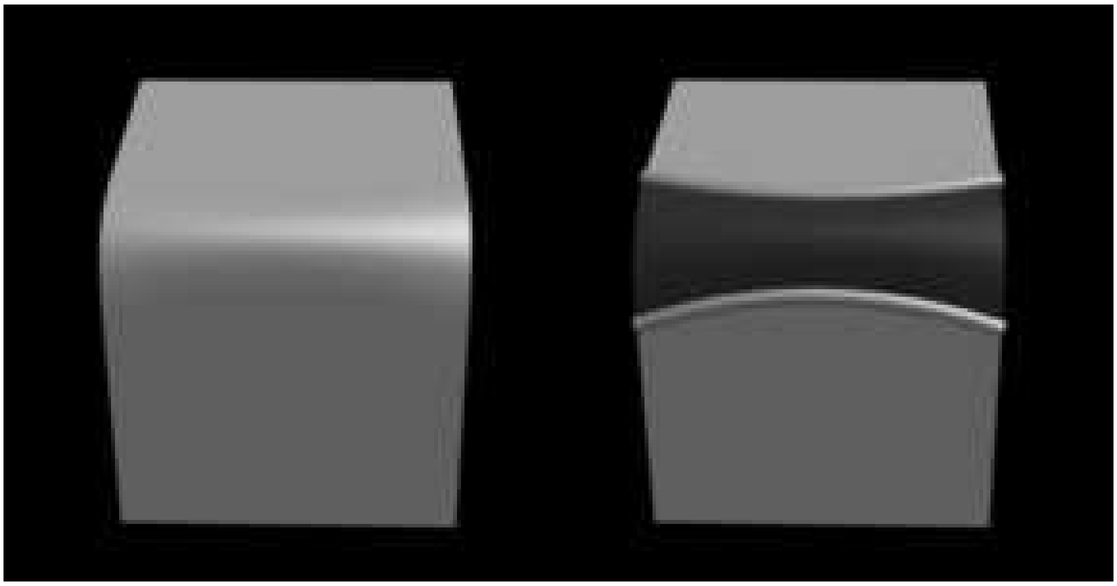


Figure 7: Blending using $T_i^5(t) = \pm\left(S^u(u(t), v(t))r(t) + S^v(u(t), v(t))s(t)\right)$.

method, which is guaranteed to be perpendicular to the rail curves in Euclidean space. Figure 9 shows the original teapot (a), the filleted teapot (b), and the filleted region enhanced (c).

4 Conclusions

The symbolic computation method proved to be a powerful tool in creating blend surfaces within machine accuracy and with a geometrically meaningful control. With the ability to symbolically compute and normalize cross tangent curves, the symbolic approach for the construction of blend surfaces can be automated. This approach is unique in that it provides an accurate tangent plane continuity for the entire boundary of the blend surface, as opposed to accurate tangent plane continuity at only a finite set of locations along this domain. Furthermore, once the tangent curves are normalized, results can be interacted with in a geometrically meaningful way.

It is our opinion that the symbolic blend surface construction, as a promising way toward a robust and fast computation of blend surfaces for freeform based models, will find its way into current solid modeling systems.

5 Acknowledgment

The authors are grateful to Elaine Cohen, Mark Bloomenthal, and Beth Cobb for their valuable remarks on the various drafts of this paper.

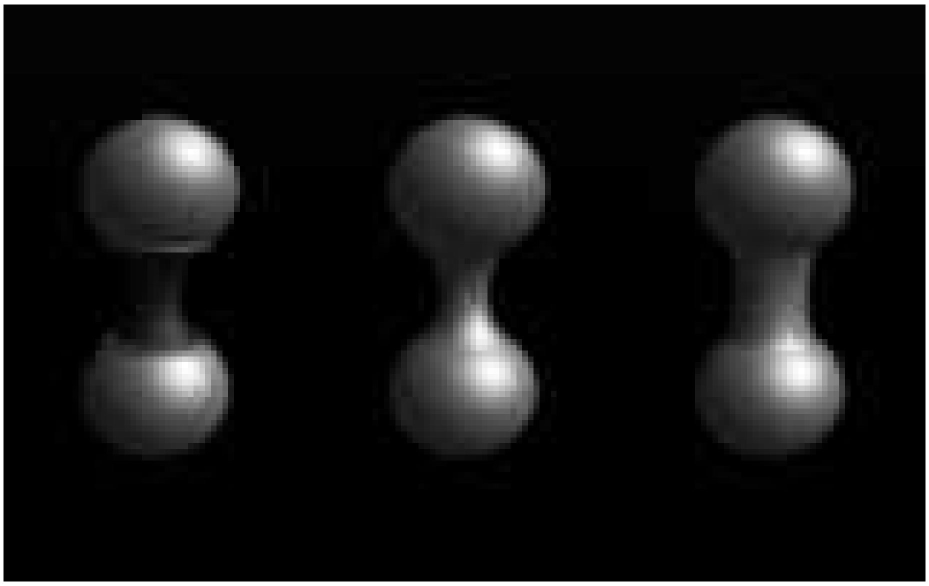
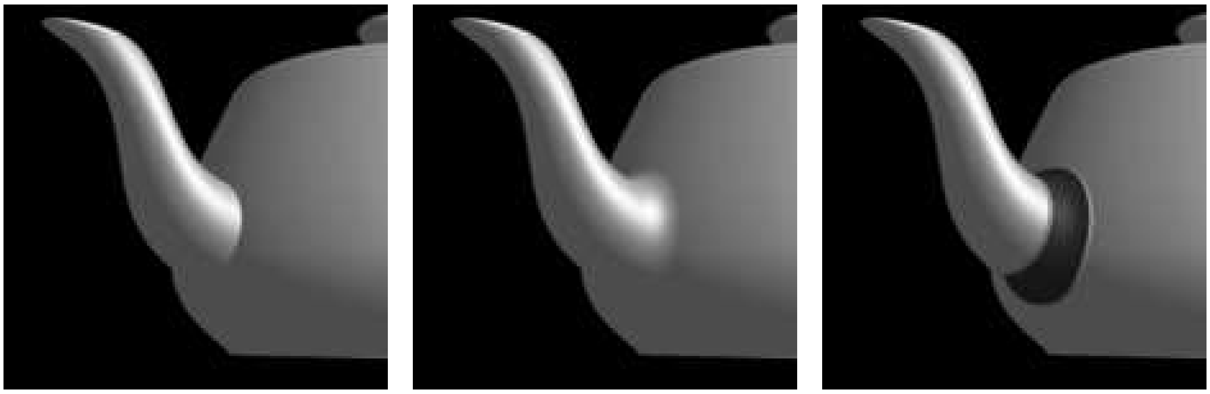


Figure 8: Blending using two different magnitudes of $T_i^2(t) = \pm\left(\frac{dC(t)}{dt} \times n(u(t), v(t))\right)$.



(a) **(b)** **(c)**

Figure 9: Spout connection to the Utah teapot rounding using $T_i^2(t) = \pm\left(\frac{dC(t)}{dt} \times n(u(t), v(t))\right)$.

References

[1] R. H. Bartels and I. Hardtke. Speed Adjustment for Key-Frame Interpolation. Graphics Interface 89 Proceedings, pp 14-19, 1989.

[2] C. L. Bajaj and I. Ihm. Algebraic Surface Design with Hermite Interpolation. ACM Transactions on graphics, Vol 11, No 1, January 1992, pages 61-91.

[3] A. P. Bien and F. Cheng. A Blending Model for Parametrically Defined Geometric Objects. Symposium on Solid Modeling Foundation and CAD/CAM Applications, 1991, pp. 339-347.

[4] B. K. Choi and S. Y. Ju. Constant Radius Blending in Surface Modeling. Computer Aided Design, Vol. 21, No. 4, pp 213-220, May 1989.

[5] M. P. DoCarmo. Differential Geometry of Curves and Surfaces. Prentice-Hall 1976.

[6] G. Elber. Free Form Surface Analysis using a Hybrid of Symbolic and Numeric Computation. Ph.D. thesis, University of Utah, Computer Science Department, 1992.

[7] G. Elber and E. Cohen. Error Bounded Variable Distance Offset Operator for Free Form Curves and Surfaces. International Journal of Computational Geometry and Applications, Vol. 1., No. 1, pp 67-78, March 1991.

[8] G. Farin. Curves and Surfaces for Computer Aided Geometric Design. Academic Press, Inc. Second Edition 1990.

[9] R. T. Farouki and V. T. Rajan. Algorithms for Polynomials in Bernstein Form. Computer Aided Geometric Design 5, pp 1-26, 1988.

[10] D. J. Fillip. Blending Parametric Surfaces. ACM Transaction on Graphics, Vol. 8, No. 3, pp 165-173, July 1989.

[11] J. Hoschek and D. Lasser. Fundamentals of Computer Aided geometric Design. A K Peters, Wellesley, Massachusetts, 1993.

[12] K. Kim. Blending Parametric Surfaces. M.Sc. thesis, University of Utah, Computer Science Department, 1992.

[13] R. K. Klass and B. Kuhn. Fillet and Surface Intersections Defined by Rolling Balls. Computer Aided Geometric Design 9, pp 185-193, 1992.

[14] P. Koparkar. Parametric Blending using Fanout Surfaces. Symposium on Solid Modeling Foundation and CAD/CAM Applications, 1991, pp. 317-327.

[15] M. Marsden and I. J. Schoenberg. On Variation Diminishing Spline Approximation Methods. Mathematica, Vol. 8(31), No. 1, pp 61-82, 1966.

[16] K. Morken. Some Identities for Products and Degree Raising of Splines. Constructive Approximation, Vol 7, pp 195-208, 1991.

[17] J. R. Rossignac and A. A. G. Requicha. Constant Radius Blending in Solid Modelling. Computers in Mechanical Engineering, pp 65-73, July 1984.

[18] K. Vida, R. R. Martin, and T. Varady. A Survey of Blending Methods that use Parametric Surfaces. Computer Aided Design, Vol. 26, No. 5, pp 341-365, May 1994.

[19] J. Warren. Blending Algebraic Surfaces. ACM Transactions on Graphics, Vol. 8, No. 4, pp 263-278, August 1989.