Controlled Blending of Procedural Implicit Surfaces

Zoran Kačić-Alesić
Brian Wyvill

Department of Computer Science
The University of Calgary
Calgary, Alberta, Canada T2N 1N4

Abstract

Implicit surfaces are becoming increasingly popular for modeling geometric objects. Procedurally defined implicit surfaces, in particular surfaces built around skeletons, provide an intuitive representation for many natural objects, and objects commonly used in geometric modeling. This paper presents a number of techniques that provide good control over the shape of the implicit surface and the way different surfaces blend together. Some extensions to these techniques provide a simple and convenient representation for "soft" *surfaces of revolution, randomly deformed surfaces, and other interesting shapes that would otherwise be difficult to model.*

Introduction

Parametric curves and surfaces have traditionally been favorite modeling primitives in computer aided geometric design and graphics. A rich literature exists on techniques for free-form modeling with these primitives. In recent years, however, implicit surfaces have become increasingly important. The major advantage of implicit surfaces that makes them very attractive for modeling and animation is the blending property. Separate implicit surfaces, unlike parametric surfaces, can blend together smoothly and form very complex, non-intersecting shapes. Implicit surfaces also conveniently define volumes. It is easy to determine whether a point is above, below, or on the surface, simply by evaluating the implicit function at the given point. Computing the surface normal is also an easier task than it is for parametric surfaces. However, interactive modeling techniques available in the past did not provide sufficient control of the shape of implicit surfaces.

In this paper we present some new techniques for providing the user with control over the way in which procedural implicit surfaces blend together. A number of new blending functions that can be customized interactively to the need of the user is introduced.

In general, the shape of an implicit surface is not very intuitive from its implicit formulation. Computer graphics techniques provide a very useful aid to the visualization of implicit surfaces. However, the effects of changing coefficients in the implicit formulation of the surface are also not intuitive, and the results are usually unpredictable and very difficult to control. The control flexibility necessary for a design environment is, thus, hardly achievable by direct manipulation of the coefficient in the implicit formulation of the surface. The goal of our research is to develop a modeling technique which will enable us to find an implicit representation of an arbitrary shape.

Skeletal Implicit Surfaces

Implicit surfaces can be built around *skeletons* [BW90]. In general, any three dimensional object can be a part of the skeleton, as long as it is possible to determine the distance from a given point in space to the skeleton. Skeletons are useful for several reasons:

- skeletons provide intuitive representation for many natural objects,
- skeletons themselves are simple and easy to manipulate and display,
- complex shapes can be modeled with few elements.

In this paper we limit our discussion to skeletons that consist of elements such as points, lines, polygons, circles, spline curves and spline patches. Each skeletal element is surrounded with an imaginary force field $\mathbf{F(r)}$, with the intensity of the field being the highest at the skeleton, and decreasing with distance $\mathbf{r}$ from the skeleton. The function $\mathbf{F(r)}$ that relates the field value (intensity) to distance from the skeleton has an impact on the shape of the surface,

and more importantly, determines how separate surfaces blend together. For that reason we call such functions **blending functions**.

The surface is defined by the set of points in space for which the intensity of the field has some chosen constant value (thus the name *iso-surface*). We call this value a **contour value**. The surface so defined is an offset surface of the skeleton. Fields from the individual elements of the skeleton add together (or subtract, in the case of negative force fields), what in turn has the blending effect on the surfaces defined by separate elements of the skeleton:

$$F_{total}(P) = \sum_{i=1}^{n} c_i F_i(r_i),$$

$$r_i = f_i(P) = dist(P, Q_i),$$

where

$F_{total}(P)$ = the intensity of the field at the point P,

c_i = a scalar value (scale factor) that represents the field magnitude due to the ith skeleton (it can be negative),

F_i = the blending function for the ith skeleton ,

r_i = distance from the point P to the skeleton i,

$P(x_P, y_P, z_P)$ = point in $\Re^3$ at which the field function is evaluated,

$Q(x_Q, y_Q, z_Q)$ = point in $\Re^3$ - the nearest point on the skeleton i to the point P.

The evaluation of the field function, thus, has two steps. The first step involves finding the nearest point on the skeleton to the given query point, and calculating the distance between them. This procedure depends on the geometry of the skeleton, and can be very simple (trivial in the case of a point skeleton), or quite complex in the case of spline curves and patches, when an iterative or numerical method has to be used. The second step involves evaluation of the blending function.

The shape of a skeletally defined implicit surface is determined by:

- the geometry of the skeleton,
- a blending function to weight the contribution of individual skeletal elements,
- modifications to the blending function that result from geometry, orientation, size, or other properties of the skeleton,
- random deformations produced by perturbation of the blending function with some three dimensional noise function.

The surface is controlled by applying local or global transformations, such as scaling, translation, and rotation, to the elements of the skeleton, and by changing the blending functions.

Many skeletally defined surfaces can be expressed analytically, but in order to keep the representation intuitive and simple these implicit functions are usually treated as **procedural**, *i.e.* defined by procedures that return a scalar field value for any point in 3-D space. Procedural implicit functions can be used to model surfaces for which analytical representation is difficult or impossible to formulate [BW90].

Blending Functions

In general, any function can be used for blending. Blinn used superimposed exponential density distribution functions to model atoms and molecules [Bli82]. A variation of the above surfaces, called *soft objects*, uses a blending function which is a cubic polynomial [WMW86b, BW90]. Soft object have been successfully used for modeling of objects and figures commonly found in nature, and for animation of objects that change shape in time [WMW86a]. Blending surfaces based on low degree polynomial functions [MS85] and super-elliptic blends [RO87] are used in solid modeling.

From this earlier work and our own experience there are several important properties of blending functions:

- each implicit function should have a limited radius of influence **R**, *i.e.* the value of the blending function should be zero beyond some distance. This enables considerable computational savings because implicit functions of distant skeletons need not be evaluated.
- blending functions have a maximum at the skeleton (zero distance), and they drop *smoothly* and *monotonously* to zero at the radius of influence. Without loss of generality, we assume that the maximum value is 1.0, and that the contour value is 0.5.
- the first, and if possible second derivative of the blending function should be continuous, and they should vanish at zero and at the radius of influence. This will ensure that surfaces blend smoothly and that there are no sharp discontinuities in the curvature of the blended surface.

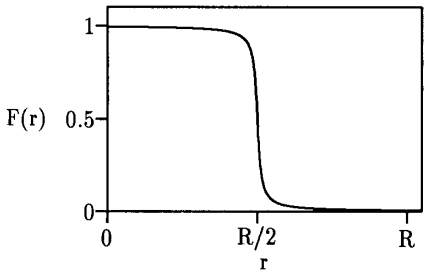

Figure 1: A "*hard*" blending function

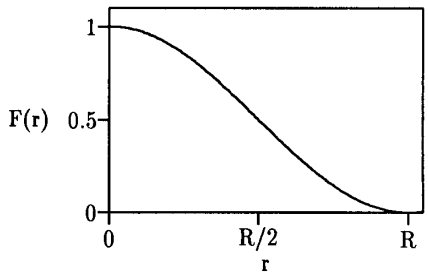

Figure 2: A "*soft*" blending function

All the blending functions presented in this paper have at least first two derivatives continuous in the interval [0, R]. This fact alone, unfortunately, does not guarantee that the blended surface will be smooth. In this paper the blended surface is defined as order k continuous if the first k derivatives of the blending function are continuous in the interval [0, R], **and** they vanish at the end points:

$$\frac{d^i}{dr^i}F(0) = 0, \frac{d^i}{dr^i}F(R) = 0, i = 0, ..., k.$$

This definition is not necessarily equivalent to that in [War89].

The shape of the blending function affects the "amount of blending." Blending functions that drop to zero shortly after they fall bellow the contour value will result in very little blending (figure 1). On the other hand, functions that drop slowly to zero will result in "soft" blending (figure 2). Too much blending is often undesirable. For example, when modeling a human body it is not desirable that the arms blend with the body anywhere but at the shoulders. As a result of using soft blending functions the blended surface is bulgy (figure 3). "Hard" blending functions can reduce the bulge (figure 4).

Low degree polynomial functions are most commonly used for blending [WMW86b, BW90]. The lowest degree polynomial that satisfies the first order continuity requirements is a cubic:

$$F_{cub}(r) = a\frac{r^3}{R^3} + b\frac{r^2}{R^2} + c\frac{r}{R} + d$$

where

r = distance from the skeleton

R = radius of influence.

A cubic in r^2 (figure 5) is more commonly used [WMW86b, BW90]:

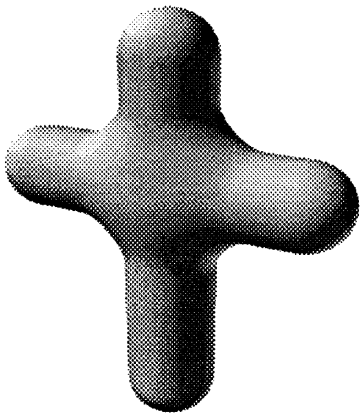

Figure 3: A soft blend of cylinders using the blending function from figure 2. Note the bulge around the intersection.

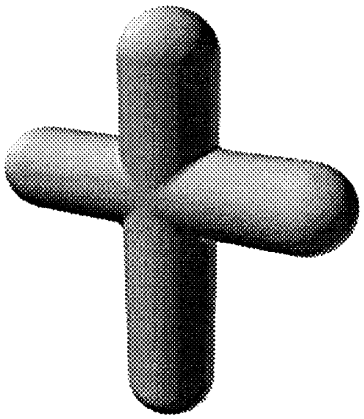

Figure 4: A hard blend of cylinders using the blending function from figure 1. The bulge is gone.

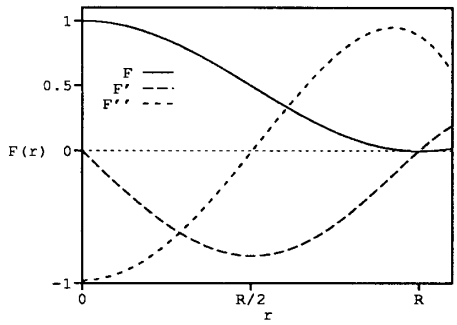

Figure 5: Cubic blending function and it's derivatives. ($F' = 0.5\frac{dF}{dr}, F'' = 0.2\frac{d^2F}{dr^2}$)

$$F_{cub2}(r) = a\frac{r^6}{R^6} + b\frac{r^4}{R^4} + c\frac{r^2}{R^2} + d$$

Since the first derivative

$$\frac{dF_{cub2}(r)}{dr} = 6a\frac{r^5}{R^6} + 4b\frac{r^3}{R^4} + 2c\frac{r}{R^2}$$

is zero at $r = 0$ regardless of the choice of coefficients, additional constraint can be put on the curve $F_{cub2}(r)$. In [WMW86b] an additional point on the curve is specified such that $F_{cub2}(0.5) = 0.5$. This results in a curve very similar to the ordinary cubic in r (figure 5). Some sort of control over the shape of the blending function can be achieved by moving this point. However, since the curve is heavily constrained at the end points, very undesirable minima and maxima with large values occur in the interval [0, R], if the point is moved considerably in any direction. Very little variation from the shape in figure 5 is achievable this way.

The blending function can be defined as an interpolating curve through a number of control points. Any numerical analysis book readily offers a number of interpolating schemes. Cubic spline interpolation is probably the most popular interpolating scheme, and it produces excellent results for a limited range of curves in our example. However, interpolating curves do not posses the convex hull property, and since they are constrained to pass through all the control points, and in our case to have zero derivatives at the end points, problems similar to those experienced with cubic curves soon arise.

Nonparametric curves in Bernstein-Bézier form [Far88] do not, in general, pass through the control points (except the end points), have a convex hull property and thus provide a very good solution to our problem:

$$F(r) = \sum_{j=0}^{n} b_j B_j^n(r/R)$$

where

$$B_j^n(r/R) = \binom{n}{j}(r/R)^j(1 - r/R)^{n-j},$$

$b_j \in [0,1]$ = the ordinate of j-th control point (a scalar value).

The k-th derivative ($k \leq n$) is also a nonparametric curve in Bernstein-Bézier form:

$$\frac{d^k}{dr^k}F(r) = \frac{n!}{(n-k)!}\sum_{j=0}^{n-k}\Delta^k b_j B_j^{n-k}(r/R),$$

$$\Delta^1 b_j = b_{j+1} - b_j,$$

$$\Delta^k b_j = \Delta^{k-1}b_{j+1} - \Delta^{k-1}b_j$$

$$= \sum_{i=0}^{k}\binom{k}{i}(-1)^{k-i}b_{i+j}.$$

The n + 1 control points $(jR/n, b_j); j = 0, ..., n$ are equally spaced, with increasing abscissae, along the r axis in the interval [0, R]. The curve is thus guaranteed to be a functional curve. Setting $b_0, ..., b_k = 1$, and $b_{n-k}, ..., b_n = 0$ results in the first k derivatives being equal to zero at the end points (r=0, and r=R). For k = 1 at least 4 control points are necessary to assure that the first derivative will vanish at the end points, and the resulting curve is a cubic. For k = 2 we need at least 6 control points to have both first and second derivative vanish at the end points, at the cost of having to evaluate a degree five polynomial (figure 6). Our examples (figures 7, 8, 9) show that a blended surface produced by a second order continuous blending function appears considerably smoother than the surface produced by a blending function which only has first order continuity. The difference is particularly noticeable on surfaces polygonized at a low level of subdivision (i.e. when fewer polygons are used to approximate the surface).

One additional property of blending functions is often required. If the point with the contour value on the graph of the blending function is moved in the r direction, the size of the objects defined by individual skeletons is changed. It is often undesirable that the choice of the blending function should affect the size or shape of a single object. It should only affect the blending of two or more separate objects. Without loss of generality we will additionally constrain our blending functions, so that every function

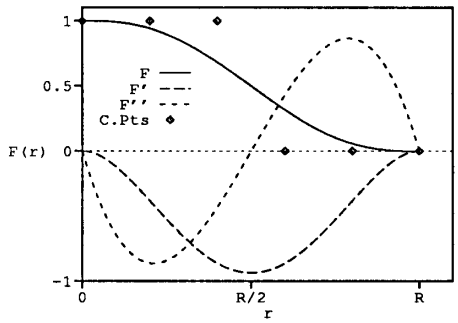

Figure 6: Degree five Bernstein-Bézier blending function, it's control points and derivatives. ($F' = 0.5\frac{dF}{dr}, F'' = 0.15\frac{d^2F}{dr^2}$) Note that both derivatives have zero values at the end points.

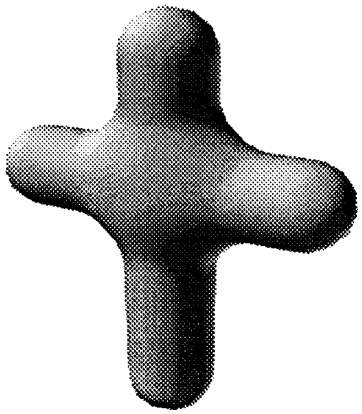

Figure 8: A blend of two cylinders using a first order continuous Bernstein-Bézier blending function with four control points.

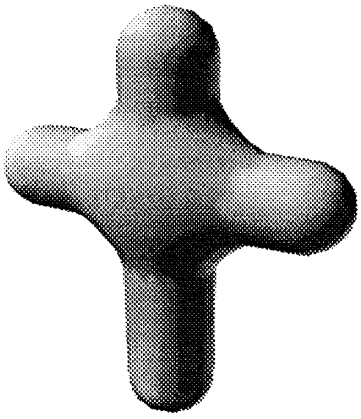

Figure 7: A blend of two cylinders using an order zero continuous Bernstein-Bézier blending function with four control points.

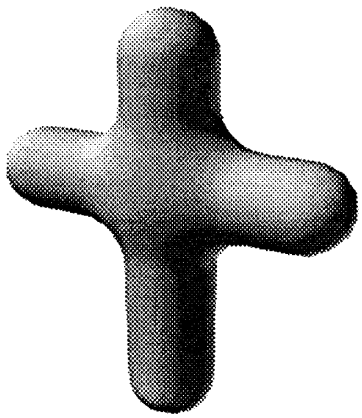

Figure 9: A blend of two cylinders using a second order continuous Bernstein-Bézier blending function with six control points (figure 6).

has the contour value exactly at one half the radius of influence

$$F(R/2) = \textit{contour value} = 0.5.$$

In the case of a single skeleton any blending function should produce the same surface, which is an offset by R/2 from the skeleton. As a result of this additional requirement most of the blending functions will be symmetrical in respect to the point (R/2, 0.5).

The control of the blending is achieved by changing the slope of the function at R/2, while maintaining the constraints at the end points, smoothness, and monotony of the function. While all the blending functions described so far can easily be constrained to pass through the point (R/2, 0.5), it is is difficult or impossible to considerably change the slope of the function at that point. A cubic function is completely defined by the end point constraints, and provides no additional flexibility. Higher order polynomials are difficult to control. Piecewise cubic interpolating splines, although not a bad choice, are not particularly suited for shape design. They suffer from the lack of the convex hull property, sometimes are difficult to control, and require many control points to get steep blending functions. Nonparametric Bézier functions have the convex hull property, and are an excellent choice for very smooth blending functions. However, the fact that the function does not pass through the control points, and that the control points are equidistant in the r direction, makes it very difficult to design steep functions with any reasonable number of control points.

A very good solution for steep blending functions is based on the arctangent function (figures 1 and 10):

$$F_{atan}(r) = 0.5 - \frac{1}{\pi}\arctan(c_1(r/R - c_2))$$

where

- c_1 = a number that controls the steepness of the function (10 – 1000). The slope of the function at the point of inflection (which also has the contour value) is $-c_1/\pi$. The function will approximate the step function as the value of c_1 increases;
- c_2 = value of r/R for which F_{atan} has the contour value (0.5).

This function is used to produce the blended surface in figure 4 ($c_1 = 100$). The function does not pass through points (0,1) and (R, 0), and it's derivatives do not vanish at the end points. However, the discrepancies are small (figure 10), and may safely be

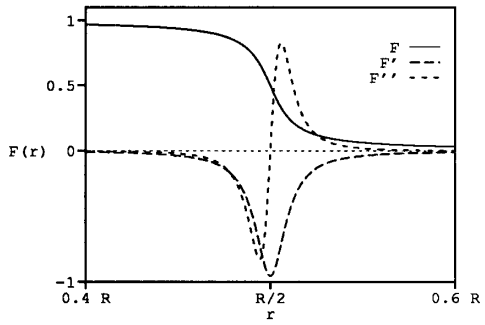

Figure 10: Arctangent blending function and it's derivatives. ($c_1 = 100, F' = 0.03\frac{dF}{dr}, F'' = 0.04\frac{d^2F}{dr^2}$)

ignored for steep functions. For not so steep functions ($c_1 < 50$) the following modification will assure that $F_{atan1,0}(0) = 1$, and $F_{atan1,0}(R) = 0$:

$$F_{atan0}(r) = F_{atan}(r) - F_{atan}(R),$$

$$F_{atan1,0}(r) = F_{atan0}(r)/F_{atan0}(0).$$

All the functions described so far in this paper are explicit, i.e. they are expressed in terms of r. It is well known that parametric curves are much better suited for shape design. We have found low degree parametric curves useful for design of our blending functions:

$$F = f_1(t),$$

$$r = f_2(t), t \in \Re.$$

For any given r it is necessary to solve f_2 for t, in order to calculate F. For quadratic and cubic parametric spline curves, a closed form solution exists. Care has to be taken to assure that the spline curve defined by the control points indeed is a functional curve, i.e. there must not be points with vertical slope or with multiple values of F for a given r. If that is provided, a single real solution to f_2 exists for any given r. The method is prone to floating point imprecisions when coefficients, particularly the coefficient with the highest power of t, become very small. That is usually the case when the control points are almost collinear, or when they are equally spaced along the r axis. Special care has to be taken to detect such situations. Figure 11 shows the useful range of a single segment cubic Bézier spline. The Cardano's formula used to solve the cubic equation in t, is numerically stable in that range. The soft blending function in figure11 also produces the blended surface in figure 3. The hard blending function produces a blended surface that is slightly softer than the surface in figure 4.

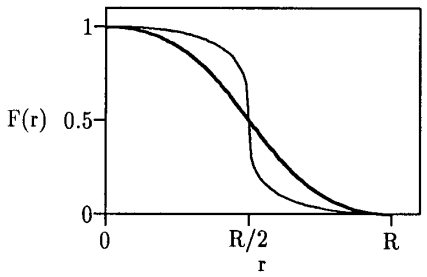

Figure 11: Single segment cubic Bézier blending functions (parametric). The four control points are (0, 1), (c, 1), (R-c, 0), (1, 0), $1/3 < c < 1$. The soft curve is identical to the cubic curve F_{cub}.

More segments of a piecewise quadratic or cubic spline curve can be used if additional flexibility is desired, at an increased cost of computation.

Selective Blending

As previously mentioned, surface blending is not always a desirable effect. There are two practical solutions to this problem. One is to group skeletons of objects which should not blend together [Bei90]. The group is then treated as a single skeleton, and the distance to the skeleton is the minimum distance to any of the elements of the group. Another solution is to use different blending functions for different skeletons. In the human body example, the body and arms would be associated with a very "hard" blending functions, while the shoulders would have a "soft" blending function.

Interactive Control of Blending Functions

An additional benefit of using spline curves (both nonparametric and parametric) is the ability to interactively design blending functions, simply by moving the control points. Similarly, the shape of the arctangent function is affected by interactively changing the value of c_1. A lot of information about the blending function is hidden in the shape of it's derivatives. We have found it very valuable for interactive design of blending function to display the first two derivatives together with the function itself.

Depending on the blending function used, the polygonization of a simple surface, such as that defined by two intersecting cylinders or by two spheres, is a relatively fast process. The effects of changing the blending function can thus be visualized sufficiently quickly for interactive design, although we cannot do this in real time with our current implementation.

Surface Normals

The gradient of the implicit function is the surface normal at a given point on the implicit surface. Procedural implicit surfaces may not have an analytical representation, or it may be very difficult to obtain, so the gradient must be approximated by evaluating the implicit function at three nearby points [BW90]. However, the following observation shows that for skeletally defined surfaces this is not necessary. For a single object, the surface is an offset in the direction perpendicular to the skeleton. The vector from the nearest point on the skeleton to the point on the surface is, thus, in the direction of the surface normal. Since the nearest point on the skeleton must be computed anyway in order to find the distance to the skeleton, the surface normals come for free. For a point in space that is affected by more then one skeleton, the surface normal is computed as a weighted average of the normals due to individual skeletons. The contribution of each skeleton to the surface normal is the same as the contribution to the total field, and is given by the blending function. This method also avoids numerical difficulties that very steep blending functions may pose to the gradient approximation technique. However, modifications to the blending function described in the next section, may limit, or at least complicate the use of this method.

Modifications to Blending Functions

Blending functions need not be defined in terms of distance only. Other parameters based on geometry and orientation of the skeleton can be used to modify the shape of the surface:

- position of the nearest point on the skeleton.
- orientation of the skeleton in space. Additional parameter (or parameters) need be specified for the reference orientation.
- size (deformation) of the skeleton. This is useful in animation when skeletons change size, either by being stretched or squashed, in order to simulate the preservation of volume. Stretched surfaces would, thus, appear thinner, and squashed surfaces would be thicker.

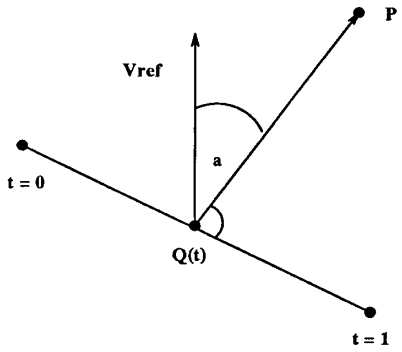

Figure 12: Parameters that can be used to modify the blending function of a line skeleton ($\alpha = a$).

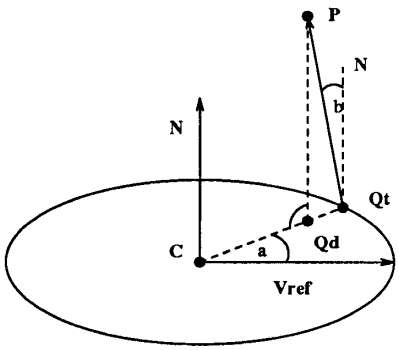

Figure 13: Parameters that can be used to modify the blending function of a circle or torus ($\alpha = a, \beta = b$).

Skeletons that have natural parametrization, such as lines, spline curves and patches, are particularly suited. Functions of the parameter(s) of the nearest point on the skeleton can be used to modify the field function.

Surfaces of Revolution and Generalized Cylinders

The offset surface of a line is a cylinder. The thickness of the cylinder is determined by the radius of influence R, and the blending function. If the radius of influence R is not treated as a constant, but rather as a function of the parameter of the nearest point on the line,

$$R = f_1(t_P),$$

$$F = f_2(r, R).$$

the result is a cylinder of varying thickness, which is actually a surface of revolution of the curve f_1 around the line.

If a cylinder is intersected with a plane perpendicular to the axis, the cross section is a circle. Noncircular cross sections can be achieved by applying yet another modification to the blending function. If an additional reference vector that is perpendicular to the line is defined, the thickness of the cylinder can be varied with a function of the angle α between the reference vector $\overline{V_{ref}}$ and the vector from the nearest point on the line to the point on the surface (figure 12):

$$R = f_1(t_P) \cdot f_3(\alpha),$$

$$F = f_2(r, R).$$

Twisted surfaces may be modeled by rotating the reference vector along the line.

Similar modifications also apply to general cylinders - offset surfaces of spline curves. However, the consistency of the reference vector along the spline is more difficult to maintain, as described in [Blo90].

Disc and Torus

A disc is an offset surface of a planar circular disc, and a torus is an offset of the circular line (figure 13) The skeleton is defined by a point C (the center of the circle), a normal to the plane of the circle, and a radius. The radius can be given by the reference vector $\overline{V_{ref}}$, which is perpendicular to the circle normal. The only difference between a disc and a torus is in the way the nearest point on the skeleton (Q_d and Q_t respectively) is calculated. The additional parameters that can be used to modify the blending function are:

- angle α between the reference vector $\overline{V_{ref}}$ and the $\overline{CQ}$ vector,
- distance between points C and Q_d (disc only),
- angle β between the circle normal $\vec{N}$ and the $\overline{Q_t P}$ vector (torus only).

The blending function is then

Random Deformations

Blending functions can be perturbed by solid noise functions, such as those described in [Per85, Pea85, Lew89]:

$$F = f_1(r_P) \cdot (1 + c_N \cdot Noise(P)).$$

Noise() is a scalar function that returns a value in the range [-1, 1] for a point in 3D space. The value

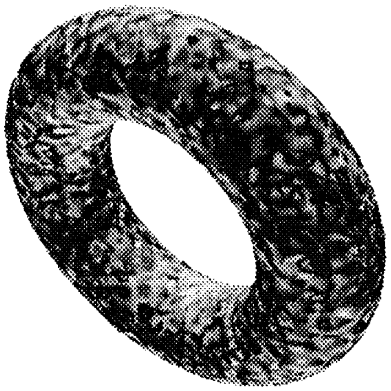

Figure 14: Bumpy Donut. The bumps are built into the surface by incorporating the solid noise function into the blending function.

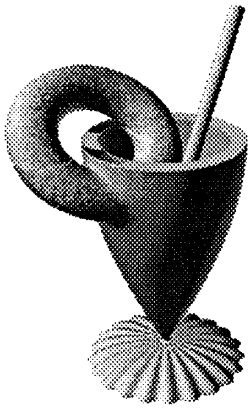

Figure 15: Cocktail á la *Doughnut de Bump.* The cup is a surface of revolution (cylinder) controlled by a sine function of the parameter. The interior is taken out by a smaller negative cylinder. The base is an offset of a circle (disc) controlled by a cosine function of the angle α and by a linear function of the distance from the center. The straw is a straight line cylinder. The cup and the straw use a very hard arctangent ($c_1 = 100$) blending function, while the base and the donut use a soft cubic function.

of the noise function is computed by smooth interpolation between the pseudorandom values assigned to the points on a 3D integer lattice [Per85]. Any blending function described in this paper can be used for such an interpolation. In order to assure that the gradient of the noise function is continuous, at least a cubic interpolation function should be used. The amplitude of the deformation thus produced is controlled by c_N, and the frequency by the size of the integer lattice. Multiple noise functions with different amplitudes and frequencies may be applied on the same surface. The result of applying such a noise function is equivalent to that produced by having a point skeleton with a random maximum intensity and a radius of influence equal to the size of the grid, at every point on the integer lattice.

Applications to Animation

Skeletally defined implicit surfaces have been used successfully in computer animation [WMW86a]. They are particularly suitable for animating objects that change shape as they move. In this earlier work models change their shape by altering the relative positions of skeletons. This shape change can be driven from a key frame approach [Wyv88], where parameters govern the kinematic relationship between skeletal elements, or using dynamic simulation [TPF89]. In this latter work a liquid like appearance was obtained using blended spherical particles.

One of the advantages of using spline curves to provide our blending functions is that the positions of the

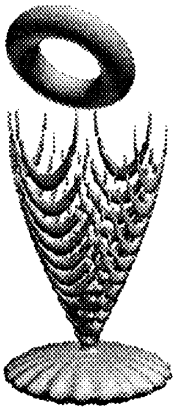

Figure 16: Bumpy Donut Trophy

Blending function	Control	Smoothness (order of continuity)	Computational Efficiency	Best used for
cubic polynomial in r^2	poor	good (0–1)	excellent	constant soft blending
interpolating cubic spline	poor (constrained) good (unconstrained)	good (0–1)	good	general unconstrained functions
nonparametric B-B curve	good	excellent (any)	poor	smooth blending
arctangent	good	good (0-2 approx.)	very good	hard blending
parametric spline	very good	good (0–1)	poor	wide range of blending

Table 1: Comparison of different blending functions.

control points can be changed over time to alter the blending from say soft to hard. Figure 11 shows suitable extremes that can be used for such an animation. The shape change may be used where a soft looking object e.g. a cloud, undergoes metamorphosis into a hard object such as the cup shown in figure 15.

The shape of the surfaces of revolution is controlled by a function of the parameter of the nearest point on the axis of rotation. Functions of other parameters can be used to modify the blending. These functions can also be changed over time by altering the positions of the control points. This leads to a very smooth alteration of shape with time and some exciting possibilities for unusual metamorphic operations for computer animation. (E.g. it would be relatively simple to change a glass into a bottle.)

Results

Of the various blending functions with which we have experimented there are certain advantages and disadvantages to each. These are summarized in table 1. A more detailed analysis of the computational efficiency of using one function over another is currently being carried out.

Conclusion

In this paper we have presented a number of techniques that provide good control over the shape of an implicitly defined surface. Various blending functions have been investigated and the results discussed. Other tools, such as the use of surfaces of revolution and noise functions, have been shown along with examples of a variety of blended shapes that can be easily produced. One of features of our approach is that all the tools work systematically so that the blending functions can be applied to any of the skeleton shape functions and any of the different shapes can be blended together to form complex shapes that would extremely difficult or tedious to make with other modeling techniques.

Acknowledgements

We would like to thank all those that have participated in the Graphicsland project at the University of Calgary, particularly Carol Wang, Mike Hermann, Geoff Wyvill (University of Otago) and Jules Bloomenthal (Xerox PARC). This work is partially supported by the Natural Sciences and Engineering Research Council of Canada.

References

[Bei90] Thaddeus Beier. Practical uses for implicit surfaces in animation. In SIGGRAPH '90 Course Notes 23, Modeling and Animating with Implicit Surfaces, August 1990.

[Bli82] J. F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(1):235–256, July 1982.

[Blo88] J. Bloomenthal. Polygonization of implicit surfaces. *Computer Aided Geometric Design*, 5(4):341–355, November 1988.

[Blo90] J. Bloomenthal. Calculation of reference frames along a space curve. In A. S. Glassner, editor, *Grephics Gems*, pages 567–571. Academic Press, 1990.

[BW90] J. Bloomenthal and B. Wyvill. Interactive techniques for implicit modeling. *Computer Graphics*, 24(2):109–116, March 1990.

[Far88] Gerald Farin. *Curves and Surfaces for Computer Aided Geometric Design.* Academic Press, 1988.

[Lew89] J. P. Lewis. Algorithms for solid noise synthesis. *Computer Graphics SIGGRAPH '89*, 23(3):263–270, July 1989.

[MS85] A. E. Midlleditch and K. H. Sears. Blend surfaces for set theoretic volume modeling systems. *Computer Graphics SIGGRAPH '85*, 19(3):161–170, July 1985.

[Pea85] D. R. Peachey. Solid texturing of complex surfaces. *Computer Graphics SIGGRAPH '85*, 19(3):279–286, July 1985.

[Per85] K. Perlin. An image synthesizer. *Computer Graphics SIGGRAPH '85*, 19(3):287–296, July 1985.

[RO87] A. P. Rockwood and J. C. Owen. Blending surfaces in solid modeling. In G. E. Farin, editor, *Geometric Modeling: Algorithms and New Trends*, pages 367–383. SIAM, 1987.

[TPF89] Demetri Terzopoulos, John Platt, and Kurt Fleischer. Heating and melting deformable models (from goop to glop). *Proc. Graphics Interface 1989*, pages 219–226, 1989.

[War89] Joe Warren. Blending algebraic surfaces. *ACM Transactions on Graphics*, 8(4):263–278, October 1989.

[WMW86a] B. Wyvill, C. McPheeters, and G. Wyvill. Animating soft objects. *Visual Computer*, 2(4):235–242, August 1986.

[WMW86b] G. Wyvill, C. McPheeters, and B. Wyvill. Data structure for soft objects. *Visual Computer*, 2(4):227–234, August 1986.

[Wyv88] Brian Wyvill. The Great Train Rubbery. *SIGGRAPH 88 Electronic Theatre and Video Review*, Issue 26, 1988.