Proceedings of DETC'2001
ASME 2001 Design Engineering Technical Conferences
and Computers and Information in Engineering Conference
Pittsburgh, Pennsylvania, September 9-12, 2001

DETC01/DAC-21065

AN EFFICIENT ALGORITHM FOR CIRCULAR BLENDING OF EDGES

Sharad K. Jaiswal
Graduate Student
Department of Mechanical Engineering
Indian Institute of Science
Bangalore - 560 012, INDIA.
Email: sharadj@mecheng.iisc.ernet.in

A. Ghosal
Professor
Department of Mechanical Engineering
Indian Institute of Science
Bangalore - 560 012, INDIA.
Email: asitava@mecheng.iisc.ernet.in

B. Gurumoorthy [1]
Associate Professor
Department of Mechanical Engineering
Indian Institute of Science
Bangalore - 560 012, INDIA.
Email: bgm@mecheng.iisc.ernet.in

ABSTRACT

This paper describes a method for constructing circular blends using geometric tools. The algorithm presented in this paper is based on marching along a characteristic direction on the tangent plane to the Voronoi surface of the two surfaces being considered for blending. Starting from any point on the edge to be blended, the algorithm converges to the spine curve. The characteristic direction of marching lies on the plane containing the points in assignment and the tangent plane to the Voronoi surface. The spine curve generation algorithm presented in this paper, does not require computing offsets of surfaces or an explicit evaluation of surface-surface intersection (SSI). The algorithm presented is computationally simple and fast, and can be used for constant and variable radius circular blending of surfaces, each of which is G^2 continuous. The algorithm can also be used to obtain the surface-surface intersection curve by setting the radius of blend to zero.

KEYWORDS

Edge blending, Voronoi surface, variable radius blending

INTRODUCTION

Blending is one of the most important editing tool provided by CAD systems for local modifications of primary models. It is often used in computer aided geometric design for aesthetic or functional reasons. Of all the blending techniques, radius blends are the most popular blends because of it's simple geometric description. A radius blend can be imagined as the trimmed envelope of a rolling ball of constant or variable radius swept along a spine curve such that the rolling ball always touches the base surfaces to be blended. From an algorithmic point of view, blending one or more edges of solid model can be divided into two subtasks. The first is to create a surface that provides smooth transition between the adjacent surfaces defining the edge. Secondly, the surfaces need to be trimmed properly and integrated into the body such that a valid solid model is maintained. While the first step is a purely geometric problem, the second one involves both geometric and topological operations. This paper addresses the first subtask.

Boundary surfaces meeting along tangent discontinuous edges are quite common in the design of mechanical components. Hence the problem of blending in computer aided geometric design has a fairly long history. Vida et. al(Vida *et al.*, 1994) provide a comprehensive review of work to date on parametric and algebraic blending. 'Rolling-ball' method was employed by Rossignac and Requicha (Rossignac and Requicha, 1984), so that the blends are approximated by a sequence of torus segments. For constant radius rolling-ball blends, most authors (see for example (Barnhill *et al.*, 1993; Choi and Ju, 1989; Klass and Kuhn, 1992; Varady *et al.*, 1989)) have tried to obtain the spine curve by tracing the intersection of offsets with an equal radius from the base surfaces. However, finding intersection of offset surfaces is

[1] Address all correspondence to this author.

a computationally intensive task, as offset surfaces have no exact representation (Farouki, 1986).

Finding surface-surface intersection(SSI) has been discussed by various authors such as Bajaj et. al(Bajaj *et al.*, 1987), Barnhill and Kersey(Barnhill and Kersey, 1989), Heo et. al(Heo *et al.*, 1999) and others. Chang et. al(Chang *et al.*, 1994) have discussed surface-surface intersection using parallelism to achieve speed and precision simultaneously. In all these methods, determining the starting points of intersection is very crucial. Karim and Yeh(Abdel-malek and Yeh, 1997) have presented two numerical algorithms for computing starting points on the curve of intersection of parametric surfaces. The offset surfaces can be represented more accurately, as a procedural surface when the parameterization is identical, but evaluation of surface intersection is not a very easy task for such procedurally defined surfaces. Hartmann(Hartmann, 1998) has presented a numerical implicitization scheme to find SSI for procedural surfaces.

Farouki and Sverrisson(Farouki and Sverrisson, 1996) have presented another approximation of constant radius rolling-ball blends for parametric surfaces where they have derived the spine curve by tracing the intersection of offset curves directly on the original surfaces. Their method does not require explicit representation of the offset surfaces, which are, in general non-rational.

The theoretical foundations of variable radius rolling-ball (VRRB) blends are described in Pegna(Pegna, 1987) and Pegna and Wilde(Pegna and Wilde, 1990). The spine curve of variable radius rolling-ball (VRRB) blends can also be obtained as an intersection of offset surfaces with varying offset distances. The offset distances are prescribed by using some radius function. In certain cases, the radius function is assumed to be monotonic between a minimum and maximum radius, both of which are chosen by the user. The intersection curve of two offsets with a radius equal to the average of the minimum and maximum radius values is traced and used as a control curve. Chuang and Hwang(Chuang and Hwang, 1997) propose several geometric constraints to specify the variable radius that constrains the variable-radius spine curve. The constraints have been generalized as $r(\theta) = c/f(\theta)$, where θ is the angle between two normals dropped from a point on the spine curve to the base surfaces. Chandru, Dutta and Hoffmann(Chandru *et al.*, 1990) use cyclides and other natural quadrics to define blend surfaces of exact tangent continuity to the base surfaces. They were also the first to introduce and explicitly use the concept of Voronoi surfaces.

Singularities of blend surfaces and their differential geometric properties have been discussed by Lukacs(Lukacs, 1998). He has introduced the concept of geodesic blend surface, which avoids local self-intersections if both surfaces locally enclose the variable radius ball.

Pegna(Pegna, 1987) also categorizes blends as *circular* blends and *spherical* blends. The former type of blends are obtained by sweeping circles of varying radius whereas the latter type of blends are generated by the envelope surface of a rolling ball. In this paper, we focus on the circular blends where the blend surface is swept by principal circles of a rolling ball and present a procedure that doesn't require offset of surfaces or explicit evaluation of surface-surface intersection(SSI). The procedure finds the spine curve of the blend by marching on the Voronoi surface of the two base surfaces. Properties of the Voronoi surface and offset surfaces are used to first find a starting point and then to march on the Voronoi surface. This approach is well suited for constant as well as variable radius circular blends of surfaces that are G^2 continuous. As a by-product, our algorithm can be also used to obtain the surface-surface intersection curve. Before we proceed further, we present, in the rest of this section, the blending terminology that has been used in this paper followed by some results that establish properties of Voronoi surfaces that will be used by the algorithm. The algorithm for circular blending of edges is presented in detail next. Results of implementation involving blending of surfaces using constant and variable radii are presented. We also present a result on obtaining the SSI curve. Finally, we present our conclusions.

Blending terminology

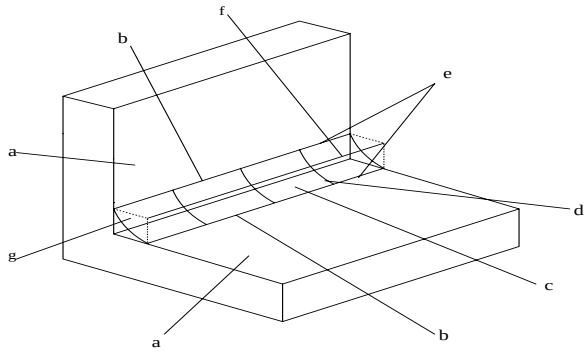

Figure 1. Terminology : (a) base surfaces, (b) trimlines, (c) blending surface, (d) profile curve, (e) pair of points in assignment, (f) spine curve, (g) plane of reference.

In the figure 1 the *base surfaces* are to be blended smoothly. The *blending surface* is considered as a swept surface, generated by sweeping a ball along a given or evaluated longitudinal trajectory. Such a trajectory is called

spine curve and projection of the spine curve on the base surfaces gives *trimlines* or the *linkage curve*. At each point of the spine curve, a cross sectional *profile curve* is associated with it which locally defines the shape of the blend. Any point on spine curve can be projected onto the base surfaces and thus on the trimlines. We refer to this process as *assignment* and the two points under consideration are called *points of assignment*. The plane containing the normals at the points of assignment is called the *plane of reference*.

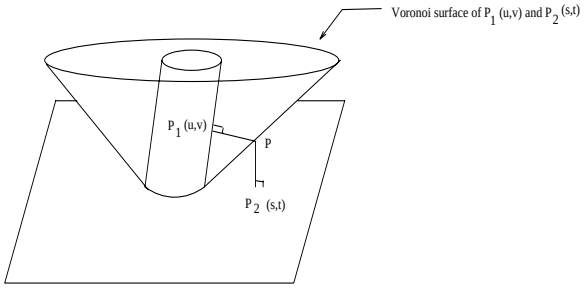

Figure 2. The Voronoi Surface of a Plane and Cylinder

THEORETICAL BACKGROUND

Rolling-ball blend can be visualized as a family of balls centered on a three dimensional curve called spine curve and maintaining contact with both the base surfaces simultaneously. It implies that the spine curve is a set of points which are locally equidistant from both the base surfaces. The spine curve in turn can be visualized as a curve lying on a surface which is locally equidistant from the both the surfaces. Such a surface is called Voronoi surface(Chandru *et al.*, 1990). Usually we don't have any closed form representation for a Voronoi surface.

Let $\mathbf{S}(u,v)$ be a smooth parametric surface. Then the offset surface of $\mathbf{S}(u,v)$ by distance r is given by

$$\mathbf{S_o}(u,v) = \mathbf{S}(u,v) + r\,\hat{\mathbf{n}}(u,v). \tag{1}$$

where r is the distance of the offset surface from the original surface and $\hat{\mathbf{n}}$ is the unit surface normal of $\mathbf{S}$. The normal is given by

$$\hat{\mathbf{n}}(u,v) = \frac{\frac{\partial \mathbf{S}}{\partial u} \times \frac{\partial \mathbf{S}}{\partial v}}{\left|\frac{\partial \mathbf{S}}{\partial u} \times \frac{\partial \mathbf{S}}{\partial v}\right|} = \frac{\mathbf{N}(u,v)}{|\mathbf{N}(u,v)|}.$$

The offset surface defined by equation(1) is smooth if the surface $\mathbf{S}(u,v)$ is well parameterized i.e. $\frac{\partial \mathbf{S}}{\partial u} \times \frac{\partial \mathbf{S}}{\partial v} \neq 0$. The offset surface can be reparameterized identically if r is less than the smallest concave principal radius of curvature at point (u,v) otherwise the parameterization will be degenerate. Hence to avoid singularities in blends, the blend radius at a point should be less than the smallest concave principal radius of curvature at that point(Lukacs, 1998).

A *Voronoi curve/surface* for two (parametric) curves/surfaces is the locus of those points which are locally equidistant from both the curves/surfaces. If we consider $\mathbf{P_1}(u,v)$ and $\mathbf{P_2}(s,t)$ as two parametric surfaces, then we can write pencil of offset surfaces to $\mathbf{P_1}$ and $\mathbf{P_2}$ as

$$\begin{aligned} \mathbf{P_1^o}(u,v,r) &= \mathbf{P_1}(u,v) + r\,\hat{\mathbf{n}}_\mathbf{1}(u,v), \\ \mathbf{P_2^o}(s,t,r) &= \mathbf{P_2}(s,t) + r\,\hat{\mathbf{n}}_\mathbf{2}(s,t). \end{aligned} \tag{2}$$

In the above equation, $\hat{\mathbf{n}}_\mathbf{1}$ and $\hat{\mathbf{n}}_\mathbf{2}$ are normal to surfaces $\mathbf{P_1}$ and $\mathbf{P_2}$ and the Voronoi surface is given as intersection of these pencil of offset surfaces with same offset distance r. Figure 2 shows a Voronoi surface of an oblique cylinder intersecting a plane.

For a G^2 continuous surface $\mathbf{S}$ it can be shown that the normal to the offset surface $\mathbf{S_o}$ is parallel to that of the original surface $\mathbf{S}$ at any parameter value (u,v) (Hermann, 1995). It follows then that the offset of an offset surface is offset of the original surface, and the offset distance gets added algebraically. This is used in the proposed algorithm to incrementally find the centre of the ball (and the point on the spine curve) at any instance.

The following theorem (Hermann, 1995; Lukacs, 1998) establishes an important result that is used in the development of the blending algorithm.

Theorem: Let $\mathbf{P_1}(u,v)$ and $\mathbf{P_2}(s,t)$ be smooth surfaces and $\mathbf{p}$ be a point on their Voronoi surface which is at a distance r from both the surfaces. Let $\mathbf{p_1}$ and $\mathbf{p_2}$ be the footprints of $\mathbf{p}$ on the base surfaces $\mathbf{P_1}(u,v)$ and $\mathbf{P_2}(s,t)$ respectively. If $\mathbf{p_1}$ and $\mathbf{p_2}$ do not coincide (i.e. $\mathbf{p_1} \neq \mathbf{p_2}$) and r is less than the smallest concave principal curvature radii then the Voronoi surface is a plane normal to vector $(\mathbf{p_2} - \mathbf{p_1})$ in the neighborhood of $\mathbf{p}$, and the Voronoi surface at a point $\mathbf{p}$ is given by vector $(\mathbf{p_2} - \mathbf{p_1})$ (Hermann, 1995; Lukacs, 1998).

This is shown schematically in figure 3.

From the above theorem it is possible to determine the direction of the march towards the spine curve. To ensure tangent plane continuity between the blend surface and the base surfaces, the blending arc through points $\mathbf{p_1}$ and $\mathbf{p_2}$ on the surfaces, $\mathbf{P_1}(u,v)$ and $\mathbf{P_2}(s,t)$, respectively, should lie on the plane containing the two normals through these

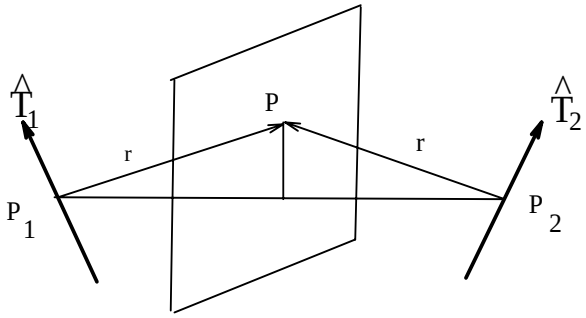

Figure 3. Tangent plane to the Voronoi Surface

points (note that this is only for circular blends). The intersection of this plane with the tangent plane to the Voronoi surface at the point $\mathbf{p}$, which is center of the blending arc, will give the tangential direction to march further on the Voronoi surface to change the offset distance r.

Let $\hat{\mathbf{n}}$ be the plane containing normals $\hat{\mathbf{n}}_\mathbf{1}$ and $\hat{\mathbf{n}}_\mathbf{2}$ at the points $\mathbf{p_1}$ and $\mathbf{p_2}$ to the surfaces $\mathbf{P_1}(u,v)$ and $\mathbf{P_2}(s,t)$. Let $\hat{\mathbf{T}}_\mathbf{1}$ and $\hat{\mathbf{T}}_\mathbf{2}$ be the tangents at points $\mathbf{p_1}$ and $\mathbf{p_2}$ to the surfaces $\mathbf{P_1}(u,v)$ and $\mathbf{P_2}(s,t)$ lying on the plane $\hat{\mathbf{n}}$. Then $\hat{\mathbf{T}}$, the tangential direction to move further, is along the angle bisector to $\hat{\mathbf{T}}_\mathbf{1}$ and $\hat{\mathbf{T}}_\mathbf{2}$ as shown in figure 4. We can write

$$\hat{\mathbf{T}} = \frac{\hat{\mathbf{T}}_\mathbf{1} + \hat{\mathbf{T}}_\mathbf{2}}{|\hat{\mathbf{T}}_\mathbf{1} + \hat{\mathbf{T}}_\mathbf{2}|} = \frac{\hat{\mathbf{n}}_\mathbf{1} + \hat{\mathbf{n}}_\mathbf{2}}{|\hat{\mathbf{n}}_\mathbf{1} + \hat{\mathbf{n}}_\mathbf{2}|} \tag{3}$$

where $\hat{\mathbf{T}}$, $\hat{\mathbf{T}}_\mathbf{1}$, $\hat{\mathbf{T}}_\mathbf{2}$, $\hat{\mathbf{n}}_\mathbf{1}$ and $\hat{\mathbf{n}}_\mathbf{2}$ are unit vectors.

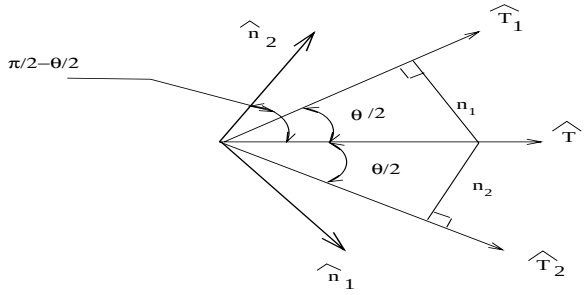

Figure 4. Tangent to the Voronoi curve

Based on the above, we formulate the radius blend algorithm in the following section.

BLEND ALGORITHM

The overall blending process involves the following steps.

- Get an initial point and plane of reference appropriately.
- Trace spine curve by marching along the spine curve.
- Refine point on spine curve.
- Terminate blending.
- Construct blend surface.

The steps to get the initial point and tracing of the spine curve makes use of the two results stated in the previous section. We describe each of the step in detail in this section.

Obtaining initial point and plane of reference

We start from a point on edge curve and try to obtain a plane containing the spine curve. However, since we don't have any *a priori* information about the spine curve, we can't decide about the reference plane. In such a situation, we proceed with any suitable plane, perform some iteration, get feedback about the plane, and finally try to converge to the plane.

To choose a suitable plane, we make use of the following reasoning: we know that the blend surface always follows the edge to be blended (i.e. intersection curve of the two surfaces to be blended). In addition, this intersection curve always lie on the Voronoi surface. This fact allows us to choose a suitable plane.

Once a suitable plane is found, we are required to maintain tangent plane continuity across the trimline. Hence, to start with, for the first plane of reference we select a plane which is normal to the intersection curve at any point on the edge (intersection curve). This point becomes the first input point to start with, and the tangent directions, $\hat{\mathbf{T}}_\mathbf{1}$ and $\hat{\mathbf{T}}_\mathbf{2}$, to the curves on the plane can be found by the intersection of the plane with the tangent planes to the base surfaces at the starting point. In fact $\hat{\mathbf{T}}_\mathbf{1}$ and $\hat{\mathbf{T}}_\mathbf{2}$ are the common vectors as well as the plane of reference, which contains normal to both the surfaces from that point. Thus we start with $\hat{\mathbf{T}}_\mathbf{1}$ and $\hat{\mathbf{T}}_\mathbf{2}$ and find the *step vector* $\mathbf{dT}$. The step vector is found directly using the two normals from the following equation

$$\begin{aligned} \mathbf{dT} &= \frac{(\hat{\mathbf{T}}_\mathbf{1} + \hat{\mathbf{T}}_\mathbf{2})}{\sqrt{1 - (\hat{\mathbf{T}}_\mathbf{1} \cdot \hat{\mathbf{T}}_\mathbf{2})^2}} dr, \\ &= \frac{(\hat{\mathbf{n}}_\mathbf{1} + \hat{\mathbf{n}}_\mathbf{2})}{1 + \hat{\mathbf{n}}_\mathbf{1} \cdot \hat{\mathbf{n}}_\mathbf{2}} dr \end{aligned} \tag{4}$$

The magnitude of the step vector is taken to be $|dT| = dr/sin(\theta/2)$, where θ is the angle between the two tangent vectors (figure 4) and dr is initially chosen to be (blend radius)/5.

The blending arc on current plane of reference may not be in a position to maintain G^1 continuity with the two

base surfaces. Hence, we correct the plane of reference by dropping normals $\mathbf{n_1}$ and $\mathbf{n_2}$ to the base surfaces from the new calculated point $\mathbf{p} = \mathbf{p} + \mathbf{dT}$. In each subsequent iteration we correct the plane of reference taking feedback from the geometry of the original base surfaces. The step vector considered here lies on the tangent plane of the Voronoi surface at point $\mathbf{p}$. After calculating the step vector we move to the next point $\mathbf{p} = \mathbf{p} + \mathbf{dT}$ on the Voronoi surface. It is worth noting that the new point $\mathbf{p}$ will not exactly lie on the Voronoi surface, and the errors in the subsequent iterations will accumulate. This implies that we also need to do point refinements at some stage, and this can be done using some higher-order predictor corrector method. A point refinement method discussed by Chuang and Hwang(Chuang and Hwang, 1997) is also very helpful in this case.

The above approach to find the initial point on the spine curve, taking a point on the edge or Voronoi surface as input, is presented as an algorithm.

ALGORITHM : *Spine*

INPUT: A point $\mathbf{p}$ on the edge curve.
OUTPUT: A point $\mathbf{p'}$ on the spine curve.
Steps:

1. Drop normals $\mathbf{n_1}$ and $\mathbf{n_2}$ from the point $\mathbf{p}$ (lying on the Voronoi surface) to the base surfaces.
2. Find the step vector $\mathbf{dT}$:
   We have used fourth order predictor-corrector method (Runge-Kutta) for this step. We write four intermediate step vectors $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\mathbf{D}$ as

$$\mathbf{A} = \frac{(\hat{\mathbf{n}}_{\mathbf{a1}} + \hat{\mathbf{n}}_{\mathbf{a2}})}{1 + \hat{\mathbf{n}}_{\mathbf{a1}} \cdot \hat{\mathbf{n}}_{\mathbf{a2}}} dr, \qquad \mathbf{B} = \frac{(\hat{\mathbf{n}}_{\mathbf{b1}} + \hat{\mathbf{n}}_{\mathbf{b2}})}{1 + \hat{\mathbf{n}}_{\mathbf{b1}} \cdot \hat{\mathbf{n}}_{\mathbf{b2}}} dr,$$

$$\mathbf{C} = \frac{(\hat{\mathbf{n}}_{\mathbf{c1}} + \hat{\mathbf{n}}_{\mathbf{c2}})}{1 + \hat{\mathbf{n}}_{\mathbf{c1}} \cdot \hat{\mathbf{n}}_{\mathbf{c2}}} dr, \qquad \mathbf{D} = \frac{(\hat{\mathbf{n}}_{\mathbf{d1}} + \hat{\mathbf{n}}_{\mathbf{d2}})}{1 + \hat{\mathbf{n}}_{\mathbf{d1}} \cdot \hat{\mathbf{n}}_{\mathbf{d2}}} dr.$$

   with $\hat{\mathbf{n}}_{\mathbf{a1}}, \hat{\mathbf{n}}_{\mathbf{a2}}, \hat{\mathbf{n}}_{\mathbf{b1}}, \hat{\mathbf{n}}_{\mathbf{b2}}, \hat{\mathbf{n}}_{\mathbf{c1}}, \hat{\mathbf{n}}_{\mathbf{c2}}, \hat{\mathbf{n}}_{\mathbf{d1}}$ and $\hat{\mathbf{n}}_{\mathbf{d2}}$ being the unit normals dropped from the points $\mathbf{p_a}, \mathbf{p_b}, \mathbf{p_c}$ and $\mathbf{p_d}$ to the base surfaces. The points are given as

$$\mathbf{p_a} = \mathbf{p}, \qquad \mathbf{p_b} = \mathbf{p} + \frac{\mathbf{A}}{2},$$

$$\mathbf{p_c} = \mathbf{p} + \frac{\mathbf{B}}{2}, \qquad \mathbf{p_d} = \mathbf{p} + \mathbf{C}.$$

   and finally we calculate the step vector as

$$\mathbf{dT} = \frac{1}{6}(\mathbf{A} + \mathbf{2B} + \mathbf{2C} + \mathbf{D}).$$

3. Calculate the next point of iteration on the Voronoi surface as $\mathbf{p} = \mathbf{p} + \mathbf{dT}$.
4. Find the distance r of $\mathbf{p}$ from the two base surfaces. $r = (|\mathbf{p} - \mathbf{P_1}(u, v)| + |\mathbf{p} - \mathbf{P_2}(s, t)|)/2$. Taking the minimum of the two distances would also work.
5. If r > blend radius

   rollback one iteration and proceed with a reduced step size dr (obtained by linear interpolation).

   else if (blend radius -r) < dr

   dr = (blend radius -r),
   if dr < TOL (where TOL is a pre-specified numerical tolerance),
   return $\mathbf{p'} = \mathbf{p}$;

   else

   go to step 1.

From, the above algorithm, we are now able to get a point which is on the spine curve within a prescribed tolerance. By repeating the same routine with different points on the edge curve, we can get different points on the spine curve. In this way, we can trace the entire spine curve but then the computation cost may be too large.

Tracing of the spine curve

To reduce the computation cost, we use the theorem defining the direction of the normal to the Voronoi surface, to march along the spine curve on the Voronoi surface of the two surfaces. We find the tangent to the spine curve on the tangent plane of the Voronoi surface at the point $\mathbf{p'}$ on the spine curve. For constant radius blend, the cross product of normals to the base surfaces at the foot point of $\mathbf{p'}$ will give the tangent direction to the spine curve. We now start marching on the spine curve to the trace the spine curve completely. In case of variable radius blends the tangent direction can't be specified easily. In this case, we take any arbitrary direction on the tangent plane of the Voronoi surface, preferably in the perpendicular direction to the plane of reference at the spine curve. In the next step the point is corrected using the algorithm *Spine* discussed before to satisfy the radius constraint.

At each step, the spine curve is traced along the tangent direction to the spine curve. By moving along the tangent to the spine curve, we may deviate from Voronoi curve. Hence we need some correction step in terms of point refinement technique.

Point refinement and blend termination

Since the spine curve and the Voronoi curve discussed above are linearly approximated along their tangent directions, the points evaluated above can't be assured to be lying exactly on the Voronoi surface or lie exactly equidistant from the base surfaces. In addition, the error in such a marching algorithm adds up, and after a while the points evaluated will be beyond the tolerance limit. Once, the tolerance limit is nearly reached, the points need to refined so as to bring it back within the tolerance limit.

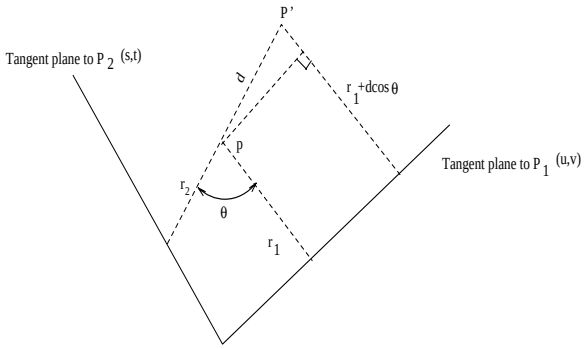

Figure 5. Point refinement

Referring to figure 5, we have

$$r_1 = |\mathbf{p} - \mathbf{p_1}|, \text{ and } r_2 = |\mathbf{p} - \mathbf{p_2}|.$$

$$\text{If } |r_1 - r_2| > \epsilon \text{ (tolerance)},$$

we refine point $\mathbf{p}$ to $\mathbf{p'}$ as follows:

$$r_1 + d\cos(\theta) = r_2 + d.$$

$$(r_1 - r_2) = d(1 - \cos(\theta)).$$

$$(r_1 - r_2) = d(1 - \hat{\mathbf{n}}_\mathbf{1} \cdot \hat{\mathbf{n}}_\mathbf{2})$$

$$d = \frac{r_1 - r_2}{1 - \hat{\mathbf{n}}_\mathbf{1} \cdot \hat{\mathbf{n}}_\mathbf{2}}$$

$$\mathbf{p}' = \mathbf{p} - d\hat{\mathbf{n}}_\mathbf{1}.$$

At any point $\mathbf{p}$ evaluated on the spine curve, we project the point to the base surfaces and if the difference of distances r_1 and r_2 is beyond tolerance limit, we refine the point by the procedure explained above.

After evaluation of points on spine curve the algorithm of spine generation needs to be terminated at a certain point. We know that the blend surface will follow the edge. Hence, for a closed edge curve the spine curve has to be closed, and for the open case it has to be open curve. If the edge is closed then from the starting point on the spine curve we calculate distance of the next evaluated points on the spine curve, and when the distance decreases to a minimum distance tolerance we terminate the algorithm. For an open curve, the algorithm is terminated when the foot prints of point $\mathbf{p}$ on the spine curve lies outside the domain of the respective base surfaces.

Construction of blend surface

While blending free-form parametric surfaces, it is difficult to get the exact representation of the blend surface. In this work, we use numerical algorithms to find the spine curve of the blend, and hence we end up with having discrete set of data points for spine and trim curves. These data points are required to be fitted in smooth curves. However, if the set of three dimensional points are fitted to a smooth curve it may not lie exactly on the surface. Hence first a curve in parametric space is fitted, then corresponding to that parametric curve which may be a non-isoparametric curve, a curve in real space is extracted from the surface. This curve will be more accurate, but it's degree may be too high. Another approach to construct the blend surface is to fit a curve in real space with a very low degree (say piecewise-cubic) and the surface is modified locally to match the trimming curve with some tangent constraint to ensure smoothness across the matching boundary(see Hu and Sun (Hu and Sun, 1997)). From either of the above procedure, we can get continuous trimline curves $\mathbf{a}(\lambda), \mathbf{b}(\lambda)$ and the spine curve $\mathbf{c}(\lambda)$.

Farouki and Sverrisson (Farouki and Sverrisson, 1996) have used some non-standard weight functions to get a rational blend surface. The blend surface can be expressed

as

$$\mathbf{S}(\lambda,\mu) = \frac{\hat{\mathbf{a}}(\lambda)(1-\mu)^2 + \hat{\mathbf{s}}(\lambda)2(1-\mu)\mu + \hat{\mathbf{b}}(\lambda)\mu^2}{w_a(\lambda)(1-\mu)^2 + w_s(\lambda)2(1-\mu)\mu + w_b(\lambda)\mu^2} \quad (5)$$

where we take the weight functions as

$$w_a(\lambda) = w_s(\lambda) = (\mathbf{a}(\lambda) - \mathbf{c}(\lambda)) \cdot (\mathbf{b}(\lambda) - \mathbf{c}(\lambda)) + r^2,$$

$$w_b(\lambda) = 2d^2.$$

and

$$\hat{\mathbf{a}}(\lambda) = [(\mathbf{a}(\lambda) - \mathbf{c}(\lambda)) \cdot (\mathbf{b}(\lambda) - \mathbf{c}(\lambda)) + r^2]\mathbf{a}(\lambda),$$

$$\hat{\mathbf{s}}(\lambda) = r^2(\mathbf{a}(\lambda)+\mathbf{b}(\lambda))+[(\mathbf{a}(\lambda)-\mathbf{c}(\lambda))\cdot(\mathbf{b}(\lambda)-\mathbf{c}(\lambda))-r^2]\mathbf{c}(\lambda),$$

$$\hat{\mathbf{b}}(\lambda) = 2d^2\mathbf{b}(\lambda).$$

and, the parameters'r' and 'd' are as defined in (Farouki and Sverrisson, 1996).

From the B-spline control points of $\hat{\mathbf{a}}(\lambda), \hat{\mathbf{s}}(\lambda)$ and $\hat{\mathbf{b}}(\lambda)$ we can find the control points of the blend surface $\mathbf{S}(\lambda,\mu)$. In our algorithm the above described method has been used for blend surface construction. It may be noted that other techniques such as lofting can also be used.

Next we describe, the implementation details of our algorithm for constant and variable radius blending.

Constant radius blending

The spine generation in case of constant radius blend is straight forward. First starting from any point on edge we get a point on spine curve using the algorithm *Spine*. Then we start moving along the spine curve by traversing along the tangent direction of the spine curve. The tangent direction of the spine curve in case of constant radius blend is the cross product of the two normals dropped from the point on the spine curve. We trace the spine curve using steps along the arc (of the spine curve) which will generate a relatively uniform distribution of point data along the spine and hence along the trimline curves. Hence the next point on the spine curve is obtained from the following relation

$$\mathbf{p}^0_{k+1} = \mathbf{p}_k + \delta\lambda \frac{\mathbf{n_1} \times \mathbf{n_2}}{|\mathbf{n_1} \times \mathbf{n_2}|}. \quad (6)$$

where in the vicinity of point $\mathbf{p}_k$, the step size $\delta\lambda$ is controlled by

$$(\mathbf{p} - \mathbf{p}_k) \cdot \frac{\mathbf{n_1} \times \mathbf{n_2}}{|\mathbf{n_1} \times \mathbf{n_2}|} - \delta\lambda < \epsilon, \quad (7)$$

with $\mathbf{p}$ being a point in the neighborhood of the point $\mathbf{p}_k$ and ϵ is distance tolerance.

The point $\mathbf{p}^0_{k+1}$ calculated may not exactly lie on the Voronoi surface i.e. distance from both the base surfaces may not be same due to complexity of the surface geometries. Hence we need to refine the point to ensure that the point lies on the Voronoi surface. We check the distance of the point from base surfaces to see if it matches with the radius of blend. If it does not match then again *Spine* algorithm is used to get the next point on the spine curve. Thus the complete spine curve can be traced out.

Variable radius blending

In the case of variable radius blending, the modeler requires more intervention from the users. Defining a suitable radius function along the spine curve for a particular case requires experience from the designers, since the geometry along the intersection of two base surfaces can be complex. Chuang and Hwang(Chuang and Hwang, 1997) has presented several constraints from which the variable radius function can be automatically generated. All these constraints can be generalized as

$$r(\theta) = \frac{c}{f(\theta)}, \quad (8)$$

where c is a user defined constant and $f(\theta)$ is a non-negative function of $\theta, 0 < \theta < \pi$. The function $f(\theta)$ controls variation of radius with respect to angle θ. An appropriate $f(\theta)$ can be chosen according to particular design need.

Using any of these constraints, we have the radius as a function of the angle θ between the two normals at any point on the spine curve. Once we are on the Voronoi surface, we can use the algorithm *Spine* to find a point on the spine curve with the given radius constraint. Since at the very first instance we don't know the angle θ or the radius of blending, we can start by using the angle between the normals at any point on the edge. From this angle we obtain the first approximation of radius corresponding to the initial points of assignment. Then we choose a step size dr, which could be half or one third of the approximate radius. Using the algorithm *Spine* we get a next point and at that

point check if any point refinement is needed as the step size dr may be large enough to deviate from the Voronoi surface of the two base surfaces. Now again we find the radius of blend using the two normals dropped from the newly evaluated point, and again calculate the step size dr as dr equal to the difference of radius of blend evaluated at this point and the distance of this point from both the base surfaces. This iteration is continued till we converge to radius satisfying the constraint.

The above procedure will give the very first point on the spine curve. From this point onwards we move along the tangential plane on the Voronoi surface taking the best direction to move as normal to the plane containing both the normals. We pick any small step size to move further, but to have a good convergence we choose a step size based on the curvature of the base surfaces at the foot points (i.e. point in assignments) of the current point on the spine curve.

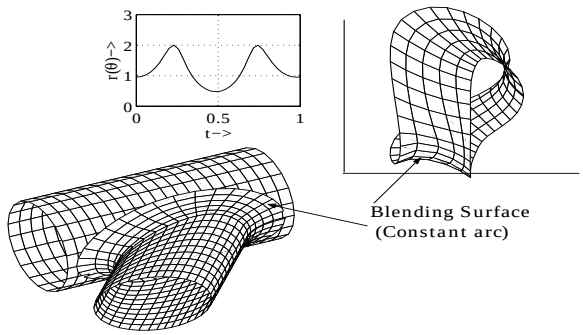

Figure 9. Blending of two intersecting cylinders

RESULTS

Tests were performed on blending of free-form surfaces as shown in figures below, with our proposed algorithm. It is found that even for fairly complicated surfaces, the algorithm works well and is able to evaluate the spine curves within few seconds($< 3s$), on a Linux-PC (Pentium III processor). The complete blending task for two bilinear surfaces meeting at two edges, was performed in about 1.24secs. For blending of two free-form surfaces, the time taken is about 2.8secs to compute the spine curve, and 4.37secs for the complete blending operation.

Variable radius blending, requires an extra input from user, i.e. the radius function along the spine curve. In our tests we have used some of the constraints discussed earlier such as constant arc length. Figure 8 shows a typical example showing blending of a cylinder intersecting a free-form surface along some non-isoparametric curve. In the inset of the figures we have shown the variation of radius along the spine curve parameter t. Figure (8a) shows constant radius blending with $r = 0.5$ and figure(8b) shows the case of variable radius with constraint of constant arc length. Another good candidate for variable radius blending is the blending of two cylinders intersecting at an angle. Figure 9 shows an example where two cylinders, intersecting at an angle $\theta = 60^o$, has been blended with constant arc length of 1.0 ($i.e., r(\theta) = \theta^{-1}$). In the figure we have also shown the blend surface along with variation of radius along the spine curve.

We have also dealt with blending cases having singularities and one of the example of such a problem is when two cylinders of same radii intersect at an angle. Figure 10 shows an example with two cylinders of same radii ($r = 2.0$) being blended with blend radius $r = 1.0$. Rendered image of the blend has been shown to visualize the continuity between the two blend surfaces.

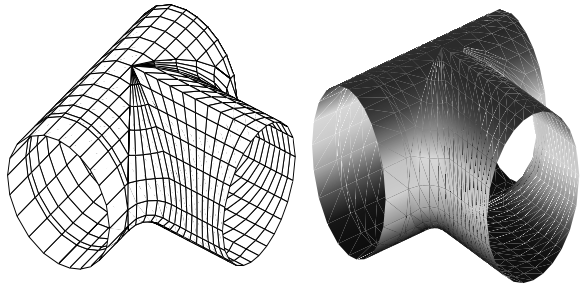

Figure 10. Radius blending for singular case

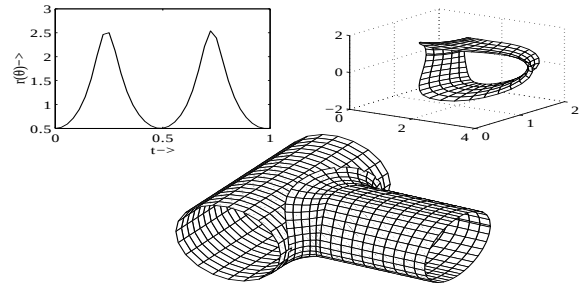

Figure 11. Blending of cylinders with constant range distance as constraint

Test of continuity between surfaces has been done by checking the visual appearance as in the figure 10 and by plotting normals at the trimlines. The figure 11, shows variable radius blending between two cylinders intersecting

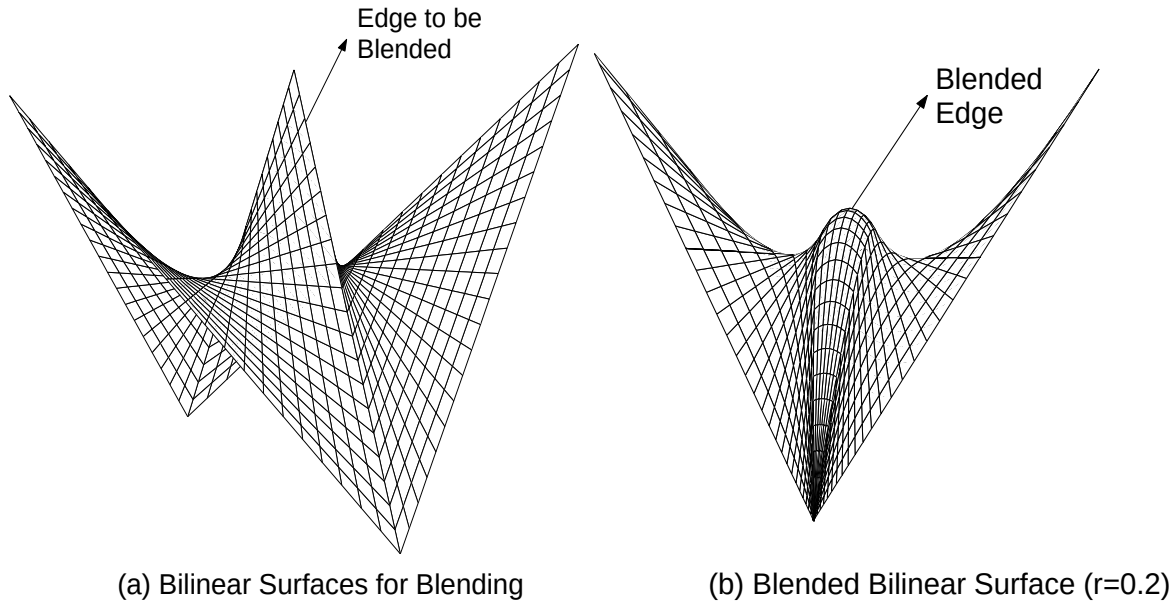

Figure 6. Blending of two bilinear surfaces meeting at two edges

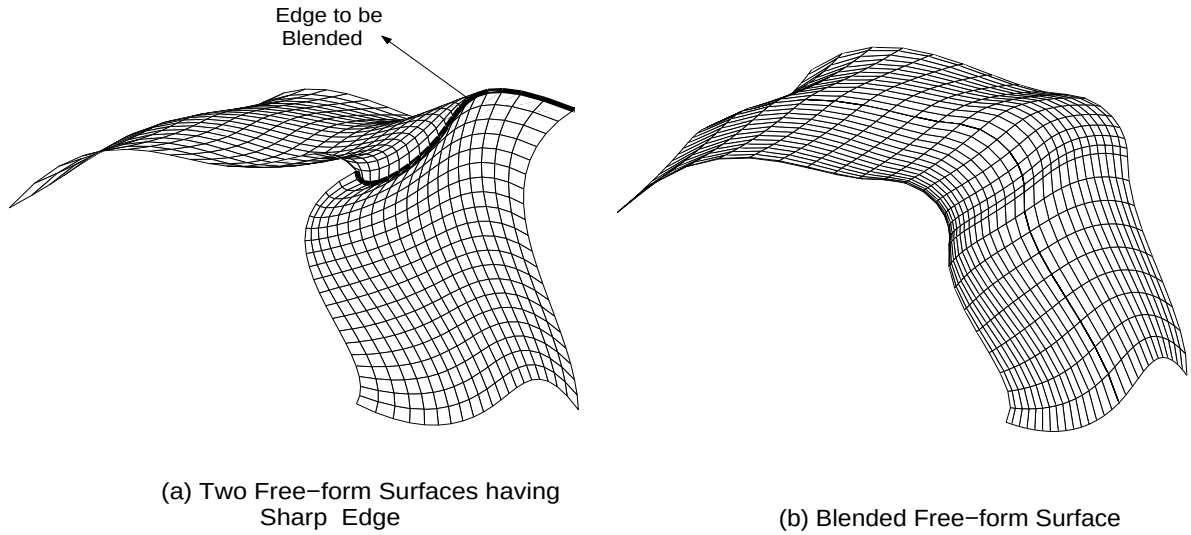

Figure 7. Blending of two free-form surfaces

perpendicularly. The constant range constraint *i.e.*, $r(\theta) = 0.5/\tan(\theta/2)$ has been used to specify the radius function as shown in figure 11. In figure 12 continuity checks along the trimlines have been shown. Figures (12a and 12b) show the plot of the angle between the normals of blend and base surface along the trimline-I and the vector plot of normals in 3-D respectively. From the figure it is found that an accuracy up to 10^{-16} has been achieved for the case of figure 11. Figures (12c and 12d) show similar plots at the trimline on the second surface.

As mentioned in the previous section, we can obtain the curve of intersection (SSI)of any two surfaces by performing a constant radius blending with $r = 0$. We tested this approach on two intersecting cylinders of radius 2.0 and 1.5 respectively. As can be seen from figure 13, our algorithms gives us the SSI curve. In the inset of figure 13, we show

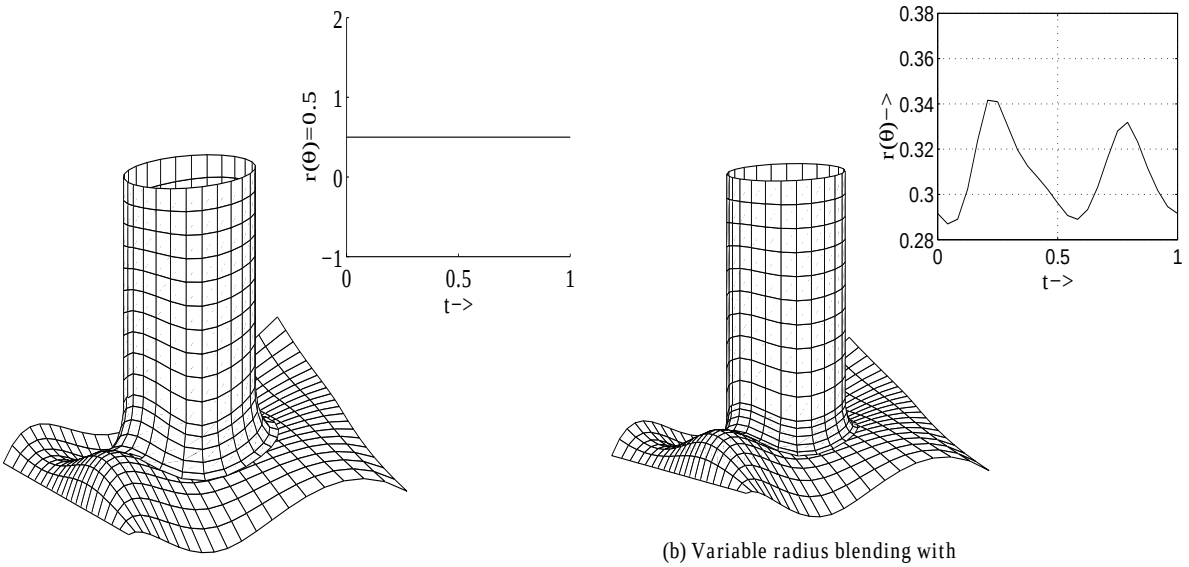

Figure 8. Blending of cylinder with a free-form surface

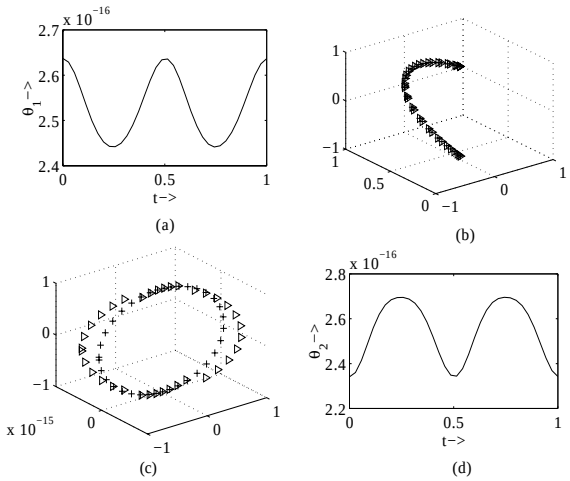

Figure 12. Continuity, (a) Angle between normals at trimline-I, (b) Vector plot of unit normals at trimline-I, (c) Vector plot of unit normals along the trimline-II, (d) Angle between the normals at trimline-II.

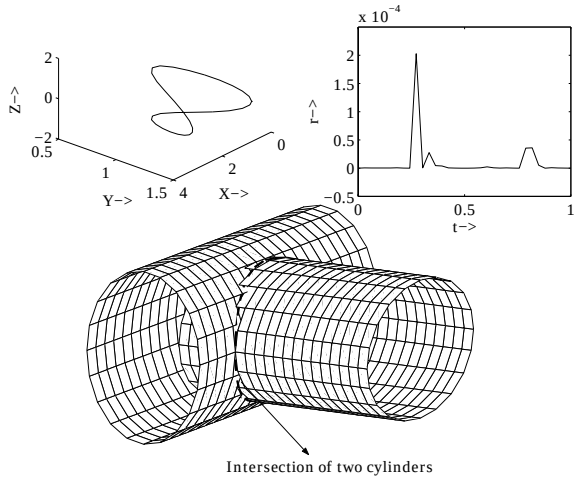

Figure 13. (a) 3D plot of the curve of intersection, (b) Variation of r from zero, (c) Intersection of two unequal cylinders

that the maximum variation of r from zero is about 2×10^{-4}.

CONCLUSION

A geometric approach for circular blending of G^2 continuous free-form surfaces meeting at an edge have been presented in this paper. The concept of Voronoi surface is fully exploited to develop an efficient algorithm. Compared to the other existing techniques, the proposed scheme can be faster since we do not have to explicitly compute surface-surface intersection. It is more accurate, since the point refinement and spine tracing techniques never let the point to deviate too much from the Voronoi surface of the two base surfaces. A byproduct of our approach is that we can also obtain the surface-surface intersection curve by performing a constant radius blend of zero radius.

REFERENCES

Bajaj, C. L., Hoffman, C. M., Lynch, R. E. and Hopcroft, J. E. H., "Tracing surface intersections", *Computer Aided Geometric Design*, Vol. 4, pp. 3-16, 1987.

Barnhill, R. E. and Kersey, S. N., "A marching method for parametric surface/surface intersection" *Computer Aided Geometric Design*, Vol. 7, pp. 257-280, 1989.

Barnhill, R. E., Farin, G. E. and Chen, Q. "Constant radius blending of parametric surfaces" in Farin, G. E., Hagen, H. and Noltemeir, H.(Eds), *Geometric Modeling* Springer 1993.

Chandru, V. Dutta, D. and Hoffman, C. M., "Variable radius blends using Dupin cyclides", *Proc. IFIP Conf. Geometric Modeling for Product Engineering*, North-Holland(1990), pp. 39-57.

Chang, L. C., Bein , W. W. and Angel, E., "Surface interaction using parallelism", *Computer Aided Geometric Design*, Vol. 11(1), pp. 39-69, 1994.

Choi, B. K. and Ju S. Y., "Constant radius blending in surface modelling", *Computer Aided Design*, Vol. 21, no. 4, pp. 213-220, 1989.

Chuang, J. H., and Hwang, W. C. "Variable-radius blending by constrained spine generation", *Visual Computer*, Vol. 13, pp. 316-329, 1997.

do Carmo, M. P. *Differential Geometry of Curves and Surfaces*, Prentice-Hall, 1976.

Farouki, R. T., "The approximation of non-degenerate offset surfaces", *Computer Aided Geometric Design*, Vol. 3, pp. 15-43, 1986.

Farouki, R. T., Sverrisson, R. "Approximation of rolling-ball blends for free-form parametric surfaces", *Computer Aided Design*, Vol. 28, no. 11, pp. 871-878, 1996.

Hartmann, E. "Numerical implicitization for intersection and G^n-continuous blending of surfaces", *Computer Aided Geometric Design*, Vol. 15, pp. 377-397, 1998.

Hermann, T., G. Lukács and T. Várady, Techniques for variable radius rolling ball blends, in *M. Dæhlen, T. Lyche and L. L. Schumaker (eds.), Mathematical Methods for Curves and Surfaces*, pp. 225-236, 1995.

Heo, H. S., Kim, M. S., Elber, G. "The intersection of two ruled surfaces", *Computer Aided Design*, Vol. 31, pp. 33-50, 1999.

Hu, Y. P. and Sun, T. C. "Moving a B-spline surfaces to a curve - a trimmed surface matching algorithm", *Computer Aided Design*, Vol. 29, no. 6, pp. 449-455, 1997.

Karim Abdel-malek and Harn-Jou Yeh, "On the determination of starting points for parametric surfaces intersections", *Computer Aided Design*, Vol. 29, no.1, pp. 21-35, 1997.

Klass, R. and Kuhn, B., "Fillet and surface intersections defined by rolling balls", *Computer Aided Geometric Design*, Vol. 9, pp. 185-193, 1992.

Lukacs, G., "The generalized inverse matrix and surface-surface intersection problem" in Strasser, W. and Seidel, H-P.(Eds), *Theory and Practice of Geometric Modelling*, Springer, pp. 167-186, 1989.

Lukacs, G. "Differential geometry of G^1 variable radius rolling ball blend surfaces", *Computer Aided Geometric Design*, Vol. 15, pp. 585-613, 1998.

Pegna, J., "Variable sweep geometric modeling", Ph.D. dissertation, *Stanford University*, 1987.

Pegna, J. and Wilde, D. J., "Spherical and circular blending of functional surfaces", *Transactions of the ASME, Journal of Offshore Mechanics and Arctic Engineering*, Vol. 112, pp. 134-142, 1990.

Rossignac, J. R. and Requicha, A. A. G. , "Constant radius blending in solid modelling", *Comput. Mech. Eng.*, Vol. 3, pp. 65-73, 1984.

Varady, T. Vida, J. and Martin, R. R. "Parametric blending in a boundary representation solid modeler", in Handscomb, D. C. (Ed.), *The Mathematics of Surfaces III* Oxford University Press, pp. 171-197, 1989.

Vida, J., Martin, R. R., and Varady, T. "A survey of blending methods that use parametric surfaces", *Computer Aided Design*, Vol. 26, no. 5, pp. 341-365, 1994.